Identifiers

Handle

ASSUMPTION (see About this wiki)

> Handle is a specific object type used to describe a particular item inside the whole Titan show structure. Examples for handles are: playbacks (the physical faders/buttons, as well as the touch buttons), palette handles (palette buttons, touch or real, as well as palettes addressed by their name/number), fixture handles (again: real buttons or touch buttons or abstract places addressed by their number).

As such, a handle has at minimum these properties (listed here as identifiers themselves as they may be used in various context):

- the titanId
- the Location
- the userNumber

These properties can also be used to identify a particular handle, e.g. to retrieve its other values, to store something onto it, or to delete its contents.

And a more general overview by Gregory Haynes:

Internally Handle is an object which contains the location of an item, i.e. what fader it is on, the internal ID of the item it references (Titan ID) along with other information such as user number and legend. Pretty much all items (playback, groups etc.) will have a handle object to represent them in the UI even if they are not visible or are unassigned. For WebAPI and macros a system was put in place to try to automatically convert/cast parameters to the correct type however for handles this will require enough context to work. So for example you can pass a string e.g. “cueHandleUN2” to a function expecting a handle and it will parse the string and find the corresponding handle, however if the parameter type is ‘object’, i.e. anything, there is no reason to convert and so the string gets passed in as is. A good example of this is the ActionScript.SetProperty function which can take any type and will try to do that even if the property it is trying to set does not support that type.

In addition you specify a specific cast using the type followed by a colon and the value e.g. “userNumber:2”. There is a cast type of titanId however this is just casting to an integer (same as int). This can be used in conjunction with the automated conversion above to yield the correct result for example a function that requires a handle and you pass in the parameter userNumber:2. This will first convert the “2” from the integer or string that it starts as into an AcwUserNumber object which represents a user number (it handles user numbers with decimals without floating point errors). Next immediately before executing the function it will find that it has a user number when it actually requires a handle so will then attempt to find the correct handle and pass this in.

Note that like object, handle is also a somewhat ambiguous item as there can be many different types of handle with the same user number i.e. you can have a fixture, a palette and a playback all with user number 2. To avoid this ambiguity some namespaces specify the default handle type so that they can correctly convert to the correct type of handle.

Examples

In general, where a function needs a handle to be passed, you pass the reference to the handle's location, titanId or userNumber - and Titan will do its best to find the correct handle, and pass this to the function.

select by location

More details for location are here.

```
<step>CueLists.GoBack("Location=Playbacks,2,1")</step>
```

This calls the function CueLists.GoBack and refers it to the handle identified by the Location string "`Location=Playbacks,2,1`": Playbacks, Page 2, Number 1. You may also want to have a look at getTitanIds in order to find the correct locations.

In general, location is passed as string, with strict formating:

```
"Location={string group_id},{int32 page},{int32 index}"
```

tbd. currently we are trying to figure out why the following code **doesn't work**:

```
<step pause="0.01">ActionScript.SetProperty("Palette.CurrentPaletteHandle",
"Location=Colours,2,2")</step>
```

select by titanId

More details for titanId are here.

```
<step>Masters.TapTempo(1607, Math.GetCurrentTimeStamp())</step>
```

This calls the function Masters.TapTempo and refers it to the handle `1607` which is the titanId of BPM Master 1 (`Math.GetCurrentTimeStamp()` refers to the second parameter of this function which is a Timestamp).

select by userNumber

More details for userNumber are here.

```
<step>Playbacks.ReleasePlayback(userNumber:2,  Playbacks.MasterReleaseTime,
true)</step>
```

This calls the function Playbacks.ReleasePlayback for the playback handle with the userNumber 2 (and again some more parameters which are not relevant for this object call).

Here, the userNumber is passed like this:

`userNumber:2`

Technically, this syntax looks like https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/named-and-optional-arguments: you have to name the argument 'userNumber', and after a colon give the value.

Another hint http://forum.avolites.com/viewtopic.php?f=20&t=5997:

> You cannot use the userNumber cast for the IsClaimed function as Handles is used for all types of handles and therefore it doesn't know to what type of handle the user number pertains.

```
Handles.IsClaimed(userNumber:83)
```

does not work but

```
Handles.IsClaimed("cueHandleUN=1")
```

does.

Also used in

- The Syntax of Functions
- AlignSelection.SelectFixture
- ConnectedPlayback.Connect
- CueLists.GoBack
- CueLists.Play
- CueLists.SetNextCue
- Group.FlashFader
- Group.QuickCreateGroup
- Group.RecallGroup
- Group.ReplaceGroupOnHandle
- Handles.ClearHighlight
- Handles.CreateHandleReference
- Handles.GetHandle
- Handles.GetIconId
- Handles.GetTitanIdFromHandle
- Handles.HighlightHandle
- Handles.IsClaimed
- Handles.SetSourceHandleFromHandle
- Include.SelectPlaybackHandle
- Masters.ClearFlash
- Masters.DeadBlackOut
- Masters.DoubleOrHalfSpeedMultiplier
- Masters.NudgeDown
- Masters.NudgeUp
- Masters.ResetSpeedMultiplier
- Masters.SetMaster
- Masters.SetSpeed

- Masters.TapTempo
- Palette.ApplyPalette
- Palette.ApplyQuickPalette
- Palette.QuickMergePalette
- PlaybackGroups.AddPlaybacksToGroup
- Playbacks.AppendOrInsertPlaybackStep
- Playbacks.ClearFlashPlayback
- Playbacks.Editor.CueSelection.SelectAll
- Playbacks.Editor.CueSelection.SelectCueByNumber
- Playbacks.FillCueLegend
- Playbacks.FirePlaybackAtLevel
- Playbacks.FlashPlayback
- Playbacks.IsCueHandle
- Playbacks.MergePlaybackCue
- Playbacks.ReleasePlayback
- Playbacks.Select.EditHandle
- Playbacks.SetCueLegend
- Playbacks.SetPlaybackLevel
- Playbacks.Timecode.ToggleEnabled.Handle
- Playbacks.ToggleBlindPlayback
- Profiles.AssignHandleProfile
- Profiles.GetHandleProfileId
- Selection.Context.Programmer.SelectFixture
- SetList.FireTrack
- Timelines.ImportMarkersFromString
- Triggers.ToggleMappingEnabledByHandle()
- Triggers.SetPendingTargetHandle
- Workspace.Record
- Identifiers
- Properties list
- Chases.ConnectedHandle
- Handles.SourceHandle
- Palette.CurrentPaletteHandle
- PlaybackGroups.CurrentPlaybackGroup
- Playbacks.Editor.Macros.Target.Handle
- Playbacks.Editor.SelectedPlayback
- Playbacks.PlaybackEdit.Handle
- Object
- Parameters