Article
Eduard Lebedyuk · May 21, 2018 10m read

Adding your own provider to MFT

Managed File Transfer (MFT) feature of InterSystems IRIS enables easy inclusion of a third-party file transfer service directly into an InterSystems IRIS production. Currently, DropBox, Box, and Kiteworks cloud disks are available.

In this article, I'd like to describe how to add more cloud storage platforms.

Here's what we're going to talk about:

- What is MFT
- Reference: Dropbox
    - Connection
    - Interoperability
    - Direct access
- Interfaces you need to implement
    - Connection
    - Logic
- Installation

What is MFT

MFT provides bidirectional file transfer, you can both download files from a cloud storage and upload files there.

Check out this video on Learning.InterSystems.com for an introduction to MFT.

Reference: Dropbox

Before we start writing our own MFT adapter, let's run an existing one. I chose Dropbox.

Basic access

First of all, we need to configure Dropbox access:

1. Register new Dropbox account.
2. Create new Dropbox App.
    - Remember your: `App Key` and `App Secret`
    - Set your Redirect URL, probably:
      `http://localhost:57772/csp/sys/oauth2/OAuth2.Response.cls`
3. Create a new SSL Configuration (if you don't have one)
4. Create a new MFT configuration at:
   `http://localhost:57772/csp/sys/sec/%25CSP.UI.Portal.MFT.Connection.zen?i...`
    - Remove "Use SSL" flag (unless your web server is available over https).
    - Email address field should match your Dropbox email address.
5. Press `Get Access Token` button (on MFT connections list page) to authorize your application.
6. If everything went alright, the status of the new MFT connection should be Authorized.

Interoperability

Next, we need to configure Interoperability production to use the new connection

To receive files from Dropbox into a local directory:

1. Create `EnsLib.MFT.Service.Passthrough` service.
    - MFT Connection Name: Dropbox
    - MFT Source Folders: the path to Dropbox folder, from which the files would be downloaded (i.e. `/inbox/`)
    - Target Config Names: BP or BO to send files to, for example, `EnsLib.File.PassthroughOperation`
2. Create `EnsLib.File.PassthroughOperation` operation.
    - File Path: the path to a local directory to which the files from Dropbox would be downloaded (i.e. `C:\\temp\in\`)
    - Charset: Binary
    - File Name: %f

That's all. After starting the production, files from the specified Dropbox folder would be downloaded into the local directory.

To upload files to Dropbox, the process should be done in reverse:

1. Create `EnsLib.MFT.Operation.Passthrough` service.
    - MFT Connection Name: Dropbox
    - Default MFT Folder: the path to Dropbox folder, into which the files would be uploaded (i.e. `/sent/`)
2. Create `EnsLib.File.PassthroughService` operation.
    - File Path: the path to a local directory from which the files would be uploaded to Dropbox (i.e. `C:\\temp\out\`)
    - Target Config Names: BP or BO to send files to, in our case `EnsLib.MFT.Operation.Passthrough`
    - Charset: Binary

And with this, files placed into `C:\\temp\out\` folder would be uploaded into `/sent/` Dropbox folder.

For a more comprehensive guide refer to MFT First look.

"Direct" access

Interoperability BS/BO are wrappers over a direct access. When writing your own adapter it's better to use direct access to check if things work as expected. Here's a sample method to get file information

```
ClassMethod dropboxInfo(file = "/1.txt")
{
    // Establishing Dropbox connection
    set mftConnectionName = "Dropbox"
    set mftConnection = ##class(%MFT.API).GetConnection(mftConnectionName, .sc)
    write:$$$ISERR(sc) $System.Status.GetErrorText(sc)

    // Getting information about file
    // Some other methods: GetFileInfo GetFolderInfo CreateFolder DeleteFile GetUser
ShareFolder UnshareFolder DownloadStream UploadFile
    set sc = $classmethod($$$EnsCoreMFTAPIClass,"GetFileInfo", mftConnection, file, .
itemInfo)
```

```
    write:$$$ISERR(sc) $System.Status.GetErrorText(sc)

    // Displaying information about file
    #dim itemInfo As %MFT.ItemInfo
    zw itemInfo
}
```

MFT interface

MFT provider consists of two parts. To create your own provider you need to implement them.

Technical API (connection)

Connection is responsible for sending requests and parsing results. It should extend `%SYS.MFT.Connection.Base` and implement the following methods:

- DefaultURL that returns the root of service API
- CreateClient that creates OAuth client
- RevokeToken to log out of the application

Other methods could be overloaded as required bit these three are a must. Additionally, for Yandex I created these three methods:

- MakeRequest - to execute normal API requests
- MakeDownloadRequest - to download files
- MakeUploadRequest - to upload files

These methods structure are completely dependent on target API architecture. In my case, a majority of requests were similar, with different requests required for downloading and uploading files so I ended up with three methods.

Logical API

Is responsible for consuming the API, it uses the connection to execute requests and it should extend `%MFT.API` and reside in the `%MFT` package (we'll talk about working around that requirement later). The methods it must overload could be separated into four categories:

Info	Create/Delete	Sharing	Load
• GetFileInfo • GetFolderInfo • GetFolderContents	• CreateFolder • DeleteFolder • DeleteFile	• ShareFolder • UnshareFolder • UnshareFolderAll	• UploadStream • DownloadStream

These methods seem self-explanatory. But please ask in comments if the purpose of these methods seems unclear. There's also documentation in %MFT.API class. I recommend implementing these methods left to right, you can also skip Sharing methods. That brings us to 8 methods that must be implemented.

Additionally, if your cloud disk provider has user management (i.e. teams and so on), you can manage that too via:

- GetUser
- GetUserById
- GetUserList

- CreateUser
- DeleteUser
- DeleteUserById

As Yandex.Disk does not have user management I skipped these methods (they should still be implemented, but left effectively empty).

Finally **GetRequestId** method should be implemented to convert paths into ids (that could be faster), otherwise, it should return path.

Along with some other methods that are about the same for all providers here's a code snippet you can start writing your adapter with:

```
/// Get the form of id for a file or folder that is most efficient for subsequent cal
ls.
/// GetRequestId will return either an id of the form "id:<id>" or a full path depend
ing on which is more efficient.
/// This method is included to allow the id for future requests to be saved in the mo
st efficient form.
ClassMethod GetRequestId(connection As %SYS.MFT.Connection.Base, itemInfo As %MFT.Ite
mInfo) As %String
{
    Quit itemInfo.GetPath()
}

/// Retrieve the %MFT.UserInfo for current user
ClassMethod GetUser(connection As %SYS.MFT.Connection.Base, username As %String, Outp
ut userInfo As %MFT.UserInfo) As %Status
{
    Set userInfo = ##class(%MFT.UserInfo).%New()
    Set userInfo.Username = connection.Username
    Quit sc
}

/// Retrieve the %MFT.UserInfo specified by the service defined Userid.
/// If the user does not exist, then $$$OK is returned as status and userInfo is retu
rned as "".
ClassMethod GetUserById(connection As %SYS.MFT.Connection.Base, userid As %String, Ou
tput userInfo As %MFT.UserInfo) As %Status
{
    Quit ..GetUser(connection, userid, .userInfo)
}

/// Return the list of all currently defined users for this team or enterprise.
ClassMethod GetUserList(connection As %SYS.MFT.Connection.Base, Output userList As %M
FT.UserList) As %Status
{
    Set sc = $$$OK
    Set userList = ##class(%MFT.UserList).%New()
    Set sc = ..GetUser(connection, "", .userInfo)
    Quit:$$$ISERR(sc) sc

    Do userList.Users.Insert(userInfo)

    Quit sc
}

/// Create a new user.
```

```
/// Unable to do it in Yandex
ClassMethod CreateUser(connection As %SYS.MFT.Connection.Base, userInfo As %MFT.UserI
nfo) As %Status
{
    Quit $$$ERROR($$$MFTBadConnection)
}

/// Delete new user.
/// Unable to do it in Yandex
ClassMethod DeleteUser(connection As %SYS.MFT.Connection.Base, username As %String) A
s %Status
{
    Quit $$$ERROR($$$MFTBadConnection)
}

/// Delete the user that is specified by the id.
ClassMethod DeleteUserById(connection As %SYS.MFT.Connection.Base, userid As %String)
 As %Status
{
    Quit $$$ERROR($$$MFTBadConnection)
}

/// Unshare a folder from everyone, user is ignored
ClassMethod UnshareFolder(connection As %SYS.MFT.Connection.Base, path As %String, us
er As %String) As %Status
{
    Quit ..UnshareFolderAll(connection, path)
}

/// MountFolder is a Dropbox specific method to mount a shared folder that was shared
 by a different user.
/// MountFolder is treated as a NOP for all other services.
ClassMethod MountFolder(connection As %SYS.MFT.Connection.Box, folderName As %String)
 As %Status
{
    // A NOP if not Dropbox
    Quit $$$OK
}

/// UnmountFolder is a Dropbox specific method to unmount a shared folder that was sh
ared by a different user.
/// UnmountFolder is treated as a NOP for all other services.
ClassMethod UnmountFolder(connection As %SYS.MFT.Connection.Box, folderName As %Strin
g) As %Status
{
    // A NOP if not Dropbox
    Quit $$$OK
}

/// Update the specified remote file with the contents of the specified local file.
/// filePath must be a file path.  An id may not be specified.
/// If replace is true, then an existing file of the same name will be replaced.
/// The default is to return an error if a replacement is attempted.
ClassMethod UploadFile(connection As %SYS.MFT.Connection.Base, localFilePath As %Stri
ng, filePath As %String, replace As %Boolean = 0, Output itemInfo As %MFT.ItemInfo) A
s %Status
{
    Set stream=##class(%FileBinaryStream).%New()
    Set stream.Filename=localFilePath
```

```
    Quit ..UploadStream(.connection,stream,filePath,replace,.itemInfo)
}

/// Download the specified remote file and store at the location given by localFilePa
th.
/// filePath may be a file path.
ClassMethod DownloadFile(connection As %SYS.MFT.Connection.Base, filePath As %String,
 localFilePath As %String) As %Status
{
    Set stream=##class(%FileBinaryStream).%New()
    Set stream.Filename=localFilePath
    Quit ..DownloadStream(.connection,filePath,stream)
}
```

Additional data

Some other classes are used to transmit information, they are:

- `%MFT.ItemInfo` is a detailed description of a file or folder - you'll need it.
- Other classes are %MFT.UserInfo for user information and %MFT.FolderContents/%MFT.UserList to list several items or users respectively.

Note, that these classes, despite being % classes, store the data in the namespace they are called from. They are storing data in ^MFT.* globals.

Installation

As our API should be in a %MFT package we would use mapping to map code from the specialized database into our target namespace (or %SYS). That way InterSystems IRIS can be safely updated and our work wouldn't be overwritten. An Interesting thing about it is that we need to load classes only into target namespace - %SYS in our case.

First of all, we need to download our code, to do that:

1. Download and import Installer into any Interoperability-enabled namespace.
2. Execute: `write $System.Status.GetErrorText(##class(MFT.Installer).Install())`

It would:

- Create `MFTLIB` database
- Add mapping of `%MFT.Addons` and `%SYS.MFT.Connection.Addons` packages into %SYS namespace from `MFTLIB` database
- Download the rest of the code from GitHub, correctly importing % and non % classes

Next, we need to configure Yandex application:

1. Register on Yandex.
2. Create Yandex App
    - Check ???-???????
    - Set Redirect URI: `http://Host:Port/csp/sys/oauth2/OAuth2.Response.cls` (https, if UseSSL = 1, for development you can set it to `http://localhost:57772/csp/sys/oauth2/OAuth2.Response.cls`)

    - Give disk access `??????.???? REST API`
    - Get `ID`, `Pass`
3. Execute: `write $System.Status.GetErrorText(##class(MFT.Yandex).Install(Login, ID, Pass, Host, Port, UseSSL))`
    - Login - your Yandex email
    - Host, Port - same as the callback
    - UseSSL - use SSL for callback? Your server needs to support https
4. Open http://Host:Port/csp/sys/sec/%25CSP.UI.Portal.MFT.ConnectionList.zen
5. Press `Get Access Token` and complete authorization.
6. If everything went fine the Status would be Authorized.
7. Execute: `write $System.Status.GetErrorText(##class(MFT.Yandex).ConfigureProduction(yandexSource, fileDestination, fileSource, yandexDestination))`
    - `yandexSource` and `fileDestination` - Yandex.Disk folder to download files from, they are stored in a local destination folder.
    - `fileSource` and `yandexDestination` - local folder from which files are uploaded to Yandex.Disk.
    - Important: Yandex.Disk folder names should end with `/` (i.e. `out` in a disk root would be `/out/`)
8. Open production `MFT.Production` and start it.
9. Add file(s) to `yandexSource` and `fileSource` to see how it works.

Note, that unlike Dropbox, I'm creating service automatically, like this:

```
ClassMethod Install(username As %String, clientId As %String, clientSecret As %String
, host As %String = "localhost", port As %Integer = {$get(^%SYS("WebServer","Port"),
57772)}, useSSL As %Boolean = {$$$NO})
{
    New $Namespace
    Set $Namespace = "%SYS"

     Do:'##class(Security.SSLConfigs).Exists(..#SSLConfig) ##class(Security.SSLConfig
s).Create(..#SSLConfig)

    Set sys = ##class(%SYS.MFT.Connection.Addons.Yandex).%New()
    Set sys.Name = "Yandex"
    Set sys.Service = "Addons.Yandex"
    Set sys.ApplicationName = "Yandex"
    Set sys.SSLConfiguration = ..#SSLConfig
    Set sys.Username = username
    Set sys.URL = ..#URL

    $$$QuitOnError(##class(%SYS.MFT.Connection.Addons.Yandex).CreateClient(sys.Name,
..#SSLConfig, clientId, clientSecret, ,host, port,,useSSL))

    Quit sys.%Save()
}
```

Note the Service value: `Addons.Yandex`. Service name, when appended to `%MFT.` should yield the name of the logical API class. Since we can't modify `%MFT` package directly we can add a subpackage to it.

Conclusion

MFT is a useful technology and can be easily extended to support cloud providers you need.

Links

- Repository
- Yandex.Disk REST API reference
- MFT First look

#Best Practices #Interoperability #InterSystems IRIS

Source URL:https://community.intersystems.com/post/adding-your-own-provider-mft