



IEventCapture plug-in interface for Morae

Purpose of interface:

This interface will be called by Morae Recorder to enable the capture of additional event data during a Morae recording session. (This interface does not provide a mechanism to read and display event data in Manager. That capability is enabled through the IMoraeEventStream, IMoraeEventSearch, and IMoraeVideoDraw interfaces.) An object that implements this interface will be expected to create event file(s) in a given directory. It will be allowed to display optional setup and calibration dialogs to allow the user to configure options and perform calibration if necessary.

This interface derives from IUnknown.

Note: Methods that return BSTRs that will be displayed to the user all use a locale ID to indicate the language. At this time (Morae release 3.2.0), Morae is only offered in an English version.

Note: To be loaded by Recorder and used for event capture, the plug-in must either:

1. Register its CLSID with the operating system and Register itself as implementing the CATID_EventCaptureSource category:
A5DC9083-35B9-411F-8E5D-54F731296D2C
2. Register its CLSID with the operating system and use the RecorderCaptureHelper to automatically register its category implementation. The CLSID of RecorderCaptureHelper is CLSID_PluginManager:
1F3E87EE-B5EC-46B3-8A6E-F08A2F 5F27AE

IEventCapture specifications:

Interface ID:

IID_IEventCapture = 78F72AF2-41A0-47F3-B753-6139A9849708

Enumerations

1. SetupStatus
GetSetupStatus should output one of these values.



LPSAFEARRAY pArParam) [in]

Recorder will use this method to apply setup parameters that it had previously stored from the GetCaptureParameters method call. If this is not applicable, the plug-in should return S_OK and ignore the pArParam argument. The caller will de-allocate this array after this method returns.

10. HRESULT SetOutputDirectory (

BSTR strDir) [in]

This call will be made before ControlCapture is called to tell the plug-in to start capturing events. The plug-in should create any files that it needs to write in this directory, making sure that it is not overwriting any existing files.

11. HRESULT GetOutputFilepaths (

LPSAFEARRAY* parNames) [out]

This call will be made after ControlCapture is called with an argument of "Capture_Stop" to tell the plug-in to stop capturing events. The plug-in should create a safearray of BSTR values, each of which supplies the full path for a file that the plug-in has created and filled with captured event data. The caller is responsible for de-allocating the array.

12. HRESULT ShowSetupDialog(

VARIANT varHwndParent, [in]
long nLocaleID) [in]

This call asks the plug-in to display a dialog to allow the user to configure capture options. The varHwndParent argument provides the handle for the parent window for the dialog. The plug-in should be able to handle a value that is either signed or unsigned, 32-bit or 64-bit. If the user cancels the dialog, this method should return S_FALSE. A return value of S_OK indicates that the user wishes to save his new settings.

13. HRESULT ShowCalibrationDialog(

VARIANT varHwndParent, [in]
long nLocaleID) [in]

This call asks the plug-in to display a dialog to allow the user to perform a calibration. The varHwndParent argument provides the handle for the parent window for the dialog. The plug-in should be able to handle a value that is either signed or unsigned, 32-bit or 64-bit. If the user cancels the dialog, this method should return S_FALSE. A return value of S_OK indicates that the user wishes to save his new settings. The plug-in is responsible for saving and applying all calibration settings.

14. HRESULT SetCaptureRect(

int nTop, [in]
int nBottom, [in]
int nLeft, [in]
int nRight) [in]



When the computer screen is serving as the primary video source for a recording, this provides the screen coordinates of the screen region that is being recorded. If the screen is not the primary video source, this method will be called with all of the parameters set to 0. The plug-in can (optionally) use this information to restrict the events that it captures and records.

15. HRESULT ControlCapture(

long nCmd, [in]
unsigned long nTickCount) [in]

This method is called to indicate when a recording starts and stops. The nCmd input will be set to one of the CaptureControl enumeration values. The nTickCount parameter will be set to the tick count (acquired via the Win32 API call GetTickCount()) of the computer when the recording was started, stopped, paused, or resumed. If a series of plug-ins are called, all will get the same nTickCount value for that call. When a recording is started, SetOutputDirectory and SetCaptureRect will be called before ControlCapture is called with nCmd set to Capture_Start. When a recording is stopped, ControlCapture will be called with nCmd set to Capture_Stop. Then GetOutputFilepaths will be called. Then ControlCapture will be called with nCmd set to Capture_Idle.

16. HRESULT GetEventReaderInfo (

GUID* puuID, [out]
unsigned long* pnMajorVersion, [out]
unsigned long* pnMinorVersion, [out]
BSTR* pstrHandlerName, [out]
BSTR* pstrURL, [out]
long nLocaleID) [in]

This method is called to get information on the event handler plug-in object that will be used by Manager to read the event file(s) created by the IEventCapture plug-in. *puuID should be set to the CLSID of the component that will read the file(s). The other information will be displayed to the user in an error message if an error occurs when Manager attempts to read the data file. *pstrURL should be set to the URL of a website that the user could go to for more information or to check for an upgrade.