Example

Playback Groups - Create and Add

by:	Sebastian Beutel, June 2021
published:	here
description:	basic functionality: creating a playback group and adding some playbacks
remarks:	with great support by Gregory Haynes

PlaybackGroup, group, create, add, ienumerable

Playback Groups were introduced with Titan v13, January 2020. This functionality is not available in earlier versions.

Gregory Haynes' email which this these examples are based on is attached as

email_07062021.pdf

for further details.

This example is meant to show how Playback Groups are created, how playbacks are added, how the various parameters are conveyed, and which caveats need to be taken care of. Particular attention should be paid to setting IEnumerables.

functions

- Handles.SetSourceHandleRange
- Handles.ClearSelection
- PlaybackGroups.CreatePlaybackGroupWithPlaybacks
- Playbacks.FilterByPlaybackHandle
- PlaybackGroups.AddPlaybacksToGroup
- PlaybackGroups.SetCurrentPlaybackGroupFromUserNumber

affected properties

- Handles.ContextHandles
- PlaybackGroups.CurrentPlaybackGroup

Code

PlaybackGroupsBasics.xml

```
<?xml version="1.0" encoding="utf-8"?>
<avolites.macros>

<!-- 1 -->
<!-- adding playbacks selected by location to a new playback group -->
```

```
  <macro id="Wiki.Macros.PlaybackGroups.Test.AddPB12tonewPBG" name="Add Playbacks 1 and 2 to new PBG">
    <sequence>
      <step>Handles.SetSourceHandleRange("Playbacks", {0, 1})</step>
      <step>PlaybackGroups.CreatePlaybackGroupWithPlaybacks("Test", Playbacks.FilterByPlaybackHandle(Handles.ContextHandles))</step>
      <step>Handles.ClearSelection()</step>
    </sequence>
  </macro>

<!-- the new PBG is also current as in PlaybackGroups.CurrentPlaybackGroup -->

<!-- 2 -->
<!-- adding playbacks selected by user numbers (string notation) to the current playback group -->

  <macro id="Wiki.Macros.PlaybackGroups.Test.AddCue3CL1tocurrPBG" name="Add Cue 3 CL 1 to curr PBG">
    <sequence>
<step>PlaybackGroups.AddPlaybacksToGroup(PlaybackGroups.CurrentPlaybackGroup, "{cueHandleUN=3,cueListHandleUN=1}")</step>
    </sequence>
  </macro>

<!-- 3 -->
<!-- adding a playback selected by its user number to the current playback group -->
<!-- this requires Playbacks.FilterByPlaybackHandle() to make sure it is a playback handle -->

  <macro id="Wiki.Macros.PlaybackGroups.Test.AddPB2tocurrPBG" name="Add PB 2 to curr PBG">
    <sequence>
<step>PlaybackGroups.AddPlaybacksToGroup(PlaybackGroups.CurrentPlaybackGroup, Playbacks.FilterByPlaybackHandle(userNumber:2))</step>
    </sequence>
  </macro>

<!-- 4 -->
<!-- adding a playback selected by its user number to a specific playback group selected by its user number -->
<!-- this requires the handle cast with a string (handle:"playbackGroupHandleUN=1") to identify the playback group -->

  <macro id="Wiki.Macros.PlaybackGroups.Test.AddPB3toPBG1" name="Add PB 3 to PBG 1">
    <sequence>
<step>PlaybackGroups.AddPlaybacksToGroup(handle:"playbackGroupHandleUN=1", "cueHandleUN=3")</step>
```

```
    </sequence>
  </macro>

<!-- 5 -->
<!-- adding a playback selected by its user number to a specific
playback group selected by its user number -->
<!-- this time the playback group is selected by pointing
PlaybackGroups.CurrentPlaybackGroup to the user number -->

  <macro id="Wiki.Macros.PlaybackGroups.Test.AddCue3tocurrPBG2"
name="Add Cue 3 to curr PBG 2">
    <sequence>
<step>PlaybackGroups.SetCurrentPlaybackGroupFromUserNumber(2)</step>
<step>PlaybackGroups.AddPlaybacksToGroup(PlaybackGroups.CurrentPlayback
Group, "cueHandleUN=3")</step>
    </sequence>
  </macro>

<!-- 6 -->
<!-- adding playbacks identified by their TitanID to the current
playback group -->
<!-- No practical use as the TitanIDs are not really accessible -->

  <macro id="Wiki.Macros.PlaybackGroups.AddPBsByTitanId" name="Adding
PBs to PBG by TitanID">
    <sequence>
<step>PlaybackGroups.AddPlaybacksToGroup(PlaybackGroups.CurrentPlayback
Group, "{ 1812; 1999 }")</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

These macros were created in order to show various ways to specify single or multiple playbacks. Basically `PlaybackGroups.CreatePlaybackGroupWithPlaybacks()` requires a name for the new group and an IEnumerable of playbacks handles to be added to the new group while `PlaybackGroups.AddPlaybacksToGroup()` requires the playback group handle and - again - an IEnumerable of playback handles.

- playback groups live only in their special window, hence there is no location to identify their handle
- you cannot set the playback group's user number when creating it. Thus you need to rely on the automatic numbering

- selecting a playback group by its user number requires extra casting
- there are some ways to pass handles as list to the required IEnumerable however some details need to be watched

Macro 1

Here the playbacks to be added to a new group are selected via SetSourceHandleRange which takes a handle group (window) and index - this operates always on the current page of this window, the index being 0-based. the set SourceHandleRange is then referenced as `Handles.ContextHandles`. Additionally this needs to be filtered by playback handles: the function `PlaybackGroups.CreatePlaybackGroupWithPlaybacks` lives in the namespace PlaybackGroups and would search for passed handles in its own namespace if not told otherwise.

- `Handles.SetSourceHandleRange("Playbacks", {0, 1})` selects playbacks 1 and 2 on the current playbacks page
- `PlaybackGroups.CreatePlaybackGroupWithPlaybacks()` creates a new playback group and adds some playbacks
  - "`Test`" is the name (legend) of the new playback group
  - `Playbacks.FilterByPlaybackHandle()` makes sure the passed list is assumed in the playbacks namespace, not in PlaybackGroups how the namespace of the function would imply
  - `Handles.ContextHandles` refers to the playbacks being selected with `Handles.SetSourceHandleRange()` in the first step
- `Handles.ClearSelection()` deselects the handles after everything is done

Macro 2

Two playbacks, identified by their user numbers in string notation, are added to the current playback group. Note that this relies on the current playback group being set previously (creating it in the macro above makes it current).

- `PlaybackGroups.AddPlaybacksToGroup()` adds playbacks to a playback group
  - `PlaybackGroups.CurrentPlaybackGroup` is the system property which denotes the currently selected playback group, e.g. still current after previously been created
  - "`{cueHandleUN=3,cueListHandleUN=1}`" is a list of playback user numbers in string notation to make clear these are playbacks of some type.

Macro 3

A playback identified by its user number in standard notation is added to the current playback group (e.g. still current from previously being created, see above). As the playback's user number is in standard notation `Playbacks.FilterByPlaybackHandle()` is required to make sure it is a playback and not a playback group.

- `PlaybackGroups.AddPlaybacksToGroup()` adds playbacks to a playback group
  - `PlaybackGroups.CurrentPlaybackGroup` is the system property which denotes the currently selected playback group, e.g. still current after previously been created
  - `Playbacks.FilterByPlaybackHandle()` makes sure the passed list is looked up in

the playbacks namespace
- `userNumber:2` is the standard notation of a user number when denotin a handle

Macro 4

A playback identified by its user number (see Macro 3) is added to a specific playback group. This is denoted by its user number which requires additional casting.

- `PlaybackGroups.AddPlaybacksToGroup()` adds playbacks to a playback group
  - `handle:“playbackGroupHandleUN=1”` takes the playback group's user number in string notation and cast this into the required handle
  - `“cueHandleUN=3”` is the playback's user number in string notation. Unlike in Macro 2 we don't need braces here as it is only one playback.

Macro 5

A playback identified by its user number (see Macro 3) is added to a specific playback group. This is denoted by pointing PlaybackGroups.CurrentPlaybackGroup to the user number of the playback group - no additional casting required, and the property PlaybackGroups.CurrentPlaybackGroup is set for future operations as well.

- `PlaybackGroups.SetCurrentPlaybackGroupFromUserNumber(2)` points the property PlaybackGroups.CurrentPlaybackGroup to the playback group with user number 2.
- `PlaybackGroups.AddPlaybacksToGroup()` adds playbacks to a playback group.
  - `PlaybackGroups.CurrentPlaybackGroup` holds the previously set reference to playback group 2.
  - `“cueHandleUN=3”` is the playback's user number in string notation, see above. Unlike in Macro 2 we don't need braces here as it is only one playback.

Macro 6

This macro has not been tested.

Similar to Macro 2. Two playbacks, identified by their TitanIDs, are added to the current playback group. Note that this relies on the current playback group being set previously (creating it in the macro above makes it current). As the TitanID is an internal thing and cannot be set manually nor predicted reliably is of limited use here.

- `PlaybackGroups.AddPlaybacksToGroup()` adds playbacks to a playback group
  - `PlaybackGroups.CurrentPlaybackGroup` is the system property which denotes the currently selected playback group, e.g. still current after previously been created, or set like in Macro 5
  - `“{ 1812; 1999 }”` is a list of playback TitanIDs.

How to use it

1. make this macro available

2. macros 1~5 have successfully been tested - simply create some playbacks with the relevant user number and fire the macros to see how they work. However these macros might be useful only in special cases, e.g. in setup macros which prepare the console for you following a fixed scheme.
3. macro 6 has not been tested as using TitanIDs in macros is only of limited use - it's not easy to determine or predict the TitanID of a particular item.

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

Permanent link:
https://www.avosupport.de/wiki/macros/example/playbackgroupcreateadd

Last update: **2026/07/24 12:46**