Colligo Contributor SDK

User Guide

CONTENTS

INTRODUCTION

Providing users a rich desktop experience for accessing and managing SharePoint content is a common issue within SharePoint deployments. Colligo Contributor provides this SharePoint integration through its core engine and main User Interface/Outlook Add-in. Support is taken one step further with Contributor providing a seamless experience in a disconnected (offline) environment. Customers have recognized the power of Contributor and demand more flexibility. To provide customers the greatest flexibility, the Contributor engine has been abstracted into a .Net API.

Limitations

Colligo reserves the right to change the functionality of the API without notice. However, we endeavor to provide customers with a reliable, consistent development environment.

Scope

The API provides developers the ability to develop applications demanding the following functionality:

- Creation of new list items and documents
- Importing existing documents in to the Contributor store
- Getting collections of sites, lists, list items, and documents from the store
- Getting metadata of list items and documents in the store
- Updating the metadata of list items and documents
- Removing list items and documents from the Contributor store
- Initiate synchronization of sites or individual lists
- Register new sites with Contributor, and removing existing registrations

API CONCEPTS

Typically an API application operates as a client to the standard Contributor server application. When the client starts it ensures the Contributor Server application is available. The server application is known as the Contributor instance. The server application provides access to the main Contributor UI through the System Tray icon.

Common Objects

Objects	
`WMLApp`	The WMLApp is the top-most object a custom application will use. It is used to access the Contributor instance.
`WebManager`	The WebManager provides access to sites registered with Contributor.
`SyncManager`	The SyncManager manages the synchronization processing within the Contributor instance.
`IList`	An IList represents a List or Document Library. Content is added-to; updated-in; and removed-from; ILists.
`IListItem`	An IListItem represents an item in a list or library. The IListItem provides access to metadata. Subfolders are IListItems.
Events	
`SyncUpdate`	Receive notifications that a synchronization has completed.
`SettingsUpdate`	Receive notifications that Sync-settings have changed. Notifies changes to: AutomaticSyncEnabled and PeriodicSyncConfigured.

Licensing and Keys

Writing and running API-applications necessitates two valid keys are available:

- **Application Key:** An API-application must connect to an API –enabled Contributor application. Most of the different flavors of Contributor may be configured to support API connections. Your Colligo Sales contact can confirm whether a particular Contributor license key allows API connections.
- **Developer Key**: A key must be supplied to API-enabled Contributor application when the external application initializes a connection to Contributor.

Program Structure

The following table itemizes the steps an application needs to perform to use the API:

Initialization	Get the WebManager from WMLApp Initialize the WebManager passing the Developer Key Initialize the SyncManager
Process	Access a site Get a list from the site
Process Items	Lock the list for sync'ing Add/Update/Delete Content Unlock the list
Release	Release references to List and Site
Shut down	Shut down the SyncManager Shut down the WebManager

API DEVELOPMENT

References and Namespaces

After installing an SDK-enabled version of Contributor a number of libraries are added to the .NET Global Assembly Cache. To develop an application with the Contributor libraries references must be added for the following libraries:

- Colligo.Properties
- Colligo.WML

The libraries provide access to a number of namespaces:

- `Colligo.Properties: Provides access to objects representing metadata`
- `Colligo.WML`: Provides access to the WMLApp, WebManager, IList and IListItem
- `Colligo.WML.Sync`: Provides access to the SyncManager

Development Environment

The Colligo Contributor API is supported in the following development IDEs:

- Visual Studio 2008 and higher

Prerequisites and Platform Support

Deployment of an application using the Colligo SDK has the following prerequisites:

- .NET Framework v3.5 Runtime.
- Colligo Contributor v3.0.0 or higher

Colligo Contributor and the API are supported on the following operating systems:

- Windows XP
- Windows Vista
- Windows 7

64-bit Support

It is possible to build applications targeting 64-bit deployments. For 64-bit compatibility, applications must set the platform target for their .NET assemblies to x86. This is required to ensure that the application is loaded in x86 compatibility mode (since the SDK includes a couple of native DLL's that are compiled for x86).

Version/Update Considerations

An application built against the Contributor API references a specific build of the Contributor libraries. As such, you should be aware that installing an update of Contributor (for example from v3.0.0 to v3.0.1) may demand a rebuild of the API client application.

Synchronization Considerations

Applications developed with the Contributor API operate in **Sync Client** mode. In this mode, the application delegates the synchronization operations to the main Contributor application; if no main Contributor application is running, it is launched.

Developers should be aware that Contributor *may* be configured to synchronize when a list is modified. The following options are available to the developer to manage this behavior:

1. Lock a list for sync while processing using the `list.GetMultipleUpdateLock()` functionality (see **Import New Document into a Library**). Synchronization will *not* occur while a list is locked.
2. Initialize synchronization (see **Initialize Synchronization**).

Performance Considerations

Most assessors within the API are fast; you are advised *not* to cache references to Contributor objects such as IWeb and IList for extended periods as they may be updated by a sync process running in another Contributor instance. We advise keeping a reference to IWeb and IList only for the duration of an Action; for example, uploading a group of files.

Code Snippets

The following sections present code demonstrating how the API can be used to perform some of the most common actions.

Initialization

```
{
    const string DEVELOPER_KEY = "0K1Q8-075YJ-Q8FEN-GS64P-45JXG";
    IWebManager m_colligoWebMgr = null;
    SyncManager m_colligoSyncMgr = null;
    private void initialize()
    {
        // get web manager
        m_colligoWebMgr = WMLApp.GetWebManager();
        // initialize web manager m_colligoWebMgr.Initialize(DEVELOPER_KEY);
        // get and initialize sync manager
        m_colligoSyncMgr = SyncManager.GetInstance();
m_colligoSyncMgr.Initialize();
    }
```

Get Reference to Downloaded Web

```
    {
        IWeb web = m_colligoWebMgr.GetWebByUrl(siteUrl);
    }
```

Url of a web registered with Contributor, eg: http://www.colligo.com/support/

Get a Reference to Downloaded List

```
    {
        IList list = web.GetListByServerName(listId);
    }
```

listId: is the GUID of the List in format "{33C6806F-13B3-414F-865D-AE49A4FD18BD}"

Locate a subfolder

Find an existing subfolder within a library. `IListItem folder =`

```
    {
        lib.GetListItemByPath("/Subfolder1/Subfolder2");
    }
```

Enumerate Downloaded Lists

```
    {
        foreach (IList list in web.Lists)
        {
            Console.WriteLine("List Title: " + list.Title); Console.WriteLine("List
GUID: " + list.ServerName); Console.WriteLine();
        }
    }
```

Enumerate Users

```
    {
        foreach (IUser user in web.Users)
        {
            Console.WriteLine("User Name: " + user.Name); Console.WriteLine("User
ID: " + user.UserID);
            Console.WriteLine();
        }
    }
```

Enumerate List Columns

```
    {
        foreach (PropertyContext ctx in list.Fields)
        {
            Console.WriteLine("Column display name: " + ctx.DisplayName);
Console.WriteLine("Column internal name: " + ctx.Name); Console.WriteLine();
        }
    }
```

Import New Document into a Library

The following code block demonstrates importing a new file into an existing List.

- **folder** is an existing ListItem representing the parent folder; this file should be added to. **null** and may be used to import the file at the root of the library
- **filename** is the name of the file to import (may include the full file path)

```
    {
        // get multiple update lock to defer auto-sync until operations are
        // complete
        using (list.GetMultipleUpdateLock())
        {
            // this call immediately commits changes to database
            IListItem imported = list.ImportFile(folder, filename, null, true);
        }
    }
```

Set Metadata

This code block demonstrates setting metadata values for an existing ListItem:

```
    void SetMetaData(IList list, IListItem item)
    {
        // set text column: single-line, multi-line, or choice;
        // for multiple choices, use ';#' as delimiter
        item.SetProperty("MyChoice", "Apple;#Orange");

        // set numeric or currency column
        item.SetProperty("Numerical_x0020_Column", 2811);

        // set date-time column
        item.SetProperty("Time_x0020_of_x0020_Incident", DateTime.Now + new
TimeSpan(0, 15, 0));

        // set hyperlink column
        item.SetProperty("Hyperlink_x0020_column", new HyperlinkPrimitive("Colligo
Networks", "www.colligo.com"));

        // set lookup or person/group column in form
        // <id>;#<value>;#<id>;#<value>
        // value can be left blank
        item.SetProperty("Task_x0020_Lookup", "53;#Task 53;#57;#Task 57");

        item.SetProperty("MultiPerson", "18;#Andrew Block;#25;#Brent Bolleman");

        // set yes/no column
        item.SetProperty("My_x0020_YesNo", true);

        item.Update();
    }
```

Initialize Synchronization

This code block demonstrates a how to initialize synchronization.

```
    {
        // get and initialize sync manager
        syncMgr = SyncManager.GetInstance();
        syncMgr.Initialize();

        // get web by URL
        IWeb web = webMgr.GetWebByUrl(siteUrl);

        // request that a site be synchronized
        syncMgr.RequestServerSync(web, false);

        // get list using its GUID identifier
        IList lib = web.GetListByServerName(listId);

        // request that an individual list or library be synchronized
syncMgr.RequestListSync(web, lib , false);
    }
```

Download a New Web

This code block demonstrates a method that synchronously downloads a new web.

```
    private SyncResults BlockingDownload(
    SyncManager mgr, string url,
    System.Net.NetworkCredential cred, int secondsTimeout)
    {
        object notifyComplete = new object();
        SyncTransaction transaction = null;
        SyncResults results = null;
        SyncUpdateHandler handleSyncResults = delegate(object source,
           SyncUpdateArgs args)
        {
            if (args.StatusInfo.Transaction.Equals(transaction) &
                args.SyncResults != null)
            {
                results = args.SyncResults;
                lock (notifyComplete)
                {
                    Monitor.Pulse(notifyComplete);
                }
            }
        };
        mgr.SyncUpdate += handleSyncResults;
        transaction = mgr.DownloadWeb(url, cred);
        lock (notifyComplete)
        {
            Monitor.Wait(notifyComplete, secondsTimeout * 1000);
        }
        mgr.SyncUpdate -= handleSyncResults;
        return results;
    }
```

Synchronize a Web

This code block demonstrates a method that synchronously synchronizes an existing web.

```
    private SyncResults BlockingWebSync(
        SyncManager mgr,
        IWeb web,
        int secondsTimeout)
    {
        object notifyComplete = new object();
        SyncTransaction transaction = null;
        SyncResults results = null;
        SyncUpdateHandler handleSyncResults = delegate(object source,
SyncUpdateArgs args)
        {
            if (args.StatusInfo.Transaction.Equals(transaction) &&
args.SyncResults != null)
            {
                results = args.SyncResults;
                lock (notifyComplete)
                {
                    Monitor.Pulse(notifyComplete);
                }
            }
        };
        mgr.SyncUpdate += handleSyncResults;
        transaction = mgr.RequestServerSync(web);
        lock (notifyComplete)
        {
            Monitor.Wait(notifyComplete, secondsTimeout * 1000);
        }
        mgr.SyncUpdate -= handleSyncResults;
        return results;
    }
```

Synchronize a List

This code block demonstrates a method that synchronously synchronizes a list. If a list is not already configured as sync-enabled, this method will enable it.

```
    private SyncResults BlockingListSync(
        SyncManager mgr,
        IWeb web, IList list, int secondsTimeout)
    {
        object notifyComplete = new object();
        SyncTransaction transaction = null;
        SyncResults results = null;
        SyncUpdateHandler handleSyncResults = delegate(object source,
SyncUpdateArgs args)
        {
            if (args.StatusInfo.Transaction.Equals(transaction) &&
args.SyncResults != null)
            {
                results = args.SyncResults;
                lock (notifyComplete)
                {
                    Monitor.Pulse(notifyComplete);
                }
            }
        };
        mgr.SyncUpdate += handleSyncResults; transaction =
mgr.RequestListSync(web, list); lock (notifyComplete) {
Monitor.Wait(notifyComplete, secondsTimeout * 1000); } mgr.SyncUpdate -=
handleSyncResults; return results;
    }
```

Shut down

It is important to shut down the connections to Contributor API after processing is complete. Typically shutting down would be employed in the Finally block of a try/catch region.

```
    public void Dispose()
    {
        // need to shutdown sync manager and web manager in the correct order
if (m_colligoSyncMgr != null)
        {
            m_colligoSyncMgr.Shutdown();
        }
        if (m_colligoWebMgr != null)
        {
            m_colligoWebMgr.Shutdown();
        }
    }
```

Support

For further details on the SDK, please contact our technical specialists at support@colligo.com.