Password Policy

Introduction

MidPoint associates password policies with an attribute that it applies to. E.g. a password policy is associated with the user password property, different password policy may be associated with a Addressbook resource password attribute, etc. This approach helps midPoint to validate and generate the value. Password policies are specified in a form of *value policy*. Value policy is a more generic term because such policies may also be applied to other values not just the passwords.

Value policies in midPoint are almost entirely declarative and there is a good reason for it. MidPoint not only needs to validate the passwords but it often needs to generate them as well. It is very easy to construct a password policy language that is used to validate the passwords. This might even contain pieces of scripting code. But that will not work for generating the value. The password generator needs to know precisely what characters are allowed in the password, how much of them is allowed and how much is required and how they are positioned (e.g. first character must be a letter). Only then can the password generator efficiently choose and shuffle the characters to generate the password.

Password Policies

There are two types of password policy:

- Global password policy used for user's password. This is referenced from the system configuration object.
- Account type password policy used for resource's account. This is referenced from the resource definition.

Value Policy

Value policy definition is an object in midPoint repository that describes rules for generating and validating passwords. The simple value policy example can be expressed in the XML as:

```
<valuePolicy oid="00000000-0000-0000-0000-000000000003" version="0" xmlns="http://midpoint.evolveum.com/xml/ns
/public/common/common-2a"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <name>Global Password Policy</name>
        <description>Global password policy</description>
        <stringPolicy>
                <description>Testing string policy</description>
                <limitations>
                        <minLength>5</minLength>
                        <maxLength>8</maxLength>
                        <minUniqueChars>3</minUniqueChars>
                        <limit>
                                <description>Alphas</description>
                                <minOccurs>1</minOccurs>
                                <maxOccurs>5</maxOccurs>
                                <mustBeFirst>false</mustBeFirst>
                                <characterClass>
                                        <value>abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
                                </characterClass>
                        </limit>
                        <limit>
                                <description>Numbers</description>
                                <minOccurs>1</minOccurs>
                                <maxOccurs>5</maxOccurs>
                                <mustBeFirst>false</mustBeFirst>
                                <characterClass>
                                        <value>1234567890</value>
                                </characterClass>
                        </limit>
                </limitations>
        </stringPolicy>
</valuePolicy>
```

The <lifetime> section describes policies for password expiration. Sections <stringPolicy> and <limitations> describe policies that the password must satisfy. The minimal, maximal length of the password and the minimal number of unique characters used in the password can be specified. The following example shows a very simple policy which is only limiting the password length and character uniqueness and not the characters that may be used (as the matter of fact, any character may be used):

```
<stringPolicy>
    <description>Testing string policy</description>
    <limitations>
        <minLength>5</minLength>
        <maxLength>8</maxLength>
        <minUniqueChars>3</minUniqueChars>
    </limitations>
</stringPolicy>
```

With the above definition, the password **p123** would be rejected because it's too short; the password **longpassword** would be rejected because it's too long and the password **bubub** would be rejected because of the inssuficient unique characters. If the above mentioned password policy is used for generating password, the password will contain alphanumeric characters only. It's a default character class when no other is specified.

To define further restrictions, you can define character set(s) by <limit> sections. Only characters explicitly defined may be used in the password and may be further restricted. All other characters are considered to be illegal or invalid. The only exception is the very simple policy above: if there are no <limit> sections, all characters are allowed.

The following policy restricts the password to contain only digits (1234567890) with at least 1 and at most 5 digits it can be specified as in the following example:

```
<stringPolicy>
    <description>Testing string policy</description>
    <limitations>
        <minLength>5</minLength>
        <maxLength>8</maxLength>
        <minUniqueChars>3</minUniqueChars>
            <limit>
                        <description>Numbers</description>
                        <minOccurs>1</minOccurs>
                        <maxOccurs>5</maxOccurs>
                        <mustBeFirst>false</mustBeFirst>
                        <characterClass>
                                <value>1234567890</value>
                        </characterClass>
            </limit>
        </limitations>
</stringPolicy>
```

With the above definition, the password **1234** would be rejected because it's too short; the password **1234567890** would be rejected because it's too long, the password **101010** would be rejected because of the inssuficient unique characters and the password **anne108** would be rejected because it contains other-than-defined characters (non-digits).

To make the password policy even more complex, you can split the character set to uppercase letters, lowercase letters, digits and special characters as in the following example:

```
<stringPolicy>
    <description>Testing string policy</description>
    <limitations>
        <minLength>5</minLength>
        <maxLength>8</maxLength>
        <minUniqueChars>3</minUniqueChars>
    <limit>
        <description>Lowercase characters</description>
        <minOccurs>1</minOccurs>
        <mustBeFirst>true</mustBeFirst>
        <characterClass>
            <value>abcdefghijklmnopqrstuvwxyz</value>
        </characterClass>
    </limit>
    <limit>
        <description>Uppercase characters</description>
        <minOccurs>1</minOccurs>
        <mustBeFirst>false</mustBeFirst>
        <characterClass>
            <value>ABCDEFGHIJKLMNOPQRSTUVWXYZ</value>
        </characterClass>
    </limit>
    <limit>
        <description>Numeric characters</description>
        <minOccurs>1</minOccurs>
        <mustBeFirst>false</mustBeFirst>
        <characterClass>
            <value>1234567890</value>
        </characterClass>
    </limit>
    <limit>
        <description>Special characters</description>
        <minOccurs>1</minOccurs>
        <mustBeFirst>false</mustBeFirst>
        <characterClass>
            <value> !"#$%&amp;'()*+,-.:;&lt;&gt;?@[]^_`{|}~</value>
        </characterClass>
    </limit>
        </limitations>
</stringPolicy>
```

With the above definition, the password **pAs1!** would be rejected, because it's too short, the password **pAssw0rd!** would be rejected, because it's too long, the password **passw0rd!** would be rejected, because it does not contain at least one uppercase letter, the password **PASSW0RD!** would be rejected, because it does not contain at least one lowercase letter and does not start with the lowercase letter, the password **Passw0rd!** would be rejected, because it does not start with the lowercase letter, the password **passWord!** would be rejected, because it does not contain any digit, and the password **pa ssW0rd** would be rejected because it does not contain at least one special character.

On the other way, with the above definition, the password **p#s5worD** would be accepted.

To disallow the usage of certain characters, you can either remove them from the <characterClass> definition, remove the <limit> section or you can set both the <minOccurs> and <maxOccurs> attribute values to 0.

Global password policy is specified in the global security policy.

The account type password policy is specified in the resource in the section schemaHandling as in the following example:

```
<c:resource oid="ef2bc95b-76e0-48e2-86d6-3d4f02d3fafe">

        <!-- Resource name. It will be displayed in GUI.  -->
        <c:name>Localhost CSVfile</c:name>

        <!-- connector configuration -->

        <!-- schema definition -->
        <schemaHandling>

            <!-- schema handling for different attributes -->
            <credentials>
                 <password>

                     <!-- outbound/inbound for password -->

                     <passwordPolicyRef oid="81818181-76e0-59e2-8888-3d4f02d3ffff" type="c:PasswordPolicyType"/>

                  </password>
            </credentials>

                ...

            </accountType>
        </schemaHandling>
    </c:resource>
```

Different account types in resource can have different password policies. If there is no password policy for the account type, the global password policy is used to validate the account password.

Check Expression

Since midPoint 3.6

Additional check expression can be specified in the string policy limitations. The value will be accepted only if the expression returns true. Additional failure message may also be specified.

There are two variables available to the expression:

variable name	content
`input`	Password to be validated. Or generated password candidate in password generation scenarios.
`object`	User in case that the user password is changed/generated. Shadow in case account password is changed/generated.

If the expression returns true then the password is accepted. If the expression returns false (or anything else) then the password is refused.

There may be more than one check expression. In that case all the expressions must pass for the value for the value to be accepted (*AND* operation is assumed).

The following example is checking password for the presence of several user properties:

```
    <stringPolicy>
        <limitations>
            ...
            <checkExpression>
                <expression>
                    <script>
                        <code>
                            if (object instanceof com.evolveum.midpoint.xml.ns._public.common.common_3.
UserType) {
                                return !basic.containsIgnoreCase(input, object.getName()) &amp;amp;&amp;amp; !basic.
containsIgnoreCase(input, object.getFamilyName()) &amp;amp;&amp;amp; !basic.containsIgnoreCase(input, object.
getGivenName()) &amp;amp;&amp;amp; !basic.containsIgnoreCase(input, object.getAdditionalName())
                             } else {
                                 return true
                             }
                         </code>
                     </script>
                </expression>
                <failureMessage>must not contain username, family name and given name and additional names<
/failureMessage>
            </checkExpression>
            ...
        </limitations>
    </stringPolicy>
```

Prohibited Values

Since midPoint 3.7

Value policy may be used to define values that are prohibited. For example if the value policy is used as a password policy, the password will be rejected if an attempt is made to set to any of those values. There is a special section of value policy for specification of prohibited values:

```
 <valuePolicy>
    ...
    <prohibitedValues>
        <item>
            <origin>persona</origin>
            <path>credentials/password/value</path>
        </item>
        <item>
            <origin>owner</origin>
            <path>credentials/password/value</path>
        </item>
    </prohibitedValues>
</valuePolicy>
```

Definition of prohibited values is composed from a set of prohibited *item* definitions. Each item defines:

- Origin object from the item values should be taken:
    - **object**: the object of the change. Usually the user whose password is changed.
    - **persona**: any persona of the object. This usually means persona linked to the user whose password is changed.
    - **owner**: owner of the object. This usually means the physical user who links to the persona in case persona password is changed. This also means owner of account (user) in case account password is changed.
    - **projection**: one of the projections of the object or "sibling" projects (if the object is a shadow). See below. (since midPoint 3.7.1)
- Path of the item. The path will be used to get the value from the origin object (user, persona, owner, ...)

The value will pass the validation only if it does not match any value of any item in any of the objects. Even a single match with any value will mean that the validation will fail.

The example above is a password policy that prohibits linked personas to have the same password. When user password is changed then it is checked that it is different than all persona passwords (this is given by the origin=persona part). This also works the other way: if persona password is changed, it is checked that the password is different that the password of a user who owns the persona (given by the origin=owner part).

This is supposed to be a generic feature. It should be theoretically used to prohibit values from the user profile. It may be extended for checking account values, make an approximate matching (in some cases) and so on. But currently (midPoint 3.7) the only supported case is the case given by the example above. This is the policy that prohibits user to have the same password as the persona.

Limited feature

This is a limited midPoint feature. This feature currently supports only some specific use-cases. We are perfectly capable to finish the feature, just the funding for the work is needed. Please consider the possibility for supporting development of this feature by using midPoint Platform subscription. If you are midPoint Platform subscriber and this feature is within the goals of your deployment you may be able to use your subscription to endorse implementation of this feature.

Prohibited Projection Values

Since midPoint 3.7.1

Prohibited value specification may be used to prohibit usage of a value from a different projection. This feature may be used for example to prohibit resource password to be the same as a password on another resource. This specific case may be specified as follows:

```
<valuePolicy>
    ...
    <prohibitedValues>
        <item>
            <origin>projection</origin>
            <path>credentials/password/value</path>
            <projectionDiscriminator>
                <resourceRef oid="f4fd7e90-ff6a-11e7-a504-4b84f92fec0e"/>
                <kind>account</kind>
            </projectionDiscriminator>
        </item>
    </prohibitedValues>
</valuePolicy>
```

This password policy will prohibit use of the password which is the same as the password for default account on resource identified by OID `f4fd7e90-ff6a-11e7-a504-4b84f92fec0e`.

Password policy specification is quite straightforward. However for this feature to work midPoint must be able to compare projection passwords (account passwords). As passwords are usually write-only attributes the comparison is not trivial. There are several ways how to compare passwords - in theory. However, midPoint currently (3.7.1) supports only one method: caching. This password policy specification works only if password value caching is enabled on the resource:

```
<resource oid="f4fd7e90-ff6a-11e7-a504-4b84f92fec0e">
    ...
    <schemaHandling>
        <objectType>
        ...
        <credentials>
                <password>
                    <compareStrategy>cached</compareStrategy>
                    <caching>
                        <cachingStategy>passive</cachingStategy>
                    </caching>
                </password>
            </credentials>
            ...
        </objectType>
    </schemaHandling>
    ...
</resource>
```

Please note that this has to be configured on the resource which is the *target* of the comparison which is **not** the resource where the password policy is used. This will also work only if all passwords of all accounts are cached. The passwords are cached only if the account password is changed by using midPoint (e.g. midPoint self-service user interface) because that is the only moment when midPoint is able to see password cleartext. The stored (cached) password are always stored in hashed form.

Limitations

Current password policy implementation has some limitations:

- Prohibited values are currently supported only to prohibit same passwords between users, personas and projections. It is possible that the prohibited values method will also work with (some) properties and attributes, but this is currently not tested and not supported.
- Prohibited projection values only work when password caching is enabled and all passwords are properly cached.
- Prohibited projection values may not work in case that the user is created together with projections. In that case it is possible to set the same password for the projections even if the policy specifies it as s prohibited value. The policy will work as expected once the user and projections are created and the password is set or changed (including account initialization scenarios when using password hashing).
- The use of prohibited projection values in user password policy is only partially tested. This feature is currently supported only when applied to resource password policy.
- Currently midPoint user interface may limit usefulness of this feature (e.g. limited capability to set account password individually using credentials self-service page).

Limited feature

This is a limited midPoint feature. This feature currently supports only some specific use-cases. We are perfectly capable to finish the feature, just the funding for the work is needed. Please consider the possibility for supporting development of this feature by using midPoint Platform subscription. If you are midPoint Platform subscriber and this feature is within the goals of your deployment you may be able to use your subscription to endorse implementation of this feature.

See Also

- Security Policy Configuration
- Password-Related Configuration