TVST3Instrument API

See UMyVST example.

You will derive a class from TVST3Instrument implementing your plugin. Let’s assume this class is called TMyVSTPlugin. First you make the plugin available to the system by implementing the function GetVST3InstrumentInfo. If all goes well your plugin will be instantiated in your DAW. From there you have control over it with this API.

Declaration:

The function GetVST3InstrumentInfo must be implemented delivering the following information:

uid A unique UUID for your plugin.

cl The class name: here: TMyVSTPlugin. It must me derived from TVST3Instrument.

Ecl The class name for your editor: here: TFormMyVST

isSynth true if this is a synth.

and a few other information fields.

API

The API is split into two parts:

**Processor**

Methods concerned with the audio processing part.

**Controller**

Methods concerned with the parameters, presets and editor

|  |  |
| --- | --- |
| **Processor** | All methods are virtual and optional for processing |
| proc NoteOn(channel,pitch,velocity:integer); |  |
| proc NoteOff(channel,pitch,velocity:integer); |  |
| proc SysexEvent(s:string); | s starts with $F0 and ends with $F7 |
| proc OnMidiCC(channel,cc,value:integer); |  |
| proc Process32(samples,channels:integer;  inputp, outputp: PPSingle); | Here you process your audio. Inputp and outputp are arrays of array with the first subscribe the channel and the second the sampleposition. See example. |
| proc SamplerateChanged(samplerate:single); |  |
| proc PlayStateChanged(playing:boolean;ppq:inte); | Called when the DAW changes play state, or position |
| proc TempoChanged(tempo:single); |  |
| proc OnAutomationReceived  (queue:TVST3AutomationQueue) | The queue contains messages to change one parameter over time. It has properties id, count and can be accesed as an array (giving you a <time,value> pair |

|  |  |
| --- | --- |
| **Controller** |  |
| proc OnInitialize; | Called after creation. Here you should add your parameters |
| proc AddParameter(id:integer;title, shorttitle,units:string;min,max,val:double) | Adds a parameter to the system. Each parameter has an unique id and some other values |
| proc getParameterValue(id:integer):double; | Return the value of the parameter with corresponding id |
| proc updateHostParameter(id,value) | Updates a parameter on the Host |
| proc updateEditorParamer(id,value) | Virtual method called when a Host parameter changes |
| proc ResendParameters | Resends all parameters thrue UpdateEditorParameter |
| function getEditorClass:TformClass; | Virtual method. You can override the default creation of the editor class (as defined in GetVST3InstrumentInfo) here (normally not needed) |
| proc OnEditOpen | Virtual method. Called when the editor opens. It could be wise to call ResendParameters and to create a mechanism so when a UI element changes you send the changed value to the host. |
| proc OnPresetChange(prgm:integer); | Virtual method. Only needed if you want to show the preset number in the UI. |
| proc OnEditClose | Virtual method. For cleanup, but normally not necessarry. |
| proc OnFinalize | Virtual method. For cleanup, but normally not necessarry. |

Missing from the API (will be implemented on request)

* Setting/Getting preset names. Setting the preset number
* MidiOut
* MidiIn stuff like MPC, aftertouch