Converter API - CES 7.0

Converter API Home

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 converters, preconversion scripts and postconversion scripts.

Converters

CES native converters support sevaral document types (see Supported File Formats). Custom converters are .NET, VBScripts, or JScripts that convert a document from an unsupported format to a format for which CES has a native converter (see Custom Converters). This functionality enables CES to index documents from any format.

Preconversion and Postconversion Scripts

Conversion scripts are VBScripts or JScripts used to manipulate document content and metadata before (preconversion) or after (postconversion) conversion (see Conversion Scripts). For example, conversion scripts can be used to update the extracted key concepts of documents.

Two types of conversion scripts exist:

- Global scripts apply to all the sources indexed (see Applying a Global Conversion Script).
- Source scripts only apply to a single source (see Applying a Source Conversion Script).

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.

Space contributors

- François Dallaire (2 minutes ago)
- Alexandre Moreau (653 days ago)
- Kevin Sampson (750 days ago)
- Benoit Bernard (1612 days ago)
- Jean-Pierre Dery (1837 days ago)

Recent space activity

Postconversion Script Samples
2 minutes ago • updated by François Dallaire • view change

Setting a Custom Document Weight
Feb 15, 2017 • updated by Alexandre Moreau • view change

Postconversion Scripts
Dec 18, 2016 • updated by François Dallaire • view change

Preconversion Scripts
Dec 18, 2016 • updated by François Dallaire • view change

Conversion Scripts
Nov 10, 2016 • updated by Kevin Sampson • view change

Custom Converters

Custom converters are .NET, VBScripts, or JScripts that convert a document from an unsupported format to a format for which the Coveo Platform 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 a custom converter to convert the document from the `.xyz` for mat 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, custom 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.

Example:
An administrator that wants to remove unwanted HTML content from a Web page document can create a custom converter in order to analyze the documents to index and remove unwanted content from it before the document is converted.

Creating a Custom Converter

Creating a custom 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 Coveo index path (ex: `C:\CES7\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 a custom 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 (see ICustomConversion Interface)
- DocumentInfo Object (see IDocumentInfo Interface)
- Mutex Object (see ICGLMutex Interface)
- IFilterWrapper Object (see ICGLIFilterWrapper Interface)

Setting Up a Custom Converter

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

1. Add the custom converter to the CES configuration (see Adding an Open Converter).
2. Assign the custom converter to the desired document type. By doing so, CES uses the open converter to convert every document that belongs to this document type. You can also change the converter associated to a specific document type (see Modifying How CES Handles a Document Type).

Custom Converter Samples

The following lists the different open converter samples available:

- Modifying an HTML Document
  Demonstrates how to modify an HTML document. It removes the HTML content that is included between specific HTML tags.
- Converting Documents to the RTF Format
  Converts a document from an unsupported format (such as Windows Write documents) to the RTF format using the Microsoft Word COM objects.
- Converting Documents to the Microsoft Excel Format
  Converts a document from an unsupported format (such as old Microsoft Excel documents) to the Microsoft Excel format using the Microsoft Excel COM objects.
- Converting Documents to the PDF Format
  Converts a document from an unsupported format (such as PostScript documents) to the PDF format using the Ghostscripttool.
- Converting Documents Using the IFilterWrapper
  Converts a document using the `IFilterWrapper` interface.
- Converting Documents to the HTML Format
  Converts a document from an unsupported format (such as Web archive documents) to the HTML format using the Microsoft Internet Explorer COM objects.

Modifying an HTML Document

Custom Converter Sample #1 (JScript Version)

```
// ***************************************************************************
//  This custom converter sample explains how to modify an HTML document.
//  It removes the HTML content included between specific HTML tags.
// ***************************************************************************
// HTML tag name to exclude.
var START_TAG = "<!-- BEGIN NOINDEX -->";
var STOP_TAG  = "<!-- END NOINDEX -->";
// The HTML charset.
var CHARSET_CP1252 = 1
// This function removes unwanted sections from the HTML content.
function CleanHTML(htmlToClean, startTag, stopTag)
{
    if (htmlToClean.length > 0) {
        // 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;
}
try {
    if (CustomConversion.InputDocument != null &&
CustomConversion.InputDocument.BytesCount != 0) {
        // Extract the document HTML content.
        var documentContent =
CustomConversion.InputDocument.ReadByteString(CustomConversion.InputDocument.BytesCoun
t, 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);
}
```

Converting Documents to the RTF Format

Custom Converter Sample #2 (JScript Version)

```
// ************************************************************************
//  This open converter sample converts a document from an unsupported format
//  (as Windows Write documents) to the RTF format using the Microsoft Word 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 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); // wdFormatRTF
    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) {
}
```

Custom Converter Sample #2 (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 inputFileName: inputFileName = CustomConversion.InputDocument.TemporaryFileName
CustomConversion.InputDocument.SaveToFile(inputFileName)
' 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(inputFileName, False)
' Save the document as a RTF document.
Dim outputFileName: outputFileName = CustomConversion.OutputDocument.TemporaryFileName
Call wordDocument.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(inputFileName)
Call CustomConversion.DeleteFile(outputFileName)
```

Converting Documents to the Microsoft Excel Format

Custom Converter Sample #3 (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, -4143); // xlWorkbookNormal
    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) {
}
```

Custom Converter Sample #3 (VBScript 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.
'************************************************************************
Option Explicit
On Error Resume Next
' Save the document to convert to a temporary file.
Dim inputFileName: inputFileName = CustomConversion.InputDocument.TemporaryFileName
CustomConversion.InputDocument.SaveToFile(inputFileName)
' 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(inputFileName)
' Save the document as a Microsoft Excel document.
Dim outputFileName: outputFileName = CustomConversion.OutputDocument.TemporaryFileName
Call excelWorkbook.SaveAs(outputFileName, -4143) ' xlWorkbookNormal
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(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)
```

Converting Documents to the PDF Format

Custom Converter Sample #4 (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 vriables.
var shellObject = null;
try {
    shellObject = CustomConversion.GetGlobalVariable("shellObject");
} catch (e) {
    // Lock here to make sure to instanciate only one instance of the shell object.
    Mutex.AcquireLock("WScriptShellMutex");
    // Retry to get the shell object, it may have been instanciated 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 += " -q -sDEVICE=pdfwrite -dNOPAUSE -dBATCH -dNOPROMPT ";
    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) {
}
```

CustomConverter Sample #4 (VBScript 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
'************************************************************************
Option Explicit
On Error Resume Next
' Get the shell object from the script global vriables.
Dim shellObject: Set shellObject = CustomConversion.GetGlobalVariable("shellObject")
If Not IsObject(shellObject) Then
    ' Lock here to make sure to instanciate only one instance of the shell object.
    Call Mutex.AcquireLock("WScriptShellMutex")
    ' Retry to get the shell object, it may have been instanciated 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)
```

Converting Documents Using the IFilterWrapper

Custom Converter Sample #5 (JScript Version)

```
 // ************************************************************************
//  This open converter sample converts a document using the IFilterWrapper
//  interface.
//
//  Note that you do not need 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 VBArray(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) {
}
```

Custom Converter Sample #5 (VBScript Version)

```
'************************************************************************
' This open converter sample converts a document using the IFilterWrapper
' interface.
'
' Note that you don't need 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.
Dim 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)
```

Converting Documents to the HTML Format

Custom Converter Sample #6 (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("");
    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 exeption("Timeout");
        }
    }
    // Set the rendered HTML as the document content.
    var UTF8_CHARSET = 2;
    var htmlContent = ieApplication.Document.documentElement.outerHTML;
    CustomConversion.OutputDocument.WriteAsByteString(htmlContent, UTF8_CHARSET);
    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) {
}
```

Custom Converter Sample #6 (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("")
Call ieApplication.Navigate(inputFileName)
' Wait for the document to be fully loaded (from 3 to 6 seconds max).
Call CustomConversion.Sleep(3000)
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.
    DocumentInfo.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 conversion process that allows administrators to fully customize the way documents are indexed.

There are two types of conversion scripts:

- Preconversion scripts are executed before a conversion occurs (see Preconversion Scripts).

- Postconversion scripts are executed after a conversion occurs (see Postconversion Scripts).

It is possible to execute a conversion script for every document of a particular source (see Applying a Source Conversion Script) or of every source of the index (see Applying a 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.

Referring to Samples

- Preconversion Script Samples
- Postconversion Script Samples

Preconversion Scripts

Coveo Enterprise Search preconversion scripts are .NET, VBScripts, or JScripts executed before the conversion, but after the files have been crawled by the connector (see What Are the 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.

> **Note:**
> A user could reject a document according to its size.

Creating a Preconversion Script

Creating a preconversion 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 a good practice is to save the file in the Coveo index configuration path (ex: `C:\CES7\Config\Scripts\`) so if you back up or move your configuration folder, scripts will follow. It is possible to create a new script file from scratch or copy and paste a preconversion script sample and modify it if necessary.

Three supported scripting languages are enabled in order to create a preconversion script: .NET, 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 a preconversion script, it is possible to use the following scripting objects:

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

Setting Up 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 (see Adding a Preconversion Script).
- Select whether the preconversion script is executed on all the documents in every source or only in a specific source :
  - On all the documents in every source (see Applying a Global Conversion Script).
  - On all the documents in a specific source (see Applying a Source Conversion Script).

Preconversion Script Samples

The following lists the different preconversion script samples available:

- Adding New Document Metadata
  Adds two new document metadata.
- Updating the Printable and Clickable URIs
  Updates the document printable and clickable URI.
- Filtering Based on Size and Date
  Filters files depending on file size and date.
- Using a Conversion Script Parameter
  Demonstrates how to use a conversion script parameter. It adds a new document metadata using the script parameter values.
- Using a Conversion Script Global Variable
  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.
- Using the CES Administration API

Uses the *CES Administration API* to duplicate every document to convert in a separate source.

Adding New Document Metadata

Preconversion Sample #1 (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;
}
```

Preconversion Sample #1 (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
```

Filtering Based on Size and Date

Preconversion Sample #3 (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;
```

Preconversion Sample #3 (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
```

Updating the Printable and Clickable URIs

Preconversion Sample #2 (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 clickes on a
search result title).
DocumentInfo.ClickableURI = DocumentInfo.ClickableURI.replace("file://", "http://");
```

Preconversion Sample #2 (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 clickes on a
search result title).
DocumentInfo.ClickableURI = Replace(DocumentInfo.ClickableURI, "file://", "http://")
```

Using a Conversion Script Global Variable

Preconversion Sample #5 (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);
```

Preconversion Sample #5 (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 variables 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)
```

Using a Conversion Script Parameter

Preconversion Sample #4 (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);
}
```

Preconversion Sample #4 (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 '" &amp;
SCRIPT_PARAMETER_METADATA_NAME  &amp; "' is '" &amp; metadataName  &amp; "'.", 1)
    Call PreConversion.Trace("The value of the script parameter named '" &amp;
SCRIPT_PARAMETER_METADATA_VALUE &amp; "' is '" &amp; metadataValue &amp; "'.", 1)
Else
    ' Unable to get the script parameters. Reject the document and report the error.
    DocumentInfo.IsValid = False
    Call PreConversion.Trace("Error: " &amp; Err.Description, 2)
End If
```

Using the CES Administration API

Preconversion Sample #6 (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.RefreshURI(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();
}
```

Preconversion Sample #6 (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.
    DocumentInfo.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

Coveo Enterprise Search (CES) postconversion scripts are .NET, VBScripts, or JScripts executed after the conversion (see What Are the Conversion Phases?).

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.

Creating a Postconversion Script

Creating a postconversion 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 a good practice is to save the file in the Coveo index configuration path (ex: `C:\CES7\Config\Scripts\`) so if you back up or move your configuration folder, scripts will follow. It is possible to create a new script file from scratch or copy and paste a postconversion script sample and modify it if necessary.

Three supported scripting languages are enabled in order to create a postconversion script: .NET, 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 a postconversion script, it is possible to use the following scripting objects:

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

Setting Up 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 (see Adding a Postconversion Script).
- Select whether the postconversion script is executed on all the documents in every source or only in a specific source.
  - On all the documents in every source (see Applying a Global Conversion Script).
  - On all the documents in a specific source (see Applying a Source Conversion Script).

Postconversion Script Samples

The following lists the different postconversion script samples available:

- Adding New Metadata Named AllFieldValues
  Adds a new metadata named `AllFieldValues`. This new metadata contains the name, value and type of all document metadata.
- Updating Document Concepts and Summary Sentences
  Updates the document concepts and summary sentences.
- Setting a Custom Document Weight
  Sets a custom document weight based on user defined rules.
- Changing View as HTML Version of Document
  Changes the default View As HTML version of the document.
- Analyzing Text to Find Metadata
  Analyses the document extracted text to find metadata and these metadata names and values are added to the document.
- Sending Email Following Indexing of a Document
  This postconversion script sample sends an email to notify a specific document has been indexed.

Adding New Metadata Named AllFieldValues

Postconversion Sample #1 (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 = "<AllFieldValues>" + NEW_LINE;
// A collection of all field names.
var fieldNames = new VBArray(DocumentInfo.Fields());

// 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 + XmlEncode(fieldName)  + DOUBLE_QUOTE;
    allValues += " value=" + DOUBLE_QUOTE + XmlEncode(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);
function XmlEncode(value) {
    var ret = "";
    if (value != null) {
        var val = value + "";
        for (var i = 0; i < val.length; ++i) {
            var ch = val.charAt(i);
            if (ch != ' ' && !(ch >= 'a' && ch <= 'z') && !(ch >= 'A' && ch <= 'Z') &&
!(ch >= '0' && ch <= '9')) {
                ch = "&#" + ch.charCodeAt(0) + ";";
            }
            ret += ch;
        }
    }
    return ret;
}
```

Postconversion Sample #1 (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 metadata.
'************************************************************************
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 & XmlEncode(fieldName)  &
DOUBLE_QUOTE
    allValues = allValues & " value=" & DOUBLE_QUOTE & XmlEncode(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)
Function XmlEncode(ByVal sVal)
    sVal = CStr(sVal)
    Dim sReturn: sReturn = ""
    Dim i
    Dim oRE: Set oRE = New RegExp : oRE.Pattern = "[ a-zA-Z0-9]"
    For i = 1 To Len(sVal)
        Dim ch: ch = Mid(sVal, i, 1)
        If (Not oRE.Test(ch)) Then
            ch = "&#" & Asc(ch) & ";"
        End If
        sReturn = sReturn & ch
    Next
    XmlEncode = sReturn
End Function
```

Updating Document Concepts and Summary Sentences

Postconversion Sample #2 (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 VBArray(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("TopSummarySentence", 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();
}
```

Postconversion Sample #2 (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
```

Setting a Custom Document Weight

Postconversion Sample #3 (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;
}
```

Postconversion Sample #3 (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
       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
```

Changing View as HTML Version of Document

Postconversion Sample #4 (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);
```

Postconversion Sample #4 (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
```

Analyzing Text to Find Metadata

Postconversion Sample #5 (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 regExpression = new RegExp("(.*)=(.*)", "g");
// Loop over all the regular expression matches in the string.
var arr;
while ((arr = regExpression.exec(documentContent)) != null) {
    var metadataName  = RegExp.$1;
    var metadataValue = RegExp.$2;
    // Add to document's metadata.
    DocumentInfo.SetFieldValue(trim(metadataName), trim(metadataValue));
}
// This method trims the given string.
function trim(p_StringToTrim)
{
    return p_StringToTrim.replace(/^\s+/,'').replace(/\s+$/,'');
}
```

Postconversion Sample #5 (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 exmpression should scan all lines.
regExpression.Global = 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
```

Sending Email Following Indexing of a Document

Postconversion Sample #6 (JScript 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
// ************************************************************************

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 VBArray(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 VBArray(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);
}
```

Postconversion Sample #6 (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 summmarySentence
For Each summmarySentence In DocumentInfo.SummarySentences
    body = body & vbTab & summmarySentence & 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
messageConfig.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
```

Add reference in a .NET Converter

I explicitly wanted to test a custom converter as a .cs script on the customer's server before compiling a working DLL. Quicker that way.

I had to use System.Web.HttpUtility in the custom converter. CES didn`t like it because PluginHost did not load the System.Web assembly.

2/4/2013 11:14:00 AM | Error loading custom converter: Compilation error: k:\CES7\Scripts\xDocumentConverter.cs(29,29): error CS0103: The name 'HttpUtility' does not exist in the current context

I dug through the PluginHost code and found this gem:

var matchReference = new Regex(@"^\s" + Regex.Escape(p_CommentPrefix) + @"\s*AddReference:(.*)$", RegexOptions.Multiline | RegexOptions.IgnoreCase);*

I tried it:

```
// AddReference:System.Web.dll
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Coveo.CES.Interops.COMCoveoConvertersWrappers;
using System.Xml;
using System.Xml.Xsl;
using System.IO;
using System.Web;


namespace Fortinet.CustomConverters
{
    public class KnovaDocumentConverter :
Coveo.CES.DotNetConverterLoader.CustomConverter
    {
        public override void RunCustomConverter(CustomConversion p_CustomConversion,
DocumentInfo p_DocumentInfo)
        {
            string input =
p_CustomConversion.InputDocument.ReadString(p_CustomConversion.InputDocument.BytesCoun
t);
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(input);

            XslCompiledTransform xsl = new XslCompiledTransform();
            xsl.Load(p_CustomConversion.GetParameter("XslPath"));

            StringWriter writer = new StringWriter();

            xsl.Transform(doc, null, writer);

            string output = HttpUtility.HtmlDecode(writer.ToString());

            p_CustomConversion.OutputDocument.WriteString(output);
        }
    }
}
```

And it worked!