

# MORAE

## Recorder COM Server Guide

Version 1.2

TechSmith Corporation



### **Morae Documentation Copyright**

TechSmith Corporation provides this manual “as is”, makes no representations or warranties with respect to its contents or use, and specifically disclaims any expressed or implied warranties of merchantability or fitness for any particular purpose.

TechSmith Corporation reserves the right to make changes to the content of this manual, at any time, without obligation to notify any person or entity of such changes.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser’s personal use, with out the express permission of TechSmith Corporation.

Copyright © 2005 TechSmith Corporation

All rights reserved. Printed in the United States of America.

### **Morae Software License Agreement**

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

### **Trademarks**

TechSmith is a registered trademark, and Morae is a trademark of TechSmith Corporation. All other registered trademarks and trademarks are the property of their respective companies. All rights reserved.

# Contents

<b>Morae Recorder COM Server Guide</b>	<b>2</b>
<b>Getting Started</b> .....	<b>4</b>
Basic Procedure .....	5
<b>Programming Reference</b> .....	<b>6</b>
Interfaces: IMoraeRecordControl .....	6
Interfaces: ICameraOptions .....	7
Interfaces: IAudioOptions .....	7
Interfaces: IKeystrokeOptions .....	7
Interfaces: ITextOptions .....	8
Interfaces: IMouseOptions .....	8
Interfaces: _IMoraeRecordControlEvents .....	8
<b>Known Issues</b> .....	<b>10</b>
<b>Appendix A: Language-Specific Examples</b> .....	<b>11</b>
C++/ATL .....	11
C# .....	13
Visual Basic 6 (VB) .....	14
Visual Basic .NET (VB.NET) .....	15
Visual Basic Script (VBScript) .....	16
<b>Appendix B: Data Type Information</b> .....	<b>17</b>
<b>Appendix C: Morae Enumerations</b> .....	<b>18</b>
<b>Appendix D: Morae Recorder Technical Support</b> .....	<b>21</b>

# Morae Recorder COM Server Guide

## Introduction

*Morae Recorder* supports an out-of-process COM server, which gives a programmer access to many of its powerful recording features. This functionality can be accessed through any programming environment that supports COM, including Visual Basic, Visual C/C++, Visual Studio.NET, Delphi, and C++ Builder.

## Downloading the COM Server Sample Applications

There are two places you can download the *Recorder* COM server sample applications (a .zip file):

1. On the *Morae* COM server page at the TechSmith Web site:  
<http://www.techsmith.com/redirect.asp?product=morae&target=moraeecom>
2. In the *Recorder COM Server* folder on the *Morae Recorder* installation CD.

## Installation

*Morae Recorder*'s setup program will automatically register the COM server with the system. However, if you need to manually register the COM server use a command prompt and type the following from the folder where *Morae Recorder* is installed:

```
MoraeRecorder.exe /register
```

To manually unregister the COM server, type the following:

```
MoraeRecorder.exe /unregister
```

When *Morae Recorder* is uninstalled, the *Recorder* COM server is automatically unregistered.

## System Requirements

	Required/Minimum	Recommended
<b>Operating System</b>	Microsoft Windows 2000, XP or later version of Windows	
<b>DirectX</b>	Version 8.1 or later	Version 9.0 or later
<b>Processor</b>	1.5 GHz or higher	1.7 GHz Pentium M or 2.8 GHz Pentium 4 processor or higher
<b>RAM</b>	512 MB RAM or more	1 GB RAM or more
<b>Video card</b> <b>Note:</b> Shared video memory graphics cards are not supported.	64 MB dedicated memory	128 MB dedicated video card
<b>Sound card, microphone, speakers</b>	Windows-compatible	
<b>Hard drive space for installation</b>	20 MB of hard drive space for program installation	
<b>Disk space for recording</b>	Approximately 20–40 MB disk space per minute of recording	
<b>Network connection</b>	10 Mbps	100 Mbps

---

# Getting Started

## Before you Begin

The *Morae Recorder* COM server should be accessible from any language that supports COM. Only basic data types are used. For more information, see *Appendix B: Data Type Information* on page 17.

---

**Note:** The COM server is not an automation server. No changes made through the COM server will be saved in the *Morae Recorder* application or the registry.

---

When *Recorder's* main UI is running, no COM client can create an interface to it. Conversely, when a COM client is connected to *Recorder*, the main UI cannot be run.

All hardware settings must be set via *Recorder's* main UI before creating an instance of the COM interface. The settings can be accessed by choosing **Record > Settings**. The hardware settings are saved when *Recorder* closes. Note that settings saved in configuration files are not used when *Recorder* is in COM mode.

## How It Works

All of the interfaces, properties and methods described in this guide are accessible from any COM-enabled language.

- **Interfaces**  
These define the properties, methods, events, and sub-interfaces for a given COM object.
- **Properties**  
Values stored in an interface that may be set or retrieved. Some properties are read only, and all sub-interfaces are read only.
- **Methods**  
Functions exposed by interfaces that interact with the COM server.
- **Events**  
Events may be sent from the COM server to the client program. The client program must "subscribe" to the event interface to receive the events. This is a language-dependent option, and not all programming languages can receive events. For more information, see *Interfaces: \_IMoraeRecordControlEvents* on page 8.
- **Enumerations**  
Enumerations are numeric values that have been assigned a type of variable. For more information, see *Appendix C: Morae Enumerations* on page 18.

In order to be able to access the COM object's functionality, you must first create an instance of the object in your client program. See *Appendix A: Language-Specific Examples* on page 11 for examples using various programming languages.

- **C++ MFC and ATL Users**  
The easiest way to make use of the COM server is using the Microsoft Visual C++ *import* statement on the **recorder.tlb** file present in the *Morae Recorder* install directory. This creates smart-pointer wrapper classes for the COM server.
- **Visual Basic 6 Users**  
A reference to the **Morae Recorder 1.0 Type Library** should be added in the *References* dialog accessed from the **Project > References** menu item.
- **Visual Basic .NET and C# Users**  
A reference to the **Morae Recorder 1.0 Type Library** should be added in the *Add Reference* dialog accessed from **Project > Add Reference** menu item. Note that the entry is on the COM property page.

### **Basic Procedure**

The following outlines the general steps taken to get an instance of the COM interface, start a recording, insert markers into a recording, and stop a recording:

1. Create an instance of the interface to the recording control object.
2. Wait for the RecorderArmed event before doing anything else.
3. Set the RecordingFilePath property on the recording control object.
4. Call the StartRecording() method to start a recording.
5. Wait for the StartedRecording event before doing anything else.
6. Optionally insert markers by calling the InsertMarker() method. Note that the InsertMarker() call is synchronous.
7. Call the StopRecording() method to shut down the recording.
8. Wait for the StoppedRecording event.
9. Optionally process the OutputProgress event(s).
10. Wait for the OutputWritten event.
11. Wait for the RecorderArmed event.
12. Go to Step 3 to start another recording or release the COM interface to clean up.

## Programming Reference

### Interfaces: *IMoraeRecordControl*

The *IMoraeRecordControl* interface is used to control all aspects of a *Morae* recording session. All properties must be set prior to running a recording. If an error is generated, a “\_com\_error” exception will be thrown. The *IMoraeRecordControl* interface contains several sub-interfaces that expose options for the various recording streams (camera, text, mouse, etc).

### Methods

Method	Arguments	Return
<b>StartCapture</b>	None	HRESULT
<b>Stop Capture</b>	None	HRESULT
<b>InsertMarker</b>  <b>Note:</b> Markers inserted via the COM interface will show up in Manager with a username of “COM”.	BSTR bstrName: The name of the marker	
	long lType: The type of marker (0 is “generic/none”, 1 is ‘A’, 2 is ‘B’, 3 is ‘C’, etc)	
	BSTR bstrAnnotation: Annotation associated with the marker	

### Properties

Property	Description
<b>RecordingFilePath</b>	The full path for storage of the recording file. This must be filled out prior to starting a recording, or an error will occur.
<b>RemoteViewersAllowed</b>	Allows or prevents <i>Remote Viewers</i> from connecting to <i>Recorder</i> during a recording. Set to zero (0) to not allow any <i>Remote Viewers</i> to connect. Set to one (1) to allow <i>Remote Viewers</i> to connect.

### Read-Only Properties

Property	Description
<b>IsRecording:</b> (VARIANT_BOOL)	This is set to false if a recording is not currently in progress. It is true if a recording is in progress.
<b>LastMoraeError:</b> (moraeError)	The most recent error encountered by the COM object. See <i>Appendix C: Morae Enumerations</i> on page 18 for possible values. You should check this for more error information if an error occurred during a COM call.



## Sub-Interfaces

These interfaces can be accessed from the IMoraeRecordControl interface. They are read-only, but their properties can be modified.

- CameraOptions (ICameraOptions)
- AudioOptions (IAudioOptions)
- KeystrokeOptions (IKeystrokeOptions)
- TextOptions (ITextOptions)
- MouseOptions (IMouseOptions)

### Interfaces: ICameraOptions

The ICameraOptions exposes options available to control the behavior of the camera stream in recordings.

Property	Description
Enable: (VARIANT_BOOL)	Set this to true to enable the camera stream during a recording. Set this to false to disable the camera stream.  Note that this cannot be changed while a recording is in progress.

### Interfaces: IAudioOptions

The IAudioOptions exposes options available to control the behavior of the audio stream in recordings.

#### Properties

Property	Description
Enable: (VARIANT_BOOL)	Set this to true to enable the audio stream during a recording. Set this to false to disable the audio stream.  Note that this cannot be changed while a recording is in progress.

### Interfaces: IKeystrokeOptions

The IKeystrokeOptions exposes options available to control the behavior of the keystroke stream in recordings.

#### Properties

Property	Description
Enable: (VARIANT_BOOL)	Set this to true to enable the keystroke stream during a recording. Set this to false to disable the keystroke stream.  Note that this cannot be changed while a recording is in progress.

**Interfaces: *ITextOptions***

The ITextOptions exposes options available to control the behavior of the text stream in recordings.

**Properties**

Property	Description
Enable: (VARIANT_BOOL)	Set this to true to enable the text stream during a recording. Set this to false to disable the text stream.  Note that this cannot be changed while a recording is in progress.

**Interfaces: *IMouseOptions***

The IMouseOptions exposes options available to control the behavior of the mouse stream in recordings.

**Properties**

Property	Description
Enable: (VARIANT_BOOL)	Set this to true to enable the mouse stream during a recording. Set this to false to disable the mouse stream.  Note that this cannot be changed while a recording is in progress.

**Interfaces: *\_IMoraeRecordControlEvents***

These events signal the completion of asynchronous operations involving recordings.

Event	Action	Parameters	DisplD
RecorderArmed	Fired when the COM object is fully initialized and it is ready to start a recording. This is also fired after <i>Recorder</i> is ready to start another recording after completing a previous recording.	None	1
StartedRecording	Fired when a recording has been started and is ready to receive markers or be shut down.	None	2
StoppedRecording	Fired when a recording has been stopped and is no longer capturing video, audio, or events.	None	3
OutputWritten	Fired when the .rdg file has been completely written to the storage medium.	None	4

Event	Action	Parameters	DisplID
OutputProgress	Fired when a percentage of the .rdg file is written to the storage medium.	<b>moraeOutputProgressState</b> <b>state:</b> State of the RDG writing process. See <i>Appendix C: Morae Enumerations</i> on page 18.  <b>long lPercent:</b> Percent of the writing or verifying process that is complete.	5
RecordingError	Fired when an error is encountered by the recording engine during a recording.	<b>moraeError error:</b> Code of the error encountered. See <i>Appendix C: Morae Enumerations</i> on page 18.	6

---

## Known Issues

- *Recorder* was not designed to handle multiple COM clients. Attaching multiple COM clients to it at once could result in unpredictable behavior.
- Attempting to release the COM object after calling `StopRecording()`, but before receiving the `RecorderArmed` event can cause *Recorder* to crash.
- Calling `StopRecording()` and then calling it again before receiving the `StoppedRecording` event will cause the RDG file to not be written out correctly.
- Manager can only handle a certain number of markers before it gets bogged down processing them when opening a recording. The number will vary based on the processor and RAM available.

---

## Appendix A: Language-Specific Examples

### C++/ATL

#### Import the table implementation file (recorder.tlb)

```
#import "Path\To\recorder.tlb" rename_namespace("Recorder")
```

#### Declare an object

```
Recorder::IMoraeRecordControlPtr pRecControl;
```

#### Create the object

```
pRecControl.CreateInstance( __uuidof( Recorder::MoraeRecordControl ) );
```

#### Handle an event

To handle events takes a few steps. Capturing error events is illustrated here:

1. At the top of your class declaration file create an external reference to an `_ATL_FUNC_INFO` object:

```
extern _ATL_FUNC_INFO RecorderErrorInfo;
```

2. Next have your class inherit from the `IDispatchSimpleImpl` templated class:

```
class CMyClass : IDispatchSimpleImpl<1, CMyClass,  
&__uuidof(Recorder::_IMoraeRecordControlEvents)>
```

3. Next create a typedef inside your class declaration to talk about the events more easily:

```
typedef IDispatchSimpleImpl<1, CMyClass,  
&__uuidof(Recorder::_IMoraeRecordControlEvents)> RecorderErrorEvents;
```

4. Now create a sink map inside your class declaration to setup the message handling:

```
BEGIN_SINK_MAP(CMyClass)  
SINK_ENTRY_INFO(1, __uuidof(Recorder::_IMoraeRecordControlEvents), 0x01,  
OnRecorderError, &RecorderErrorInfo )  
END_SINK_MAP()
```

5. Declare the message handler function inside your class declaration:

```
void __stdcall OnRecorderError( Recorder::moraeError nErrorCode );
```

6. Now in your class implementation file, define the RecorderErrorInfo \_ATL\_FUNC\_INFO object:

```
_ATL_FUNC_INFO RecorderErrorInfo = { CC_STDCALL, VT_EMPTY, 1, { VT_I4 } };
```

7. Before you start getting events, you must advise the object that you are listening for events somewhere in the implementation file:

```
RecorderErrorEvents::DispEventAdvise( pRecControl );
```

8. Lastly, implement the handler function:

```
void _stdcall CMyClass::OnRecorderError( Recorder::moraeError nErrorCode )  
{  
    /* implementation */  
}
```

## C#

### Declare an object

```
private MoraeRecorderLib.MoraeRecordControl RecControl;
```

### Create the object

```
RecControl = new MoraeRecorderLib.MoraeRecordControlClass();
```

### Handle callbacks

To handle callbacks, create a function that handles the callback and attach it to the object using the correct delegate.

```
RecControl.RecordingError += new  
MoraeRecorderLib.IMoraeRecordControlEvents_RecordingErrorHandler(  
this.OnRecordingError );
```

## Visual Basic 6 (VB)

### Declare an object able to receive events

```
Dim WithEvents RecControl As MoraeRecorderLib.MoraeRecordControl
```

### Create the object

```
Set RecControl = CreateObject("MoraeRecorder.MoraeRecordControl")
```

### Declare an event handler

```
`This function handles the OnRecordingError event fired by the  
`MoraeRecordControl interface  
`Event handling is done simply by putting the event after an _ after the object  
`that fires the event. That is, Object_Event  
Private Sub RecControl_OnRecordingError(ByVal error As  
MoraeRecorderLib.moraeError)  
...  
...  
End Sub
```



## Visual Basic .NET (VB.NET)

### Declare an object able to receive events

```
Public WithEvents RecControl As MoraeRecorderLib.MoraeRecordControl
```

### Create the object

```
RecControl = New MoraeRecorderLib.MoraeRecordControlClass()
```

### Declare an event handler

```
`This function handles the OnRecordingError event from the MoraeRecordControl  
`object  
Private Sub RecordingError(ByVal error As MoraeRecorderLib.moraeError) Handles  
RecControl.OnRecordingError  
...  
...  
End Sub
```

## Visual Basic Script (VBScript)

Objects are created using the CreateObject() function:

```
'Create and record control object  
set RecControl = CreateObject("MoraeRecorderLib.MoraeRecordControl")
```

If the script is allowed to exit before the recording has completed, the capture object will go out of scope and will exit. To keep the script running while the capture finishes, a sleeping loop can be used.

```
Do Until RecControl.IsRecording  
    WScript.Sleep 10  
Loop
```

## Appendix B: Data Type Information

Only simple data types are used in the *Morae Recorder* COM server. This allows it to be used from many different programming languages. This section provides more specifics on each of the basic data types.

### Boolean

Note that many languages have defines for true and false values, these may or may not be translated to the correct `VARIANT_BOOL` values. For example C++/ATL programs should use the **VARIANT\_TRUE** and **VARIANT\_FALSE**, whereas VB, VB.NET, VBScript and C# correctly translate the Default: **true** and **false** values.

### Long

32-Bit signed integer

### String

Variable length string

C++/ATL – **BSTR** or **CComBSTR**

VB, VB.NET, VBScript, and C# – **string**

### Interface

All interfaces are derived from `IDispatch` to allow for use in scripting languages such as VBScript.

## Appendix C: Morae Enumerations

### **moraeOutputProgressState**

The following table lists all the possible values of the moraeOutputProgressState enumeration used in the OutputProgress event.

Value	Description
mopsWrite	The RDG file is currently being written to the storage medium.
mopsVerify	The RDG file contents are being verified.

### **moraeError**

The following table lists all the possible error codes that LastMoraeError could be set to by a COM method.

Error Code	Description
merrNone	No error has occurred.
merrInvalidTempFolder	The temporary recording storage folder is invalid.
merrOutputFileExists	The output RDG file already exists.
merrTempAccessDenied	<i>Recorder</i> cannot write to the temporary storage folder.
merrMemError	No more memory could be allocated.
merrEmptyTempFolder	No temporary storage folder has been specified.
merrEmptyOutputFilename	No output RDG filename has been specified.
merrUnkownError	An unknown error has occurred.
merrInvalidFilename	The filename given was invalid.
merrInvalidStartTime	The automatic start time is invalid.
merrCreateScreenCapture	Screen recorder could not be initialized.
merrSetScreenInput	Failed to set the screen recorder input.
merrSetScreenOutput	Failed to set the screen recorder output.
merrSetScreenOutFile	Failed to set the screen recorder output file.
merrSetScreenSaveDlg	Failed to set the screen recorder save dialog.
merrSetScreenTempDir	Failed to set the screen recorder temporary storage directory.
merrSetAudioDevice	Failed to initialize the audio input device.
merrSetAudioFormat	Failed to set the audio stream format.
merrSetAviInterleave	Failed to set the AVI interleave property.

Error Code	Description
merrSetFrameCallback	Failed to set the frame capture callback.
merrSetSdkStateCallback	Failed to set the SDK state callback.
merrSetScreenFrameRate	Failed to set the screen AVI frame rate.
merrSetScreenKeyframe	Failed to set the screen AVI keyframe.
merrSetScreenDataRate	Failed to set the screen AVI data rate.
merrSetAviQuality	Failed to set the AVI quality property.
merrSetScreenVideoCodec	Failed to set the screen video codec.
merrSetScreenVideoCodecState	Failed to set the screen video codec state.
merrSetScreenHideCapRect	Failed to hide the screen video capture rectangle.
merrNoScreenConfigInfo	There was no screen configuration information.
merrSetAudioFileOption	Failed to set audio file options.
merrStartScreenCapture	The screen capture failed to start.
merrStopTimerSetup	An error occurred while setting up the stop timer.
merrCameraStart	Failed to start the camera capture.
merrStopTimeInvalid	The automatic stop time is invalid.
merrSetScreenStartPaused	Failed to set the screen recorder to pause when started.
merrScreenResume	Failed to resume the screen recorder.
merrEmptyOutputFolder	No output folder for the RDG has been specified.
merrOutputAccessDenied	<i>Recorder</i> does not have permission to write to the output folder.
merrInvalidOutputFolder	The folder to put the output RDG file in is invalid.
merrInvalidOutputFileName	The name of the output RDG file is invalid.
merrCameraInitialization	The camera could not be initialized.
merrEmptyTempFileName	The temporary filename was empty.
merrTempFileExists	The temporary file already exists.
merrStartTimerSetup	An error occurred while setting up the stop timer.
merrSdkAllocation	Memory allocation in the SDK failed.
merrSdkRendering	An error occurred while the SDK was trying to render something.
merrSdkFile	A file error occurred while manipulating a file in the SDK.
merrSdkAudioDevice	The SDK could not initialize the audio device.
merrSdkAudioCodec	The SDK could not set the audio codec.
merrSdkVideoCodec	The SDK could not set the video codec.

Error Code	Description
merrSetSdkErrorCallback	Failed to set the error callback for the SDK.
merrPrestartTimerSetup	Failed to set the prestart timer.
merrReleaseCaptureHooks	Failed to release capture hooks.
merrSetCaptureHooks	Failed to set capture hooks.
merrCreateFrameThread	Failed to create the frame thread.
merrCantFindScreenVideoFile	The screen recorder failed to create the screen video file.
merrSetCursor	Failed to set the cursor.
merrSetLayeredWndCapture	Failed to initialize layered window capture.
merrMainWndDoesNotExist	<i>Recorder</i> 's main window does not exist.
merrRecordingInProgress	The operation could not be completed because a recording is in progress.
merrNoRecordingInProgress	The operation could not be completed because no recording is in progress.
merrRecordingStartFatalFailure	<i>Recorder</i> encountered a fatal error while attempting to start the recording.
merrRecordingStopFatalFailure	<i>Recorder</i> encountered a fatal error while attempting to stop the recording.
merrInsertMarkerFatalFailure	<i>Recorder</i> encountered a fatal error while attempting to insert a marker.
merrInterfacesNotInitialized	Internal interfaces that the COM object uses were not initialized correctly.

## Appendix D: Morae Recorder Technical Support

### Contacting TechSmith Technical Support

If at any time you experience problems with *Morae Recorder*, we encourage you to contact TechSmith Technical Support at:

<http://support.techsmith.com>

The *Morae* Essential Plan provides full maintenance, major upgrades and priority support services. For additional information, contact your *Morae* representative at 888-750-0685, or visit [www.morae.com](http://www.morae.com).

### Mailing Address

TechSmith Corporation  
2405 Woodlake Dr.  
Okemos, MI 48864 USA

### Morae Software License Agreement

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

### Trademarks

TechSmith is a registered trademark, and *Morae* is a trademark of TechSmith Corporation. All other registered trademarks and trademarks are the property of their respective companies. All rights reserved.