Morae®
TechSmith®

Morae Manager Extensions

Introduction:

Plug-ins can extend the functionality of Morae Manager by adding commands to the main menu of Manager. These menu commands will appear under "Plugins" in Manager's main menu. A plug-in will be able to specify which, if any, of its commands will be displayed on each tab in Manager's main user interface (UI). If desired, the plug-in can receive notification of events occurring in the main UI that result from user-driven actions. The plug-in can query Manager for information and issue commands for Manager to carry out specific actions.
Three COM interfaces are involved in extending Manager's functionality:

1. IManagerExtension – This is the primary extension interface that must be implemented by the plug-in. Manager will call this interface to request a list of commands implemented by the plug-in and to provide the plug-in with an IMoraeManagerApp interface.

2. IMoraeManagerApp – This interface is implemented by Morae Manager. The plug-in can use this interface to query for information and to issue commands.

3. IMoraeManagerAppEvents – This optional interface can be implemented by plug-ins that require notification of user-driven events that occur in Manager.

These three interfaces are described in more detail below. To provide an extension for Manager the plug-in must register itself as a COM object, providing its CLSID, and indicate that it implements the CATID_ManagerAppLevelPlugin category (789F23C7-B3DB-47DA-8C57-15243D4A8CB9). Manager will load the plug-in and provide it with an IMoraeManagerApp interface via a call to IManagerExtension::SetApplication.

If the plug-in wishes to receive event notifications, it must implement the IMoraeManagerAppEvents interface and attach to Manager's connection point. To do this, the plug-in must:

1. Call QueryInterface on the IMoraeManagerApp interface to get its IConnectionPointContainer interface.

2. Call IConnectionPointContainer::FindConnectionPoint with the IID_IMoraeManagerAppEvents id (defined below) to get an IConnectionPoint interface.

3. Call IConnectionPoint::Advise to subscribe to notifications. Note that the plug-in is expected to call IConnectionPoint::Unadvise to allow Manager to release its interface when it receives a call to IManagerExtension::SetApplication with a NULL interface pointer as an argument.

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 3.3.0), Morae is only offered in an English version.

Interfaces:

1. IManagerExtension (derives from IUnknown)

The plug-in must implement this interface. Manager will call the appropriate methods on this interface to provide an IMoraeManagerApp interface and to determine what menu commands the plug-in would like to add to Manager's main menu.

Interface ID:

IID_IManagerExtension = 956A298A-CC6D-4F00-9C10-A28303B12DF2

Methods:

a. HRESULT GetCmdCount(

REFGUID guidView,	**[in]**
int* pnCmdCount)	**[in]**

This method asks for the number of commands that will be added to the plug-in menu for this view.

Arguments:

i. guidView – Ref to a GUID that identifies the tab in the Manager UI that these commands will be displayed on. The possible values are:
   a. 56447BD5-097D-4038-8442-A2005A6D3D80 (Analyze tab)
   b. 67F4DE0F-57DF-430C-8F34-32070C90F13B (Graph tab)
   c. 33488129-2CAC-49AF-83D2-F997ED137C97 (Present tab)

ii. pnCmdCount – Pointer to an int value that should be set to the number of commands.

b. HRESULT GetCmdParam(

REFGUID guidView,	**[in]**
int nCmdIndex,	**[in]**
BSTR* pstr,	**[out]**
int* pnCmdID,	**[out]**
long nLocaleID)	**[in]**

This method is called to get the details of each command that will be added to the menu. It will be called for each command, based on the output from GetCmdCount.

Arguments:

i. guidView – Ref to a GUID that identifies the tab in the Manager UI that these commands will be displayed on. The possible values are listed above under GetCmdCount.

ii. nCmdIndex –1-based index into the list of commands.

iii. pstr – Pointer to a BSTR value. It should be set to the string that should be displayed in the menu. The plug-in should allocate the system string. The caller will de-allocate.
iv. pnCmdID - Pointer to an int value that should be set to the ID that the plug-in wants to assign to this command. When the user selects this command, this is the ID that will be provided in the call to ExecuteCmd.
v. nLocaleID – This is a LCID value that indicates the language that the string should be in.

c. HRESULT GetExtensionName(

BSTR* pstr,	**[out]**
long nLocaleID)	**[in]**

This method requests the name of the extension in a form that can be displayed to the user.

Arguments:

i. pstr –Pointer to a BSTR value. This should be set to a string that can be displayed to the user to identify the plug-in. It will appear as the label of the flyout menu item that will hold the list of commands. The plug-in allocates this system string. The caller will de-allocate.
ii. nLocaleID – This is a LCID value that indicates the language that the string should be in.

d. HRESULT ExecuteCmd(

int nCmdID,	**[in]**
VARIANT varHwnd)	**[in]**

This method requests that the plug-in execute the command that was selected by the user.

Arguments:

i. nCmdID – Specifies the ID of the command that the user selected. This ID was provided by the plug-in in the GetCmdParam method.
ii. varHwnd - Provides the HWND of Manager's main window. If the command requires display of a dialog, this HWND should be the owner of the dialog. The datatype of this variant may be VT_UI4 or VT_UI8.

e. HRESULT SetApplication(

IMoraeManagerApp* pApp)	**[in]**

This method provides the plug-in with Manager's IMoraeManagerApp interface. This interface (described below) can be used to query information from and issue commands to Manager.

Arguments:

i. pApp – Pointer to Manager's IMoraeManagerApp interface. If the plug-in stores this pointer, it should call AddRef on it. Each AddRef call must be balance by a Release call when the plug-in no longer needs the interface.

This pointer may be NULL. In that case, the plug-in must release an existing pointer and call Unadvise on Manager's IConnectionPoint interface if it has requested event notification.

2. IMoraeManagerApp (derives from IUnknown)

This interface is implemented by Manager to provide information to plug-ins and to execute command from plug-ins. Plug-ins can also call QueryInterface on IMoraeManagerApp to get an IConnectionPointContainer interface as described above to subscribe to event notification.

Interface ID:

IID_IMoraeManagerApp = FE8F168E-E75C-4015-9A6A-FFE93B1EEE16

Enumerations:

a. ManCmndType

i. ManCmndSetPlayerTime
Commands the video display in Manager to move to a particular time in the recording.

ii. ManCmndEditItemName
Commands Manager to bring up the in-place editor for an item in the treeview.

iii. ManCmndPlayEvent
Commands Manager to play the time segments encompassed by an event from the recording.

iv. ManCmndZoomToEvent
Commands Manager to zoom its timeline display to the outer boundaries of the time segments encompassed by an event from the recording.

v. ManCmndAddWndToViewMenu
Commands Manager to add a window to its View menu so that the user can choose to show or hide the window.

vi. ManCmndRemoveWndFromViewMenu
Commands Manager to remove a window from its view menu that had previously been added via a ManCmndAddWndToViewMenu command.

b. ManParamType

i. ManParamGetPlayerTime
Requests the current time on display in the player.

ii. ManParamIsPlayerPlaying
Asks Manager whether the player is currently playing.

iii. ManParamGetPlayInOutTimes
Requests the current times of the in and out markers in the player's timeline.

iv. ManParamGetMainHwnd
Requests the HWND of Manager's main window.

Methods:

a. HRESULT RunManagerCommand(

ManCmndType nCmndType,	**[in]**
VARIANT varParam1,	**[in]**
VARIANT varParam2)	**[in]**

This requests that Manager run one of the commands identified in the ManCmndType enumeration.

Arguments:

i. nCmndType – Identifies the desired command.
ii. varParam1 – VARIANT to provide additional information. This depends on the command type.
iii. varParam2 - VARIANT to provide additional information. This depends on the command type.

Supported values for nCmndType and the VARIANT types:

i. ManCmndSetPlayerTime
    a) varParam1 – Type = VT_I8: Set the llVal to the target time expressed in 100 nanosecond units from the start of the recording.
    b) varParam2 – Not used.
ii. ManCmndEditItemName
    a) varParam1 – Type = VT_UNKNOWN: Set the punkVal to a pointer to the IUnknown interface of an object that also implements IDisplayableItem.
    b) varParam2 – Not used.
iii. ManCmndPlayEvent
    a) varParam1 – Type = VT_UNKNOWN: Set the punkVal to a pointer to the IUknown interface of an object that also implements IMoraeEvent.
    b) varParam2 – Not used.
iv. ManCmndZoomToEvent
    a) varParam1 – Type = VT_UNKNOWN: Set the punkVal to a pointer to the IUknown interface of an object that also implements IMoraeEvent.
    b) varParam2 – Not used.
v. ManCmndAddWndToViewMenu
    a) varParam1 – Type = VT_BSTR: Set the bstrVal to the string that should be added to the View menu.
    b) varParam2 – Type = VT_UI4: Set the ulVal to the HWND of the window that the user should be allowed to show or hide.
vi. ManCmndRemoveWndFromViewMenu
    a) varParam1 – Type = VT_BSTR: Set the bstrVal to the string that should be removed from the View menu.
    b) varParam2 – Type = VT_UI4: Set the ulVal to the HWND of the window that the user should no longer be allowed to show or hide.

b. HRESULT GetManagerParameter (

ManParamType nParamType,	**[in]**
VARIANT varModifier,	**[in]**
VARIANT* pvarParam)	**[out]**

This requests information from Manager. Arguments will depend on the type of information requested.

Arguments:

i. nParamType – Member of the ManParamType enumeration that identifies the type of information requested.
ii. varModifier –VARIANT whose type depends on the type of information requested.
iii. pvarParam – VARIANT whose value will be set to the requested information.

Supported values for nCmndType and the VARIANT types:

i. ManParamGetPlayerTime
    a) varModifier – Not used.
    b) pvarParam – Will be set to a type of VT_I8. The llVal will be set to the current player time in 100 nanosecond units from the start of the recording.
ii. ManParamIsPlayerPlaying
    a) varModifier – Not used.
    b) pvarParam – Will be set to a type of VT_BOOL. The boolVal will be set to VARIANT_TRUE if the player is currently playing, VARIANT_FALSE if not.
iii. ManParamGetPlayInOutTimes
    a) varModifier – Not used.
    b) pvarParam – Will be set to a type of VT_I8 | VT_ARRAY. The parray value will be set to a LPSAFEARRAY of REFERENCE_TIME values that give the in and out points in 100 nanosecond units from the start of the video.
iv. ManParamGetMainHwnd
    a) varModifier – Not used.
    b) pvarParam – Will be set to a type of VT_UI4. The ulVal will be set to the HWND of Manager's main window.

3. IMoraeManagerAppEvents(derives from IUnknown)

A Manager extension plug-in can optionally implement this interface to receive notifications from Manager of user-driven events.

Interface ID:

IID_IMoraeManagerAppEvents = 92E4739A-A551-40FC-B2A5-6C27B4B2FB38

Methods:

a. HRESULT NotifyEvent(

REFGUID idEventType, [in]
VARIANT varData) [in]

Arguments

i. idEventType – Reference to a GUID that identifies the event type. Details of possible values are listed below.
ii. varData – VARIANT that contains the data that describe the event. Details are provided below.

The types of events that plugins can listen for are listed in the following table:

Event Name	Event GUID	Description of Event
Object Created	C394F78B-D40B-4DBC-8EEB-9186720DFF65	Occurs after a new object has been created
Object Deleted	D39CFAEA-E81C-4609-A961-1B069884FD66	Occurs after an object has been deleted
Object Properties Changed	38B72223-8B98-48E5-ADC1-03F0FE6BE17B	Occurs after an object has been modified
Object Selected	0CF2EF72-94F1-41D2-B9F8-245687A35BA1	Occurs after a new object has been selected in the project tree view
Player End Reached	C31DB0E9-A4CE-499F-A1F9-24BC5D02C3AB	Occurs after an object has been played to its end
Player Object Changed	17002E7B-BFDA-438C-95CE-6E6EB5A681C4	Occurs after a new object has been loaded into the playing screen
Player Paused	D1304D27-3517-4D43-A58D-1C9A7AFC2724	Occurs after the object playing has been paused
Player Skipped	FB8278F1-C3C2-416F-81A5-2CFE1710BEF2	Occurs after an object playing skips forward or backward
Player Started	D1304D27-3517-4D43-A58D-1C9A7AFC2724	Occurs after an object starts playing
Project (Workspace) Closing	4CC86A83-09E9-45AA-9558-6FFD06A44215	Occurs before a project closes. Allows plugins time to perform cleanup operations
Project (Workspace) Opened	E12DCC26-0A3D-4149-A568-86FCFFFD1E08	Occurs immediately after a project has been opened
Tab Changed	FDA5D326-23DE-4BEE-BF2-D6C61797F7B7	Occurs after a different tab is selected (i.e. Analyze, Graph, Present)

idEventType is set to the GUID corresponding to the event that has occurred.
varData is set to a value specific to the type of event that has occurred.

The values for varData for each event are shown in the table below:

Event Name	VARIANT Type	Description of VARIANT Data
Object Created	VT_UNKNOWN	Contains an IUnknown pointer to the object that was created
Object Deleted	VT_UNKNOWN	Contains an IUnknown pointer to the object that was deleted. Note: IMoraeObj functions are still safe to call on this object.
Object Properties Changed	VT_UNKNOWN	Contains an IUnknown pointer to the object whose properties were changed
Object Selected	VT_UNKNOWN	Contains an IUnknown pointer to the object that was selected in the project tree
Player End Reached	VT_I8	Contains the time reached at the end of the object before the player time is reset
Player Object Changed	VT_UNKNOWN or VT_EMPTY	Contains an IUnknown pointer to the object that was loaded into the player or is empty if no new object was loaded (e.g. a non-playable object was selected)
Player Paused	VT_I8	Contains the time at which the player has stopped
Player Skipped	VT_I8	Contains the time at which the player is resuming playing
Player Started	VT_I8	Contains the time at which the player has started playing
Project (Workspace) Closing	VT_UNKNOWN	Contains an IUnknown pointer to the project that is about to close
Project (Workspace) Opened	VT_UNKNOWN	Contains an IUnknown pointer to the project that has been opened
Tab Changed	VT_BSTR	Contains the GUID of the new tab stored as a BSTR

For the Tab Changed event, the GUIDs for the views are listed in the table below

Tab Name	Tab GUID
Analyze	56447BD5-097D-4038-8442-A2005A6D3D80
Graph	67F4DE0F-57DF-430C-8F34-32070C90F13B
Present	33488129-2CAC-49AF-83D2-F997ED137C97

Note that if an event returns an IUnknown pointer, an IMoraeObj pointer can be obtained by calling QueryInterface on the IUnknown pointer. Other interfaces that are available depend on the type of object.