Von: Gregory Haynes
Gesendet: Mittwoch, 13. April 2022 23:26
An: Sebastian Beutel
Betreff: RE: timeline macro questions

Hello,

I'm doing pretty well, we seem to be getting out of winter now and soon it will be the summer and festival season, so it will be good to see that happening again.

I hope you are keeping well, I assume you might be doing something at Prolight and Sound in a couple of weeks (not that I'm involved as I believe we are keeping things pretty low key).

I will try to explain what all these different time objects are:

Type	Description
System.TimeSpan	Is a general type built into the Microsoft .NET Framework which stores a length of time which can be retrieved as hours, minutes, seconds, milliseconds either as totals or as separate components. A lot of our time values make use of this internally and add additional properties.
Avolites.Acw.Titan.AcwTimeSpan	This is the time value that we use for most things such as fade times, release times etc.. It is basically a System.TimeSpan plus a field which indicates special values such as AsIn, NotUsed, Global etc. which, indicates that this isn't a real value and should be ignored or looked up elsewhere.
Avolites.Acw.Titan.TimecodeTime	The name pretty much gives away what this is for, this is basically a System.TimeSpan but with an additional field to indicate the frame rate and whether or not it is enabled. We used to change how we saved the frame value depending on what the frame rate was however more recently the time value is always stored in milliseconds and it is converted to the appropriate frame rate only when we display it. This is usually displayed in full with hours and minutes so is usually preferred for larger time values.
Avolites.Titan.Controllers.Editor.Playbacks.ObservableTimecodeTime	This is practically speaking identical to Avolites.Acw.Titan.TimecodeTime however provides some necessary linking that ensures that whenever part of the value is changed (even if it is just the minutes or seconds part and not the whole value) the necessary events trigger to propagate the change. This is only relevant the UI where you might be wheeling them or typing in and when that happens this ensures that the engine gets updated.

System.Int32 (int) System.Single (float)	Sometimes just having a simple number makes sense, either if it is a value being typed in for the first time by the user or in low level functions. However they are generally avoided when passing data around as they could mean different things in different places, could be a number of minutes, seconds or milliseconds and that sort of ambiguity is the sort of thing that leads to bugs.

I had a look at the `UpdateTimelineViewProperties` function and I don't think it is really usable by WebAPI or macros; you can blame me from 4.5 years ago for marking it as a menu function for some unknown reason. It isn't really possible to convert to or change `System.TimeSpan` values in WebAPI or macros, the best option would be to change it so it uses our standard types, ideally the same as the corresponding context timeline properties.

What I can suggest as an alternative is taking advantage of the functions provided for the wheel view. This will require changing the current 'page' of timeline wheels but doesn't actually require the Timeline window to be open or active and the wheels can still be showing attributes or connected playback etc..

To set the horizontal position to a specific time you can do the following:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.SetWheelValue(0,
Timecode.AsObservable(Timecode.MakeTimecodeTime(1, 0, 24, 0, false, 25)), false)
```

To set the zoom to the length of time you want to display:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.SetWheelValue(2, time:15, false)
```

The values for incrementing aren't quite as straightforward but the following will move a screen full (i.e. `Editor.Timelines.ContextTimeline.ViewWidth`) to the left:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.IncrementTriggerWheelParameter(0, 64, 0, true)
```

And to move to the right:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.IncrementTriggerWheelParameter(0, -64, 0, true)
```

You can adjust the second parameter as required, 64 is the number of wheel counts for a full rotation if you were wondering where that comes from.

Vertical scrolling, second parameter is number of pixels:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.IncrementTriggerWheelParameter(1, 10, 0, true)
```

As it will limit scroll values that are too big or small you can scroll to the very beginning or end by making use of the `MaxVerticalOffset` property:

```
Editor.Timelines.IncrementTriggerWheelParameter(1,
float:Editor.Timelines.MaxVerticalOffset, 0, true)
Editor.Timelines.IncrementTriggerWheelParameter(1, 0 -
float:Editor.Timelines.MaxVerticalOffset, 0, true)
```

Note `SetWheelValue` has no function for vertical scrolling as we didn't expect people to enter that as a value directly.

Incrementing the zoom goes in 5% increments or decrements, the second parameter determines the direction but is otherwise ignored:

```
ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.IncrementTriggerWheelParameter(2, 1, 0, false)

ActionScript.SetProperty.Enum("Editor.Timelines.WheelView", "ScrollZoom")
Editor.Timelines.IncrementTriggerWheelParameter(2, -1, 0, false)
```

Enabling the fast option will change the increments to 15.8%.

Regarding the `Timelines.ToggleTrackViewMode` function, as far as I can tell it was never implemented, prior to it being removed the function was entirely empty apart from a comment saying “TIMELINETODO”. Presumably we thought that we wanted this function at some point but never got around to writing it and was subsequently removed as it wasn’t doing anything. Unfortunately from what I can see the only way of changing the view mode of the tracks is from within the Timeline window itself which means it is out of reach of macros and WebAPI. In principle it would be possible to write that function but actually writing it would be more helpful than leaving it empty.

Hopefully some of this information will be helpful to you.

Gregory

--
Gregory Haynes
Senior Developer

www.avolites.com

From: Sebastian Beutel
Sent: 11 April 2022 08:18
To: Gregory Haynes
Subject: timeline macro questions

Hi Gregory,

May I please bother you with a few more macro questions? I was helping a guy to do some timeline stuff. I.e. make macros to set the zoom and offset of the timeline view, minimize tracks, etc. Here are the two macros in question:

```
  <macro id="rp.Macros.SetTimelineViewWidthTo120"
name="SetTimelineViewWidthTo120">
      <sequence>
        <step>ActionScript.SetProperty("EditorTimelines.ContextTimeline.ViewWid
th", time:120)</step>
        <step>Editor.Timelines.UpdateTimelineViewProperties(Timelines.ContextTi
meline.ViewTimeOffset, Timelines.ContextTimeline.ViewVerticaOffset,
Timelines.ContextTimeline.ViewWidth)</step>
      </sequence>
  </macro>

  <macro id="rp.Macros.ToggleTrackViewMode1" name="ToggleTrackViewMode1">
    <description>Toggle Track View Mode 1.</description>
      <sequence>
      <step>Timelines.ToggleTrackViewMode(timelineId1, trackId1)</step>
      </sequence>
  </macro>
```

The first macro is intended to set zoom and offset. However regardless which values I put in there it doesn't work. I tried numbers like

```
Editor.Timelines.UpdateTimelineViewProperties(0, 20, 120)
```

as well as times

```
Editor.Timelines.UpdateTimelineViewProperties(time:1, 20, time:120)
```

But neither works. The most interesting error was 'Cannot convert type "Avolites.Acw.Titan.AcwTimeSpan" into type "System.TimeSpan".' Is there a way how offset and width can be set by macro at all, and if so, how?

The second macro is even more interesting: the function '`Timelines.ToggleTrackViewMode`' is mentioned in the v14 API documentation, but has gone in v15, and the logs say the same ("The action script function was not found…"). Again, is there a way to set or toggle track view modes in v15 at all?

Thanks in advance for a little help.

And another question only out of interest: what is the 'observable' property for timecodes? (We made a macro to set the cursor to a specified position with

```
ActionScript.SetProperty("Editor.Timelines.ContextTimeline.CursorPosition",
Timecode.AsObservable(Timecode.MakeTimecodeTime(1, 00, 00, 00, false, 100)))
```

and I am just wondering what this additional thing is…

Best regards, Sebastian