Example

Playback - Set fade-in time - modular

by:	Sebastian Beutel, April 2021
published:	here
description:	set some playbacks' fade-in time, modular macros
remarks:	these macros incorporate some newer syntax details: instruction blocks, custom variables, macros refered in other macros

playback, fade-in, time, modular, variable, new, syntax, custom, blocks

The basic functions are best understood when comparing with Playback - Set fade-in time. When using `Handles.SetSourceHandle()` which always refers to the handle index on the current page you need to make sure that there is actually something suitable present in this location - hence, some steps need to be conditional. Additionally when you want to make such macros for n different times and m different playbacks you need to repeat many lines of code.

Using newer syntax with code blocks and custom variables this is way easier: here we create only one macro to set all relevant playbacks' fade time to a variable value (using the block syntax to put a few instructions under one condition). Additionally a number of rather short macros simply sets the time variable to a certain value and then calls the 'apply' macro:

functions

- Handles.SetSourceHandle
- Playbacks.IsCueHandle
- ActionScript.SetProperty
- Playbacks.Editor.SelectLiveCue
- Handles.ClearSelection
- UserMacros.RecallMacroById

affected properties

- Handles.SourceHandle
- Playbacks.Editor.SelectedPlayback
- Playbacks.Editor.Times.CueFadeInTime

control structures

- step condition

specials

- code blocks
- custom variables
- refering macros

Code

SetFadeInNew.xml

```
<?xml version="1.0" encoding="utf-8"?>
<avolites.macros>
  <macro id="Wiki.Macros.SetFadeIn" name="Set Fade In (2)">
    <variables>
      <object id="Time" type="Avolites.Acw.Titan.AcwTimeSpan"
assembly="Avolites.Acw.Titan" value="2" />
    </variables>
    <sequence>
      <step>
        {
          Handles.SetSourceHandle("PlaybackWindow", 0);
          if (Playbacks.IsCueHandle(Handles.SourceHandle) == true)
          {
ActionScript.SetProperty("Playbacks.Editor.SelectedPlayback",
Handles.SourceHandle);
            Playbacks.Editor.SelectLiveCue();
ActionScript.SetProperty("Playbacks.Editor.Times.CueFadeInTime",
Wiki.Macros.SetFadeIn.Time);
          }
          Handles.ClearSelection();
        }
      </step>
      <step>
        {
          Handles.SetSourceHandle("PlaybackWindow", 1);
          if (Playbacks.IsCueHandle(Handles.SourceHandle) == true)
          {
ActionScript.SetProperty("Playbacks.Editor.SelectedPlayback",
Handles.SourceHandle);
            Playbacks.Editor.SelectLiveCue();
ActionScript.SetProperty("Playbacks.Editor.Times.CueFadeInTime",
Wiki.Macros.SetFadeIn.Time);
          }
          Handles.ClearSelection();
        }
      </step>
      <step>
        {
          Handles.SetSourceHandle("PlaybackWindow", 2);
          if (Playbacks.IsCueHandle(Handles.SourceHandle) == true)
          {
ActionScript.SetProperty("Playbacks.Editor.SelectedPlayback",
Handles.SourceHandle);
            Playbacks.Editor.SelectLiveCue();
```

```
ActionScript.SetProperty("Playbacks.Editor.Times.CueFadeInTime",
Wiki.Macros.SetFadeIn.Time);
          }
          Handles.ClearSelection();
        }
      </step>
    </sequence>
  </macro>
  <macro id="Wiki.Macros.SetFadeIn0" name="Set Fade In 0">
    <sequence>
      <step>ActionScript.SetProperty("Wiki.Macros.SetFadeIn.Time",
time:0)</step>
      <step>UserMacros.RecallMacroById("Wiki.Macros.SetFadeIn")</step>
    </sequence>
  </macro>
  <macro id="Wiki.Macros.SetFadeIn1" name="Set Fade In 1">
    <sequence>
      <step>ActionScript.SetProperty("Wiki.Macros.SetFadeIn.Time",
time:1)</step>
      <step>UserMacros.RecallMacroById("Wiki.Macros.SetFadeIn")</step>
    </sequence>
  </macro>
</avolites.macros>
```

Explanation

This explains the functional steps within the sequence. For all the other XML details please refer to Formats and syntax

The **Main macro** `Wiki.Macros.SetFadeIn` defines a custom variable `Wiki.Macros.SetFadeIn.Time`, than selects a number of playbacks, and sets them to this time.

- defining the variable is done in the block <variables>...</variables>
- in order to define a variable you need to
  - give it an **id** (this also defines the 'full name' of the variable: it's `macro id.variable id` This also means that if you change the id of the macro you need to change the variable name wherever it is called, e.g. where the variable is set, changed, or read.
  - the **type** defines the type of the variable, see also Types
  - the **assembly** is some dotnet specific thing (maybe the variable's scope). I'd guess that `Avolites.Acw.Titan` should be a good starting point when testing other variables.
  - the **value** is the initial value the variable is set to when being defined.
- after this, selecting a playback and setting it's time is all done in one step thanks to the code blocks syntax:
  - multiple instructions inside one step are bundled with { curly braces }
  - instructions inside code blocks are separated with semicolons ;
  - code blocks are also possible as conditional statements, in a very traditional way like

if (condition)

```
    {
      code being executed only if condition is true
    }
```

How to use it

1. make this macro available

tbd

From:
https://avosupport.de/wiki/ - **AVOSUPPORT**

Permanent link:
https://avosupport.de/wiki/macros/example/setplaybackfadeintimemodular?rev=1626876577

Last update: **2021/07/21 14:09**