



Coveo Enterprise Search 6.0

Open Converter API

Contents

Overview	4
Open Converters	4
Conversion Scripts	4
Conversion Phases	4
Open Converters	6
How to Create an Open Converter	6
How to Setup an Open Converter	6
Open Converter Samples	7
Sample 1 – Modify an HTML Document (JScript Version)	7
(VBScript Version)	8
Sample 2 – Convert Document to RTF Format (JScript Version)	9
(VBScript Version)	10
Sample 3 – Convert Document to Microsoft Excel Format (JScript Version)	11
(VBScript)	12
Sample 4 – Convert Document to PDF Format (JScript Version)	13
(VBScript Script)	14
Sample 5 – Convert Document Using IFilterWrapper (JScript Version)	15
(VBScript Version)	16
Sample 6 – Convert Document to HTML Format (JScript Version)	17
(VBScript Version)	18
Conversion Scripts	19
Preconversion Scripts	20
How to Create a Preconversion Script	20
How to Setup a Preconversion Script	20
Preconversion Script Samples	20
Sample 1 – Add New Document Metadata (JScript Version)	20
(VBScript Version)	21
Sample 2 – Update Printable and Clickable URI (JScript Version)	22
(VBScript Version)	22
Sample 3 – Filter Based on Size and Date (JScript Version)	22
(VBScript Version)	23
Sample 4 – Use a Conversion Script Parameter (JScript Version)	23
(VBScript Version)	24
Sample 5 – Use a Conversion Script Global Variable (JScript Version)	24
(VBScript Version)	25
Sample 6 – Use CES Administration API (JScript Version)	26
(VBScript Version)	27
Postconversion Scripts	28

How to Create a Postconversion Script	28
How to Setup a Postconversion Script	28
Postconversion Script Samples	29
Sample 1 – Add New Metadata Named AllFieldValues (JScript Version)	29
(VBScript Version)	29
Sample 2 – Update Document Concepts and Summary Sentences (JScript Version)	30
(VBScript Version)	31
Sample 3 – Set a Custom Document Weight (JScript Version)	32
(VBScript Version)	33
Sample 4 – Change View as HTML Version of Document (JScript Version)	34
(VBScript Version)	34
Sample 5 – Analyze Text to Find Metadata (JScript Version)	35
(VBScript Version)	35
Sample 6 – Send Email to Specify Indexing (JScript Version)	36
(VBScript Version)	38
Sample 7 – Update Document Permissions (JScript Version)	39
(VBScript Version)	40

Overview

Coveo Enterprise Search (CES) converters are responsible for extracting content and metadata from the documents gathered by the CES connectors. Once a document is converted, it can be added to the index to make it available for user queries. While CES comes with a fully functional conversion process, it can easily be enhanced and tweaked using open converters, preconversion scripts and postconversion scripts.

Open Converters

CES native converters support the document types listed in [Supported File Formats](#). [Open Converters](#) are VBScripts or JScripts that convert a document from an unsupported format to a format for which CES has a native converter. This functionality enables CES to index documents from any format.

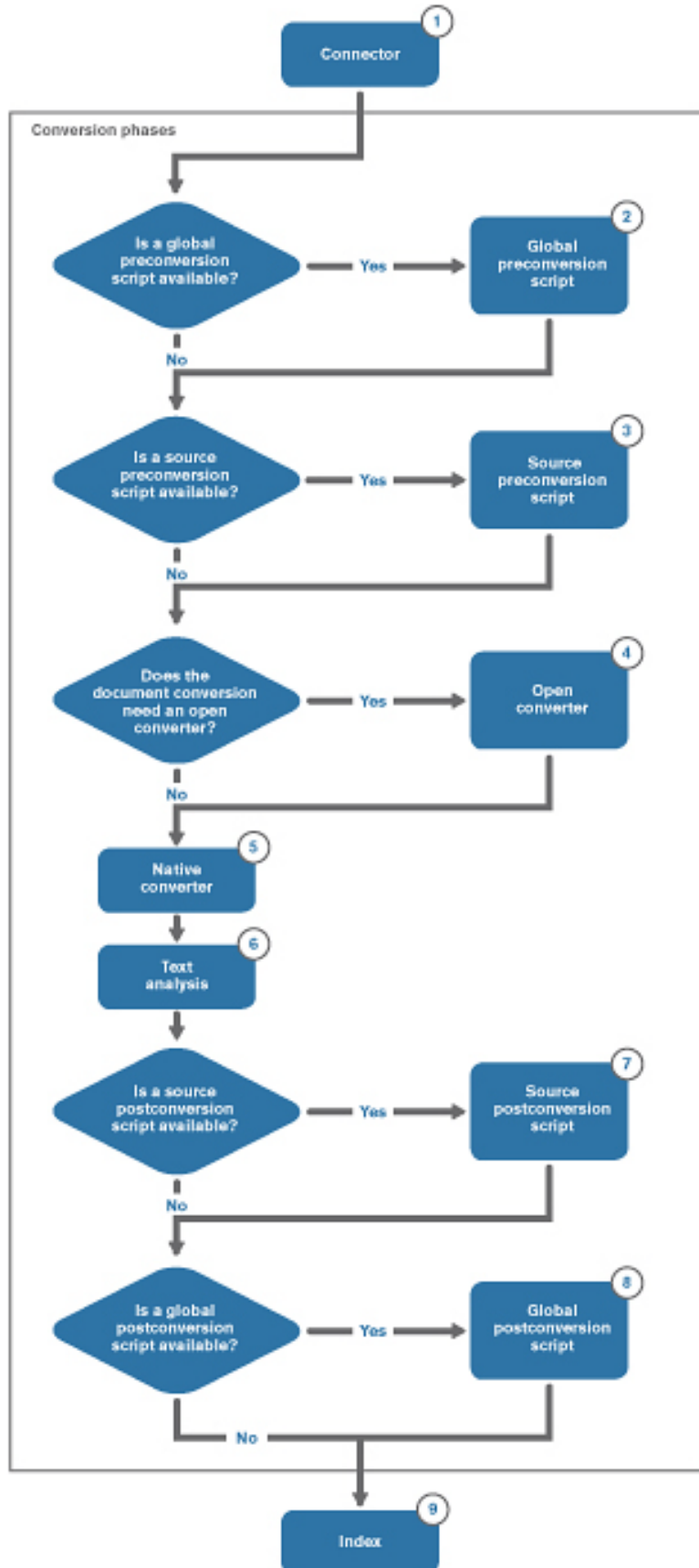
Conversion Scripts

[Conversion Scripts](#) are VBScripts or JScripts used to manipulate document content and metadata before (preconversion) or after (postconversion) conversion. For example, conversion scripts can be used to update the extracted key concepts of documents. Two types of conversion scripts exist: global and source scripts. [Global scripts](#) apply to all the sources indexed; whereas, [source script](#) only apply to a single source. Only one global preconversion and postconversion script can be applied to the index at a time; moreover, only one source preconversion and postconversion script can be applied per source.

Conversion Phases

The following lists the different phases of the conversion process:

1. **Crawling:** In this phase, the document will be fetched using a connector (file, Exchange, SharePoint, etc.). Some metadata will already be made available by the connector, including the URI, modification date and much more.
2. **Global Preconversion script (optional phase):** If a global preconversion script has been set, this script will be executed in this phase. Common script tasks: Read document content, read and modify metadata, update document permissions, etc.
3. **Source Preconversion script (optional phase):** If a preconversion script has been set on the document's source, this script will be executed in this phase. Common script tasks: Read document content, read and modify metadata, update document permissions, etc.
4. **Open Converter (optional phase):** If the Document Type has been set to convert the document using an open converter, this script will be executed in this phase. Common script tasks: Convert the document from an unsupported format to a CES native format, read and modify metadata, update document permissions, etc.
5. **Native Converter:** In this phase, the document content and properties will be extracted using one of the native CES converters.
6. **Document Text Analysis:** Many proprietary analysis technologies will be used to generate a document's summary and key concepts, perform a language and title detection, and much more.
7. **Source Postconversion script (optional phase):** If a postconversion script has been set on the document's source, this script will be executed in this phase. Common script tasks: Update the document extracted text and HTML, read and modify metadata, update document permissions, etc.
8. **Global Postconversion script (optional phase):** If a global postconversion script has been set, this script will be executed in this phase. Common script tasks: Update the document extracted text and HTML, read and modify metadata, update document permissions, etc.
9. **Index the document.** This phase will put the document extracted content and properties into the CES index to make it available for user queries.



Open Converters

Open converters are VBScripts or JScripts that convert a document from an unsupported format to a format for which CES has a native converter. The main purpose of the open converter functionality is to let CES index documents from any format.

A good example would be an administrator that needs to index a document from a XYZ company. Because CES has no native converter to extract the content and properties of the .xyz format, the administrator can create an open converter to convert the document from the .xyz format to the HTML format (or any other native format, such as Word, PDF, ZIP, etc); therefore, CES can index the content and properties of the .xyz document.

In addition to converting documents from an unsupported format to a native format, open converters can also:

- modify the document's metadata in order to tweak the data to index;
- update the document's permissions;
- reject the document if corrupted or deemed necessary;
- convert a document from a native format to another native format. For example, an administrator that wants to remove unwanted HTML content from a Web page document can create an open converter in order to analyze the documents to index and remove unwanted content from it before the document is converted.

How to Create an Open Converter

Creating an open converter script implies creating a new script file (which is a basic text file) and adding code. This file can be created anywhere on the server; however the common location is in the CES index path (ex: `C:\CES5\Scripts\`). It is possible to create a new script file from scratch or copy and paste an open converter script sample and modify it if necessary.

Two supported scripting languages are enabled in order to create an open converter: [VBScript language](#) (.vbs file) and [JScript language](#) (.js file). Note that the script file extension does not have to match its scripting language type.

To build an open converter script, it is possible to use the following scripting objects:

- CustomConversion Object (ICustomConversion Interface)
- DocumentInfo Object (IDocumentInfo Interface)
- Mutex Object (ICGLMutex Interface)
- IFilterWrapper Object (ICGLFilterWrapper Interface)

How to Setup an Open Converter

Once the open converter is created, perform the following steps in order to use it:

- Add the open converter to the CES configuration (refer to [How to Add an Open Converter](#)).
- Assign the open converter to the desired document type. By doing so, CES uses the open converter to convert every document that belongs to this document type. For more information on how to change the converter associated to a specific document type, refer to [How to Add and Modify Document Types](#).

Open Converter Samples

The following lists the different open converter samples available:

Sample 1 – Modify an HTML Document (JScript Version)

```
// *****
// This open converter sample explain is how to modify an HTML document.
// It removes the HTML content included between specific HTML tags.
// *****
// HTML tag name to exclude.
var START_TAG = "<moindex>";
var STOP_TAG = "<noindex>";

//The HTML charset.
var CHARSET_CP1252 = 1

try {
    // Extract the document HTML content.
    var documentContent =
CustomConversion.InputDocument.ReadByteString(CustomConversion.InputDocument.BytesCount,
CHARSET_CP1252);

    // Remove unwanted sections from the HTML content.
    documentContent = CleanHTML(documentContent, START_TAG, STOP_TAG);

    // Set the new document content.
    CustomConversion.OutputDocument.WriteAsByteString(documentContent, CHARSET_CP1252);
} catch (e) {
    // Unable to modify the document. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.description, 2);
}

// This function removes unwanted sections from the HTML content.
function CleanHTML(htmlToClean, startTag, stopTag)
{
    // Find the beginning of the section to remove.
    var startIdx = htmlToClean.indexOf(startTag);

    // While there are sections to remove.
    while (startIdx != -1) {
        // Find the end of the section and remove it from the HTML content.
        var stopIdx = htmlToClean.indexOf(stopTag, startIdx);
        htmlToClean = htmlToClean.substring(0, startIdx) + htmlToClean.substring(stopIdx +
stopTag.length, htmlToClean.length);
        startIdx = htmlToClean.indexOf(startTag, startIdx);
    }

    return htmlToClean;
}
}
```

(VBScript Version)

```

'*****
' This open converter sample explains how to modify an HTML document.
' It removes the HTML content included between specific HTML tags.
'*****

Option Explicit

' HTML tag name to exclude.
Const START_TAG = "<noindex>"
Const STOP_TAG = "</noindex>"

' The HTML charset.
Const CHARSET_CP1252 = 1

' Extract the document HTML content.
Dim documentContent
documentContent =
CustomConversion.InputDocument.ReadByteString(CustomConversion.InputDocument.BytesCount,
CHARSET_CP1252

' Remove unwanted sections from the HTML content.
documentContent = CleanHTML(documentContent, START_TAG, STOP_TAG)

' Set the new document content.
Call CustomConversion.OutputDocument.WriteAsByteString(documentContent, CHARSET_CP1252)

If Err.Number <> 0 Then
  ' Unable to modify the document. Reject the document and report the error.
  DocumentInfo.IsValid = False
  Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' This function removes unwanted sections from the HTML content.
Function CleanHTML(htmlToClean, startTag, stopTag)
  ' Find the beginning of the section to remove.
  Dim startIdx: startIdx = InStr(htmlToClean, startTag)

  'While there are sections to remove.
  Do While (startIdx > 0)
    ' Find the end of the section and remove it from the HTML content.
    Dim stopIdx: stopIdx = InStr(startIdx, htmlToClean, stopTag)
    htmlToClean = Left(htmlToClean, startIdx - 1) & Right(htmlToClean, Len(htmlToClean)
- (stopIdx + Len(stopTag)) + 1)
    startIdx = InStr(htmlToClean, startTag)
  Loop

  CleanHTML = htmlToClean
End Function

```

Sample 2 – Convert Document to RTF Format (JScript Version)

```
// *****
// This open converter sample converts a document from an unsupported format
// (as Windows Write documents) to the RFT format using the Microsoft Word COM
// objects.
// *****

// Save the documents to convert to a temporary file.
var inputFileName = CustomConversion.InputDocument.TemporaryFileName();
var outputFileName = CustomConversion.OutputDocument.TemporaryFileName();
CustomConversion.InputDocument.SaveToFile(inputFileName);

// Lock here to instantiate the Microsoft Word COM objects one at a time.
Mutex.AcquireLock("MicrosoftWordMutex");

try {
    // Open the document using Microsoft Word COM objects.
    var wordApplication = new ActiveXObject("Word.Application");
    var wordDocuments = wordApplication.Documents;
    var wordDocument = wordDocuments.Open(inputFileName, false);

    // Save the document as a RTF document.
    wordDocument.SaveAs(outputFileName, 6); // wFormatRTF
    wordDocument.Close();
    wordApplication.Quit(0); // Do not save changes

    // The document format was successfully changed to RTF. Convert this document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileName);
} catch (e) {
    // Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.description, 2);
}

// Release our mutex in order for other scripts to be able to use the Microsoft Word COM
objects.
Mutex.ReleaseLock();

// Clear our temporary files.
try {
    CustomConversion.DeleteFile(inputFileName);
} catch (e) {
}
try {
    CustomConversion.DeleteFile(outputFileName);
} catch (e) {
}
```

(VBScript Version)

```

*****
' This open converter sample converts a document from an unsupported format
' (such as Windows Write documents) to the RTF format using the Microsoft
' Word COM objects.
*****

Option Explicit
On Error Resume Next

' Save the document to convert to a temporary file.
Dim inputFileNames: inputFileNames = CustomConversion.InputDocument.TemporaryFileName
CustomConversion.InputDocument.SaveToFile(inputFileNames)

' Lock here to instantiate the Microsoft Word COM objects one at a time.
Call Mutex.AcquireLock("MicrosoftWordMutex")

' Open the document using Microsoft Word COM objects.
Dim wordApplication: Set wordApplication = CreateObject("Word.Application")
Dim wordDocuments: Set wordDocuments = wordApplication.Documents
Dim wordDocument: Set wordDocument = wordDocuments.Open(inputFileNames, False)

' Save the document as a RTF document.
Dim outputFileName.SaveAs(outputFileName, 6) ' wdFormatRTF
Call wordDocument.Close()
Call wordApplication.Quit(0) ' Do not save changes

' Release our mutex in order for other scripts to be able to use the Microsoft Word COM
objects.
Call Mutex.ReleaseLock()

If Err.Number = 0 Then
    ' The document format was successfully changed to RTF. Convert this document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileName)
Else
    ' Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' Clear our temporary files.
Call CustomConversion.DeleteFile(inputFileNames)
Call CustomConversion.DeleteFile(outputFileName)

```

Sample 3 – Convert Document to Microsoft Excel Format (JScript Version)

```
// *****
// This open converter sample converts a document from an unsupported format
// (such as old Microsoft Excel documents) to the Microsoft Excel format using
// the Microsoft Excel COM objects.
// *****

// Save the document to convert to a temporary file.
var inputFileName = CustomConversion.InputDocument.TemporaryFileName();
var outputFileName = CustomConversion.OutputDocument.TemporaryFileName();
CustomConversion.InputDocument.SaveToFile(inputFileName);

// Lock here to instantiate the Microsoft Excel COM objects one at a time.
Mutex.AcquireLock("MicrosoftExcelMutex");

try {
    // Open the document using Microsoft Excel COM objects.
    var excelApplication = new ActiveXObject("Excel.Application");
    excelApplication.UserControl = false;
    var excelWorkbooks = excelApplication.Workbooks;
    var excelWorkbook = excelWorkbooks.Open(inputFileName);

    // Save the document as a Microsoft Excel document.
    excelWorkbook.SaveAs(outputFileName);
    excelWorkbook.Saved = true; // Prevent prompting
    excelWorkbook.Close();
    excelApplication.Quit();

    // The document format was successfully changed to Microsoft Excel. Convert the
    document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileName);
} catch (e) {
    // Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.name, 2);
    CustomConversion.Trace("Error: " + e.description, 2);
}

// Release our mutex in order for other scripts to be able to use the Microsoft Excel COM
objects.
Mutex.ReleaseLock();

// Clear our temporary files.
try {
    CustomConversion.DeleteFile(inputFileName);
} catch (e) {
}
try {
    CustomConversion.DeleteFile(outputFileName);
} catch (e) {
}
```

(VBScript)

```

'*****
' This open converter sample converts a document from an unsupported format
' (such as old Microsoft Excel documents) to the Microsoft Excel format using
' the Microsoft Excel COM objects.
'*****

OptionExplicit
On Error Resume Next

'Save the document to convert to a temporary file.
Dim inputFileNames: inputFileNames = CustomConversion.InputDocument.TemporaryFileName
CustomConversion.InputDocument.SaveToFile(inputFileNames)

'Lock here to instantiate the Microsoft Excel COM objects one at a time.
Call Mutex.AcquireLock("MicrosoftExcelMutex")

' Open the document using Microsoft Excel COM objects.
Dim excelApplication: Set excelApplication = CreateObject("Excel.Application")
excelApplication.UserControl = False
Dim excelWorkbooks: Set excelWorkbooks = excelApplication.Workbooks
Dim excelWorkbook: Set excelWorkbook = excelWorkbooks.Open(inputFileNames)

' Save the document as a Microsoft Excel document.
Dim outputFileNames: outputFileNames = CustomConversion.OutputDocument.TemporaryFileName
Call excelWorkbook.SaveAs(outputFileNames)
excelWorkbook.Saved = True ' Prevent prompting
Call excelWorkbook.Close()
Call excelApplication.Quit()

' Release our mutex in order for other scripts to be able to use the Microsoft Excel COM
objects.
Call Mutex.ReleaseLock()

If Err.Number = 0 Then
    ' The document format was successfully changed to Microsoft Excel. Convert the
document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileNames)
Else
    ' Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' Clear our temporary files.
Call CustomConversion.DeleteFile(inputFileNames)
Call CustomConversion.DeleteFile(outputFileNames)

```

Sample 4 – Convert Document to PDF Format (JScript Version)

```

// *****
// This open converter sample converts a document from an unsupported format
// (such as PostScript documents) to the PDF format using the Ghostscript
// tool.
//
// Required parameter:
//   - GhostScriptExecutablePath: The Ghostscript tool executable path.
//                               Example: c:\gs\gs7.04\bin\gswin32c.exe
// *****

// Get the shell object from the script global variables.
var shellObject = null;
try {
    shellObject = CustomConversion.GetGlobalVariable("shellObject");
} catch (e) {
    // Lock here to make sure to instantiate only one instance of the shell object.
    Mutex.AcquireLock("WScriptShellMutex");

    // Retry to get the shell object, it may have been instantiate by another script.
    try {
        shellObject = CustomConversion.GetGlobalVariable("shellObject");
    } catch (e) {
        // Create the shell object and add it to the global variables.
        CustomConversion.Trace("Creating the system shell object...", 0);
        shellObject = new ActiveXObject("WScript.Shell");
        CustomConversion.SetGlobalVariable("shellObject", shellObject);
    }

    // Release our mutex in order for other scripts to retrieve the shell object.
    Mutex.ReleaseLock();
}

// Save the document to convert to a temporary file.
var inputFileName = CustomConversion.InputDocument.TemporaryFileName();
var outputFileName = CustomConversion.OutputDocument.TemporaryFileName();
CustomConversion.InputDocument.SaveToFile(inputFileName);

// Lock here to instantiate GhostScript executable one at a time.
Mutex.AcquireLock("GhostScriptMutex");

try {
    // Perform the document format conversion.
    var ghostScriptCommandLine = '\"' +
CustomConversion.GetParameter("GhostScriptExecutablePath") + '\"';
    ghostScriptCommandLine += " -sOutputFile=\"" + outputFileName + "\"";
    ghostScriptCommandLine += '\"' + inputFileName + '\"';
    shellObject.Run(ghostScriptCommandLine, 0, true);

    // The document format was successfully changed to PDF. Convert this document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileName);
} catch (e) {
    // Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.description, 2);
}

```

```

}

// Release our mutex in order for other scripts to be able to use the GhostScript
executable.
Mutex.ReleaseLock();

// Clear our temporary files.
try {
    CustomConversion.DeleteFile(inputFileName);
} catch (e) {
}
try {
    CustomConversion.DeleteFile(outputFileName);
} catch (e) {
}

```

(VBScript Script)

```

*****
' This open converter sample converts a document from an unsupported format
' (such as PostScript documents) to the PDF format using the Ghostscript
' tool.
'
' Required parameter:
'   - GhostScriptExecutablePath: The Ghostscript tool executable path.
'                               Example: c:\gs\gs7.04\gswin32c.exe
*****

Option Explicit
On Error Resume Next

' Get the shell object from the script global variables.
Dim shellObject: Set shellObject = CustomConversion.GetGlobalVariable("shellObject")
If Not IsObject(shellObject) Then
    ' Lock here to make sure to instantiate only one instance of the shell object.
    Call Mutex.AcquireLock("WScriptShellMutex")

    ' Retry to get the shell object, it may have been instantiated by another script.
    Set shellObject = CustomConversion.GetGlobalVariable("shellObject")
    If Not IsObject(shellObject) Then
        ' Create the shell object and add it to the global variables.
        Call CustomConversion.Trace("Creating the system shell object... ", 0)
        Set shellObject = CreateObject("WScript.Shell")
        Call CustomConversion.SetGlobalVariable("shellObject", shellObject)
    End If

    ' Release our mutex, so other scripts can retrieve the shell object.
    Call Mutex.ReleaseLock()
End If

' Save the document to convert to a temporary file.
Dim inputFileName: inputFileName = CustomConversion.InputDocument.TemporaryFileName
CustomConversion.InputDocument.SaveToFile(inputFileName)

' Lock here to instantiate GhostScript executable one at a time.
Call Mutex.AcquireLock("GhostScriptMutex")

```

```

' Perform the document format conversion.
Err = 0
Dim DOUBLE_QUOTE: DOUBLE_QUOTE = chr(34)
Dim outputFileName: outputFileName = CustomConversion.OutputDocument.TemporaryFileName
Dim ghostScriptCommandLine: ghostScriptCommandLine =
CustomConversion.GetParameter("GhostScriptExecutablePath")
ghostScriptCommandLine = DOUBLE_QUOTE & ghostScriptCommandLine & DOUBLE_QUOTE
ghostScriptCommandLine = ghostScriptCommandLine & " -sOutputFile=" & DOUBLE_QUOTE &
outputFileName & DOUBLE_QUOTE
ghostScriptCommandLine = ghostScriptCommandLine & " -q -sDEVICE=pdfwrite -dnOPAUSE -dBATCH
-dNOPROMPT "
ghostScriptCommandLine = ghostScriptCommandLine & DOUBLE_QUOTE & inputFileName &
DOUBLE_QUOTE
Call shellObject.Run(ghostScriptCommandLine, 0, True)

' Release our mutex in order for other scripts to be able to use the GhostScript
executable.
Call Mutex.ReleaseLock()

If Err.Number = 0 Then
    ' The document format was successfully changed to PDF. Convert this document.
    CustomConversion.OutputDocument.LoadFromFile(outputFileName)
Else
    ' Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' Clear our temporary files.
Call CustomConversion.DeleteFile(inputFileName)
Call CustomConversion.DeleteFile(outputFileName)

```

Sample 5 – Convert Document Using IFilterWrapper (JScript Version)

```

// *****
// This open converter sample converts a document using the IFilterWrapper
// interface.
//
// Note that you do not need to create an open converter to convert a document
// using an IFilter, as CES manages IFilters. This sample only demonstrates
// another example of what can be achieved using CES open converters.
// *****

// Save the document to convert to a temporary file.
var inputFileName =
CustomConversion.InputDocument.TemporaryFileName(DocumentInfo.Extension);
var outputFileName = CustomConversion.OutputDocument.TemporaryFileName();
CustomConversion.InputDocument.SaveToFile(inputFileName)

try {
    // Convert the file using the appropriate IFilter.
    var tempDirectory = inputFileName.substr(0, inputFileName.lastIndexOf("\\"));
    IFilterWrapper.ConvertFromFile(inputFileName, tempDirectory);

    // Set the IFilter extracted text as the document content.
    IFilterWrapper.GetText().SaveToFile(outputFileName);
    CustomConversion.OutputDocument.LoadFromFile(outputFileName);
}

```

```

// Set the document properties.
var propertyIDs = new VBAArray(IFilterWrapper.GetPropertyList());
for (var i = propertyIDs.lbound(); I <= propertyIDs.ubound(); ++i) {
    var propertyID    = propertyIDs.getItem(i);
    var propertyName  = IFilterWrapper.GetNameForProperty(propertyID);
    var propertyValue = IFilterWrapper.GetValueForProperty(propertyID);
    if (propertyName.toLowerCase() == "title") {
        DocumentInfo.Title = propertyValue;
    } else if (propertyName.toLowerCase() == "subject") {
        DocumentInfo.Title = propertyValue;
    } else if (propertyName.toLowerCase() == "author") {
        DocumentInfo.Author = propertyValue;
    } else {
        DocumentInfo.SetFieldValue(propertyName, propertyValue);
    }
}
} catch (e) {
    // Unable to extract data from the document. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.description, 2);
}

// Clear our temporary files.
try {
    CustomConversion.DeleteFile(inputFileName);
} catch (e) {
}
try {
    CustomConversion.DeleteFile(outputFileName);
} catch (e) {
}

```

(VBScript Version)

```

'*****
' This open converter sample converts a document using the IFilterWrapper
' interface.
'
' Note that you don't need to create an open converter to convert a document
' using an IFilter, as CES manages IFilters. This sample only demonstrates
' another example of what can be achieved using CES open converters.
'*****

Option Explicit
On Error Resume Next

' Save the document to convert to a temporary file.
Dim inputFileName: inputFileName =
CustomConversion.InputDocument.TemporaryFileName(DocumentInfo.Extension)
CustomConversion.InputDocument.SaveToFile(inputFileName)

' Convert the file using the appropriate IFilter.
Dim tempDirectory: tempDirectory = Left(inputFileName, InStrRev(inputFileName, "\"))
Call IFilterWrapper.ConvertFromFile(inputFileName, tempDirectory)

' Set the IFilter extracted text as the document content.

```

```

Dir outputFileName: outputFileName = CustomConversion.OutputDocument.TemporaryFileName
IFilterWrapper.GetText.SaveToFile(outputFileName)
CustomConversion.OutputDocument.LoadFromFile(outputFileName)

' Set the document properties.
Dim propertyIDs: propertyIDs = IFilterWrapper.GetPropertyList
Dim propertyID
For Each propertyID In propertyIDs
    Dim propertyName: propertyName = IFilterWrapper.GetNameForProperty(propertyID)
    Dim propertyValue: propertyValue = IFilterWrapper.GetValueForProperty(propertyID)
    If LCase(propertyName) = "title" Then
        DocumentInfo.Title = propertyValue
    ElseIf LCase(propertyName) = "subject" Then
        DocumentInfo.Title = propertyValue
    ElseIf LCase(propertyName) = "author" Then
        DocumentInfo.Author = propertyValue
    Else
        Call DocumentInfo.SetFieldValue(propertyName, propertyValue)
    End If
Next

If Err.Number <> 0 Then
    ' Unable to extract data the document. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' Clear our temporary files.
Call CustomConversion.DeleteFile(inputFileName)
Call CustomConversion.DeleteFile(outputFileName)

```

Sample 6 – Convert Document to HTML Format (JScript Version)

```

// *****
// This open converter sample converts a document from an unsupported format
// (such as Web archive documents) to the HTML format using the Microsoft
// Internet Explorer COM objects.
// *****

// Save the document to convert to a temporary file.
var inputFileName = CustomConversion.InputDocument.TemporaryFileName(".mht");
CustomConversion.InputDocument.SaveToFile(inputFileName);

// Lock here to instantiate the Microsoft Internet Explorer COM objects one at a time.
Mutex.AcquireLock("MicrosoftInternetExplorerMutex");

try {
    // Open the document using Microsoft Internet Explorer COM objects.
    var ieApplication = new ActiveXObject("InternetExplorer.Application ");
    ieApplication.Visible = false;
    ieApplication.Silent = true;
    ieApplication.Navigate("about:blank ");
    ieApplication.Navigate(inputFileName);

    // Wait for the document to be fully loaded (from 3 to 6 seconds max).
    CustomConversion.Sleep(3000);
    var loopCount = 0;

```

```

while (ieApplication.Busy) {
    CustomConversion.Sleep(100);
    If (++loopCount > 30) {
        ieApplication.Stop();
        throw new exception("Timeout");
    }
}

// Set the rendered HTML as the document content.
var UTF8_CHARSET = 2;
var htmlContent = ieApplication.Document.documentElement.outerHTML;
CustomConversion.OutputDocument.WriteAsByteString(htmlContent, 2);
ieApplication.Quit();
} catch (e) {
    // Unable to change the document format. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    CustomConversion.Trace("Error: " + e.description, 2);
}

// Release our mutex in order for other scripts to be able to use the Microsoft Internet
Explorer COM objects.
Mutex.ReleaseLock();

// Clear our temporary file.
try {
    CustomConversion.DeleteFile(inputFileName);
} catch (e) {
}

```

(VBScript Version)

```

'*****
' This open converter sample converts a document from an unsupported format
' (such as Web archive documents) to the HTML format using the Microsoft
' Internet Explorer COM objects.
'*****

Option Explicit
On Error Resume Next

' Save the document to convert to a temporary file.
Dim inputFileName: inputFileName =
CustomConversion.InputDocument.TemporaryFileName(".mht")
CustomConversion.InputDocument.SaveToFile(inputFileName)

' Lock here to instantiate the Microsoft Internet Explorer COM objects one at a time.
Call Mutex.AcquireLock("MicrosoftInternetExplorerMutex ")

' Open the document using Microsoft Internet Explorer COM objects.
Dim ieApplication: Set ieApplication = CreateObject("InternetExplorer.Application")
ieApplication.Visible = False
ieApplication.Silent = True
Call ieApplication.Navigate("about:blank")
Call ieApplication.Navigate(inputFileName)

' Wait for the document to be fully loaded (from 3 to 6 seconds max).
Dim loopCount: loopCount = 0

```

```

Do While (ieApplication.Busy)
    Call CustomConversion.Sleep(100)
    loopCount = loopCount + 1
    If loopCount > 30 Then
        Call ieApplication.Stop()
        Call Err.Raise(vbObjectError + 2, "Timeout", "Timeout")
        Exit Do
    End If
Loop

' Set the rendered HTML as the document content.
Dim UTF8_CHARSET: UTF8_CHARSET = 2
Dim htmlContent: htmlContent = ieApplication.Document.documentElement.outerHTML
Call CustomConversion.OutputDocument.WriteAsByteString(htmlContent, UTF8_CHARSET)
Call ieApplication.Quit()

' Release our mutex in order for other scripts to be able to use the Microsoft Internet
Explorer COM objects.
Call Mutex.ReleaseLock()

If Err.Number <> 0 Then
    ' Unable to change the document format. Reject the document and report the error.
    Document.Info.IsValid = False
    Call CustomConversion.Trace("Error: " & Err.Description, 2)
End If

' Clear our temporary file.
Call CustomConversion.DeleteFile(inputFileName)

```

Conversion Scripts

Conversion scripts are hooks in the CES conversion process that allows administrators to fully customize the way documents are indexed.

There are two types of conversion scripts:

- [Preconversion Scripts](#), executed before a conversion occurs;
- [Postconversion Scripts](#), executed after a conversion occurs.

It is possible to execute a conversion script for every document of a particular source ([source conversion script](#)) or of every source of the index ([global conversion script](#)).

Scripting Possibilities

Possibilities are vast when using conversion scripts. The following lists examples of tasks performed using scripting objects:

- Override the language of the document;
- Override or modify the content of the document;
- Override or modify the HTML rendering of the document;
- Add or modify metadata to the document;
- Update the permissions of the document;
- Reject documents;
- Modify the ranking value of a document.

Preconversion Scripts

Preconversion scripts are VBScripts or JScripts executed before the conversion, but after the files have been crawled by the connector (refer to [Conversion Phases](#) for an overview of the various conversion phases). Preconversion scripts can alter or add metadata related to the document using the DocumentInfo Object. This phase is often used to determine if a document is rejected based on the information provided in the DocumentInfo Object. For example, a user could reject a document according to its size.

How to Create a Preconversion Script

To build a preconversion script, it is possible to use the following scripting objects:

- PreConversion Object (IPreConversion Interface)
- DocumentInfo Object (IDocumentInfo Interface)
- Mutex Object (ICGLMutex Interface)

How to Setup a Preconversion Script

Once the preconversion script is created, perform the following steps in order to use it:

- Add the preconversion script to the CES configuration (refer to [How to Add a Preconversion Script](#)).
- Select whether the script is executed on all the documents in every source or only in a specific source.
 - If the preconversion script is executed on all the documents in every source, refer to [How to Apply a Global Conversion Script](#).
 - If the preconversion script is executed on all the documents in a specific source, refer to [How to Apply a Source Conversion Script](#).

Preconversion Script Samples

The following lists the different preconversion script samples available:

Sample 1 – Add New Document Metadata (JScript Version)

```
// *****
// This preconversion script sample adds two new document metadata:
//   fileSize - The number of bytes of the file to convert.
//   filename - The name of the file to convert.
//*****

// Add the "fileSize" metadata.
DocumentInfo.SetFieldValue("fileSize", PreConversion.InputDocument.BytesCount);

// Extract and add the "fileName" metadata.
var documentFileName = GetFileName(DocumentInfo.URI);
DocumentInfo.SetFieldValue("fileName", documentFileName);

// Output information to the CES console.
PreConversion.Trace("The file '" + documentFileName + "' has " +
PreConversion.InputDocument.BytesCount.toString() + " bytes.", 1);

// This method extracts the file name from a given URI.
function GetFileName(URI)
{
    var fileName = URI;
```

```

var pos = filename.lastIndexOf("/");
if (pos == -1) {
    pos = filename.lastIndexOf("\\");
}
If (pos != -1) {
    fileName = fileName.substring(pos + 1, filename.length);
}
pos = fileName.indexOf("?")
if (pos != -1) {
    filename = filename.substring(0, pos);
}
Return filename;
}

```

(VBScript Version)

```

'*****
' This preconversion script sample adds two new document metadata:
'   fileSize - The number of bytes of the file to convert.
'   filename - The name of the file to convert.
'*****

Option Explicit

' Add the "fileSize" metadata.
DocumentInfo.SetFieldValue "fileSize", PreConversion.InputDocument.BytesCount

' Extract and add the "fileName" metadata.
Dim filename: filename = GetFileName(DocumentInfo.URI)
DocumentInfo.SetFieldValue "fileName", filename

' Output information to the CES console.
Call PreConversion.Trace("The file '" & filename & "' has " &
CStr(PreConversion.InputDocument.BytesCount) & " bytes ", 1)

' This method extracts the file name from a given URI.
Function GetFileName(ByVal URI)
    Dim pos: pos = InStrRev(URI, "/")
    If pos = 0 Then
        Pos = InStrRev(URI, "\")
    End If
    If pos = 0 Then
        GetFileName = URI
    Else
        GetFileName = Right(URI, Len(URI) - pos)
    End If
    pos = InStr(GetFileName, "?")
    If pos > 1 Then
        GetFileName = Left(GetFileName, pos - 1)
    End If
End Function

```

Sample 2 – Update Printable and Clickable URI (JScript Version)

```
// *****
// This preconversion script sample updates the document printable and
// clickable URI.
// *****

// Update the document printable URI (the URI the user sees on a search result).
DocumentInfo.PrintableURI = DocumentInfo.PrintableURI.replace("file://", "http://");

// Update the document clickable URI (the URI that is opened when a user clicks on a search
// result title).
DocumentInfo.ClickableURI = DocumentInfo.ClickableURI.replace("file://", "http://");
```

(VBScript Version)

```
' *****
' This preconversion script sample updates the document printable and
' clickable URI.
' *****

Option Explicit

' Update the document printable URI (the URI the user sees on a search result).
DocumentInfo.PrintableURI = Replace(DocumentInfo.PrintableURI, "file://", "http://")

' Update the document clickable URI (the URI that is opened when a user clicks on a search
// result title).
DocumentInfo.ClickableURI = Replace(DocumentInfo.ClickableURI, "file://", "http://")
```

Sample 3 – Filter Based on Size and Date (JScript Version)

```
// *****
// This preconversion script sample filters files depending on file size and
// date:
// - If the file's size is greater than 1MB, it is rejected.
// - If the file's last modification date is older than 01/01/2007, it is
// rejected.
// *****

// Max file size = 1 MB. (In Bytes)
var MAX_FILE_SIZE = 1024 * 1024;

// The minimum date to index.
var MIN_DATE = new Date("01/01/2007");

// Retrieves the size and last modification date of the document.
var fileSize = PreConversion.InputDocument.BytesCount;
var fileDate = DocumentInfo.Date;

// Index this document only if
// - the file size is acceptable.
// - the file date is acceptable.
DocumentInfo.IsValid = fileSize <= MAX_FILE_SIZE && fileDate >= MIN_DATE;
```

(VBScript Version)

```

*****
' This preconversion script sample filters files depending on file size and
' date:
'   - If the file's size is greater than 1MB, it is rejected.
'   - If the file's last modification date is older than 01/01/2007, it is
'     rejected.
*****

```

Option Explicit

```

' Max file size = 1 MB. (In Bytes)
Dim MAX_FILE_SIZE: MAX_FILE_SIZE = 1024 * 1024

' The minimum date to index.
Dim MIN_DATE: MIN_DATE = #01/01/2007#

' Retrieves the size and last modification date of the document.
Dim fileSize: fileSize = PreConversion.InputDocument.BytesCount
Dim fileDate: fileDate = DocumentInfo.Date

' Index this document only if
'   - the file size is acceptable.
'   - the file date is acceptable.
DocumentInfo.IsValid = fileSize <= MAX_FILE_SIZE And fileDate >= MIN_DATE

```

Sample 4 – Use a Conversion Script Parameter (JScript Version)

```

// *****
// This preconversion script sample demonstrates how to use a conversion
// script parameter. It adds a new document metadata using the script
// parameter values.
//
// Required parameters:
//   - MetadataName: The value to use as the new metadata name.
//                   Example: MyMetadataName
//   - MetadataValue: The value to use as the new metadata value.
//                   Example: MyMetadataValue
// *****

var SCRIPT_PARAMETER_METADATA_NAME = "MetadataName";
var SCRIPT_PARAMETER_METADATA_VALUE = "MetadataValue";
try {
  // Get the name and value of the metadata to add.
  var metadataName = PreConversion.GetParameter(SCRIPT_PARAMETER_METADATA_NAME);
  var metadataValue = PreConversion.GetParameter(SCRIPT_PARAMETER_METADATA_VALUE);

  // Add a new metadata using these script parameter values.
  DocumentInfo.SetFieldValue(metadataName, metadataValue);

  // Output the script parameter values to the CES console.
  PreConversion.Trace("The value of the script parameter named '" +
    SCRIPT_PARAMETER_METADATA_NAME + "' is '" + metadataName + "'.", 1);
  PreConversion.Trace("The value of the script parameter named '" +
    SCRIPT_PARAMETER_METADATA_VALUE + "' is '" + metadataValue + "'.", 1);
} catch (e) {

```

```

    // Unable to get the script parameters. Reject the document and report the error.
    DocumentInfo.IsValid = false;
    PreConversion.Trace("Error: " + e.description, 2);
  }

```

(VBScript Version)

```

'*****
' This preconversion script sample demonstrates how to use a conversion
' script parameter. It adds a new document metadata using the script
' parameter values.
'
' Required parameters:
'   - MetadataName: The value to use as the new metadata name.
'                   Example: MyMetadataName
'   - MetadataValue: The value to use as the new metadata value.
'                   Example: MyMetadataValue
'*****

Option Explicit
On Error Resume Next

Dim SCRIPT_PARAMETER_METADATA_NAME: SCRIPT_PARAMETER_METADATA_NAME = "MetadataName"
Dim SCRIPT_PARAMETER_METADATA_VALUE: SCRIPT_PARAMETER_METADATA_VALUE = "MetadataValue"

' Get the value of the script parameter named "MetadataValue".
Dim metadataName: metadataName =
PreConversion.GetParameter(SCRIPT_PARAMETER_METADATA_NAME)
Dim metadataValue: metadataValue =
PreConversion.GetParameter(SCRIPT_PARAMETER_METADATA_VALUE)

' Validate the script parameter was found.
If Err.Number = 0 Then
    ' Add a new metadata using these script parameter values.
    DocumentInfo.SetFieldValue metadataName, metadataValue

    ' Output the script parameter values to the CES console.
    Call PreConversion.Trace("The value of the script parameter named '" &
SCRIPT_PARAMETER_METADATA_NAME & "' is '" & metadataName & "'.", 1)
    Call PreConversion.Trace("The value of the script parameter named '" &
SCRIPT_PARAMETER_METADATA_VALUE & "' is '" & metadataValue & "'.", 1)
Else
    ' Unable to get the script parameters. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call PreConversion.Trace("Error: " & Err.Description, 2)
End If

```

Sample 5 – Use a Conversion Script Global Variable (JScript Version)

```

// *****
// This preconversion script sample demonstrates how to use a conversion
// script global variable. It increments a global variable that contains
// the document position and adds a new document metadata using this position
// value.
// *****

// Our global variable name.
var GLOBAL_VARIABLE_NAME = "Js-DocumentPosition";

```

```

// We need a mutex to make sure only one script updates the global variable at a time.
// Global variables are multithread safe, but this mutex assures the document's position
// is accurate.
var MUTEX_NAME = GLOBAL_VARIABLE_NAME + "Mutex";
Mutex.AcquireLock(MUTEX_NAME);

// Get the value of the global variable named "DocumentPosition".
var documentPosition;
try {
    documentPosition = PreConversion.GetGlobalVariable(GLOBAL_VARIABLE_NAME);
} catch (e) {
    // If the global variable was not found, it means it is the first this script is
    // running.
    documentPosition = 0;
}

// We must increment the document position.
++documentPosition;

// Set the new document position.
PreConversion.SetGlobalVariable(GLOBAL_VARIABLE_NAME, documentPosition);

// Release the mutex in order for other scripts to be able to safely use the global
// variable.
Mutex.ReleaseLock();

// Add a new metadata using this global variable.
DocumentInfo.SetFieldValue(GLOBAL_VARIABLE_NAME, documentPosition);

// Output the document position to the CES console.
PreConversion.Trace("The position of the document '" + DocumentInfo.URI + "' is '" +
documentPosition.toString() + "'.", 1);

```

(VBScript Version)

```

*****
' This preconversion script sample demonstrates how to use a conversion
' script global variable. It increments a global variable that contains
' the document position and adds a new document metadata using this position
' value.
*****

Option Explicit
On Error Resume Next

' Our global variable name.
Dim GLOBAL_VARIABLE_NAME: GLOBAL_VARIABLE_NAME = "Vbs-DocumentPosition"

' We need a mutex to make sure only one script updates the global variable at a time.
' Global variable are multithread safe, but this mutex assures the document's position
' is accurate.
Dim MUTEX_NAME: MUTEX_NAME = GLOBAL_VARIABLE_NAME & "Mutex"
Mutex.AcquireLock MUTEX_NAME

' Get the value of the global variable named "DocumentPosition".

```

```

Dim documentPosition: documentPosition =
PreConversion.GetGlobalVariable(GLOBAL_VARIABLE_NAME)

' If the global variable was not found, it means it is the first this script is running.
If Err.Number <> 0 Then
    documentPosition = 0
End If

' We must increment the document position.
DocumentPosition = documentPosition + 1

' Set the new document position.
PreConversion.SetGlobalVariable GLOBAL_VARIABLE_NAME, documentPosition

' Release the mutex in order for other scripts to be able to safely use the global
variable.
Mutex.ReleaseLock()

' Add a new metadata using this global variable.
DocumentInfo.SetFieldValue GLOBAL_VARIABLE_NAME, documentPosition

' Output the document position to the CES console.
Call PreConversion.Trace("The position of the document '" & DocumentInfo.URI & "' is '" &
CStr(documentPosition) & "'.", 1)
  
```

Sample 6 – Use CES Administration API (JScript Version)

```

// *****
// This preconversion script sample uses the CES Administration API to
// duplicate all of the documents to convert in a separate source.
// *****

// Update this information to match your needs.
var CES_ADMIN_API_CLASS_ID = "CESAdmin.Admin.6.0";
var DUPLICATE_SOURCE_NAME = DocumentInfo.Source + "Duplicate";
var MUTEX_NAME = "CESAdminAPIMutex";

// Lock to make sure there is only one instance of the COM Admin object.
Mutex.AcquireLock(MUTEX_NAME);

try {
    // Connect to the CES Admin API.
    var admin = new ActiveXObject(CES_ADMIN_API_CLASS_ID);
    admin.Connect("localhost", "default");

    // Get the collection.
    var physicalIndex = admin.PhysicalIndexes.Item(1);
    var collection = physicalIndex.Collections.ItemByName(DocumentInfo.Collection);

    // Get the duplicate source.
    var duplicateSource = null;
    try {
        duplicateSource = collection.Sources.ItemByName(DUPLICATE_SOURCE_NAME);
    } catch (e) {
        // The source does not exist, create a new one.
        duplicateSource = collection.Sources.DuplicateByName(DocumentInfo.Source);
        duplicateSource.Name = DUPLICATE_SOURCE_NAME;
    }
}
  
```

```

        duplicateSource.PreConversionScriptID = -1;
        admin.Commit();
    }
    // Command duplicate-source to index this document.
    duplicateSource.RefreshURL(DocumentInfo.URI);

    // Release the CES Admin API objects.
    duplicateSource = null;
    collection = null;
    physicalIndex = null;
    admin.Disconnect();
    admin = null;
} finally {
    // Release the mutex in order for other scripts to be able to safely use the CES Admin
    API.
    Mutex.ReleaseLock();
}

```

(VBScript Version)

```

'*****
' This preconversion script sample uses the CES Administration API to
' duplicate all of the documents to convert in a separate source.
'*****

Option Explicit
On Error Resume Next

' Update this information to match your needs.
Dim CES_ADMIN_API_CLASS_ID:      CES_ADMIN_API_CLASS_ID = "CESAdmin.Admin.6.0"
Dim DUPLICATE_SOURCE_NAME:      DUPLICATE_SOURCE_NAME = DocumentInfo.Source &
"Duplicate"
Dim MUTEX_NAME:                 MUTEX_NAME = "CESAdminAPIMutex"

' Lock to make sure there is only one instance of the COM Admin object.
Mutex.AcquireLock MUTEX_NAME

' Connect to the CES Admin API.
Dim admin: Set admin = CreateObject(CES_ADMIN_API_CLASS_ID)
Admin.Connect "localhost", "default"

' Get the collection.
Dim physicalIndex: Set physicalIndex = admin.PhysicalIndexes.Item(1)
Dim collection: Set collection =
physicalIndex.Collections.ItemByName(DocumentInfo.Collection)

If Err.Number = 0 Then
    ' Try to get the duplicate source.
    Dim duplicateSource: Set duplicateSource = Nothing
    Set duplicateSource = collection.Sources.ItemByName(DUPLICATE_SOURCE_NAME)

    ' Validate the source exists.
    If duplicateSource Is Nothing Then
        ' Clear the error state.
        Call Err.Clear()

        ' The source does not exist, create a new one.

```

```

    Set duplicateSource = collection.Sources.DuplicateByName(DocumentInfo.Source)
    duplicateSource.Name = DUPLICATE_SOURCE_NAME
    duplicateSource.PreConversionScriptID = -1
    admin.Commit()
  End If

  ' Command duplicate-source to index this document.
  duplicateSource.RefreshURI(DocumentInfo.URI)

  ' Release the CES Admin API objects.
  Set duplicateSource = Nothing
End If
Set collection = Nothing
Set physicalIndex = Nothing
Call admin.Disconnect()
Set admin = Nothing

If Err.Number <> 0 Then
  ' Unable to duplicate the source.
  Document Info.IsValid = False
  Call PreConversion.Trace("Error: " & Err.Description, 2)
End If

' Release the mutex in order for other scripts to be able to safely use the CES Admin API.
Mutex.ReleaseLock()

```

Postconversion Scripts

Postconversion scripts are VBScripts or JScripts executed after the conversion. Preconversion scripts have full access to the extracted text and HTML from the document, as well as its metadata; while postconversion scripts modify the extracted information of the converted documents, which is not available in preconversion. Postconversion enables the addition, rejection or modification of the metadata from a document by using the DocumentInfo Object. It can also override the extracted text or HTML in order to send it to CES or reject a document based on the results of the conversion.

How to Create a Postconversion Script

To build a postconversion script, it is possible to use the following scripting objects:

- PostConversion Object (IPostConversion Interface)
- DocumentInfo Object (IDocumentInfo Interface)
- Mutex Object (ICGLMutex Interface)

How to Setup a Postconversion Script

Once the postconversion script is created, perform the following steps in order to use it:

- Add the postconversion script to the CES configuration (refer to [How to Add a Postconversion Script](#)).
- Select whether the script is executed on all the documents in every source or only in a specific source.
 - If the postconversion script is executed on all the documents in every source, refer to [How to Apply a Global Conversion Script](#).
 - If the postconversion script is executed on all the documents in a specific source, refer to [How to Apply a Source Conversion Script](#).

Postconversion Script Samples

The following lists the different postconversion script samples available:

Sample 1 – Add New Metadata Named AllFieldValues (JScript Version)

```
// *****
// This postconversion script sample adds a new metadata named
// "AllFieldValues". This new metadata contains the name, the value and the
// type of all document metadata.
// *****

// A double quote and a new line.
var DOUBLE_QUOTE = '"';
var NEW_LINE     = '\r\n';

// This buffer variable holds all metadata names and values.
var allValues = "<AllFiledValues>" + NEW_LINE;

// A collection of all field names.
var filedNames = new VArray(DocumentInfo.Fileds());

// For each document field name.
For (var I = fieldnames.lbound(); I <= fieldnames.ubound(); ++i) {
    // Get the field value.
    var fieldName = fieldnames.getItem(i);
    var fieldValue = DocumentInfo.GetFieldValue(fieldName);

    // Add the metadata name and its value to the buffer variable.
    allValues += "<Field";
    allValues += " name=" + DOUBLE_QUOTE + fieldName + DOUBLE_QUOTE;
    allValues += " value=" + DOUBLE_QUOTE + fieldValue + DOUBLE_QUOTE;
    allValues += " type=" + DOUBLE_QUOTE + typeof(fieldValue) + DOUBLE_QUOTE;
    allValues += " />" + NEW_LINE;
}

// Add the new metadata.
DocumentInfo.SetFieldValue("AllFieldValues", allValues + "</AllFieldValues>" + NEW_LINE);
```

(VBScript Version)

```
' *****
' This postconversion script sample adds a new metadata named
' "AllFieldValues". This new metadata contains the name, the value and the
' type of all document metabata.
' *****

Option Explicit

' A double quote.
Dim DOUBLE_QUOTE: DOUBLE_QUOTE = chr(34)

' This buffer variable holds all metadata names and values.
Dim allValues: allValues = "<AllFieldValues>" & vbNewLine

' A collection of all field names.
Dim fieldnames: fieldnames = DocumentInfo.Fields
```

```

' For each document field name.
Dim fieldname
For Each fieldname In fieldnames
  ' Get the field value.
  Dim fieldValue: fieldValue = DocumentInfo.GetFieldValue(CStr(fieldname))

  ' Add the metadata name and its value to the buffer variable.
  allValues = allValues & "<Field"
  allValues = allValues & " name=" & DOUBLE_QUOTE & CStr(fieldname) & DOUBLE_QUOTE
  allValues = allValues & " value=" & DOUBLE_QUOTE & fieldValue & DOUBLE_QUOTE
  allValues = allValues & " type=" & DOUBLE_QUOTE & TypeName(fieldValue) & DOUBLE_QUOTE
  allValues = allValues & " />" & vbNewLine
Next

' Add the new metadata.
Call DocumentInfo.SetFieldValue("AllFieldValues", allValues & "</AllFieldValues>" &
vbNewLine)

```

Sample 2 – Update Document Concepts and Summary Sentences (JScript Version)

```

// *****
// This postconversion script sample updates the document concepts and summary
// sentences.
//
// It first adds four new metadata fields:
// - TopConcept:           The original top concept.
// - NumberOfConcepts:    The number of concepts.
// - TopSummarySentence:  The original summary sentence.
// - NumberOfSummarySentences: The number of summary sentences.
//
// Then it changes the top concept and the top summary sentence to a custom
// value.
// *****

// Get the document concepts.
var concepts = new VBArray(DocumentInfo.SummaryConcepts);

// Add the number of concept metadata.
DocumentInfo.SetFieldValue("NumberOfConcepts", concepts.ubound() - concepts.lbound() + 1);

if (concepts.ubound() >= concepts.lbound()) {
  // Add the top concept metadata.
  var topConcept = concepts.getItem(concepts.lbound());
  DocumentInfo.SetFieldValue("TopConcept", topConcept);
  PostConversion.Trace("The original top concept was '" + topConcept + "'.", 1);

  // Update the top concept.
  var newConcepts = new ActiveXObject("Scripting.Dictionary");
  newConcepts.add(0, "NewTopConcept");
  for (var I = concepts.lbound() + 1; I <= concepts.ubound(); ++i) {
    newConcepts.add(I, concepts.getItem(i));
  }
  DocumentInfo.SummaryConcepts = newConcepts.Items();
}

```

```
// Get the document summary sentences.
var summarySentences = new VBAArray(DocumentInfo.SummarySentences);

// Add the number of summary sentences metadata.
DocumentInfo.SetFieldValue("NumberOfSummarySentences", summarySentences.ubound() -
summarySentences.lbound() + 1);

If (summarySentences.ubound() >= summarySentences.lbound()) {
    // Add the top summary sentence metadata.
    var topSummarySentence = summarySentences.getItem(summarySentences.lbound());
    DocumentInfo.SetFieldValue("TopSummarySentences", topSummarySentence);
    PostConversion.Trace("The original top summary sentence was '" + topSummarySentence +
    "'.", 1);

    // Update the top summary sentence.
    var newSummarySentences = new ActiveXObject("Scripting.Dictionary");
    newSummarySentences.Add(0, "New Top Summary Sentence");
    for (var I = summarySentences.lbound() + 1; i <= summarySentences.ubound(); ++i) {
        newSummarySentences.add(I, summarySentences.getItem(i));
    }
    DocumentInfo.SummarySentences = newSummarySentences.Items();
}
```

(VBScript Version)

```
*****
' This postconversion script sample updates the document concepts and summary
' sentences.
'
' It first adds four new metadata fields :
' - TopConcept:           The original top concept.
' - NumberOfConcepts:    The number of concepts.
' - TopSummarySentence:  The original summary sentence.
' - NumberOfSummarySentences: The number of summary sentences.
'
' Then it changes the top concept and the top summary sentence to a custom
' value.
*****

Option Explicit

' Get the document concepts.
Dim concepts: concepts = DocumentInfo.SummaryConcepts

' Add the number of concept metadata.
DocumentInfo.SetFieldValue "NumberOfConcepts", (UBound(concepts) - Lbound(concepts)) + 1

If UBound(concepts) >= LBound(concepts) Then
    ' Add the top concept metadata.
    Dim topConcept: topConcept = concepts(LBound(concepts))
    DocumentInfo.SetFieldValue "TopConcept", topConcept
    Call PostConversion.Trace("The original top concept was '" & topConcept & "'.", 1)

    'Update the top concept.
    concepts(LBound(concepts)) = "NewTopConcept"
    DocumentInfo.SummaryConcepts = concepts
End If
```

```

' Get the document summary sentences.
Dim summarySentences: summarySentences = DocumentInfo.SummarySentences

' Add the number of summary sentences metadata.
DocumentInfo.SetFieldValue "NumberOfSummarySentences", (UBound(summarySentences) -
Lbound(summarySentences)) + 1

If UBound(summarySentences) >= LBound(summarySentences) Then
  ' Add the top summary sentence metadata.
  Dim topSummarySentence: topSummarySentence = summarySentences(LBound(summarySentences))
  DocumentInfo.SetFieldValue "TopSummarySentence", topSummarySentence
  Call PostConversion.Trace("The original top summary sentence was '" &
topSummarySentence & "'.", 1)

  ' Update the top summary sentence.
  summarySentences(LBound(summarySentences)) = "New Top Summary Sentence"
  DocumentInfo.SummarySentences = summarySentences
End If

```

Sample 3 – Set a Custom Document Weight (JScript Version)

```

// *****
// This postconversion script sample sets a custom document's weight depending
// on user defined rules.
//
// Required parameter:
//   - DocumentNameToBoost: The file name of the document to boost.
//   Example: MyDocument.txt
// *****

try {
  // In this example, we boost the document's weight only when the
  // document's URI is set as a script parameter.
  // Note: You should update this example to add your own rules.
  var SCRIPT_PARAMETER_NAME = "DocumentNameToBoost";
  var boostDocument = GetFileName(DocumentInfo.URI) ==
PostConversion.GetParameter(SCRIPT_PARAMETER_NAME);

  // Check whether we must boost the document's rank.
  if (boostDocument) {
    // The document weight property is an integer and the possible values
    // vary from 0 to 15 inclusively; where 0 sets the lowest rank and 15
    // the highest one (the default value is 7).
    DocumentInfo.CustomWeight = 10;
    PostConversion.Trace("The rank was boosted for this document: '" + DocumentInfo.URI
+ "'.", 1);
  }
} catch (e) {
  // Unable to boost the document. Reject the document and report the error.
  DocumentInfo.IsValid = false;
  PostConversion.Trace("Error: " + e.description, 2);
}

// This method extracts the file name from a given URI.
function GetFileName(URI)
{

```

```

var fileName = URI;
var pos = fileName.lastIndexOf("/");
if (pos == -1) {
    pos = fileName.lastIndexOf("\\");
}
if (pos != -1) {
    fileName = fileName.substring(pos + 1, fileName.length);
}
pos = fileName.indexOf("?")
if (pos != -1) {
    fileName = fileName.substring(0, pos);
}
Return fileName;
}

```

(VBScript Version)

```

'*****
' This postconversion script sample sets a custom document's weight depending
' on user defined rules.
'
' Required parameter:
'   - DocumentNameToBoost: The file name of the document to boost.
'       Example: MyDocument.txt
'*****

```

Option Explicit

```

' In this example, we boost the document's weight only when the
' document's URI is set as a script parameter.
' Note: You should update this example to add your own rules.
Dim SCRIPT_PARAMETER_NAME: SCRIPT_PARAMETER_NAME = "DocumentNameToBoost"
Dim boostDocument: boostDocument = GetFileName(DocumentInfo.URI) =
PostConversion.GetParameter(SCRIPT_PARAMETER_NAME)

' Check whether we must boost the document's rank.
If boostDocument Then
    ' The document weight property is an integer and the possible values
    ' vary from 0 to 15 inclusively; where 0 sets the lowest rank and 15
    ' the highest one (the default value is 7).
    DocumentInfo.CustomWeight = 10
    Call PostConversion.Trace("The rank was boosted for this document: " &
DocumentInfo.URI & ". ", 1)
End If

If Err.Number <> 0 Then
    ' Unable to boost the document. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call PostConversion.Trace("Error: " & Err.Description, 2)
End If

' This method extracts the file name from a given URI.
Function GetFileName(ByVal URI)
    Dim pos: pos = InStrRev(URI, "/")
    If pos = 0 Then
        GetFileName = URI
    Else

```

```

    GetFileName = Right(URI, Len(URI) - pos)
  End If
  pos = InStr(GetFileName, "?")
  If pos > 1 Then
    GetFileName = Left(GetFileName, pos - 1)
  End If
End Function

```

Sample 4 – Change View as HTML Version of Document (JScript Version)

```

// *****
// This postconversion script sample changes the default View As HTML version
// of the document.
// *****

// Build a new document content stream.
Var documentContent = "This is a custom document content set by a postconversion script! ";

// Set the document content stream.
PostConversion.TextToOverride.WriteString(documentContent);

// Build a new View As HTML stream.
var viewAsHTML = "";
viewAsHTML += "<html><body>";
viewAsHTML += "This is a custom View As HTML stream set by a postconversion script! ";
viewAsHTML += "</body></html>";
// Set the View As HTML stream.
PostConversion.HTMLOutputToOverride.WriteString(viewAsHTML);

```

(VBScript Version)

```

' *****
' This postconversion script sample changes the default View As HTML version
' of the document.
' *****

Option Explicit

' Build a new document content stream.
Dim documentContent
documentContent = "This is a custom document content set by a postconversion script!"

' Set the document content stream.
PostConversion.TextToOverride.WriteString documentContent

' Build a new View As HTML stream.
Dim viewAsHTML: viewAsHTML = ""
viewAsHTML = viewAsHTML & "<html><body>"
viewAsHTML = viewAsHTML & "This is a custom View As HTML stream set by a postconversion
script!"
viewAsHTML = viewAsHTML & "</body></html>"

' Set the View As HTML stream.
PostConversion.HTMLOutputToOverride.WriteString viewAsHTML

```

Sample 5 – Analyze Text to Find Metadata (JScript Version)

```

// *****
// This postconversion script sample analyses the document extracted text to
// find metadata. The metadata names and values are added to the
// document.
//
// This script assumes the document content has the format of old Windows
// .ini files.
//
// Format sample:
//
//     Name1 = Value1
//     Name2 = Value2
//     Name3 = Value3
// *****

// Get the document extracted content.
PostConversion.Text.SeekReadPointerInChars(0);
var documentContent = PostConversion.Text.ReadString(PostConversion.Text.CharsCount);

// New regular expression object used to find the metadata name and value pairs.
var regExp = new RegExp("(.*)=(.*)", "g");

// Loop over all the regular expression matches in the string.
var arr;
while ((arr = regExp.exec(documentContent)) != null) {
    var metadataName = RegExp.$1;
    var metadataValue = RegExp.$2;

    // Add to document's metadata.
    DocumentInfo.SetFieldValue(trim(metadataName), trim(metadataValue));
}

// This method trims the given string.
function trim(p_StringToTrim)
{
    return p_StringToTrim.replace(/^\s+/, '').replace(/\s+$/, '');
}

```

(VBScript Version)

```

' *****
' This postconversion script sample analyses the document extracted text to
' find metadata. These metadata names and values are added to the
' document.
'
' This script assumes the document content has the format of old Windows
' .ini files.
'
' Format sample:
'
'     Name1 = Value1
'     Name2 = Value2
'     Name3 = Value3
' *****

```

Option Explicit

```

' Get the document extracted content.
Call PostConversion.Text.SeekReadPointerInChars(0)
Dim documentContent: documentContent =
PostConversion.Text.ReadString(PostConversion.Text.CharsCount)

' New regular expression object used to find the metadata name and value pairs.
Dim regExpression: Set regExpression = New RegExp

' The regular expression should scan all lines.
regExpression.Globa = True

' Must find a '=' enclosed by any number of any character.
' Front and trail strings are tagged to be used later with SubMatches.
regExpression.Pattern = "(.*)=(.*)"

' Get the matches.
Dim regExpMatches: Set regExpMatches = regExpression.Execute(documentContent)

' Add metadata if some matches were found.
If regExpMatches.Count > 0 Then
    Dim regExpMatch
    For Each regExpMatch In regExpMatches
        Dim metadataName: metadataName = Trim(regExpMatch.SubMatches(0))
        Dim metadataValue: metadataValue = Trim(regExpMatch.SubMatches(1))

        ' Add to document's metadata.
        Call DocumentInfo.SetFieldValue(metadataName, metadataValue)
    Next
End If

```

Sample 6 – Send Email to Specify Indexing (JScript Version)

```

// *****
// This postconversion script sample sends an email to notify a specific
// document has been index.
//
// Required parameters:
//
// - SMTPServer: The SMTP server to use to send the email.
//               Example: smtp.myserver.com
//
// - From:       The email address of the email sender.
//               Example: myself@mycompany.com
//
// - To:        The email addresses of the email recipients.
//               Example: someone@mycompany.com
// *****

try {
    // Get the email settings from the script parameters.
    var SMTP_SERVER = PostConversion.GetParameter("SMTPServer");
    var FROM_EMAIL  = PostConversion.GetParameter("From");
    var TO_EMAIL    = PostConversion.GetParameter("To");
    var NEW_LINE    = "\r\n";

```

```

// Generate the email body.
var body = "";
body += "Document information:" + NEW_LINE;
body += NEW_LINE;
body += "URI: " + DocumentInfo.URI + NEW_LINE;
body += "Source: " + DocumentInfo.Source + NEW_LINE;
body += "Concepts:" + NEW_LINE;
var concepts = new VArray(DocumentInfo.SummaryConcepts);
for (var I = concepts.lbound(); i <= concepts.ubound(); ++i) {
    body += "\t" + concepts.getItem(i) + NEW_LINE;
}
body += "Summary Sentences:" + NEW_LINE;
var summarySentences = new VArray(DocumentInfo.SummarySentences);
for (var I = summarySentences.lbound(); i <= summarySentences.ubound(); ++i) {
    body += "\t" + summarySentences.getItem(i) + NEW_LINE;
}
body += NEW_LINE;
body += "Document content:" + NEW_LINE;
body += NEW_LINE;

PostConversion.Text.SeekReadPointerInChars(0);
Body += PostConversion.Text.ReadString(PostConversion.Text.CharsCount);

// Create the message object.
var messageObject = new ActiveXObject("CDO.Message");
var messageConfig = messageObject.Configuration;

// Set the SMTP server.
var messageField =
messageConfig("http://schemas.microsoft.com/cdo/configuration/sendusing");
messageField.Value = 2; // CDO.CdoConfiguration.cdoSMTPServer
messageField =
messageConfig("http://schemas.microsoft.com/cdo/configuration/smtpserver");
messageField.Value = SMTP_SERVER;
messageConfig.Fields.Update();

// Set the message information and send it.
messageObject.TextBody = body;
messageObject.Subject = "A document was just indexed in the source '" +
DocumentInfo.Source + "'";
messageObject.From = FROM_EMAIL;
messageObject.To = TO_EMAIL;
messageObject.Send();
} catch (e) {
// Unable to send the email. Reject the document and report the error.
DocumentInfo.IsValid = false;
PostConversion.Trace("Error: " + e.description, 2);
}

```

(VBScript Version)

```

*****
' This postconversion script sample sends an email to notify a specific
' document has been indexed.
'
' Required parameters:
'
' - SMTPServer: The SMTP server to use to send the email.
'               Example: smtp.myserver.com
'
' - From:       The email address of the email sender.
'               Example:myself@mycompany.com
'
' - To:        The email addresses of the email recipients.
'               Example:someone@mycompany.com
*****

```

Option Explicit

On Error Resume Next

```

' Get the email settings from the script parameters.
Dim SMTP_SERVER: SMTP_SERVER = PostConversion.GetParameter("SMTPServer")
Dim FROM_EMAIL:  FROM_EMAIL  = PostConversion.GetParameter("From")
Dim TO_EMAIL:    TO_EMAIL    = PostConversion.GetParameter("To")

' Generate the email body.
Dim body: body = ""
body = body & "Document information:" & vbNewLine
body = body & vbNewLine
body = body & "URI: " & DocumentInfo.URI & vbNewLine
body = body & "Source: " & DocumentInfo.Source & vbNewLine
body = body & "Concepts: " & vbNewLine
Dim concept
For Each concept In DocumentInfo.SummaryConcepts
    body = body & vbTab & concept & vbNewLine
Next
body = body & "Summary Sentences:" & vbNewLine
Dim summarySentence
For Each summarySentence In DocumentInfo.SummarySentences
    body = body & vbTab & summarySentence & vbNewLine
Next
body = body & vbNewLine
body = body & "Document content:" & vbNewLine
body = body & vbNewLine

Call PostConversion.Text.SeekReadPointerInChars(0)
body = body & PostConversion.Text.ReadString(PostConversion.Text.CharsCount)

' Create the message object.
Dim messageObject: Set messageObject = CreateObject("CDO.Message")
Dim messageConfig: messageConfig = messageObject.Configuration

' Set the SMTP server.
Dim messageField: Set messageField =
messageConfig("http://schemas.microsoft.com/cdo/configuration/sendusing")

```

```

messageField.Value = 2 ' CDO.CdoConfiguration.cdoSMTPServer
Set messageField =
messageConfig("http://schemas.microsoft.com/cdo/configuration/smtpserver")
messageField.Value = SMTP_SERVER
message Config.Update

' Set the message information and send it.
messageObject.TextBody = body
messageObject.Subject = "A document was just indexed in the source '" &
DocumentInfo.Source & "'"
messageObject.From      = FROM_EMAIL
messageObject.To        = TO_EMAIL
messageObject.Send

If Err.Number <> 0 Then
  ' Unable to send the email. Reject the document and report the error.
  DocumentInfo.IsValid = False
  Call PostConversion.Trace("Error: " & Err.Description, 2)
End If

```

Sample 7 – Update Document Permissions (JScript Version)

```

// *****
// This postconversion script sample updates the document permissions.
//
// Required parameters :
//
// - UserName: The name of the user to add.
// - DomainName: The name of the domain for this user.
// *****

try {
  // Get the email settings from the script parameters.
  var username  = PostConversion.GetParameter("UserName");
  var domainName = PostConversion.GetParameter("DomainName");

  // Display initial permissions.
  DisplayPermissions("Initial");

  // Clear existing permissions.
  DocumentInfo.AllowedPermissions.Clear();
  DocumentInfo.DeniedPermissions.Clear();

  // Allow or deny the user.
  if (GetFileName(DocumentInfo.URI) == "AllowedDocument.txt") {
    // Allow this user.
    DocumentInfo.AllowedPermissions.AddWindowsNTPermission(username, domainName);
  } else if (GetFileName(DocumentInfo.URI) == "DeniedDocument.txt") {
    // Allow everyone except this user.
    DocumentInfo.AllowedPermissions.AddWindowsNTPermission("everyone", "");
    DocumentInfo.DeniedPermissions.AddWindowsNTPermission(username, domainName);
  }

  // Display final permissions.
  DisplayPermissions("Final");
} catch (e) {
  // Unable to update the document permissions.

```

```

    DocumentInfo.Is Valid = false;
    PostConversion.Trace("Error: " + e.description, 2);
  ]

// This method extracts the file name from a given URI.
Function GetFileName(URI)
{
  var fileName = URI;
  var pos = fileName.lastIndexOf("/");
  if (pos == -1) {
    pos = fileName.lastIndexOf("\\");
  }
  if (pos != -1) {
    filename = filename.substring(pos + 1, filename.length);
  }
  pos = fileName.indexOf("?")
  if (pos != -1) {
    fileName = FileName.substring(0, pos);
  }
  Return fileName;
}

// This method displays the document permissions.
function DisplayPermissions(message)
{
  // Get the current document allowed permissions.
  var allowedPermissionsString = "";
  for (var index = 1; index <= DocumentInfo.AllowedPermissions.Count; ++index) {
    var permission = DocumentInfo.AllowedPermissions.Item(index);
    allowedPermissionsString += permission.Type + ": " + permission.Details + " ";
  }
  PostConversion.Trace(GetFileName(DocumentInfo.URI) + " -> " +
    message + " AllowedPermissions: " + allowedPermissionsString, 1);

  // Get the current document denied permissions.
  var deniedPermissionsString = "";
  for (var index = 1; index <= DocumentInfo.DeniedPermissions.Count; ++index) {
    var permission = DocumentInfo.DeniedPermissions.Item(index);
    deniedPermissionsString += permission.Type + ": " + permission.Details + " ";
  }
  PostConversion.Trace(GetFileName(DocumentInfo.URI) + " -> " +
    message + " DeniedPermissions: " + deniedPermissionsString, 1);
}

```

(VBScript Version)

```

'*****
' This postconversion script sample updates the document permissions.
'
' Required parameters:
'
'   - UserName:   The name of the user to add.
'   - DomainName: The name of the domain for this user.
'*****

```

```

Option Explicit
On Error Resume Next

```

```

' Get the user and domain name from the script parameters.
Dim userName:  username  = PostConversion.GetParameter("UserName")
Dim domainName: domainName = PostConversion.GetParameter("DomainName")

' Display initial permissions.
Call DisplayPermissions("Initial")

' Clear existing permissions.
Call DocumentInfo.AllowedPermissions.Clear()
Call DocumentInfo.DeniedPermissions.Clear()

' Allow or deny the user.
If GetFileName(DocumentInfo.URI) = "AllowedDocument.txt" Then
    ' Allow this user.
    Call DocumentInfo.AllowedPermissions.AddWindowsNTPermission(userName, domainName)
ElseIf GetFileName(DocumentInfo.URI) = "DeniedDocment.txt" Then
    ' Allow everyone except this user.
    Call DocumentInfo.AllowedPermissions.AddWindowsNTPermission("everyone", "")
    Call DocumentInfo.DeniedPermissions.addWindowsNTPermission(userName, domainName)
End If

' Display final permissions.
Call DisplayPermissions("Final")

If Err.Number <> 0 Then
    ' Unable to update the document permissions.
    DocumentInfo.IsValid = False
    Call PostConversion.Trace("Error: " & Err.Description, 2)
End If

' This method extracts the file name from a given URI.
Function GetFileName(ByVal URI)
    Dim pos: pos = InStrRev(URI, "/")
    If pos = 0 Then
        pos = InStrRev(URI, "\")
    End If
    If pos = 0 Then
        GetFileName = URI
    Else
        GetFileName = Right(URI, Len(URI) - pos)
    End If
    pos = InStr(GetFileName, "?")
    If pos > 1 Then
        GetFileName = Left(GetFileName, pos - 1)
    End If
End Function

' This method displays the document permissions.
Sub DisplayPermissions(ByVal message)
    ' Get the current document allowed permissions.
    Dim allowedPermissionsString: allowedPermissionsString = ""
    Dim allowedPermission
    For Each allowedPermission In DocumentInfo.AllowedPermissions
        allowedPermissionsString = allowedPermissionsString & _

```

```

        allowedPermission.Type & ": " &
allowedPermission.Details& " "
    Next
    Call PostConversion.Trace(GetFileName(DocumentInfo.URI) & " -> " & _
        message & " AllowedPermissions: " & allowedPermissionsString,
1)

    ' Get the current document allowed permissions.
    Dim deniedPermissionsString: deniedPermissionsString = ""
    Dim deniedPermission
    For Each deniedPermission In DocumentInfo.DeniedPermissions
        deniedPermissionsString = deniedPermissionsString & _
            deniedPermission.Type & ": " & deniedPermission.Details &
" "
    Next
    Call PostConversion.Trace(GetFileName(DocumentInfo.URI) & " -> " & _
        message & " DeniedPermissions: " & deniedPermissionsString,
1)
End Sub

```