



Everest Software International®

# ONE ATTACHMENT MANAGER

AN EASY DOCUMENT MANAGEMENT INTERFACE FOR JDEDWARDS® ENTERPRISEONE®

## Table of Contents

Copyright And Disclaimer .....	3
Introduction: .....	4
One Attachment Manager API's .....	5
MOReader Object .....	5
MOWriter Object .....	6
Print Attachment Utility .....	8
AutoAttach Utility .....	9
Appendix .....	11
A sample VB Script To Create New FILE/URL Attachments .....	11
A sample VB Script to Open All IMAGE Attachments .....	12
A sample VB Script to Add a Text Attachment .....	14
DLL Interfaces .....	15

## Copyright And Disclaimer

This Guide is Copyright © 2005 Everest Software International Pty Ltd. All Rights Reserved.

### LIMIT OF LIABILITY / DISCLAIMER OF WARRANTY:

THE PUBLISHER AND AUTHOR HAVE USED THEIR BEST EFFORTS IN PREPARING THIS GUIDE. THE PUBLISHER AND AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THERE ARE NO WARRANTIES WHICH EXTEND BEYOND THE DESCRIPTIONS CONTAINED IN THIS PARAGRAPH. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES OR WRITTEN SALES MATERIALS. THE ACCURACY AND COMPLETENESS OF THE INFORMATION PROVIDED HEREIN AND THE OPINIONS STATED HEREIN ARE NOT GUARANTEED OR WARRANTED TO PRODUCE ANY PARTICULAR RESULTS, AND THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY INDIVIDUAL. NEITHER THE PUBLISHER NOR AUTHOR SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

## Introduction:

JDEdwards® EnterpriseOne® software offers powerful Document Management framework, somewhat restricted by the lack of easy interfaces.

To illustrate, let us consider the P0801 – “Work With Employee Information” program as an example:

This program is a standard user interface for the employee data, it's a part of the HR module. All company employees will have an entry in this program and it's a natural place to look for employees, change their details, etc.

Each employee may have a number of documents associated with them. These may be contracts, photographs, various documents and even copies of all communications with the employees.

This program is a perfect place to add these documents to the employees as attachments – since it's a natural starting point for an employee search. Once the record is located, no additional searches are required and all paperwork associated with the employee is now at hand.

The only issue, and we consider this a significant one, is that all this is only applicable to manual searches. For example, there is no standard way to print all or some of any given employee's attachments from a UBE, or to bulk-attach copious files in an automated manner.

The same, of course, applies to all the other standard programs, capable of handling attachments.

On the other hand, the many available third-party document management systems offer great flexibility in processing such documents, but will require a separate piece of software with a separate interface and a separate search to find all these documents.

One Attachment Manager seamlessly integrates into JDEdwards® EnterpriseOne®, making all attachments readily accessible from within native JDEdwards® EnterpriseOne® code, as well as from any external systems.

The core components of this solution are OAM\_READER.DLL and OAM\_WRITER.DLL, plus a collection of Windows® COM Automation Objects, which expose a set of methods for accessing JDEdwards® EnterpriseOne® attachments.

On top of these objects, we have also included a few complimentary auxiliary utilities, which can be used to illustrate the power of the solution, as well as to actually facilitate a number of attachment-related business tasks:

- A Windows® console (command-line) utility to print attachments;
- A set of native JDEdwards® EnterpriseOne® Business Functions to add/retrieve attachments directly from JDEdwards® EnterpriseOne® code.
- A few VBS scripts to add and read different types of attachments

# One Attachment Manager API's

## *MOR*Reader Object

As its name suggests, this Object offers an interface for reading Media Object Attachments.

Using VB to illustrate, this Object can be declared and created as follows:

```
DIM OMR as new IMORReader
```

Or

```
DIM OMR as Object  
Set OMR = Create('OneAttachmentManager.MORReader')
```

Once created, the following functions published by this Object can be used:

```
` Login and intialize:  
function LoginIntoEnterpriseOne(User: WideString; Password: WideString; Environment: WideString): HRESULT  
  
` An auxiliary function to build a Key, given a GT DSTR name and a populated DSTR by reference:  
function BuildKeyFromDSTR(DSTRName: WideString; lpDSTR: Pointer; out KEY: WideString): HRESULT  
  
` Execute the search, creating an in-memory list for the next function:  
function QueryByKey(OBNM: WideString; TXKY: WideString; LNGP: WideString; Count: integer): HRESULT  
  
` Retrieve the search results one by one, until this function returns a non-0 result:  
function GetNextAttacment(out MOTYPE: LongWord; out DATA: WideString): HRESULT  
`/////////////////////  
` MOTYPE value will be either:  
`/////////////////////  
` "0" for TEXT,  
` "1" - IMAGE,  
` "2" - OLE,  
` "3" - SHORTCUT,  
` "4" - VENDOR_IMAGE or  
` "5" - FILE/URL  
` ,  
` DATA will be either a full file name in UNC format for MOTYPE's of "1" and "5", or  
` a Unicode string with RTF data. The latter can be written into a new file with an extension  
` of "RTF", if necessary, for subsequent manipulations. These three types - 0, 1 and 5 are the only  
` types supported at present (for either Read or Write).  
  
` Clean-up and log-out:  
function Logout: HRESULT
```

All functions will return "0", if the respective calls were successful.

Once finished, the Objects should be freed, like so:

```
Set Object = Nothing
```

This software requires licensing for each installation, per Object, per Computer. The license key is linked to the target computer hardware configuration. An included Licensing Utility can be used to retrieve the software's serial number and to enter the License Key, received from Everest Software International.

If the Object was not properly licensed before, the calls to "LoginIntoEnterpriseOne" function will generate a licensing prompt, until a valid License Key has been supplied.

## ***MOWriter Object***

As its name suggests, this Object offers an interface for creating new Media Object Attachments.

Using VB to illustrate, this Object can be declared and created as follows:

```
DIM OMW as new IMOWriter
```

Or

```
DIM OMW as Object  
Set OMW = Create('OneAttachmentManager.MOWriter')
```

Once created, the following functions published by this Object can be used:

```
` Login and intialize:  
function LoginIntoEnterpriseOne(User: WideString; Password: WideString; Environment: WideString): HRESULT  
  
` An auxiliary function to build a Key, given a GT DSTR name and a populated DSTR by reference:  
function BuildKeyFromDSTR(DSTRName: WideString; lpDSTR: Pointer; out KEY: WideString): HRESULT  
  
` Create a new TEXT Attachment, using RTF:  
function AttachText(GTName: WideString; KEY: WideString; RTFString: WideString): HRESULT  
` Pass RTF text in a Unicode string as the last parameter  
  
` Create a new TEXT Attachment, using TXT:  
function AttachPlainText(GTName: WideString; KEY: WideString; TXTString: WideString): HRESULT  
` Pass TXT text in a Unicode string as the last parameter  
  
` Create a new TEXT Attachment, using Plain Text File:  
function AttachPlainTextFromFile(GTName: WideString; KEY: WideString; FileName: WideString): HRESULT  
` Pass File Name in a Unicode string as the last parameter  
  
` Create a new IMAGE Attachment:  
function AttachImage(GTName: WideString; KEY: WideString; FileName: WideString; const QueueName:  
WideString): HRESULT  
` Pass the short File Name and the target MO Queue Name in Unicode strings as the last two parameters.  
` The file will not be copied into the MO Queue by this function, it should already be there.  
  
` Create a new FILE/URL Attachment:  
function AttachFile(GTName: WideString; KEY: WideString; FileName: WideString; QueueName: WideString):  
HRESULT  
` Pass the short File Name and the target MO Queue Name in Unicode strings as the last two parameters.  
` The file will not be copied into the MO Queue by this function, it should already be there.  
  
` Copy FILE function:  
function CopyFile(FromName: WideString; ToName: WideString; bFailIfExists: BOOL): HRESULT  
` Pass two full File Names. The result will be non-0, if the last param is TRUE and  
` the operation was not successful.  
  
` Convert Plain Text into RTF function:  
function ConvertTextStringToRTFString(PlainText: WideString; BytesNeeded, BytesCopied: integer;  
RTFStringBack: WideString): HRESULT  
` Pass the plain text in a string, create BytesNeeded-sized storage for RTFStringBack and  
` call it again to convert the text into RTF.  
  
` Retrieve the full path to any given MO Queue function:  
function GetQueuePath(QueueName: WideString; BytesNeeded, BytesCopied: integer; DirNameBack: WideString):  
HRESULT  
` Pass the MO Queue Name in and read the full UNC path to the corresponding folder.  
  
` Clean-up and log-out:  
function Logout: HRESULT
```

All functions will return "0", if the respective calls were successful.

Once finished, the Objects should be freed, like so:

```
Set Object = Nothing
```

This software requires licensing for each installation, per Object, per Computer. The license key is linked to the target computer hardware configuration. An included Licensing Utility can be used to retrieve the software's serial number and to enter the License Key, received from Everest Software International.

If the Object was not properly licensed before, the calls to "LoginIntoEnterpriseOne" function will generate a licensing prompt, until a valid License Key has been supplied.

## Print Attachment Utility

This handy Windows® console utility will print any required attachments to the specified printer from a command-line or a script.

Syntax:

```
PrintAttachment <USER> <PASSWORD> <ENVIRONMENT> <GT_DSTR_NAME> <KEY> <MO_TYPE> <PRINTER>
```

Where:

- The first three parameters are the JDEdwards® EnterpriseOne® **login details** to be used;
- <GT\_DSTR\_NAME> is the name of the GT data structure, used to identify the Attachments. I.e.: for the P0801 program, this will be “GT0801”;
- <KEY> is a string of all GT DSTR values concatenated with “|” characters in between the values. In the simplest case of GT0801 DSTR, this will be the A/B Number of the Employee, i.e.: “6001”;
- <MO\_TYPE> is the type of the attachment(s) to be printed – “0” – for TEXT, “1” – for IMAGE or “5” – for any other miscellaneous file type, like PDF, HTM, DOC, etc.;
- <PRINTER> is the UNC name of the target printer, i.e.:  
\\PRINTSERVER\PRINTER\_SHARE\_NAME

Please note, that no other types of attachments are supported.

This utility sets the Windows® %ERRORLEVEL% Environment Variable on exit. If this value is “0” when the tool finishes, then it was successful. If the value was different, there were errors processing your request.

Running this tool without any parameters will print a help screen.

# AutoAttach Utility

This very powerful Windows® console utility will bulk-attach Image, File or Text (TXT or RTF) attachments to the respective JDEdwards® EnterpriseOne® record, using the source file names as sources of the MO Key data.

Running the tool without any parameters will produce this help screen:



```
AutoAttach - creating OW/EO MO Attachments from external files.
-----
Copyright (C) 2005 Everest Software International(TM). All Rights Reserved.
-----

!!! Not enough, or incorrect parameters !!!

Usage:
=====

AutoAttach <<switch> <parameter>>, where <switches> and <parameters> are:

-U <OW_SIGNON_USER_NAME>
-P <OW_SIGNON_PASSWORD>
-E <OW_SIGNON_ENVIRONMENT>
-G <OW_MO_DSTIR_NAME>
-T <ATTACHMENT_TYPE> <IMAGE, FILE, TXT or RTP>
-F <FROM_DIRECTORY>
-Q <MO_QUEUE_NAME> <for FILE or IMAGE - TEXT does not require this parameter>
-C <Y/N> - should we copy the file into the MO Queue?
-CC <TARGET_DIRECTORY_NAME> - carbon-copy the original file
-D <Y/N> - should we delete the original file after the attachment is created?
-NF <FILE_NAME_FORMAT> - how to extract the key elements from the file name
-KF <KEY_FORMAT> - how to use the key elements to build the MO Key

Switches can be prefixed with "-" or "/". Parameters can follow the switches
after any number of space[s], or immediately (without a delimiter). There
must be space between any parameter and the next switch. Any parameters with
spaces must be enclosed in ""'s. Every switch must have a corresponding
parameter. Directories must exist and be accessible. The number of keys used
in the format parameters must match.

Key Format:
Key elements are denoted by keywords "\KEY#", i.e.: "\KEY1", "\KEY2" and so on (no
quotes, no spaces, leading back-slash).
For example, a 3-part key will have a format string of "\KEY1!\KEY2!\KEY3" - please,
note the separating "!" characters, used in actual MO Keys.
The actual values for the "\KEY#" keywords will be determined at run-time from the
file names, used to create the attachments, using the File Name Format for parsing.

File Name and Key Formats are very similar, with the exception of File Names, possibly,
containing non-key parts, which are described differently, plus the field type/length
specifications:
The file name can contain literal text (fixed, or dynamic) and key elements (again fixed,
or dynamic) in any order.
Literal text is used in the format "as is", the key elements are denoted with "\KEY#=",
followed by a type/length specification, while any other text is denoted with "\TEXT=",
followed by a type/length specification.
The type/length specification uses a "#" character to denote any numeric fields and "@" -
to denote any alpha fields, followed by an optional number, representing the field length
for fixed-length fields, so that "#5" is interpreted as a "5-digit numeric field", "@10" - as
a "10-character alpha-numeric string", while a single "#" - any number of sequential
digits and "@" - any number of sequential alpha (not special, alpha-numeric or mixed)
characters.
For example, "HR \KEY3=#\KEY1=@ - \KEY2=@10 (\TEXT=@10).JPG" File Name Format contains:
- Literal "HR " - 3 chars, including 1 space, followed without any more spaces by
- KEY3 element, comprised of digits and followed without spaces by
- KEY2 element, comprised of alpha chars and followed without spaces by
- literal " - "
- KEY2 - fixed length 10-char alpha (or alpha-numeric) field
- literal " <"
- 10-char long non-key text
- literal ">"
- extension ".JPG"

The Key Format must use all of the KEY# elements from the File Name Format and no more,
in any order, separated by "!"-characters, i.e.: "\KEY1!\KEY2!\KEY3"
```

This utility sets the Windows® %ERRORLEVEL% Environment Variable on exit. If this value is “0” when the tool finishes, then it was successful. If the value was different, there were errors processing your request.

Typically, all files to be attached will be dumped into a pre-defined folder.

The files can be manually created, or automatically generated (i.e.: by Create!Print, or similar software).

These files should all be named in a uniform fashion, with the correct identifiers in the names, i.e. if the files are to be attached to the Address Book entries (using DSTR “ABGT” with 1 Key Element of the AB Number), then the file names must contain the target AB Numbers.

When AutoAttach is subsequently executed (i.e.: by Windows Scheduler), it will process all files in this folder, interpreting the names to extract the MO Key data and attach the files to the correct AB entries.

The files can also optionally be copied into the specified MO Queue and deleted afterwards, if the attachment and the copy were both successful.

If the MO DSTR contains more elements than one, then the file names must contain all of the required elements.

This tool accepts two format strings: one for the file names, telling it which parts of the file names correspond to which GT DSTR elements and another one – to tell it how these key elements are to be combined together to construct the MO Key to be used for the given attachment.

# Appendix

## A sample VB Script To Create New FILE/URL Attachments

```
*****
'*
'* File:          AddAttachment.Vbs
'* Created:       April 2004
'* Version:       1.0
'*
'* Main Function: Add a new EnterpriseOne FILE/URL MO Attachment.
'*
'* AddAttachment.vbs <username> <password> <environment> <GT DSTR Name> <Key> <MO Queue Name> <File Name>
'*
'* Copyright (C) 2005 Everest Software International. All Right Reserved.
'*
*****

ON ERROR RESUME NEXT

DIM MOW

DIM Usr
DIM Pwd
DIM Env
DIM Dsn
DIM Key
DIM Que
DIM Fil

If Wscript.Arguments.Count <> 7 Then
    Wscript.Echo "Wrong number of parameters." & vbCRLF & _
        "" & vbCRLF & _
        "Usage:" & vbCRLF & _
        "===== & vbCRLF & _
        "AddAttachment.vbs <username> <password> <environment> <GT DSTR Name>" & vbCRLF & _
        "          <Key> <MO Queue Name> <File Name>" & vbCRLF & _
        ""
Else
    Usr = Wscript.arguments.Item(0)
    Pwd = Wscript.arguments.Item(1)
    Env = Wscript.arguments.Item(2)
    Dsn = Wscript.arguments.Item(3)
    Key = Wscript.arguments.Item(4)
    Que = Wscript.arguments.Item(5)
    Fil = Wscript.arguments.Item(6)

    Set MOW = CreateObject("OneAttachmentManager.MOWriter")

    Select Case MOW.LoginIntoEnterpriseOne(Usr, Pwd, Env)
        Case 0
            Wscript.Echo "Logged In"

            if MOW.AttachFile(Dsn, Key, Fil, Que) = 0 then
                Wscript.Echo "SUCCESS!"
            Else
                Wscript.Echo "Failed to Attach File"
            End If

            MOW.Logout
        Case 100
            Wscript.Echo "Licensing Failed"
        Case 1
            Wscript.Echo "Login Failed"
        Case 2
            Wscript.Echo "Login Crashed"
    End Select
End If

Set MOW = Nothing
```

## A sample VB Script to Open All IMAGE Attachments

```
*****
'*
'* File:          OpenAllImageAttachments.vbs
'* Updated:      September 2005
'* Version:      1.3
'*
'* Main Function: Add a new EnterpriseOne FILE/URL MO Attachment.
'*
'* Syntax:
'* cscript.exe OpenAttachment.vbs <username> <password> <environment> <GT DSTR Name> <Key>
'*
'* Copyright (C) 2005 Everest Software International. All Right Reserved.
'*
*****

ON ERROR RESUME NEXT

DIM MOR
dim objShell

DIM Usr
DIM Pwd
DIM Env
DIM Dsn
DIM Key
DIM Fil
DIM Typ
DIM Cnt
DIM Res
DIM Lang
DIM i

If Wscript.Arguments.Count <> 5 Then
    Wscript.Echo "Wrong number of parameters." & vbCRLF & _
        "" & vbCRLF & _
        "Usage:" & vbCRLF & _
        "===== " & vbCRLF & _
        "cscript.exe OpenAllImageAttachments.vbs <username> <password> <environment> <GT DSTR Name> <Key>" &
vbCRLF & _
        ""
Else
    Usr = Wscript.arguments.Item(0)
    Pwd = Wscript.arguments.Item(1)
    Env = Wscript.arguments.Item(2)
    Dsn = Wscript.arguments.Item(3)
    Key = Wscript.arguments.Item(4)

    Set MOR = CreateObject("OAM_OLEWRAPPER.MOReader")

    Select Case MOR.LoginIntoEnterpriseOne(Usr, Pwd, Env)
        Case 0
            Wscript.Echo "Logged In"

            ' Default Language:
            Lang = " "

            Res = MOR.QueryByKey(Dsn, Key, Lang, Cnt)
            Select Case Res
                Case 0
                    if Cnt = 0 then
                        Wscript.Echo "No Attachments Found"
                    Else
                        Wscript.Echo "Found " & Cnt & " Attachments"
                    End If
                Case 1
                    Wscript.Echo "Open Table Failed"
                Case 2
                    Wscript.Echo "Not Connected"
            End Select

            For i = 1 to Cnt
                Call MOR.GetNextAttacment(Typ, Fil)

                Select Case Typ
                    Case 1
                        Wscript.Echo "Found: " & Fil & ", Opening..."
                        set objShell = CreateObject("Shell.Application")
                        objShell.ShellExecute Fil, "", "", "open", 1
                        set objShell = nothing
                    Case 0
                        Wscript.Echo "Found: TEXT - wrong type, skipping."
                    Case 5
                        Wscript.Echo "Found: FILE/URL - wrong type, skipping."
                End Select
            Next

            MOR.Logout
        Case 100
```

```
        Wscript.Echo "Licensing Failed"
    Case 1
        Wscript.Echo "Login Failed"
    Case 2
        Wscript.Echo "Login Crashed"
End Select
End If

Set MOW = Nothing
```

## A sample VB Script to Add a Text Attachment

```
*****
'*
'* File:          AttachText.vbs
'* Created:       September 2005
'* Version:       1.0
'*
'* Main Function: Add a new EnterpriseOne TEXT MO Attachment to "GT0005" (UDC).
'*               This GT DSTR requires 3 elements in a KEY. Because we cannot
'*               use a "|" character in a string in a command line, we supply
'*               the three elements as separate parameters, i.e.: 98 JQ QBATCH.
'*               This will be converted into a "98|JQ|QBATCH" in the code below.
'*
'* Syntax:
'* cscript.exe AddAttachment.vbs <username> <password> <environment> <GT DSTR Name> <Key1> <Key2> <Key3> <Text>
'*
'* Copyright (C) 2005 Everest Software International. All Right Reserved.
'*
*****

ON ERROR RESUME NEXT

DIM MOW

DIM Usr
DIM Pwd
DIM Env
DIM Dsn
DIM Key
DIM Txt
DIM Num

If Wscript.Arguments.Count <> 8 Then
    Wscript.Echo "Wrong number of parameters." & vbCRLF & _
        " " & vbCRLF & _
        "Usage:" & vbCRLF & _
        "===== " & vbCRLF & _
        "cscript.exe AddAttachment.vbs <username> <password> <environment> <GT DSTR Name>" & vbCRLF & _
        " <Key1> <Key2> <Key3> <Text>" & vbCRLF & _
        " "
Else
    Usr = Wscript.arguments.Item(0)
    Pwd = Wscript.arguments.Item(1)
    Env = Wscript.arguments.Item(2)
    Dsn = Wscript.arguments.Item(3)
    Key = Wscript.arguments.Item(4)+"|"+Wscript.arguments.Item(5)+"|"+Wscript.arguments.Item(6)
    Txt = Wscript.arguments.Item(7)

    Set MOW = CreateObject("OAM_OLEWRAPPER.MOWriter")

    Num = MOW.LoginIntoEnterpriseOne(Usr, Pwd, Env)

    Select Case Num
        Case 0
            Wscript.Echo "Logged In"

            ' "AttachPlainText" method converts plain text to RTF first and then attaches it:
            if MOW.AttachPlainText(Dsn, Key, Txt) = 0 then
                Wscript.Echo "SUCCESS!"
            Else
                Wscript.Echo "Failed to Attach Text"
            End If

            MOW.Logout
        Case 100
            Wscript.Echo "Licensing Failed"
        Case 1
            Wscript.Echo "Login Failed"
        Case 2
            Wscript.Echo "Login Crashed"
        Case 99
            Wscript.Echo "Login Crashed"
        Case 101
            Wscript.Echo "Failed To Load DLL"
        Case 102
            Wscript.Echo "Failed To Get Procedure Address(es)"
    End Select
End If

Set MOW = Nothing
```

## DLL Interfaces

Once you have installed the native EnterpriseOne® functions from the included Boomerang Package, you can use the .H and .C files to see exactly how these calls are implemented:

```
// Use this function from an already-connected EnterpriseOne® session:
typedef int (__stdcall *ExternalAttachFunction) (HUSER U);
// Because once you are in EnterpriseOne®, there's no need to login again.
// This DLL also offers a usual Login function, which accepts
// User/Password/Environment for use in external C applications.
// For external applications in VB, the OLE Wrapper installed with
// OAM offers a superior interface.

// Call once, when finished:
typedef int (__stdcall *ExternalLogoutFunction) ();

// Call first, after logging in:
typedef int (__stdcall *ExternalReaderQueryByKeyFunction) (char *lpDSTRName, char *lpKey, char *lpLang, ID
*lpCount);

// Call in a loop to retrieve attachments, until it returns error:
typedef int (__stdcall *ExternalReaderGetNextAttachmentFunction) (_MOType_ *eMOType, char *lpData);

// Attach ready RTF text:
typedef int (__stdcall *ExternalWriterAttachText) (char *lpDSTRName, char *lpKey, char *lpRTFText);

// Attach Plain Text (it's transparently converted to RTF):
typedef int (__stdcall *ExternalWriterAttachPlainText) (char *lpDSTRName, char *lpKey, char *lpPlainText);

// Attach Plain (ANSI) Text from a file:
typedef int (__stdcall *ExternalWriterAttachPlainTextFromFile) (char *lpDSTRName, char *lpKey, char
*lpFileName);

// Attach IMAGE:
typedef int (__stdcall *ExternalWriterAttachImage) (char *lpDSTRName, char *lpKey, char *lpFileName, char
*lpMOQueueName);

// Attach File/URL:
typedef int (__stdcall *ExternalWriterAttachFile) (char *lpDSTRName, char *lpKey, char *lpFileName, char
*lpMOQueueName);

// Copy File - no "Attach" functions copy the files into the MO Queue directory.
// They assume the file is already there and accept a short file name as a
// parameter - without the path part, because the path is implied by the MO Queue:
typedef int (__stdcall *ExternalWriterCopyFile) (char *szFullFilePathFrom, char *szFullPathTo, BOOL
bFailIfExists);
```

These native BSFN's can be used anywhere in ER or C code throughout your system, i.e.: you will be able to create UBE's and APPL's to read and/or write attachments.