



IMoraeEventStream interface

Purpose of interface:

This interface will be called by Morae Manager to read a data file that was created by an IEventCapture plug-in in Morae Recorder. The base functionality described in this interface simply reads the file(s) and allows creation of an enumerator that permits a caller to step through all of the captured events. Manager does not directly expose these events anywhere in the UI. To make these events accessible to the user, the object that implements IMoraeEventStream should also implement IMoraeEventSearch and / or IMoraeVideoDraw. IMoraeEventSearch allows the user to search for events and display them in the search results table. IMoraeVideoDraw allow the user to display events on the video as it plays.

Objects that implement this interface need to have the CLSID registered with the system. The IMoraeEventStream interface is not associated with any implemented category in the registry. Note that the CLSID under which an object is registered should be the same as the GUID supplied by the event source in the IEventCapture::GetEventReaderInfo call.

This interface derives from IUnknown.

Interface ID:

IID_IMoraeEventStream = C8CC8AD7-F920-4167-A397-8FAC322D619D

Methods:

1. HRESULT EnumEvents(
IEnumMoraeEvent** ppIEnum) [out]

The plug-in should provide an enumerator that allows the caller to step through all of the captured events.

2. HRESULT GetEventCount(
ULONG* pCount) [out]

The plug-in should output the number of events that the enumerator acquired in EnumEvents contains.

3. HRESULT GetEventStreamID(
IID* pID) [out]



This IID identifies the type of event in the stream. It does not identify the object that is implementing IMoraeEventStream.

4. HRESULT SetRecording(

IMoraeRecording* pRecording) [in]

This provides the plug-in instance with an interface that can be used to get information about the recording in which it found. Note that the IMoraeRecording interface will not change during the object's lifetime. If the plug-in stores this pointer for later use, it should call AddRef on it to ensure that it remains valid. In that case, the plug-in must call Release on this pointer when it is no longer using it.

5. HRESULT SetProject(

IMoraeProject* pProject) [in]

This provides the plug-in object with an interface to the project in which its recording is found. (Note that the IMoraeProject interface relates to a Study rather than a project in the Manager UI.) This method may be called multiple times if the user drags a recording from one study to another in the Manager project. In that case, only the most current IMoraeProject interface pointer should be considered valid. If the plug-in stores this pointer for later use, it should call AddRef on it to ensure that it remains valid. In that case, the plug-in must call Release on this pointer when it is no longer using it.

6. HRESULT ReadEventFile(

BSTR bstrFile) [in]

The bstrFile will contain the full path to the event file that the plug-in should read. This may be called multiple times if the event capture object generated multiple files.

7. HRESULT GetStreamName(

LCID localeID, [in]
BSTR* pbstrName) [out]

This asks the plug-in for a name that could be displayed to the user

8. HRESULT SetTempDirectory(

BSTR bstrDir) [in]

This call directs the plug-in to a directory that it can use to store temporary data. The plug-in should clean up any files that it writes to this directory.

9. HRESULT SetStartTime(



Morae[®]
TechSmith

VARIANT varTickCount, [in]
VARIANT varSysTime) [in]

This call is used to inform the plug-in of the start time for the recording in two different formats. It is intended to allow the plug-in to be able to offset any timestamps that were used internally when the data was captured to calculate the time into the recording at which each event occurred. The varTickCount gives the system tick count at which the recording started. The plug-in should accept types of VT_I4, VT_U4, VT_I8, or VT_U8. The varSysTime is a VT_DATE type, giving the system time at the start.