White Paper

infoRouter Integration

Documents, features, functionality and other information held within infoRouter can be accessed in a number of ways from external applications. The aim of this document is to outline these options, and highlight the merits of each approach.

NOTE: *No external integration may be created with the aim of bypassing the infoRouter licensing model. Where a specific integration is created the licensing model may be adapted if not already adequate ~ Please speak to your account manager.*

1. Simple Web browser access
2. Direct or generated URL access to individual documents or folders
3. Access to core infoRouter functionality programmatically via the provided Web Services API

Simple Web browser access

The linking ERP system can trigger the creation of a browser object or simply open a browser pointed at the infoRouter Server.

Once opened, the user will independently logon to infoRouter and have access to the infoRouter interface as they would by opening the browser manually, and navigating to the infoRouter server URL. The user will be controlled by the security model applied within infoRouter.

Merits: This is extremely simple and makes all features of infoRouter available to a user

Cons: Degree of integration limited

Direct or generated URL access to individual documents or folders

Documents and folders may be accessed directly by a URL. Below are some examples which will actually link to documents on our infoRouter demo website.

Whether a document can be viewed, or the contents of a folder listed is controlled by the security permissions of the user supplied.

Note: depending upon settings links may not work directly from this document, so copy the link into the browser to see behaviour.

User: **url_demo**

Pwd: **urldemo**

Public access document by direct URL

http://demo.inforouter.co.uk/InfoRouter/docs/Public/URL_DEMO_PUBLIC.pdf

This is a non-controlled document not requiring any login to view

Controlled access document by direct URL

http://demo.inforouter.co.uk/InfoRouter/docs/URL_IntegrationDemo/URL_DEMO_PRIVATE.pdf

This is a controlled document and you will be asked to login before you can see the document

DOCUMENT genetics
business automation software

Controlled access document by direct URL with embedded username and password

http://demo.inforouter.co.uk/InfoRouter/docs/URL_IntegrationDemo/URL_DEMO_PRIVATE.pdf?_UID=url_demo&_PWD=urldemo

This is a controlled document but with user ID and password supplied in the URL so you will not be asked to login before you can see the document.

Note: mind underscores in spaces,
/URL_DEMO_PRIVATE.pdf?_UID=url_demo&_PWD=urldemo

These URLs are to the full virtual path within infoRouter. Documents and folders within Formate are also allocated a unique identifier ~D1234, ~F5678 with the "D" signifying a document and "F" a folder. If Known this short form URL's can be used, so the above 3 links are also; The short cut URL is available from the properties page of a document or folder.

http://demo.inforouter.co.uk/InfoRouter/docs/~D1171

http://demo.inforouter.co.uk/InfoRouter/docs/~D1172

http://demo.inforouter.co.uk/InfoRouter/docs/~D1172?_UID=url_demo&_PWD=urldemo

Similarly for folders

Controlled access folder by direct URL

http://demo.inforouter.co.uk/InfoRouter/docs/URL_IntegrationDemo

This is a controlled folder and you will be asked to login before you can see the document. Access to a public folder would be immediate.

Controlled access folder by direct URL with embedded username and password

http://demo.inforouter.co.uk/InfoRouter/docs/URL_IntegrationDemo?_UID=url_demo&_PWD=urldemo

This is a controlled folder but with user ID and password supplied in the URL so you will not be asked to login before you can see the contents of the folder.

Again the URL shortcut can be used

http://demo.inforouter.co.uk/InfoRouter/docs/~F1498

http://demo.inforouter.co.uk/InfoRouter/docs/~F1498?_UID=url_demo&_PWD=urldemo

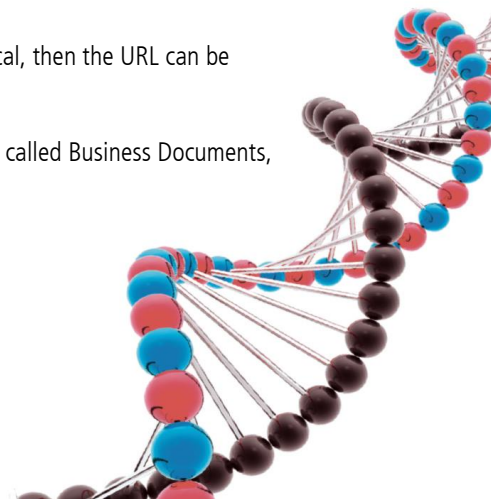
Thus direct URLs for individual documents or department, team or project folders could be stored within the ERP system either in the full or short forms.

The Clever Bit!

The beauty of this approach is that if the infoRouter folder path and document name are logical, then the URL can be generated on the fly. For example

An invoice (Invoice #12345), for Smith & Sons (Account #009999). If this is stored in a library called Business Documents, with the invoice titled Inv_12345.pdf, then the URL can be generated as (dummy link):

http://servername/inforouter/docs/Business Documents/009999/Inv_12345.pdf



The elements in green are fixed, with only the account and invoice numbers (in red) being variables supplied by the ERP system. In a similar manner the ERP system could access the user details of the ERP user to embed the infoRouter credentials with the URL as well.

Merits: enables very simple direct viewing of documents, or access to department, team, or project folders.

Cons: Direct viewing of a document is effectively read-only and prevents access to normal infoRouter functionality for that document.

Access to core infoRouter functionality programmatically via the provided Web Services API

infoRouter exposes a series of Web Service Calls allowing a subset of functionality to be driven programmatically from any language that can interface with a Microsoft .net Web service as exposed by infoRouter.

The API is free to use and documented at http://www.inforouter.com/web-services-80/default.aspx

In addition there is a simple MS visual studio project available to demonstrate a basic integration.

Or individual Service Calls information and a test facility are available directly from the infoRouter server.

http://[servername]/inforouter/srv.asmx

infoRouter is Web service driven and the add-ins, and plug-ins supplied with the product such as the Office plug-in, Scan Station, Email Scanner and Hot Folders services all use these web services to connect to infoRouter.

No specific programming support is available for the API, beyond the normal bug reporting procedures.

Currently there are 185 Web Service calls available, and this number is growing steadily. They provide access to a number of core features and functions of infoRouter, such as metadata search, and result retrieval, document upload, and download (including multiple files as a zip), plus library, folder, and user creation.

The basic workflow of any integration is to:

Create an infoRouter Server object

Authenticate a user against the server object to establish the security context (i.e. to set what files and folders are accessible and what actions are allowed for this user)

If authentication is successful then any number of the other available web services calls may be made as required.)

Note: the authentication ticket generated when successfully authenticating a user has a limited lifetime of 20 minutes.

Below are two VB.net Code samples by way of an introduction

Example 1; Search for, and download a document if found

The code below, authenticates a user and then commits a search (in this case for a fixed document) Then it checks to see if there is a single document in the results, and if so downloads it to the local drive.

```
Dim IRServername As String = "localhost" 'Address of Inforouter Server

Dim AuthenticationTicket As String = ""
Dim infoRouterServerAddress As String
Dim xmlresponse As Xml.XmlElement

Dim IR_OBJ As Inforoutertest.localhost.srv = New Inforoutertest.localhost.srv
infoRouterServerAddress = "http://" & IRServername & "/inforouter/srv.asmx"

IR_OBJ.Url = (infoRouterServerAddress)

'Authenticate User
xmlresponse = IR_OBJ.AuthenticateUser("IRuser", "IRuser")

If xmlresponse.GetAttribute("success") = "true" Then
  AuthenticationTicket = xmlresponse.GetAttribute("ticket")
Else
End If

Dim DocPath As String = ""
Dim DocID As String = ""
Dim f As String

Try
   Application.DoEvents()

'Fixed string for example searching for Invoice # CN17529152 in Account # 313474

   f = "<search><ITEM NAME = ""SEARCHFOR"" VALUE = ""DOCUMENTSONLY""></ITEM><ITEM NAME=""PROPERTYSETNAME""
VALUE=""INVOICE""><ATTR NAME=""INVNO"" OPERATOR=""EQ"" VALUE=""CN17529152"" /><ATTR NAME=""ACCOUNT""
OPERATOR=""EQ"" VALUE=""313474"" /></ITEM></search>"

   xmlresponse = IR_OBJ.Search(AuthenticationTicket, f, "DOCUMENTNAME", True)

   If xmlresponse.GetAttribute("success") = "true" Then
     MsgBox("The search prepared successfully.")
   Else
     MsgBox("The search cannot be prepared.")
     MsgBox("server response:" & xmlresponse.GetAttribute("error"))
   End If
    xmlresponse = Nothin

xmlresponse = IR_OBJ.GetNextSearchPage(AuthenticationTicket, False, False, False, False, False)


   If xmlresponse.GetAttribute("success") = "true" Then
     Dim g As String
          g = xmlresponse.GetAttribute("to") & "/" & xmlresponse.GetAttribute("from") & "/" &_
          xmlresponse.GetAttribute("FirstPage") & "/" & xmlresponse.GetAttribute("LastPage")
     Dim xmlitem As System.Xml.XmlElement

  If g <> "1/1/TRUE/TRUE" Then 'Check for single entry in search list

    For Each xmlitem In xmlresponse
      If xmlitem.Name = "document" Then
        DocID = xmlitem.GetAttributeNode("DocumentID").Value
       DocPath = xmlitem.GetAttributeNode("Path").Value & "\" & xmlitem.GetAttributeNode("Name").Value

       End If
     Next
    Else
```


```
            MsgBox("More than one result for search")
    End If

    Else

        If xmlresponse.GetAttribute("error").ToLower = "no search available." Then
            MsgBox("Document Not Available")
        Else
            MsgBox("server response:" & xmlresponse.GetAttribute("error"))
        End If

     End If
     xmlresponse = Nothing

If DocPath <> "" And DocID <> "" Then

   xmlresponse = IR_OBJ.DocumentExists(AuthenticationTicket, DocPath)
   If xmlresponse.GetAttribute("success") = "true" Then

'Temp path for download (will overwrite existing file of same name)

       Const DownloadPath As String = "c:\temp\"

       Try

        Dim FileName As String = System.IO.Path.GetFileName(DocPath)
        'call downloaddocument method to get bytearray
        Dim buffer As Byte() = IR_OBJ.DownloadDocument(AuthenticationTicket, DocPath)
         If IsArray(buffer) Then
             Dim fs As System.IO.FileStream = New System.IO.FileStream(DownloadPath & FileName,_
            System.IO.FileMode.Create, IO.FileAccess.Write)
            Dim bw As System.IO.BinaryWriter = New system.IO.BinaryWriter(fs)
            'write bytearray to the local path
            bw.Write(buffer)
            bw.Close()
            fs.Close()
            bw = Nothing
            fs = Nothing

    Else
            Dim xml_response As System.Xml.XmlNode = IR_OBJ.GetDownloadInfo(AuthenticationTicket, DocPath)
            If Not (xml_response.Attributes("success").Value.ToUpperInvariant() = "TRUE") Then
             Console.WriteLine("Server Response :" & ml_response.Attributes("error").Value)
    End If
                xml_response = Nothing
     End If

Catch EX As Exception

            Console.WriteLine("error:" & EX.Message)
   End Try

   Else

        MsgBox("server response:" & xmlresponse.GetAttribute("error"))
    End If
   End If

Catch ex As Exception
        MsgBox("error:" & ex.Message)
Finally
        IR_OBJ = Nothing
End Try
```

Example 2; Listing folders and documents

The code below, uses the getfoldersanddocuments1 api call to list all documents and folders in the "invoice" library or root folder if you prefer. The response is a list of folders. You would then need to recurse through these folders identifying subfolders in each folder, and then recurse again to get documents in each subfolder, as the getfodlersanddocuments1() only returns a list at the specified level not everything below it.

```
Dim IRServername As String = "localhost"
Dim AuthenticationTicket As String = ""
Dim infoRouterServerAddress As String
Dim xmlresponse As Xml.XmlElement


Dim IR_OBJ As Inforoutertest.localhost.srv = New Inforoutertest.localhost.srv
infoRouterServerAddress = "http://" & IRServername & "/inforouter/srv.asmx"

IR_OBJ.Url = (infoRouterServerAddress)

'Authenticate User
xmlresponse = IR_OBJ.AuthenticateUser("IRuser", "IRuser")

If xmlresponse.GetAttribute("success") = "true" Then
  AuthenticationTicket = xmlresponse.GetAttribute("ticket")

  xmlresponse = IR_OBJ.GetFoldersAndDocuments1(AuthenticationTicket, "/invoice")

      If xmlresponse.GetAttribute("success") = "true" Then
        Dim xmlitem As System.Xml.XmlElement
        Dim gg As String
        Dim gf As String
        Dim fg As Int32

        MsgBox(xmlresponse.OuterXml)

        For Each xmlitem In xmlresponse

          gg = xmlitem.GetAttribute("n")
          gf = gf & vbCrLf & gg

        Next

        MsgBox(gf)
      Else
        MsgBox(xmlresponse.OuterXml.ToString)
      End If

    Else

End If
Label1.Text = ""
Me.Refresh()

xmlresponse = Nothing
```

Contact Document Genetics

Document Genetics is an established UK based company providing a comprehensive range of business automation software. We focus on improving document automation, workflow and collaboration within our client organisations, and our range of innovative solutions and specialist services help to save time and money by processing documents and data more efficiently. If you'd like to discuss your supplier invoice application with Document Genetics, we'll be delighted to help.

Author - © Phil Castle, Technical Director at Document Genetics

Hall Farm, Sywell Aerodrome, Sywell, Northampton NN6 0BN

t: 01604 671177 f: 0844 557 6431

e: info@document-genetics.co.uk w: www.document-genetics.co.uk

DOCUMENT genetics
business automation software