KWizCom Corporation

iMUSH

Information Management Utilities for SharePoint

Printing Feature

Application Programming Interface (API)

Company Headquarters

KWizCom

95 Mural Street, Suite 600

Richmond Hill, Ontario

L4B 3G2, Canada

E-mail: info@KWizCom.com

Web site: http://www.kwizcom.com

Sales

E-mail: sales@KWizCom.com
Telephone: +1-905-370-0333

CONTENTS

INTRODUCTION

GENERAL

iMUSH-Print enables end-users to easily merge and convert list items and documents to a .PDF printable document.

This document describes iMUSH-Print's Application Programming Interface (API), which enables using iMUSH-Print's services from other software.

TARGET AUDIENCE

SharePoint developers.

JAVASCRIPT API (AVAILABLE FROM VERSION 12.1.56)

The API provides a single javascript method to merge items/documents, convert the merged document to PDF. The result PDF document is then sent back to the client via HTTP (end-user sees 'save as' dialog enabling saving the converted document on the client desktop.

SYNTAX AND PARAMETERS

KWizCom.iMUSH.Print.Scripts.PrintItems('SiteUrl', 'ListID', 'ItemIDs', HideOrShowSetingsDialod);

Parameters:

SiteUrl – absolute URL of a SharePoint site

ListID – List Guid ID. For Example: {245a0d7c-1147-4d92-a54b-dc35127ee86d}

ItemIDs – list of item ID's to print delimited by comma. For example: 1,2,3

HideOrShowSetingsDialod – Boolean. Indicates show or hide print settings dialog. Default is false (silent print)

HTTP GET/POST API

PROTOCOL

The API provides a single web method. Calling this method is done by sending a POST or GET http request to "*KWizComPrintFile.aspx*" page.

The method merges a SharePoint page with optionally *connected resources**, into a single .PDF document, and returns the merged document to the client with the appropriate http response header (which causes the local browser to use the local PDF reader to display the returned document).

* **Connected resources definition**:

A SharePoint page may include list/document view web parts. These web parts are connected to resources which are not part of the page. The following are considered resources that are connected to a web part page:

- Links list items – these items link to resources such as pages, list items and documents
- File attachments – files attached to list items included in a list view which is displayed on the page.
- Documents – documents included in a view which is displayed on the page.

PARAMETERS

"*KWizComPrintFile.aspx*" receives the following parameters by query string (GET) or in the http request's header (POST):

Parameter	Required?	Description
PageUrl	No	Full url of a SharePoint page. The page can be a web part page, publishing page or Wiki page, on the same farm.
IncludeLists	No	This parameter is used to define what exactly should be printed along with the page. This parameter can be equal to one of the following values: 1. A collection of ListId/ViewId/CamlQuery - to print only specific list views that appear on the page, provide for each list its id and the requested view id. 1.1. If additional filtering was applied on a list view web part (by using query string or manually by end-user), then use the CamlQuery parameter to pass the exact CAML query that needs to be used to get the exact items for printing. 1.2. The query will be merged with the view existing query using "And" logic, unless you use "OverrideQuery=true" parameter, which will ignore the view query and replace it with the provided query. 1.3. ViewId and CamlQuery are optional, if omitted default view is used / no query is used. 1.4. This parameter may be long, so consider posting it rather than using query string. 1.5. The format of this parameter will be JSON representation of an object with a collection of list instances (using JSON.stringify). See example below. The API can print files, item properties, or both. By default only files are printed, but you can use "PrintProperties" and "PrintFiles" parameters to change that. If both are set to false, files will still be printed. For list items – the item attachments will be considered as files. 1.6. What will be printed? 1.6.1. For any list provided: any item in that list that matches the filter. 1.6.2. For Links lists only: for each link, the link target page/file will be printed. Only internal links are supported (links to items on the same SharePoint farm).

		1.6.3.For document libraries – any document in that library that matches the filter. 2. None (Default) – Print only the page, don't print any of the connected resources.
PrintingSettings	No	A JSON representation of a string key-value collection. This optional property allows developers to override print settings for this request. Such properties will be overrides for the entire print request and allows you to set a custom watermark, change page size, print quality or any other print setting. See documentation and complete list of properties below.

USAGE EXAMPLE

The following is a usage example using C# code. The API is available from client script as well as any other language that can make post requests.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using KWizCom.Utilities.Utility.Serialization;
using System.Net;

namespace KWizCom.SharePoint.iMUSH
{
    /// <summary>
    /// This is an example class with description on how to use the iMush Print API
    /// This example shows how to print any page in SharePoint
    /// Add a collection of lists, filter each list by a view and/or custom query,
    ///       and print files or item properties from each list
    /// You can also override (optionally) any print settings with custom values for this specific print job
    /// What is printed:
    ///       Document library - file, item properties, or both
    ///       Lists - attachments, item properties, or both
    ///       Links lists - if the link target is to a file inside SharePoint, the target file is printed.
    /// To specify printing options you will need to add a reference to KWizCom.Foundation.dll to get
    ///       SerializableDictionary class, or to provide your own JSON string in this format:
    ///       {"InfoPathViewNames":"","PageSize":"LETTER"}
    /// To specify included lists, copy the ListInfo class definition to your application,
    ///       or provide your own JSON string in this format:
    ///       [{"ListId":"","ViewId":"","CamlQuery":"","OverrideQuery":false,"PrintFile":true,"PrintProperties":true}]
    /// </summary>
    public class PrintPublicAPI
    {
        public void PrintSample()
        {
            //Printing settings are all optional. Settings are taken from iMush settings, unless overridden here.
            DisctionarySerializeHelper.SerializableDictionary<string, string>
                printingSettingsOverride = new DisctionarySerializeHelper.SerializableDictionary<string, string>();
            //Override infopath view names to use, delimited by ;
            printingSettingsOverride.Add("InfoPathViewNames", "");
            //Any page size from dropdown
            printingSettingsOverride.Add("PageSize", "LETTER");
            //send true to print in landscape
            printingSettingsOverride.Add("RotatePageToLandscape", "false");
            //send true to print html in hi resolution
```



```
            printingSettingsOverride.Add("HiResHtmlOutput", "false");
            //send false to print item properties as pdf text instead of image
            printingSettingsOverride.Add("PrintViewPropertiesAsHiFiImage", "true");
            //Specify different format string for headers
            printingSettingsOverride.Add("PrintHeader", "List: [List]*[Item:Title|Item: {0}]*[FileName|File name: {0}]");
            //Specify different format string for footers
            printingSettingsOverride.Add("PrintFooter",
                    "Printed on: [Now|MMMM dd, yyyy h:mmtt][Me| By: {0}]*[PageNum|{0} of {1}]");
            //Specify different format string for footers
            printingSettingsOverride.Add("PrintWatermark", "");
            //Send int > 5
            printingSettingsOverride.Add("TopMargin", "20");
            //Send int > 5
            printingSettingsOverride.Add("BottomMargin", "20");
            //Send int > 5
            printingSettingsOverride.Add("LeftMargin", "20");
            //Send int > 5
            printingSettingsOverride.Add("RightMargin", "20");
            //Any font name from iTextSharp.text.pdf.BaseFont
            printingSettingsOverride.Add("WatermarkFontName", "Courier");
            //Send 1<float<800
            printingSettingsOverride.Add("WatermarkFontSize", "40");
            //Send 0.1f<=float<=1
            printingSettingsOverride.Add("WatermarkFontOpacity", "0.4f");
            //Any color name from settings page dropdown
            printingSettingsOverride.Add("WatermarkFontColor", "BLACK");
            //Send true to render watermark horizontal and not diagonal
            printingSettingsOverride.Add("WatermarkLayoutHorizontal", "false");

            //Add a collection of lists
            List<ListInfo> includedLists = new List<ListInfo>();
            includedLists.Add(new ListInfo() { ListId = Guid.Empty.ToString(), ViewId = "",
CamlQuery = "<Where><Neq><FieldRef Name='ID'/><Value Type='Integer'>1</Value></Neq></Where>",
PrintFile = true, PrintProperties = true });

            byte[] pdf = GetPDF("http://server/", "http://server/sitepages/page.aspx", includedLists, printingSettingsOverride);

            if (pdf != null)
                System.IO.File.WriteAllBytes("c:\\temp\\print.pdf", pdf);

            return;
        }

        /// <summary>
        /// Get PDF file byte array
        /// </summary>
        /// <param name="WebUrl">Full URL of the web to work under</param>
        /// <param name="PageUrl">Page URL to print, this parameter accepts and prints a URL with query string</param>
        /// <param name="IncludedLists">Collection of ListInfo</param>
```



```
        /// <param name="PrintingSettings">Collection of print settings to override for this print request</param>
        /// <returns></returns>
        public byte[] GetPDF(string WebUrl, string PageUrl, List<ListInfo> IncludedLists,
KWizCom.Utilities.Utility.Serialization.DisctionarySerializeHelper.SerializableDictionary<string, string> PrintingSettings)
        {
            var jsn = new System.Web.Script.Serialization.JavaScriptSerializer();

            string printingSettingsSerialized = (PrintingSettings == null || PrintingSettings.Count < 1) ? "" :
jsn.Serialize(PrintingSettings);
            string includedListsSerialized = (IncludedLists == null || IncludedLists.Count < 1) ? "" : jsn.Serialize(IncludedLists);
            string actionPageUrl = WebUrl.TrimEnd('/') + "/_layouts/KWizCom_iMushFeature/KWizComPrintFile.aspx";


            string postData = "";
            if (!string.IsNullOrEmpty(PageUrl))
            {
                postData += "PageUrl=" + PageUrl + "&";
            }
            if (!string.IsNullOrEmpty(includedListsSerialized))
            {
                postData += "IncludedLists=" + includedListsSerialized + "&";
            }
            if (!string.IsNullOrEmpty(printingSettingsSerialized))
            {
                postData += "PrintingSettings=" + printingSettingsSerialized + "&";
            }

            postData = postData.TrimEnd('&');
            byte[] postDataByteArray = Encoding.UTF8.GetBytes(postData);


            var request = (HttpWebRequest)WebRequest.Create(actionPageUrl);
            request.Credentials = CredentialCache.DefaultCredentials;
            //request.UseDefaultCredentials = true;
            request.Method = "POST";
            request.ContentLength = postDataByteArray.Length;
            request.ContentType = "application/x-www-form-urlencoded";
            using (System.IO.Stream dataStream = request.GetRequestStream())
            {
                dataStream.Write(postDataByteArray, 0, postDataByteArray.Length);
            }

            byte[] result = null;
            using (var response = request.GetResponse())
            {
                using (var responseStream = response.GetResponseStream())
                {
                    using (var memoryStream = new System.IO.MemoryStream())
                    {
```



```
                    int bytesToReadEachTry = 4096;
                    int bytesRead = 0;
                    byte[] buffer = new byte[bytesToReadEachTry];
                    do
                    {
                        bytesRead = responseStream.Read(buffer, 0, bytesToReadEachTry);
                        memoryStream.Write(buffer, 0, bytesRead);

                    } while (bytesRead > 0);

                    result = memoryStream.ToArray();
                }
            }
        }
        return result;
    }

    [Serializable]
    public class ListInfo
    {
        /// <summary>
        /// GUID list id. List should be on the same web of file URL if provided, or context web if not provided.
        /// </summary>
        public string ListId = "";
        /// <summary>
        /// Optional view id. empty or invalid views will use default view.
        /// </summary>
        public string ViewId = "";
        /// <summary>
        /// Optional Where and OrderBy CAML to merge with the selected view's query.
        /// OrderBy - will replace the view's OrderBy. Where will be added to existing Where statement using And logic.
        /// </summary>
        public string CamlQuery = "";
        /// <summary>
        /// Send true to replace existing view Query, and not merge to it.
        /// </summary>
        public bool OverrideQuery = false;
        /// <summary>
        /// Print file in document libraries or attachments for listitems.
        /// If PrintProperties is false - this will be printed regardless to this value.
        /// </summary>
        public bool PrintFile = false;
        /// <summary>
        /// Print item properties page.
        /// </summary>
        public bool PrintProperties = false;
    }
  }
}
```

