

DirectMusicTool Wizard: Overview

The DirectMusicTool Wizard simplifies the task of creating a custom tool that can be inserted in Microsoft® DirectMusic® Producer or in a DirectMusic application to intercept and process performance messages. It creates a ready-to-compile project containing C++ code that handles the basic COM component creation and DLL registration as well as the entry point for the DLL. In addition, the wizard generates the tool's implementation class with basic services already in place.

After the project is created, you must add code to do the actual processing of messages and, optionally, to expose parameters of the tool. For more information, see [Completing the Implementation](#).

Quick links: [Step 1](#), [Step 2](#), [Step 3](#), [Completing the Implementation](#).

DirectMusicTool Wizard: Step 1

In this step, you provide information about the tool and choose to build either an empty tool that you can customize later or a complete sample tool.

Enter the following information.

<u>Tool information</u>	<u>Description</u>
Tool Name	Name of the tool.
Class Name	Name of the implementation class for the tool. This name is derived from Tool Name and cannot be changed independently.
Interface Name	Name of the interface for the tool. This name is derived from Tool Name and cannot be changed independently.
DLL Name	Name of the DLL file for the tool. The .dll extension is added automatically.
Author	Your name.
Description	Short description of the tool.

Note The following symbols are not allowed in the tool name or DLL name:

` ~ ! @ # \$ % ^ & * () - + = { } [] | \ : ; ' " < > , . ? /

Click one of the following options.

<u>Tool option</u>	<u>Description</u>
Empty Tool	An empty tool framework. Choosing this option optionally leads to additional steps that help you define the functionality of the tool.
Echo Tool	Sample tool that adds an echo effect to DirectMusic sounds. When this option is selected, values in the Tool Information area are ignored, and no more steps are available in the wizard.

Quick links: [Overview](#), [Step 2](#), [Step 3](#), [Completing the Implementation](#).

DirectMusicTool Wizard: Step 2

In this step, you select the types of performance messages the tool receives. The message types correspond to members of the **DMUS_PMSGT_TYPES** enumeration in the DirectMusic application programming interface.

Select one or more of the following options.

<u>Message type</u>	<u>Description</u>
MIDI	Message containing data for a standard MIDI message, such as a control change or pitch bend.
Note	Message for a note. Includes duration, so MIDI note-on and note-off messages are combined in this type.
SysEx	Message containing data for a MIDI system-exclusive message.
Notification	Message for a notification.
Tempo	Message that controls the performance tempo.
Curve	Message for a controller curve.
TimeSig	Message that controls the time signature of the performance.
Patch	Message for a MIDI program change.
Transpose	Message for a transposition.
Channel_Priority	Message for a channel priority change.
Stop	Message that stops the performance at the specified time.
Dirty	Message sent to tools when a control segment starts or ends.
Wave	Message containing control information for playing a waveform.
Lyric	Message containing text.
ScriptLyric	Lyric message sent by a script.
User	Application-defined message.

Quick links: [Overview](#), [Step 1](#), [Step 3](#), [Completing the Implementation](#).

DirectMusicTool Wizard: Step 3

In this step, you specify when you want to receive messages, what to do with messages after processing them, and which extra interfaces to support in the tool.

PMsg Delivery

In this area, select one option to specify when performance messages are to be delivered. In most cases, you should select **DMUS_PMSGF_TOOL_IMMEDIATE**, which causes the DirectMusic performance to deliver messages to the tool as soon as they become available.

<u>Delivery type</u>	<u>Description</u>
DMUS_PMSGF_TOOL_IMMEDIATE	Delivers messages immediately.
DMUS_PMSGF_TOOL_QUEUE	Delivers messages just before the time at which they are supposed to play, taking latency into account.
DMUS_PMSGF_TOOL_ATTIME	Delivers messages at exactly the time at which they are to play.

PMsg Post-Process

In this area, select an **HRESULT** to return from the method that processes messages. This value specifies what to do with messages after they are processed by the tool.

<u>HRESULT</u>	<u>Description</u>
DMUS_S_REQUEUE	Puts the message back in the queue for delivery to other tools, including the output tool.
DMUS_S_FREE	Deletes the message so that it is not delivered to any more tools, including the output tool.
S_OK	Does not delete the message or put it back in the queue. You should select this option only if the tool has freed the message itself or if the tool needs to hold onto it.

Compatibility

In this area, select one option to specify which additional interfaces the tool will support for extra functionality.

<u>Option</u>	<u>Description</u>
DirectMusic	Supports only the tool's interface and IDirectMusicTool8 . The tool can be manually inserted in an application but cannot be used in an authored audiopath configuration.
DirectMusic + Real Time Parameter Control	Supports all the interfaces in the previous option plus the IMediaParams and IMediaParamInfo interfaces, which enable tool parameters to be controlled in real time (for example, by a parameter control track); and the IPersistStream interface, which enables the tool to be dragged from the DirectMusic Producer tool palette and embedded in an audiopath configuration.

DirectMusic + Authoring

Supports all the interfaces in the previous options plus the **ISpecifyPropertyPages** and **IPropertyPageImpl** interfaces, and create a basic property page resource to which you can add your own controls. The property page enables editing of the default tool parameters in an environment such as DirectMusic Producer.

The wizard implements the **RegisterTool** function to register the component under the key defined in Dmplugin.h as DMUS_REGSTR_PATH_TOOLS. This key makes the tool visible in DirectMusic Producer. However, the tool cannot actually be used in an audiopath configuration unless it supports at least **IPersistStream**.

For more information on the interfaces in the table, see [Completing the Implementation](#).

Quick links: [Overview](#), [Step 1](#), [Step 2](#), [Completing the Implementation](#).

Completing the Implementation

The wizard creates a partial implementation of the **IDirectMusicTool8** interface. You must add functionality to at least the **ProcessPMsg** method, which is where the work of the tool is done. In addition, you may want to expand or alter the implementation of other methods.

The following table describes the purpose of each method and the default implementation provided by the wizard. For more information, see **IDirectMusicTool8** in DirectX Help.

<u>Method</u>	<u>Description</u>	<u>Default implementation</u>
Init	Performs any needed initialization of the object. This method is called when the application adds the tool to a graph by calling IDirectMusicPerformance::InsertTool . It should always return S_OK.	Sets a global variable.
GetMsgDeliveryType	Specifies when the performance should deliver messages to the tool by calling its ProcessPMsg method.	Specifies the message delivery type selected in the PMsg Delivery area.
GetMediaTypes	Returns an array of message types that the tool processes. This method is called by the DirectMusic performance to determine which messages to pass to the ProcessPMsg method.	Returns the message types selected in the wizard.
GetMediaTypeArraySize	Specifies the number of message types that the tool processes.	Specifies the number of message types selected in the wizard.
ProcessPMsg	Processes each message. This method is called by the performance each time a message that matches the requested types is available.	Stamps the message for the next tool and returns the success code you set for the PMsg Post-Process option. You must add code to this method to process messages, unless the tool is simply discarding them.
Flush	Specifies the behavior of the tool when it receives a message sent as a result of an invalidation. This can happen, for example, when a note or curve is in progress and the segment stops unexpectedly.	Requeues the message. Most tools don't need to do anything else.

In addition, the wizard optionally provides implementations of the following interfaces:

IMediaParamInfo and **IMediaParams**

These interfaces expose parameters supported by the tool so that they can be manipulated in real time. For example, these interfaces make it possible to control the tool by means of a curve in the parameter control track of a segment.

The interfaces are fully implemented by the wizard but rely on your declaration of the **g_params** array in the class header file for your tool. In this array, you can specify minimum and maximum values and other properties for each parameter.

IPersistStream

This interface enables DirectMusic Producer to save and load your tool as part of an audiopath configuration and enables the DirectMusic loader to set up the tool when the audiopath is created for playback. You are responsible for streaming a RIFF chunk of the file that contains custom properties for the tool. You save and load this chunk by implementing the **Write** and **Read** methods of the **IStream** interface that is passed to your implementations of **IPersistStream::Save** and **IPersistStream::Load**.

Even if your tool doesn't have any properties that need to be persisted, **IPersistStream::Save** must be implemented at least to the extent of saving a chunk identifier and a data size. The data size can be zero. If you do not do this, DirectMusic Producer will not allow the user to drag your tool into the Toolgraph Designer window.

The wizard provides a default implementation of **IPersistStream** that relies on your implementation of **GetAllParameters**, **SetAllParameters**, and **SetParamInternal**. You must add code to these methods to get and set global variables for all persistent parameters of your tool.

ISpecifyPropertyPages

This interface exposes a property page for the tool, so that its parameters can be edited within DirectMusic Producer or another application. A sample property page with a single control is created by the wizard, and the **ISpecifyPropertyPages** interface is fully implemented to expose that page.

IPropertyPageImpl

This interface enables the tool to set and retrieve the values of controls on the property page. You must add your own implementation to the **OnInitDialog** and **Apply** methods.

Quick links: [Overview](#), [Step 1](#), [Step 2](#), [Step 3](#).

