



## Morae Manager native event interfaces

### Purpose of interfaces:

These interfaces provide access to the native RRT event streams that are normally captured in a Morae recording session. They can be used by plug-ins that implement `IMoraeEventSearch` or `IMoraeVideoDraw`. This allows development of plug-ins that use the native Morae RRT data to provide enhanced search or draw functionality to Morae Manager.

To access and use these events, the plug-in is provided with an `IMoraeRecording` interface. A call to `IMoraeRecording::GetDataStream` can be used to get an `IMoraeEventStream` interface for the data stream of interest. (The IID values that apply to each stream are given below.)

`IMoraeEventStream::EnumEvents` can be called to get an enumerator for all of the events captured as part of the stream. The plug-in can call `QueryInterface` with the appropriate interface ID (given below) on each `IMoraeEvent` interface retrieved from the enumerator to get access to the full details captured with the event.

All of these interfaces derive from `IMoraeEvent`.

### `IMoraeEventMouse` specifications:

#### Description:

This stream captures mouse clicks and movement.

#### Data stream ID:

`IID_MOUSEEVENTTYPE = 87DFC9CA-BFBA-47a0-8A0F-60ED99813CA3`

#### Interface ID:

`IID_IMoraeEventMouse = 55BB8B15-E5E5-4778-90D0-C75A9861F29D`

#### Methods

1. `HRESULT FindString (`

<code>BSTR str,</code>	<code>[in]</code>
<code>int* pResult,</code>	<code>[out]</code>
<code>int nCriteria )</code>	<code>[in]</code>

This call searches for a string within the event. `str` is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. `nCriteria` is a bitwise set of flags that determine how the search is made:

- a. `TXT_MATCH_CASE_SENSITIVE (0x00000001)` should be set to look for a case-sensitive match.



- b. TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
  - c. TXT\_MATCH\_OWN\_WND (0x00000004) should be set to search for the string within the window text of the window that received the mouse click.
  - d. TXT\_MATCH\_PARENT\_WND (0x00000008) should be set to search for the string within the window text of the parent window of the window that received the mouse click.
- \*pResult will be set to a bitwise combination of the search results:
- a. MFS\_WND\_TEXT (0x00000001) indicates that the text was found in the window text of the window that received the mouse click.
  - b. MFS\_PARENT (0x00000002) indicates that the text was found in the window text of the parent to the window that received the event.

2. HRESULT GetLocation (

int\* pX, [out]  
int\* pY ) [out]

This call gets the location (in pixels) at which the event occurred, relative to the upper left corner of the screen recording area. Note that X increases going to the right, Y increases going down.

3. HRESULT GetButtonFlags (

int\* pFlags ) [out]

This call retrieves the type of mouse event that was recorded. \*pFlags will be set to a bitwise combination of (All values except MCT\_NC are mutually exclusive.):

- a. MCT\_NC (0x01000000) indicates that this event occurred in the non-client area of the window. It can be combined with any of the other flags listed below.
- b. MCT\_LDOWN (0x00000001) indicates a left-button down event.
- c. MCT\_RDOWN (0x00000002) indicates a right-button down event.
- d. MCT\_MDOWN (0x00000004) indicates a middle-button down event.
- e. MCT\_XDOWN (0x00000008) indicates a button down event for either X button (in recordings made prior to Morae version 2.0).
- f. MCT\_X1DOWN (0x00010000) indicates an X1-button down event.
- g. MCT\_X2DOWN (0x00020000) indicates an X2-button down event.
- h. MCT\_LUP (0x00000010) indicates a left-button up event.
- i. MCT\_RUP (0x00000020) indicates a right-button up event.
- j. MCT\_MUP (0x00000040) indicates a middle-button up event.
- k. MCT\_XUP (0x00000080) indicates a button-up event for either X button (in recordings made prior to Morae version 2.0).
- l. MCT\_X1UP (0x00040000) indicates an X1-button up event.
- m. MCT\_X2UP (0x00080000) indicates an X2-button up event.
- n. MCT\_LDBCLCK (0x00000100) indicates a left-button double-click event.
- o. MCT\_RDBCLCK (0x00000200) indicates a right-button double-click event.



- p. MCT\_MDBCLCK (0x00000400) indicates a middle-button double-click event.
- q. MCT\_XDBCLCK (0x00000800) indicates a double-click event for either X button (in recordings made prior to Morae version 2.0).
- r. MCT\_X1DBCLCK (0x00100000) indicates an X1-button double-click event.
- s. MCT\_X2DBCLCK (0x00200000) indicates an X2-button double-click event.
- t. MCT\_WHEEL\_PLUS (0x00001000) indicates a positive mouse wheel click event.
- u. MCT\_WHEEL\_MINUS (0x00400000) indicates a negative mouse wheel click event.
- v. MCT\_MOVE (0x00002000) indicates a mouse move event.

4. HRESULT GetModKeyFlags (

int\* pFlags ) [out]

This call retrieves which, if any, modifier keys were depressed when the mouse event was recorded. \*pFlags will be set to a bitwise combination of:

- a. KCF\_SHIFT (0x00000004) indicates that the shift key was down.
- b. KCF\_CONTROL (0x00000008) indicates that the control key was down.
- c. KCF\_SHIFTLOCK (0x00010000) indicates that shift-lock was in effect.
- d. KCF\_LWINDOW (0x00020000) indicates that the left windows key was down.
- e. KCF\_RWINDOW (0x00040000) indicates that the right windows key was down.
- f. KCF\_MENU (0x20000000) indicates that the menu (alt) key was down.

5. HRESULT GetParentHWND (

VARIANT\* pVar ) [out]

This gets the window handle of the parent to the window that received the mouse event notification.

6. HRESULT GetWndText (

BSTR\* pstr ) [out]

This gets the window text of the window that received the mouse event. The caller is responsible for freeing the string.

7. HRESULT GetParentWndText (

BSTR\* pstr) [out]

This gets the window text of the parent of the window that received the mouse event. The caller is responsible for freeing the string.



## IMoraeEventAA specifications:

### Description:

This stream captures Active Accessibility events, including windows focus, resize, and move events.

### Data stream ID:

IID\_AAEVENTTYPE = 796ADB4D-E4A5-405c-85E0-6A5F57D65CF4

### Interface ID:

IID\_IMoraeEventAA = AFEE7ECF-713A-4801-B3BD-1C9BE90DBC56

### Methods

#### 1. HRESULT FindString (

BSTR str,	[in]
int* pResult,	[out]
int nCriteria )	[in]

This call searches for a string within the event. str is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. nCriteria is a bitwise set of flags that determine how the search is made:

- TXT\_MATCH\_CASE\_SENSITIVE (0x00000001) should be set to look for a case-sensitive match.
- TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
- TXT\_MATCH\_OWN\_WND (0x00000004) should be set to search for the string within the window text of the window that received the AA event.
- TXT\_MATCH\_PARENT\_WND (0x00000008) should be set to search for the string within the window text of the parent window of the window that received the AA event.
- TXT\_MATCH\_TOP\_WND (0x00000010) should be set to search for the string within the window text of the top parent window of the window that received the AA event.

\*pResult will be set to a bitwise combination of the search results:

- MFS\_WND\_TEXT (0x00000001) indicates that the text was found in the window text of the window that received the event.
- MFS\_PARENT (0x00000002) indicates that the text was found in the window text of the parent to the window that received the event.
- MFS\_TOP\_PARENT (0x00000004) indicates that the text was found in the window text of the top parent to the window that received the event.

#### 2. HRESULT GetLocation (

int* pLeft,	[out]
int* pTop,	[out]
int* pRight,	[out]
int* pBottom )	[out]



This call gets the location (in pixels) at which the event occurred, relative to the upper left corner of the screen recording area. Note that X increases going to the right, Y increases going down.

3. HRESULT GetEventDetails (
- |                  |       |
|------------------|-------|
| int* pEventType, | [out] |
| int* pObjType,   | [out] |
| int* pDepth )    | [out] |

This call retrieves details about the AA event.

\* pEventType will be set to the type of AA event. Possible values (defined in WinUser.h) include:

- EVENT\_OBJECT\_FOCUS (0x8005) indicates that an object has received focus.
- EVENT\_SYSTEM\_MOVESIZEEND (0x0000000B) indicates a window has been moved or resized.

\* pObjType will be set to the AA object type. This may be application-specific or refer to a system object. (See documentation for the SetWinEventHook windows SDK function.) Possible values for system objects (defined in WinUser.h) include:

- OBJID\_WINDOW (0x00000000)
- OBJID\_SYSMENU (0xFFFFFFFF)
- OBJID\_MENU (0xFFFFFFFDD)

\* pDepth will be set to the number of steps that the window that fired this event is removed from a top-level window. In other words, a value of 0 indicates that a top-level window was moved or resized. A value of 1 indicates that a child of a top-level window was moved or resized, etc.

4. HRESULT GetParentHWND (
- |                 |       |
|-----------------|-------|
| VARIANT* pVar ) | [out] |
|-----------------|-------|

This gets the window handle of the parent to the window that fired the Active Accessibility event.

5. HRESULT GetTopParentHWND (
- |                 |       |
|-----------------|-------|
| VARIANT* pVar ) | [out] |
|-----------------|-------|

This gets the window handle of the top-most parent to the window that fired the Active Accessibility event.

6. HRESULT GetWndText (
- |              |       |
|--------------|-------|
| BSTR* pstr ) | [out] |
|--------------|-------|

This gets the window text of the window that fired the Active Accessibility event. The caller is responsible for freeing the string.

7. HRESULT GetParentWndText (
- |             |       |
|-------------|-------|
| BSTR* pstr) | [out] |
|-------------|-------|





IMoraeEventKeystroke specifications:

Description:

This stream captures keystroke events. At this time, only keydown events are recorded.

Data stream ID:

IID\_KEYEVENTTYPE = 008D470B-493C-4974-9BD6-1112D90AC68B

Interface ID:

IID IMoraeEventKeystroke = F8E9DAB6-4645-43A7-A82D-6AC56ADD108A

Methods

1. HRESULT GetLocation ( 

int* pLeft,	[out]
int* pTop,	[out]
int* pRight,	[out]
int* pBottom )	[out]

This call gets the location (in pixels) at which the event occurred, relative to the upper left corner of the screen recording area. Note that X increases going to the right, Y increases going down.

2. HRESULT GetEventDetails ( 

int* pEventType,	[out]
int* pKey,	[out]
int* pModifiers,	[out]
int* pDepth,	[out]
int* pCodes )	[out]

This call retrieves the keystroke event that was recorded.

\* pEventType will be whether this is a key-up or key-down event. A value of '1' indicates key down, '0' indicates key up. Currently, Morae only captures key down events.

\* pKey will be set to the virtual key code of the keystroke. Values of virtual key codes are defined in WinUser.h.

\* pModifiers receives bit-wise flags to indicate any modifier keys that were depressed along with the keystroke. Possible values are:

- a. KCF\_SHIFT (0x00000004)
- b. KCF\_CONTROL (0x00000008)
- c. KCF\_SHIFTLOCK (0x00010000)
- d. KCF\_LWINDOW (0x00020000)
- e. KCF\_RWINDOW (0x00040000)
- f. KCF\_MENU (0x20000000)

\* pDepth will be set to the number of steps that the window that fired this event is removed from a top-level window. In other words, a value of 0 indicates that a top-level window received



this keystroke. A value of 1 indicates that a child of a top-level window received this keystroke, etc.

\*pCodes receives information about the repeat count, scan code, and other flags. For details, see the documentation for the SetWindowsHookEX Windows API call.



## IMoraeEventText specifications:

### Description:

This stream captures text that applications display on the screen.

### Data stream ID:

IID\_TEXTEVENTTYPE = 21351BCA-4590-40fc-B470-F6066D81C59A

### Interface ID:

IID\_IMoraeEventText = B47BEB4A-AE9E-451f-9CEF-7E9BC4D00825

### Methods

#### 1. HRESULT FindString (

BSTR str,	[in]
int* pResult,	[out]
int nCriteria )	[in]

This call searches for a string within the event. str is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. nCriteria is a bitwise set of flags that determine how the search is made:

- TXT\_MATCH\_CASE\_SENSITIVE (0x00000001) should be set to look for a case-sensitive match.
- TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
- TXT\_MATCH\_CAPTURED\_TXT (0x00000020) should be set to search for the string within the text captured as this event.
- TXT\_MATCH\_TOP\_WND (0x00000010) should be set to search for the string within the window text of the top parent window of the window that captured the text event.

\*pResult will be set to a bitwise combination of the search results:

- MFS\_CAPTURED\_TEXT (0x00000020) indicates that the text was found in the captured text of the event.
- MFS\_TOP\_PARENT (0x00000004) indicates that the text was found in the window text of the top parent to the window that received the event.

#### 2. HRESULT GetLocation (

int* pLeft,	[out]
int* pTop,	[out]
int* pRight,	[out]
int* pBottom )	[out]

This call gets the location (in pixels) at which the event occurred, relative to the upper left corner of the screen recording area. Note that X increases going to the right, Y increases going down.

#### 3. HRESULT GetEventDetails (

```

int* pFlags, [out]
int* pDepth, [out]
int* pAveWidth, [out]
int* pAveHeight ) [out]

```

This call retrieves details about the text event that was recorded.

\* pFlags will be set to a bitwise combination of flags that specify whether this text output occurred in the active application or active window. Possible values include:

- a. TCF\_ACTIVE\_APP (0x00000001) indicates that the application was active at the time of the capture.
- b. TCF\_ACTIVE\_WND (0x00000002) indicates this text was captured from the active window.
- c. TCF\_ACTIVE\_ACC (0x00000004) indicates that this text was captured as part of an Active Accessibility event.

\*pDepth will be set to the number of steps that the window that fired this event is removed from a top-level window. In other words, a value of 0 indicates that a top-level window displayed this text. A value of 1 indicates that a child of a top-level window displayed this text, etc.

\*pAveWidth and \*pAveHeight will be set to the average width and height (in pixels) of the text characters as they were drawn on the screen.

4. HRESULT GetTopParentHWND ( VARIANT\* pVar ) [out]

This gets the window handle of the top-most parent to the window that displayed the text.

5. HRESULT GetText ( BSTR\* pstr ) [out]

This gets the text that was captured as this event. The caller is responsible for freeing the string.

6. HRESULT GetTopParentWndText ( BSTR\* pstr ) [out]

This gets the window text of the top-most parent of the window that displayed text on the screen. The caller is responsible for freeing the string.



## IMoraeEventBrowser specifications:

### Description:

This stream captures navigation events in web browsers, including tab changes and page navigation.

### Data stream ID:

IID\_BROWSEEVENTTYPE = C18F48AF-2DE8-40fd-AA87-DAAC8D198625

### Interface ID:

IID\_IMoraeEventBrowser = 79E8074E-D643-4853-AF92-1DB74EF88AE5

### Methods

#### 1. HRESULT FindString (

BSTR str,	[in]
int* pResult,	[out]
int nCriteria )	[in]

This call searches for a string within the event. str is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. nCriteria is a bitwise set of flags that determine how the search is made:

- TXT\_MATCH\_CASE\_SENSITIVE (0x00000001) should be set to look for a case-sensitive match.
- TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
- TXT\_MATCH\_CURRENT\_URL (0x00000040) should be set to search for the string within the URL of this navigation event.
- TXT\_MATCH\_ANY\_WND (0x0000001C) should be set to search for the string within the title of the page.

\*pResult will be set to a bitwise combination of the search results:

- MFS\_CURRENT\_URL (0x00000010) indicates that the text was found in the URL text of the event.
- MFS\_WND\_TEXT (0x00000001) indicates that the text was found in the page title for the navigation event.

#### 2. HRESULT GetLocation (

int* pLeft,	[out]
int* pTop,	[out]
int* pRight,	[out]
int* pBottom )	[out]

This call gets the location (in pixels) at which the event occurred, relative to the upper left corner of the screen recording area. Note that X increases going to the right, Y increases going down.



3. HRESULT GetEventType ( [out]  
int\* pType )  
This call gets the type of event that was recorded. Possible values are:
  - a. BCT\_TABCHANGE (0x00000080) The user switched to a different tab in a tabbed browser.
  - b. BCT\_PAGECHANGE (0x00000001) The browser has finished loading a page.
  
4. HRESULT GetEventDetails ( [out]  
int\* pEventDetails, [out]  
int\* pEventFlags, [out]  
int\* pTargetFlags )  
This method is not currently implemented.
  
5. HRESULT GetURL ( [out]  
BSTR\* pstr )  
This gets the URL of the navigation event. The caller is responsible for freeing the string.
  
6. HRESULT GetLinkURL ( [out]  
BSTR\* pstr )  
This method is not currently implemented.
  
7. HRESULT GetObjectText ( [out]  
BSTR\* pstr )  
This method is not currently implemented.
  
8. HRESULT GetObjectAltText ( [out]  
BSTR\* pstr )  
This method is not currently implemented.
  
9. HRESULT GetObjectTipText ( [out]  
BSTR\* pstr )  
This method is not currently implemented.
  
10. HRESULT GetWndTitleText ( [out]  
BSTR\* pstr )  
This gets the page title of the URL site for this navigation event.



## IMoraeEventMarker specifications:

### Description:

This stream records markers that are entered during a recording or analysis.

### Data stream ID:

IID\_MARKEREVENTTYPE = 84B82AB8-D096-43ef-A043-D30E2382C448

### Interface ID:

IID\_IMoraeEventMarker = D69059AF-0660-4768-AEEB-2E8764015E53

### Methods

#### 1. HRESULT FindString (

BSTR str,	[in]
int* pResult,	[out]
int nCriteria )	[in]

This call searches for a string within the event. str is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. nCriteria is a bitwise set of flags that determine how the search is made:

- TXT\_MATCH\_CASE\_SENSITIVE (0x00000001) should be set to look for a case-sensitive match.
- TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
- TXT\_MATCH\_NAME (0x00000800) should be set to search for the string within the marker name.
- TXT\_MATCH\_NOTE (0x00001000) should be set to search for the string within the note of the marker.
- TXT\_MATCH\_CREATOR (0x00002000) should be set to search for the string within the name of the creator of the marker.

\*pResult will be set to a bitwise combination of the search results:

- MFS\_NAME (0x00000200) indicates that the text was found in the name of the marker.
- MFS\_NOTE (0x00000400) indicates that the text was found in the note of the marker.
- MFS\_CREATOR (0x00000800) indicates that the text was found in the creator name of the marker.

#### 2. HRESULT GetMarkerType (

int* pType )	[out]
--------------	-------

This call gets the type of the marker.

#### 3. HRESULT GetName (

BSTR* pstr )	[out]
--------------	-------

This gets the name of the marker. The caller is responsible for freeing the string.

4. HRESULT GetCreatorName ( [out]  
BSTR\* pstr )  
 This gets the name of the creator of the marker. The caller is responsible for freeing the string.

5. HRESULT GetNote ( [out]  
BSTR\* pstr )  
 This gets the marker note. The caller is responsible for freeing the string.

6. HRESULT GetAttr ( [in]  
int nAttrID,  
int\* pAttr ) [out]  
 This method retrieves a specific attribute of the marker. nAttrID specifies the attribute.  
 Currently supported values include:

- a. MRK\_ATTR\_TYPE (0x00000001) This asks for the marker type (same information as a call to GetMarkerType).
- b. MRK\_ATTR\_CREATOR\_TYPE (0x00000002) This asks for the maker creator type. Possible values that are output in \*pAttr are:
  - i. MRK\_CREATOR\_TYPE\_MANAGER (0x00000000) The marker was created in Manager.
  - ii. MRK\_CREATOR\_TYPE\_RV (0x00000001) The marker was created by a Remote Viewer.
  - iii. MRK\_CREATOR\_TYPE\_PARTICIPANT (0x00000004) The marker was created by the participant.
  - iv. MRK\_CREATOR\_TYPE\_LOCAL (0x00000003) The marker was created by a local logger (while recording with a camera as the main video source).
  - v. MRK\_CREATOR\_TYPE\_COM (0x00000002) The marker was created by a COM client.

7. HRESULT GetDisplayPath ( [out]  
BSTR\* pstr )  
 This method gets the folder path within the Manager treeview where this marker is found. The caller is responsible for freeing the string.

8. HRESULT GetAudioNoteFileName ( [out]  
BSTR\* pstr )  
 This gets the full path to the audio note that is associated with this marker. Note that this path will change if a new audio note is recorded for the marker. The caller is responsible for freeing the string.

9. HRESULT GetMarkerScoreID ( [out]  
int\* pScore )





IMoraeEventTask specifications:

Description:

This stream records tasks that are logged during a recording or its analysis in Manager.

Data stream ID:

IID\_TASKEVENTTYPE = D0F89A2B-DF21-47f9-8507-9E538B6A41A7

Interface ID:

IID\_IMoraeEventTask = 904CB9D1-8A61-4326-9949-D84B887741F1

Methods

1. HRESULT FindString (

BSTR str, [in]  
int\* pResult, [out]  
int nCriteria ) [in]

This call searches for a string within the event. str is the string to find. Note that if a case-insensitive search is to be made and the full-string flag is not set, this should be a string of all lower-case characters. nCriteria is a bitwise set of flags that determine how the search is made:

- a. TXT\_MATCH\_CASE\_SENSITIVE (0x00000001) should be set to look for a case-sensitive match.
- b. TXT\_MATCH\_FULL\_STRING (0x00000002) should be set to match an entire string (as opposed to finding a substring within a larger string).
- c. TXT\_MATCH\_NAME (0x00000800) should be set to search for the string within the task name.
- d. TXT\_MATCH\_NOTE (0x00001000) should be set to search for the string within the note of the task.

\*pResult will be set to a bitwise combination of the search results:

- d. MFS\_NAME (0x00000200) indicates that the text was found in the name of the task.
- e. MFS\_NOTE (0x00000400) indicates that the text was found in the note of the task.

2. HRESULT GetTaskID(

UINT\* pType ) [out]

This call gets the type of the task.

3. HRESULT GetName (

BSTR\* pstr ) [out]

This gets the name of the task. The caller is responsible for freeing the string.

4. HRESULT GetTaskScoreID(

int\* pScore ) [out]

This call gets the score that has been set for the task.





11. HRESULT GetAudioNoteFileName (

BSTR\* pstr)

[out]

This gets the full path to the audio note that is associated with this task. Note that this path will change if a new audio note is recorded for the task. The caller is responsible for freeing the string.

12. HRESULT SetTaskID(

UINT nType )

[in]

This call sets the type of the task.

13. HRESULT SetTaskScoreID(

int nScore )

[in]

This call sets the score that has been set for the task.

14. HRESULT GetTaskDuration(

VARIANT\* pVarDuration)

[out]

\*pVarDuration will be set to a VT\_I8 type. Its lVal will be set to the total duration of the task expressed in 100 nanosecond units.





