Updating pre v6.8 Projects

Will Pirkle

This document explains how to update older RackAFX projects to take advantage of new features in the VST plugin support code. This is for both RAFX.DLL as VST.DLL and the Make VST function.

NOTE: If you use the *processAudioFrame( )* function, you don't need to update anything for sample accurate MIDI - this paradigm does not require any extra code.

Sample Accurate MIDI and *processVSTAudioBuffers( )*

RackAFX v6.8 added sample accurate MIDI to its VST projects (previously there was a 32-sample slop). When using *processAudioFrame( )* everything works as expected. However, when using *processVSTAudioBuffer( )*, some code was added to implement the sample accurate MIDI events**. Newly created projects have this code in place**, however older projects need to be updated manually - fortunately it's easy. This update only concerns the case of using *processVSTAudioBuffer( )* with VST plugins.

To update, you need to add some code to handle the *IMidiEventList* interface, which is declared in plugin-constants.h

Step 1: add a member variable and virtual function override for *processRackAXMessage( )* to your .h file; you can insert it after the declaration for *userInterfaceChange( )*

<yourplugin.h>

```
// 5. userInterfaceChange() occurs when the user moves a control.
virtual bool __stdcall userInterfaceChange(int nControlIndex);

// --- declare an EventList interface
IMidiEventList* m_pMidiEventList = nullptr;

// --- grab MIDI interface
virtual void __stdcall processRackAFXMessage(UINT uMessage, PROCESS_INFO&
                                             processInfo);
```

Step 2: pick up the interface pointer in your newly declared message function; add this to your .cpp file:

<yourplugin.cpp>

```
void __stdcall <yourplugin>::processRackAFXMessage(UINT uMessage, PROCESS_INFO&
                                                   processInfo)
{
    // --- always call base class first
    CPlugIn::processRackAFXMessage(uMessage, processInfo);

    // --- for MIDI Event list handling (AU, VST, AAX with processVSTBuffer())
    if (uMessage == midiEventList)
    {
        m_pMidiEventList = processInfo.plMidiEventList;
    }
}
```

Step 3: alter the code in *processVSTAudioBuffer( )* to use the interface. Notice you need to declare a separate counter variable for it, outside the while( )loop:

```
bool __stdcall <yourplugin>::processVSTAudioBuffer(float** inBuffer, ...
{
	// declarations and other stuff
	// …

	// - the loop
	unsigned int uSample = 0;
	while (--inFramesToProcess >= 0)
	{
		// --- fire midi events (AU, VST2, AAX buffer processing only)
		if (m_pMidiEventList)
			m_pMidiEventList->fireMidiEvent(uSample++);

		--- rest of loop as normal
```

That's it - you'll now have sample accurate MIDI for all RackAFX projects (VST, AU, AAX, RAFX) while using the *processVSTAudioBuffer( )* function.

Sample Accurate Parameter Support (Automation in VST3 hosts)

RackAFX v6.8 added sample accurate parameter processing its VST projects, requiring a bit of extra code on the plugin side. This is only for VST3 plugins as VST3 currently is the only API with this feature built in. **Newly created projects have this code in place,** however older projects need to be updated manually - fortunately it's easy. This document is only for updating older, pre v6.8 projects.

To update, you need to add a function called *doVSTSampleAccurateParamUpdates()* to your plugin, and then call it at the top of your processing function.

Step 1: add a member function to your plugin's .h file, you can add it with the other functions at the top of the file:

```
<yourplugin.h>
	// --- function to handle VST sample accurate parameter updates
	void doVSTSampleAccurateParamUpdates();
```

Step 2: implement the function in your .cpp file:

<yourplugin.cpp>

```
/* doVSTSampleAccurateParamUpdates
	Short handler for VST3 sample accurate automation added in v6.8.0.5
	There is nothing for you to modify here.
*/
void <yourplugin>::doVSTSampleAccurateParamUpdates()
{
	// --- for sample accurate parameter automation in VST3 plugins; ignore otherwise
	if (!m_ppControlTable) return; /// should NEVER happen
	for (int i = 0; i < m_uControlListCount; i++)
	{
		if (m_ppControlTable[i] && m_ppControlTable[i]->pvAddlData)
		{
			double dValue = 0;
			if (((IParamUpdateQueue *)m_ppControlTable[i]->
							pvAddlData)->getNextValue(dValue))
			{
				setNormalizedParameter(m_ppControlTable[i], dValue, true);
			}
		}
	}
}
```

Step 3: add the code to use the new object to the top of your processing frame

For processAudioFrame( )

```
bool __stdcall <yourplugin>::processAudioFrame(float* pInputBuffer,...
{
	// --- for VST3 plugins only
	doVSTSampleAccurateParamUpdates();

	--- rest of function as normal
```

For processVSTAudioBuffer( )

```
bool __stdcall <yourplugin>::processVSTAudioBuffer(float** inBuffer, ...
{
	// declarations and other stuff
	// …

	// - the loop
	unsigned int uSample = 0;
	while (--inFramesToProcess >= 0)
	{
		// --- fire midi events (AU, VST2, AAX buffer processing only;)
		if (m_pMidiEventList)
			m_pMidiEventList->fireMidiEvent(uSample++);
```

```
// --- sample accurate automation for VST3 only
doVSTSampleAccurateParamUpdates();

// --- smooth parameters (if enabled) DO NOT REMOVE
smoothParameterValues(); // done on a per-sample-interval basis

--- rest of function as normal
```

That's it - you'll now have sample accurate parameter support (automation in a VST3 host) as long as the host DAW supports this feature as well.