PolyString Improvements

Planned feature

This page describes a feature planned for future midPoint versions. Part of this functionality is implemented in midPoint 4.0, but it is strictly limited only to a very specific use-cases.

However, there is currently no specific plan for a complete implementation because there is no funding for this development yet. In case that you are interested in supporting development of this feature, please consider activating midPoint Platform subscription.

- Introduction
- Future
  - PolyString Localization
    - Implementation
- PolyStrings in MidPoint 5.0
- See Also

Introduction

PolyString is an unusual kind of animal. It is a built-in data type for *polymorphic string*. This string maintains extra values in addition to its original value. The extra values are derived from the original value automatically using a *normalization* code.

See PolyString page for an introduction to PolyStrings.

Future

PolyString Localization

MidPoint has a fully localized user interface. This means that all the strings used in the user interface are localized. User interface is using keys instead of final strings and the keys are translated to a specific language by using translation catalogs. Such localization works well for all the strings that are present at *compile time*. But there are many things in midPoint that are configured during the deployment. E.g. it is often desired to name objects in multiple languages. Maybe we need names of the roles in several languages. Or names of organizational units. Those objects are not present at compile time. Therefore ordinary localization techniques cannot be used on their own.

We have considered this issue very early in midPoint design. And it was one of the motivations to introduce PolyString. The idea is that PolyString could be used to store language mutations. Simplest form is to specify strings for all supported languages:

```xml
<role>
  <name>
    <orig>System administrator</orig> <!-- This will be used for languages that do not have translations. -->
    <en>
      <orig>System administrator</orig>
    </en>
    <sk>
      <orig>Systémový správca</orig>
    </sk>
    <cz>
      <orig>Správce systémů</orig>
    </cz>
  </name>
  ...
</role>
```

There may be a cases, when the object name is in fact already part of the localization catalogs, e.g. if the object refers to some well-known object in midPoint. In that case the PolyString can simply refer to a localization key:
Or there may be a more complex case that includes translation parameters:

```
<case>
  <name>
    <translation>
      <key>ObjectSpecification</key>
      <argument>
        <translation>
          <key>ObjectTypeLowercase.RoleType</key>
          <fallback>RoleType</fallback>
        </translation>
      </argument>
      <value>Role1</value>
    </translation>
  </name>
  ...  
</case>
```

**Implementation**

The algorithm is supposed to work like this:

1. Values from `lang` are processed. If there is a suitable value for current language then this value is displayed. The values of `lang` are supposed to always override anything else in polystring.
2. Translations are processed. MidPoint looks in the catalog files for translation keys. In case that a particular translation key is not found, the `fallback` will be used. In case that fallback is not specified for any inner (nested) translations, then the translation key itself will be used. In case of top-level translation the behavior is different: if the key cannot be found and there is no fallback then the `orig` value will be used.
3. The `orig` value of the polystring is used as default value. It will be used in case that there is no suitable `lang` for a particular language, there is no translation or that the translation does not provide meaningful result (e.g. the translation key cannot be found).

**PolyStrings in MidPoint 5.0**

It is likely that PolyString will become containers in midPoint 5.0. It has to be 5.0 because major schema improvements are needed to support PolyString as container.

See [MidPoint 5.0 Vision](#) for more information.

**See Also**

- Using Prism Objects
- PolyString Normalization Configuration
- Localization Improvements