```
#region Assembly Microsoft.VisualStudio.Help.Runtime.dll, v2.0.0.0
// C:\Program Files\Microsoft Help Viewer\v2.0\Microsoft.VisualStudio.Help.Runtime.dll
#endregion

using System;
using System.ComponentModel;
using System.Runtime.InteropServices;

namespace Microsoft.VisualStudio.Help.Runtime
{
    [Guid("947FE65C-0636-4CBD-A96D-5D5F4FDD40EA")]
    [InterfaceType(ComInterfaceType.InterfaceIsDual)]
    [ComVisible(true)]
    [Description("State object, holds open catalog and all info about catalog")]
    public interface ICatalog
    {
        string ContentPath { get; }
        bool IsOpen { get; }

        [DispId(3)]
        [Description("method Close - closes a currently open catalog")]
        void Close();
        [Description("Method GetReadWriteLock - Return a reference to the catalog read write lock object for a given location")]
        [DispId(7)]
        ICatalogReadWriteLock GetReadWriteLock(string catalogPath);
        [Description("Method IsLanguageSupportAvailable - Check to see if the natural language support is installed on the system for a given language")]
        [DispId(6)]
        int IsLanguageSupportAvailable(string lang);
        [DispId(1)]
        [Description("method Open - opens a catalog given a valid catalog directory path and list of prioritized languages for fallback")]
        void Open(string path, string[] prioritizedLocales);
        [Description("method OpenMshx - opens an MSHX catalog given a valid path to the MSHX file")]
        [DispId(2)]
        void OpenMshx(string path);
    }

    [Guid("CB26DC0A-7E69-4BDB-A82F-37A785D51B93")]
    [Description("Stateless object, does fetch action based on catalog object passed in")]
    [InterfaceType(ComInterfaceType.InterfaceIsDual)]
    [ComVisible(true)]
    public interface ICatalogRead
    {
        [Description("method GetTopic - returns a data stream containing the XHTML data for a topic identified by its topic id in an open catalog")]
        [DispId(1)]
        IStream GetIndexedTopic(ICatalog catalog, string topicId, IHelpFilter filter);
        [Description("method GetIndexedTopicDetails - returns an ITopic interface containing the details of a topic identified by its topic id in an open catalog")]
        [DispId(7)]
        ITopic GetIndexedTopicDetails(ICatalog catalog, string topicId, IHelpFilter filter);
        [SuppressMessage("Microsoft.Design", "CA1026:DefaultParametersShouldNotBeUsed", Justification = "Used to support the COM interface")]
        [DispId(6)]
        [Description("method GetKeywords - returns a collection containing the catalogs keywords")]
        IKeywordCollection GetKeywords(ICatalog catalog, bool useCache = true);
        [Description("method GetLinkedAsset - returns a data stream containing the requested asset.  Asset is identified by its package name and file path within the package within the currently open
catalog")]
        [DispId(2)]
        IStream GetLinkedAsset(ICatalog catalog, string packageName, string path, string locale);
        [Description("method GetSearchResults - returns a data stream of search results in open search XML format for a given query within a catalog.  filterCriteria is an array of filter criteria to refine
the search results.  Supports paging.")]
        [DispId(3)]
        ITopicCollection GetSearchResults(ICatalog catalog, string query, IHelpFilter filter, SearchOptions options, int pageSize, int pageNumber, out int totalSearchResults);
        [DispId(4)]
        [Description("method GetTableOfContents - returns a data stream of ToC data in XML format.  ToC is from the starting point of the topic id and matching the topic locale and topic version (to support
multiple locale and versions in a catalog).  The return detail can be one of four levels (Children of the topicId, Siblings of the topicId, Ancestors of the topicId, or all ToC Root Nodes)")]
        ITopicCollection GetTableOfContents(ICatalog catalog, string topicId, IHelpFilter filter, TocReturnDetail returnDetail);
        [DispId(8)]
        [Description("method GetTopicDetailsForF1Keyword - returns an ITopic interface containing the matching topic for the F1 keyword.  F1 keywords consists of an array of prioritized F1 keywords")]
        ITopic GetTopicDetailsForF1Keyword(ICatalog catalog, string[] prioritizedF1Keywords, IHelpFilter filter);
        [DispId(5)]
        [Description("method GetTopicForF1Keyword - returns a stream containing the matching topic for the F1 keyword in XHTML format.  F1 keywords consists of an array of prioritized F1 keywords")]
        IStream GetTopicForF1Keyword(ICatalog catalog, string[] prioritizedF1Keywords, IHelpFilter filter);
    }



    [Description("Catalog lock state object, holds catalog lock and all info about catalog lock")]
```

```
[ComVisible(true)]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
[Guid("E087C981-50BC-4081-9CC7-6B8077D354BE")]
public interface ICatalogReadWriteLock
{
    bool IsCurrentlyNonBlockingWriteLocked { get; }
    bool IsCurrentlyWriteLocked { get; }

    [Description("method EnterReadLock - creates a read lock")]
    [DispId(5)]
    bool EnterReadLock(int timeout);
    [Description("method EnterWriteLock - creates a write lock")]
    [DispId(7)]
    bool EnterWriteLock(int timeout);
    [DispId(2)]
    [Description("method ExitNonBlockingWriteOperation - exits a non blocking write lock")]
    void ExitNonBlockingWriteOperation();
    [Description("method ExitReadLock - exits the read lock")]
    [DispId(6)]
    void ExitReadLock();
    [DispId(8)]
    [Description("method ExitWriteLock - exits the write lock")]
    void ExitWriteLock();
    [DispId(1)]
    [Description("method StartNonBlockingWriteOperation - creates a non blocking write lock")]
    bool StartNonBlockingWriteOperation(int timeout);
}



[Description("Collection of filter criteria name/value pairs")]
[ComVisible(true)]
[Guid("03B187B9-926C-4FDE-91D0-BE84467696F5")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IHelpFilter
{
    [DispId(2)]
    [Description("method Add - Adds a filter key/value pair to the filter list")]
    void Add(string key, string value);
    [DispId(1)]
    [Description("method AddRange - Adds a criteria key and matching range of values to the filter list")]
    void AddRange(string key, string[] value);
    [Description("method ContainsKey - Check to see if a filter key is in the filter list")]
    [DispId(4)]
    bool ContainsKey(string key);
    [Description("method Count - returns the number of filter elements in the collection")]
    [DispId(6)]
    int Count();
    [Description("method ElementAt - returns filter value at a particular index in the collection")]
    [DispId(7)]
    IHelpKeyValuePair ElementAt(int index);
    [DispId(5)]
    [Description("method Item - returns the filter value of a provided filter key")]
    string[] Item(string key);
    [DispId(3)]
    [Description("method Remove - Remove a criteria key/value pair from the filter list")]
    bool Remove(string key);
}


[Guid("599A8FE5-EF2C-4B7F-80AE-B84855BDB2BA")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
[Description("Key/value pair for IHelpFilter objects")]
[ComVisible(true)]
public interface IHelpKeyValuePair
{
    string Key { get; }

    [DispId(2)]
    [Description("method Value - get the value of a key/value pair")]
    string[] Value();
}


[ComVisible(true)]
```

```
[Guid("9BE144E4-99D9-4F72-91FE-D0C0C6F6BCC3")]
[Description("Help Keyword interface, contains methods to retrieve keyword information")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IKeyword
{
    string DisplayValue { get; }
    bool IsSubkey { get; }
    ITopicCollection Topics { get; }
    string Value { get; }
}


[InterfaceType(ComInterfaceType.InterfaceIsDual)]
[Guid("E0B62A51-5073-497A-A3E9-A3BF118B75F5")]
[Description("Collection of IKeywords")]
[ComVisible(true)]
public interface IKeywordCollection : IEnumerable, IEnumerator
{
    int Count { get; }
    object Current { get; }
    int PageSize { get; }

    [DispId(4)]
    [Description("Returns an enumerator that iterates through a collection")]
    IEnumerator GetEnumerator();
    [DispId(10)]
    [Description("Loads the keyword index from the provided filename")]
    void Load(ICatalog catalog, string fileName);
    [DispId(1)]
    [Description("Advances the enumerator to the next element of the collection")]
    bool MoveNext();
    [DispId(6)]
    [Description("Sets current element in the collection the specified index")]
    void MoveTo(int index);
    [DispId(7)]
    [Description("Sets current element in the collection the first occurrence of the supplied string")]
    int MoveToKeyword(string partialKeyword);
    [Description("Sets the enumerator to its initial position, which is before the first element in the collection")]
    [DispId(2)]
    void Reset();
    [Description("Saves the keyword index to the provided filename")]
    [DispId(9)]
    void Save(string fileName);
}


[InterfaceType(ComInterfaceType.InterfaceIsDual)]
[Guid("C9BFBF64-A2BC-4925-BFFB-776E76CF4D17")]
[Description("Help topic interface, contains methods to retrieve topic information")]
[ComVisible(true)]
public interface ITopic
{
    ICatalog Catalog { get; }
    string Description { get; }
    string DisplayVersion { get; }
    string Id { get; }
    string Locale { get; }
    string Package { get; }
    string ParentId { get; }
    string ParentTopicLocale { get; }
    string ParentTopicVersion { get; }
    bool TableOfContentsHasChildren { get; }
    string TableOfContentsPosition { get; }
    string Title { get; }
    string TopicLocale { get; }
    string TopicVersion { get; }
    string Url { get; }
    string Vendor { get; }

    [DispId(8)]
    [Description("Returns topic category (if defined in the content)")]
    string[] Category();
    [Description("Returns the content filter value for the topic")]
    [DispId(12)]
    string[] ContentFilter();
```

```
    [DispId(9)]
    [Description("Returns topics content type (if defined in the content)")]
    string[] ContentType();
    [Description("method FetchContent - returns a data stream containing the XHTML data for the topic")]
    [DispId(14)]
    IStream FetchContent();
}


[Guid("32F9A41A-CA73-4FAA-839C-5D5DF67CB6D7")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
[Description("Collection of ITopics")]
[ComVisible(true)]
public interface ITopicCollection : IEnumerable, IEnumerator
{
    int Count { get; }
    object Current { get; }

    [Description("Returns an enumerator that iterates through a collection")]
    [DispId(4)]
    IEnumerator GetEnumerator();
    [DispId(1)]
    [Description("Advances the enumerator to the next element of the collection")]
    bool MoveNext();
    [DispId(6)]
    [Description("Sets current element in the collection the specified index")]
    void MoveTo(int index);
    [DispId(2)]
    [Description("Sets the enumerator to its initial position, which is before the first element in the collection")]
    void Reset();
}


[Guid("42BC2BBB-B6E9-4E6F-BC5F-9336AA2630D6")]
[Flags]
[ComVisible(true)]
public enum SearchOptions
{
        None = 0,
        SearchTermHighlight = 1,
        OrSearchOverride = 2,
    }


    [Guid("15EFDFE2-E3CC-4C86-A58F-B12D622782BF")]
    [ComVisible(true)]
    public enum TocReturnDetail
    {
        TocChildren = 0,
        TocSiblings = 1,
        TocAncestors = 2,
        TocRootNodes = 3,
        TocDescendants = 4,
    }

}
```