Morae®
TechSmith

Morae Enumeration Interfaces

There are several enumeration interfaces in the Morae plug-in interface set that act as typical COM enumerators. Each allows the caller to step through a set of objects. The interfaces are:

1. **IEnumMoraeRecording**
   IID_IEnumMoraeRecording = FE614A21-2396-41a2-AAAF-61F4E6116B4D
   object interface type = IMoraeRecording
   This enumerator can be acquired by calling IMoraeProject::EnumRecordings (described in Information Supplied by Manager). It is not implemented by plug-ins. Rather, plug-ins that implement the IMoraeEventStream interface can call this method on the IMoraeProject interface provided to it in the IMoraeEventStream::SetProject method (described in Reading and Displaying Events).
2. **IEnumMoraeEvent**
   IID_IEnumMoraeEvent = A0DE954C-40B9-47f4-8FC2-F61D990BC4B1
   object interface type = IMoraeEvent
   Plug-ins that implement IMoraeEventStream or IMoraeEventSearch (described in Reading and Displaying Events) will need to implement this interface to provide an enumerator for events in response to a call to IMoraeEventStream::EnumEvents or IMoraeEventSearch::RunSearch.
3. **IEnumMoraeTaskInfo**
   IID_IEnumMoraeTaskInfo = E07834DC-0479-4fb4-AF09-2F7955E4EF0F
   object interface type = IMoraeTaskInfo
   This enumerator is used in plug-ins that implement IMoraeAnalysis (described in Data Analysis for Graphing) to enumerate through the IMoraeTaskInfo interfaces that provide information on the tasks involved in an analysis. An IEnumMoraeTaskInfo interface can be acquired from calling EnumTasks on the IMoraeTaskGroupInfo interface that is provided as an input in the IMoraeAnalysis::SetAnalysisTargets method.
4. **IEnumMoraeTaskGroupInfo**
   IID_ IEnumMoraeTaskGroupInfo = A68A46AF-3EBA-470a-9B3C-B0CB1E0AA777
   object interface type = IMoraeTaskGroupInfo
   This enumerator is provided to plug-ins that implement IMoraeAnalysis (described in Data Analysis for Graphing) as an argument in the IMoraeAnalysis::SetAnalysisTargets method. It can be used to enumerate through the groups of tasks that make up an analysis set.

5. **IEnumMoraeRecordingInfo**
   IID_ IEnumMoraeRecordingInfo = 7F565C0D-EB49-424e-9E99-0868850D13F8
   object interface type = IMoraeRecordingInfo
   This enumerator is provided to plug-ins that implement IMoraeAnalysis (described in Data Analysis for Graphing) as an argument in the IMoraeAnalysis::SetAnalysisTargets method. It can be used to enumerate through the recordings that make up an analysis set.
6. **IEnumMoraeProject**
   IID_ IEnumMoraeProject = 76079676-C1ED-4E95-A51C-F708E04D1624
   object interface type = IMoraeProject
   This interface allows a plug-in to enumerate through the projects that are loaded into a workspace. It is acquired by calling IMoraeWorkspace::EnumProjects (described in Information Supplied by Manager). The IMoraeWorkspace interface is provided to Manager plug-ins that implement the IMoraeManagerAppEvents interface (described in Morae Manager Extensions) when a workspace is opened or closed.
7. **IEnumDisplayableItem**
   IID_ IEnumDisplayableItem = 847ECB88-F0F6-44D8-B696-BD0EC114F732
   object interface type = IDisplayableItem
   This enumerator should be implemented by Manager plug-ins that implement IDisplayableItem (described in Reading and Displaying Events). Manager will use this interface to enumerate through the subitems that should be displayed.

These interfaces derive from IUnknown.

Methods:

1. **HRESULT Skip(**
   ULONG nCount) **[in]**
   This call skips over nCount objects in the enumerator. If the nCount is more than the remaining number of items in the enumerator, this method returns S_FALSE. A subsequent call to Next is affected by this call.

2. **HRESULT Reset()**
   This call resets the enumerator to the first item in its collection. A subsequent call to Next is affected by this call.

3. **HRESULT Next(**

    ULONG nCount, [in]
    <object interface type> ppIObject, [out]**
    ULONG* pFetched) [out]

    nCount indicates the number of items to fetch in this call. If nCount is > 1, the ppIObject input should be a pointer to an array that is sized to hold at least nCount interfaces. The enumerator does not perform any memory allocation. *pFetched will be set to the number of interfaces that were actually output. If nCount == 1, pFetched can be NULL. This method returns S_OK if at least 1 interface was fetched in this call. If none were fetched (usually indicating that the caller has stepped through all of the available objects), S_FALSE is returned. (For this reason, if Next is called in a loop to iterate through all of the objects in the collection, the SUCCEEDED macro should not be used on the return value to determine whether to continue the loop.) The caller must call Release on each interface when finished with it.

4. **HRESULT Clone(**

    <enum interface type> ppIEnum) [out]**

    This call outputs a copy of the enumerator. Note that this new enumerator will have the same internal state as the existing one. The caller must call Reset on it if items are to be enumerated from the beginning of the collection. The new enumerator must be released by the caller when no longer needed.