Script Expression

Languages

Script expressions can use a variety of scripting languages. Currently supported languages are:

- Groovy
- ECMAScript (JavaScript)
- Python

(XPath version 2 was supported until midPoint 3.5, but it is currently not supported)

Script Expression Structure

All the script expressions have the same internal structure no matter where and how they are used. The structure is illustrated in the following code snippet.

```
<expression>
    <script>
        <language>http://midpoint.evolveum.com/xml/ns/public/expression/language#Groovy</language>
        <code>
            'Mr. ' + user.getFamilyName();
        </code>
    </script>
</expression>
```

Meaning of individual script expression fields is as follows:

Field		Description
`language`	optional	Language URL. Specifies the language in which the expression is written. If not specified it defaults to Groovy language.
`trace`	optional	Explicitly trace the execution of this expression. The execution will be logged on INFO level if this property is set to true to make sure that it will be visible in the logs. Available since midPoint 3.5.
`returnType`	optional	The type of the expression result, either "list" or "scalar". If not set it will be determined automatically from the target data type for the expression. This setting should be used only if the automatic mechanism does not work as expected.
`relativityMode`	optional	Values: relative (default) or absolute. See below for more details. If not specified and allowed by the expression usage it defaults to relative mode.
`includeNullInputs`	optional	If set to true (which is the default) the script will be evaluated with null value as argument when the input is changed from or to an empty value. This generally works well for most cases. It may be set to false as an optimization.
`code`	mandatory	Expression code. Depends on the script language used for the expression. This is usually a string, but it also may be an XML. Please note that the code is embedded in the XML therefore proper XML escaping is required

Variables

The expressions used in midPoint are usually using variables that are set up by midPoint expression engine. For example, the following expression will evaluate to the content of the `fullName` property of a user (Groovy and Javascript):

```
<code>
   user.getFullName()
</code>
```

See Expression page for more generic information about the use of variables in expressions.

Functions

See MidPoint Script Library for more details.

Absolute and Relative Script Expressions

MidPoint always works with relative changes. MidPoint expressions are built for this mode of operation. The expression will receive every individual value on input and it has to transform that to output. MidPoint will take care of the logic around it. If the value was removed in the input, the result will be delete delta. If the value was added in the input the result will be add delta. MidPoint takes care of all that. The administrator only needs to specify an expression that transforms one value.

However, in some cases it might be useful to handle all values at once in an absolute way. Imagine, that you have multi-value UID attribute in LDAP and you want to select the right value based on DN. Then you need list of all the values. On the other hand, in relative approach, each value in UID attribute would be processed individually as a String.

```
<expression>
        <script>
                <relativityMode>absolute</relativityMode>
                <code>
             //...
                </code>
        </script>
</expression>
```

The input to this script is a list of all input values. The output is a list of new values. MidPoint always operates in relative mode, therefore at the end of the evaluation midPoint will diff the value to create a delta. This does not change anything about that. However this mode of expression operation can be an advantage if you need to process all values as a group instead of processing every value one by one. E.g. in case that you want to choose a particular value or your algorithm depends on other values in some way.

Security of Script Expressions

Script expressions are a code that runs inside midPoint servers. As such, script expressions are incredibly powerful. But with great powers comes great responsibility. Script expressions can do a lot of useful things, but they can also do a lot of harm. There are just a few simple internal safeguards when it comes to expression evaluation. E.g. midPoint script libraries will properly enforce authorization when executing the functions. However, script languages are powerful and a clever expression can find a way around this safeguards. MidPoint is **not** placing expressions in a sandbox, therefore expressions are free to do almost anything. The sandbox is not enforced from complexity and performance reasons, but it may be applied in future midPoint versions if necessary. For the time being, please be very careful who can define expressions in midPoint. Do not allow any untrusted user to modify the expressions.

See Script Expression Sandboxing for more details.

Future

The expressions are designed to be extensible and the expression language is not fixed. New expression languages may come in the future if there is a demand for them.

See Also

- Mappings and Expressions
- Expression
- Mapping
- XPath2 Tutorial