Doug Rindfleisch

From:	Doug Rindfleisch
Sent:	Thursday, March 25, 2021 13:46
To:	Scott Kruse; Andriy Lysak; Chase Kruse; Doug Rindfleisch
Subject:	File Transfer Documentation: CoreFileUpload, CoreFileDownload, CoreFileImgShow, CoreFileCreateThumbnail, BackEndFileUpload

Updated 2021-03-25

If you add this line to the top of your JS file, all of this shows up in the intellisense in VS2017:

```
/// <reference path="../Core2/Core2.js" />
```

BE CAREFUL WITH OPTIONAL PARAMETERS! In most use cases, they are NOT needed.

JS CoreFileUpload

Function Definition:

```
/** @description CoreFileupload function - upload files to server
* @param {string} FormName FormName for CoreDataSave Security - Also (App Name) Folder name for save path (E:\Data\FormName\YYYY\YYYYMM\)
* @param {string} SecurityObject SecurityObject name for Security
* @param {string} ReturnFunction Returnfunction name - fired once when all file(s) are uploaded
* @param {int} MaxFiles Max number of files to allow during upload
* @param {string} FileTypes File types allowed, | delimited ("jpg|png|gif"), null or "*" for any file type

* @param {bool=} PrependUploadTime OPTIONAL 'true': prepend yyyyMMddHHmmssffffxxxx_ to filename(s) OR 'false': keep orig file name. Default is true
* @param {string=} OverrideFileName OPTIONAL Overrides given filename, enter file name WITHOUT extension, forces PrependUploadTime = "false" and Maxfiles = 1; Default is null
* @param {string=} ThumbHeight OPTIONAL int pixel height, create thumbnail image if > 0 - default is 0
*/

$.CoreFileUpload = function (FormName, SecurityObject, ReturnFunction, MaxFiles, FileTypes, PrependUploadTime, OverrideFileName, ThumbHeight)
```

Developers should record the FileID (bigint) and FileSize (int) in your application to download the file later.

Returns:

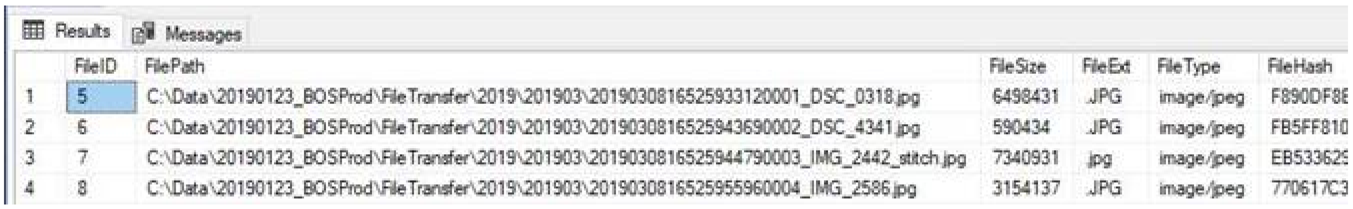
msg[0] :

```
FileExt: ".png"
FileID: "1771719"
FileName: "20210325094601244600001_2021-03-25_09-45-21.png"
FileSaved: "Success"
FileSize: "13216"
FileType: "image/png"
ThumbFileID: "1771720"
ThumbFileSize: "1408"
```

File Upload Example:

```
        $("#ExampleFileUpload").on("click", function () {
               $.CoreFileUpload(FN, "FileTransfer", "ExampleFileUploadReturn", 50, "xls|csv"); //
        additional parameters are optional after these needed ones.
        });

More Examples:
        $.CoreFileUpload(FN, "FileTransfer", "ExampleFileUploadReturn", 1000,
        "jpg|png|gif|mp4|xls|xlsx|csv", true, null);
        $.CoreFileUpload(FN, "FileTransfer", "ExampleFileUploadReturn", 1, "jpg", false,
        "TheBestFileName.jpg", 100);
        $.CoreFileUpload(FN, "FileTransfer", "ExampleFileUploadReturn", 50, "xls|csv", true, null);
```

File Upload checks if user is logged in and has access to FormName/SecurityObject. It records which form it was uploaded from along with file metadata:



	FileID	FilePath	FileSize	FileExt	FileType	FileHash
1	5	C:\Data\20190123_BOSProd\FileTransfer\2019\201903\20190308165259331200001_DSC_0318.jpg	6498431	.JPG	image/jpeg	F890DF8E
2	6	C:\Data\20190123_BOSProd\FileTransfer\2019\201903\20190308165259436900002_DSC_4341.jpg	590434	.JPG	image/jpeg	FB5FF810
3	7	C:\Data\20190123_BOSProd\FileTransfer\2019\201903\20190308165259447900003_IMG_2442_stitch.jpg	7340931	.jpg	image/jpeg	EB53362
4	8	C:\Data\20190123_BOSProd\FileTransfer\2019\201903\20190308165259559600004_IMG_2586.jpg	3154137	.JPG	image/jpeg	770617C3

Example with return function saving FileID's:

```
$("#AttachmentUploadBtn").on("click", function () {
    $.CoreFileUpload(FN, "CAEdit", "AttachmentUploadReturn", 1, "*");
});

function AttachmentUploadReturn(msg) {

    if (msg[0].FileSaved == "Success") {
        var Parameters = {
            "CAGoalID": $("#CAGoalID").text(),
            "FileID": msg[0].FileID,
            "FileSize": msg[0].FileSize
        };

        $.CoreDataSave(DB, FN, "Access", "AttachmentUploadSaveReturn",
"PlantGoals_AttachmentUploadSave", Parameters);
    }
    else
        alert("FileUpload Error: " + msg[0].FileSaved);
}


----------------------------------------------------------------------------------------------------------
---
JS CoreFileDownload

/** @description CoreDownload function - download a file
* @param {string} FormName FormName for CoreDataSave Security
* @param {string} FileID FileID is a unique identifier for the file(s), single Int or comma separated
list of FileID Integers
* @param {string} FileSize FileSize is the file size(s). It can be a single Int or comma separated list
of FileSize Integers

* @param {bool=} zipbit OPTIONAL zipbit boolean to zip a single file, multiple files are automatically
zipped. Default = false
```

```
* @param {bool=} KeepOrigFileName OPTIONAL KeepOrigFileName true: download the file with the filename
it was uploaded with. | false: download filename as written on server (with override filename or
prepended timestamp filename) | Default = true
* @param {string=} OverrideFileName OPTIONAL OverrideFileName string with a length > 0 will override
the downloaded file name and extension.  ex: "filename.ext"
*/
$.CoreFileDownload = function (FormName, FileID, FileSize, zipbit, KeepOrigFileName, OverrideFileName)
{
// FileSize is used as a 2nd form of authentication to stop malicious mass downloads
```

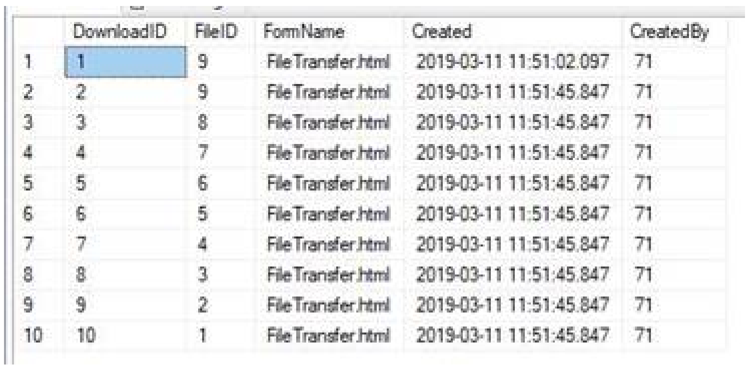
Examples:

```
        $.CoreFileDownload(FN, 5, 197834); //additional parameters are optional

        $.CoreFileDownload(FN, "5,6,7" , "189134,93884,76153" , true); // 3 files in a zip file

        $("#FileDownloadBtn").on("click", function () {  // write parameters based on selected files on
        screen
            var FileID = "",
                FileSize = "";

            $(".Selected").each(function () {
                FileID += this.id.split("_")[1] + ",";
                FileSize += this.id.split("_")[2] + ",";
            });

            $.CoreFileDownload(FN, FileID, FileSize, $("#ExampleZipCB").prop("checked"));
        });
```

File Download uses the UserKey to check if the user is logged in – this is the only BOS security check

The FileID and FileSize are used to look up the server download path.
This gives us the ability to upload in one form, and download in another.
I added the FileSize requirement for download to stop malicious user from iterating over all the FileIDs and downloading files they don't have access to.
Developers will store the file id and size with their detail data, so the file download will be covered under the umbrella of core security that controls detail selects

User downloads are recorded:



	DownloadID	FileID	FormName	Created	CreatedBy
1	1	9	FileTransfer.html	2019-03-11 11:51:02.097	71
2	2	9	FileTransfer.html	2019-03-11 11:51:45.847	71
3	3	8	FileTransfer.html	2019-03-11 11:51:45.847	71
4	4	7	FileTransfer.html	2019-03-11 11:51:45.847	71
5	5	6	FileTransfer.html	2019-03-11 11:51:45.847	71
6	6	5	FileTransfer.html	2019-03-11 11:51:45.847	71
7	7	4	FileTransfer.html	2019-03-11 11:51:45.847	71
8	8	3	FileTransfer.html	2019-03-11 11:51:45.847	71
9	9	2	FileTransfer.html	2019-03-11 11:51:45.847	71
10	10	1	FileTransfer.html	2019-03-11 11:51:45.847	71

SQL example of FileID and FileSize storage in the CPNDA app :

Save FileID and FileSize to your program's tables, then join on them to
FileTransfer.dbo.Files to get file info:

```
SELECT      NDA.RecordID,
            NDA.CPNDAID,
            NDA.FilePath,
            NDA.FileID,
            NDA.FileSize,
            NDA.Active,
            NDA.Created,
            U1.UserName AS CreatedBy,
            NDA.Updated,
            U2.UserName AS UpdatedBy,
            F.OrigFileName

FROM  CPNDA_Files NDA Left Join BOS2.dbo.tblSysUser U1
                        ON U1.UserID = NDA.CreatedBy
                  Left Join BOS2.dbo.tblSysUser U2
                        ON U2.UserID = NDA.UpdatedBy
                  Left Join FileTransfer.dbo.Files F
                        ON NDA.FileID = F.FileID AND NDA.FileSize = F.FileSize

Where @ID = NDA.CPNDAID AND NDA.Active = 1
```


JS CoreFileImgShow

Used to display images in the DOM instead of downloading them
Images are in data folder are not accessible to IIS directly for security reasons, so they are hosted in a temp location that is cleaned out on regular intervals only after being called by an authorized user.

```
/** @description CoreFileImgShow function - get the FileID and source path of an image for display in
browser.
* @param {string} FormName FormName to associate the download with a form for user activity tracking
* @param {string} FileID FileID is a unique identifier for the file, a single FileID BigInt
* @param {string} FileSize FileSize is the file size. It can be a single FileSize Integer
* @param {string} ReturnFunction Returnfunction name
* @return {object} Retruns msg.FileID and msg.src
*/
$.CoreFileImgShow = function (FormName, FileID, FileSize, ReturnFunction) {
```

Returns:

```
   msg[0] =

   FileExt: ".png"
   FileID: "1771719"
   src: "https://bosprod.coilcraft.com/Temp/404176204EF7BCABB21FBA2F899BD06E87DC536F/1771719.png"
```


JS CoreFileCreateThumbnail   (not used anywhere in BOSPROD production yet – should just be for migration from CARYDB08)

An independent thumbnail creator for files already uploaded (you’ll need the FileID and FileSize)

```
/** @description CoreFileCreateThumbnail function - create a thumbnail from an image.
* @param {string} FormName FormName to associate the download with a form for user activity tracking
* @param {string} SecurityObject SecurityObject name for Security
* @param {string} FileID FileID is a unique identifier for the file, a single FileID Integer
* @param {string} FileSize FileSize is the file size. It can be a single FileSize Integer
* @param {string} ThumbHeight ThumbHeight is the number of pixels
* @param {string} ReturnFunction Returnfunction name
* @return {object} Retruns msg.FileID and msg.src
*/
$.CoreFileCreateThumbnail = function (FormName, SecurityObject, FileID, FileSize, ThumbHeight,
ReturnFunction)
```


C# CoreFileUpload_201907BE
BACK END SERVER SIDE FILE UPLOAD (for files created on the fly)

An independent class in the App_Code folder: /App_code/Core2BackEndFileTransfer.cs

```
/** @description CoreFileUpload_201907BE function - Back End Fileupload,
* @param {string} FormName FormName - for CoreDataSave Security - Also (App Name) Folder name for save
path (E:\Data\FormName\YYYY\YYYYMM\)
* @param {string} FileName FileName - name for the file to be uploaded
* @param {byte[]} ByteArray ByteArray -  byte array of the created file
* @param {bool} PrependUploadTime PrependUploadTime - 'true' = prepend yyyyMMddHHmmssffffxxxx_ to
filename(s) OR 'false' = keep orig file name.
* @param {int} UserID UserID - id of the uploader / file creator
*/
public SerializableDictionary<string, string> CoreFileUpload_201907BE(string FormName, string FileName,
byte[] ByteArray, bool PrependUploadTime, int UserID)
```

EXAMPLE:

```
    public void BuildTestFile()
    {
        Document Doc = new Document(iTextSharp.text.PageSize.LETTER, 0, 0, 0, 0);
        MemoryStream MS = new MemoryStream();
        PdfWriter Writer = PdfWriter.GetInstance(Doc, MS);

        //open pdf to print dialog:
        PdfAction PrintAction = new PdfAction(PdfAction.PRINTDIALOG);
        Writer.SetOpenAction(PrintAction);

        //Writer.SetEncryption(PdfWriter.STANDARD_ENCRYPTION_128, "thisisasuperamazingpassword", null,
PdfWriter.AllowCopy);
        //Writer.SetPdfVersion(PdfWriter.PDF_VERSION_1_5);
        //Writer.CompressionLevel = PdfStream.BEST_COMPRESSION;

        Doc.Open();

        // ---------------------------- print a list of available Fonts on the server ----------------
-----------

        Doc.NewPage();
        string fontfolder = Environment.GetEnvironmentVariable("SystemRoot") + "\\Fonts";
        Doc.Add(new Paragraph(fontfolder));
        int totalfonts = FontFactory.RegisterDirectory(fontfolder, true);
        Doc.Add(new Paragraph("All " + totalfonts.ToString() + " Fonts:\n"));
        foreach (string fontname in FontFactory.RegisteredFonts)
        {
            Doc.Add(new Paragraph(fontname + "\n"));
        }
        // -------------------------------------------------------------------------------------
------------

        Doc.Close();

        byte[] pdfBytes = MS.ToArray();

        int UserID = 71; // userid of the the uploader/file creator

        //initialize the class from /App_code/Core2BackEndFileTransfer.cs:

        Core2BackEndFileTransfer FileTransfer = new Core2BackEndFileTransfer();

        SerializableDictionary<string, string> UploadedFile =
FileTransfer.CoreFileUpload_201907BE("SampleKitLabels.html", "ServerFontList.pdf", pdfBytes, true,
UserID);

        MS.Dispose();

        // UploadedFile is a dictionary with these strings in it:

        string  FileSaved = UploadedFile["FileSaved"],  // success/failure message
                FileSize = UploadedFile["FileSize"],    // file size
                FileType = UploadedFile["FileType"],    // mime file type
                FileExt = UploadedFile["FileExt"],      // file extension
                FileID = UploadedFile["FileID"],        // file id
                FileName = UploadedFile["FileName"];    // file name on server


        // -------------------------------------------------------------------------------------
--
        // return to browser (download file) if you need it:

        //HttpContext Context = HttpContext.Current;
        //Context.Response.ClearContent();
        //Context.Response.ClearHeaders();
        //Context.Response.ContentType = "application/pdf";
        //Context.Response.AddHeader("Content-Disposition", "attachment; filename=ServerFontList.pdf");
        //Context.Response.BinaryWrite(pdfBytes);
        //Context.ApplicationInstance.CompleteRequest();

    }
```

Doug Rindfleisch | Coilcraft Inc.
Systems Engineering Web Developer
Cell 847-691-3880
Office 847-516-7337

Support Cub Scouts! Buy Popcorn:
https://www.trails-end.com/store/scout/2OQY05S8?share=SPG1YF12