Enabling Productions to Use Managed File Transfer Services

Version 2023.1
2024-07-11

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Enabling Productions to Use Managed File Transfer Services
InterSystems IRIS Data Platform Version 2023.1 2024-07-11

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

1

Introduction to MFT Services

This topic introduces Managed File Transfer (MFT) services and explains how an InterSystems IRIS® production can communicate with these services, so that the production can send and receive files securely.

1.1 Overview of MFT Services

Managed File Transfer (MFT) services are third-party services that support secure transmission of files. These services generally include the following features:

- Use of encrypted channels for the actual data transmission.
- Scalability and high availability.
- Ability to control and monitor information about senders and recipients. Only authenticated and authorized users or applications can send and receive files, and all transmissions are fully logged for auditing purposes.
- Ability to easily integrate with other applications, so that files can be sent directly between applications.
- Server backup. End users do not have to manage backups.

1.2 InterSystems IRIS and MFT Services

InterSystems IRIS provides support for using MFT services directly from a production, so that the production can receive files from and send files to an MFT service. Specifically, InterSystems IRIS supports the following MFT services:

- Box – http://www.box.com
- Dropbox – http://www.dropbox.com
- Accellion kiteworks – http://www.kiteworks.com

Integration with an MFT service causes the production message log to keep an audit trail of file transmissions, so that there is never any question about the sender, recipient, or contents. This ability is useful when the production needs to route or process sensitive files.

The InterSystems IRIS support for MFT is particularly useful for secure communication between a large organization (using an InterSystems IRIS production) and small facilities that may have less technical staff.

1.3 Support for MFT in InterSystems IRIS

Formally, the support for MFT in InterSystems IRIS consists of the following items:

- Business hosts for sending files to and receiving files from the supported MFT services:
  - The MFT business service class, EnsLib.MFT.Service.Passthrough, for receiving files from an MFT service
  - The MFT business operation class, EnsLib.MFT.Adapter.Outbound, for sending files to an MFT service
- A web interface for configuring a secure, authorized connection to an MFT service. The InterSystems IRIS Management Portal provides a way to create and save reusable connections to MFT services, for use by productions in that InterSystems IRIS server.
- An API for managing resources within an account at an MFT service. This API includes a complete set of methods for managing users, files, and directories within the account.

The following subsections describe how the business hosts classes work.

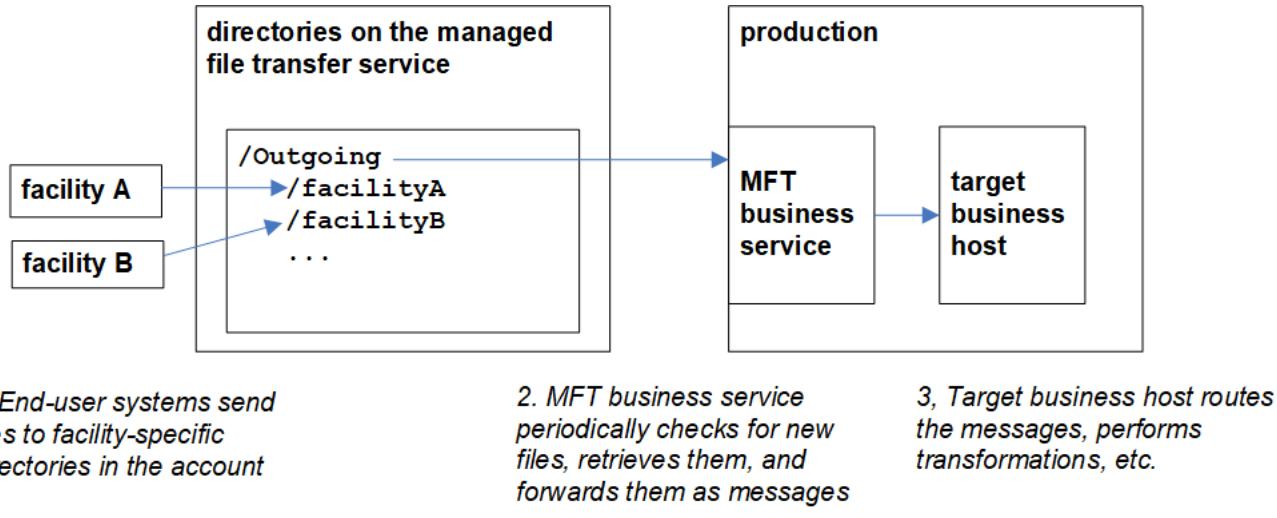
1.3.1 Overall Behavior of the MFT Business Service Class

The MFT business service class, EnsLib.MFT.Service.Passthrough, provides runtime settings that you use to specify items *including* the following:

- Connection information that enables the business service to make an authorized connection to a specific account at an MFT service
- Location of the directory where the service should look for files
- A regular expression that identifies the filename to look for
- A polling interval, which controls how frequently the service checks for new input
- The name of one or more target business hosts within the production

The MFT business service periodically checks the given directory, and then iterates through the matching files it finds. For each file, the business service creates a message (an instance of Ens.MFT.StreamContainer) that contains the file contents and then sends that message to the configured target elsewhere in the production.

The following figure shows the overall flow:

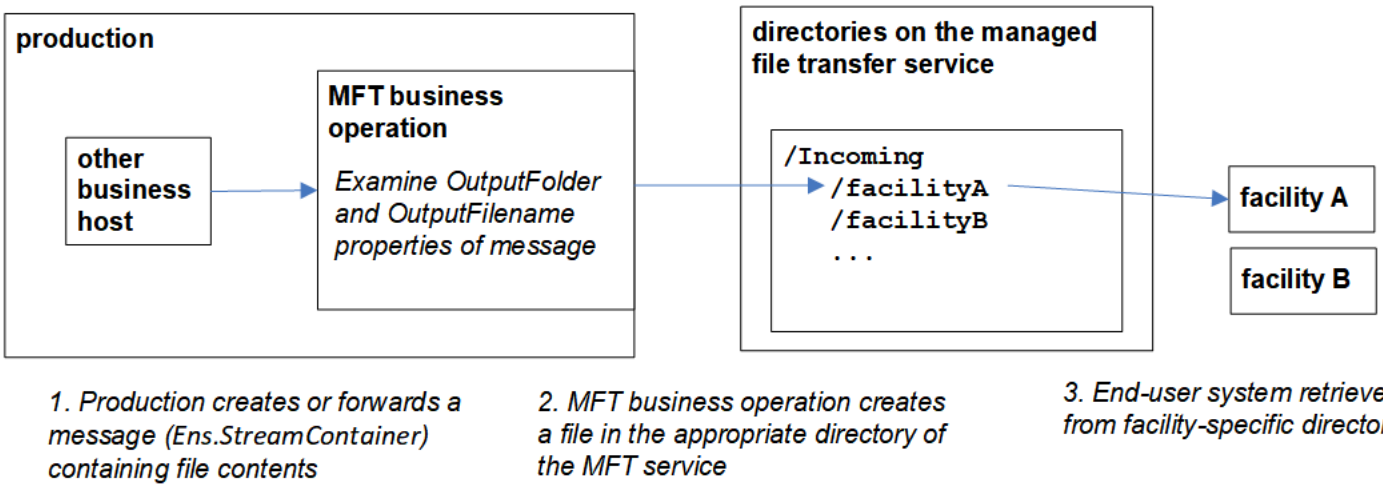


1.3.2 Overall Behavior of the MFT Business Service Operation

The MFT business operation class, EnsLib.MFT.Adapter.Outbound, provides runtime settings that you use to specify items *including* the following:

- Connection information that enables the business service to make an authorized connection to a specific account at an MFT service
- Directory within that account where the operation should place files
- A filename specification that describes the format of generated filenames

When an MFT business operation receives a message (an instance of Ens.StreamContainer or any subclass of that class), it creates a file and writes that to the specified directory within the MFT account. The following shows a sketch of the process:



2

Prerequisites and Configuration for MFT

An InterSystems IRIS® production can include business hosts that communicate directly with Managed File Transfer (MFT) services. Before adding these business hosts, perform the following prerequisites:

- Preparing the MFT service account
- Creating configuration items in InterSystems IRIS

InterSystems IRIS supports the following MFT services:

- Box – http://www.box.com
- Dropbox – http://www.dropbox.com
- Accellion kiteworks – http://www.kiteworks.com

2.1 Preparing the MFT Service Account

Before you can use an MFT service with InterSystems IRIS, you must perform the following tasks:

- Create an account and subaccounts
- Create a custom application for use by the production
- Set up a directory structure for incoming and outgoing files

The following subsections provide details.

2.1.1 Setting Up Accounts

For each MFT service you intend to use, you must create two types of accounts:

- One main administrative account, which manages all users and all directories
- Multiple subaccounts, as needed for the end users

An end user account is needed for each person (or organization) expected to send and receive files. These end users can access only the files in their own folders or in folders to which they have explicitly been granted access, either by the actual owner or by an administrator. Consult the documentation for the MFT service for instructions on how to create these sub-accounts, and how to set and verify permissions.

When you create the main account, make a note of the root URL that is meant for use in transferring files (for this account). You will need this information later (to use as the **Base URL** for the connection to the account).

Also choose an administrator for the account and make a note of the email address of that person.

2.1.2 Creating a Custom Application

After creating accounts at the MFT service, you must create a custom application within the main account, for use by the InterSystems IRIS production. Within this custom application, specify the following details:

- A unique name.
- The redirect URL for the InterSystems IRIS server. This URL has the following form:

  ```
  http://hostname:port/prefix/csp/sys/oauth2/OAuth2.Response.cls
  ```

  Where:

  - **hostname** is the fully qualified domain name (FQDN) or IP address of the InterSystems IRIS server on which the production is running.
  - **port** is the web server port specified in the Web Gateway configuration, if any. If the URL does not include a port number, there is no colon after the host name.
  - **prefix** is an optional prefix if needed by the Web Gateway configuration.

  See Creating a Managed File Transfer Connection.
- A pair of generated values (an OAuth 2.0 client ID/client secret pair), for use in authorizing the connection. The MFT services each use different names for these values:

MFT Service	Name for the OAuth 2.0 client ID	Name for the OAuth 2.0 client secret
Box	**Client ID**	**Client Secret**
Dropbox	**App key**	**App secret**
kiteworks	**Client Application ID**	**Client Secret Key**

Important: When the transfer service generates this information, record it immediately and keep it in a secure location. The client secret, in particular, is meant to be held privately.

2.1.3 Setting Up the Directory Structure

You must also set up a directory structure so that each subaccount has a designated area for sending files and for receiving files.

InterSystems recommends using one parent “Incoming” and one parent “Outgoing” directory at the top level of the account, with individual incoming and outgoing subdirectories for the subaccounts within those directories. This organization makes it easier for the InterSystems IRIS production to locate all files that need to be transferred in either direction.

If you are already using an MFT service, then you may already be using a different directory structure. If so, and you do not wish to modify the directory structure, then you might need to add multiple MFT business services and operations to the production, each configured to find or place files in different directories.

2.2 Creating Configuration Items in InterSystems IRIS

In addition to preparing the MFT account, you need to create specific configuration items on the InterSystems IRIS server. Specifically, you must:

- Create a TLS configuration for use by the production
- Create a MFT connection for use by the production
- Authorize the MFT connection

The following subsections describe the details.

2.2.1 Creating a TLS Configuration

InterSystems IRIS uses TLS to connect to an MFT service, so you must create a TLS configuration to use. InterSystems recommends that you create a separate configuration to use only for MFT connections, even if it uses default settings.

For details on creating a new TLS configuration, see About Configurations.

2.2.2 Creating a Managed File Transfer Connection

A *managed file transfer (MFT) connection* is a configuration item that the production can use to connect securely to an MFT service. If you have multiple productions running on a single InterSystems IRIS server, create a separate MFT connection for each production. In each case, the MFT connection must contain the OAuth 2.0 information you received from the MFT service.

To create an MFT connection:

1. From the Management Portal, go to the **Managed File Transfer Connections** page (**System Administration** > **Security** > **Managed File Transfer Connections**).
2. Click **Create Connection** to bring up the configuration page.
3. Specify values for the fields as follows, and then click **Save**:
   - **Connection Name**—Name for this connection for use within the production.
   - **File management service**—MFT service used for this connection.
   - **SSL/TLS configuration**—Name of the TLS configuration to use for this connection.
   - **Email address**—Email address of the administrator of the MFT account.
   - **Base URL**—Root URL meant for use in transferring files (specific to your account).
   - **OAuth 2.0 application name**—Name of the custom application created within the MFT service.
   - **OAuth 2.0 client ID**—The application identifier as provided by the MFT service. Each MFT service uses a different name for this item:
     - Box: **Client ID**
     - Dropbox: **App key**
     - kiteworks: **Client Application ID**
   - **OAuth 2.0 client secret**—The password that the MFT service provided. Each MFT service uses a different name for this item:

  - Box: **Client Secret**
  - Dropbox: **App secret**
  - kiteworks: **Client Secret Key**

- **OAuth 2.0 redirect URL**—URL used by the MFT service to connect to InterSystems IRIS. Enter the following values to automatically generate this URL:
  - **Use TLS/SSL**—Whether to use TLS to connect to the MFT service. In general, you should select this option.
  - **Host name**—The fully qualified domain name (FQDN) or IP address of the InterSystems IRIS server.
  - **Port**—The web server port specified in the Web Gateway configuration.
  - **Prefix**—Typically blank. Specify this if needed to accommodate any changes in the Web Gateway configuration.

4. Verify that the generated redirect URL has the following form:

   ```
   http://hostname:port/prefix/csp/sys/oauth2/OAuth2.Response.cls
   ```

   If you omit **Port**, the colon is omitted in the generated URL. Similarly, if you omit **Prefix**, there is only one slash between `hostname:port` and `csp`.

   This URL must match the one you supplied to the MFT service when creating the custom application for the production.

5. If the generated URL does not match what you had provided to the MFT service, then log in to the MFT service and edit the app definition to use the generated URL.

2.2.3 Authorizing an MFT Connection

The next step is to authorize the newly created MFT connection. To do so, obtain and save an access token from the **Managed File Transfer Connections** page (**System Administration** > **Security** > **Managed File Transfer Connections**), as follows:

1. Click the **Get Access Token** link for the connection you want to authorize.

   When you do so, the Management Portal displays the login page for the MFT service.

2. Log in with the credentials for the administrative account.

   Once the MFT service has authenticated the credentials, you see a page that displays the authorization request from the MFT service, listing the types of access that are to be granted to the production.

3. Click **Grant Access** to authorize the access. This redisplays the **Connections** list, and the MFT connection is now listed as `Authorized`.

3

Configuring the Production to Use MFT Services

After completing the prerequisites, modify the production to include MFT business hosts as needed. This topic describes how to do this.

3.1 Enabling a Production to Retrieve Files

To enable a production to retrieve files from an MFT service:

1. From the **Production Configuration** page, add a business service based on the class EnsLib.MFT.Service.Passthrough.
2. Configure the following items for this business service:
   - **MFT Connection Name**—Specifies the MFT connection to use.
   - **MFT Source Folders**—Specifies the top-level *sending* directory at the MFT service, such as `/Outgoing`
   - **Files to Retrieve**—Optionally specifies a regular expression for the names (types) of files to retrieve. An empty value means all files.
   - **Include Sub Folders**—Specifies whether the production should recursively examine all subfolders.
   - **Delete From Server**—Specifies whether items successfully imported from the MFT service should be removed from there after completion. The options are:
     - **No**—Default value. Do not remove the item. Only keep track of the last date and time of import.
     - **Trash**—Causes the MFT service to delete the item. In this case, the MFT service moves the item to a trash folder where it places all deleted items. Depending on the MFT service, items in this trash folder may be recoverable manually for a period of time, so consult the documentation for the MFT service for specifics.
     - **Permanent**—Causes the MFT service to permanently delete the items that have been successfully received by the production.
   - **Find ModifiedBy Username**—Specifies whether the production should also retrieve the user who last modified the file at the MFT service.
   - **Call Interval**—Specifies how frequently (in seconds) the production should check for new files. The default value is 5 seconds.

- **Target Config Names**—Specifies a comma-separated list of business hosts to which this business service should send information.

For information on other settings, see Settings for the File Inbound Adapter

3.2 Enabling a Production to Send Files

To enable a production to send files to an MFT service:

1. From the **Production Configuration** page, add a business operation based on the class EnsLib.MFT.Operation.Passthrough.
2. Configure the following items for this business operation:
   - **MFT Connection Name**—Specifies the MFT connection to use.
   - **Default MFT Folder**—Specifies the top-level *receiving* directory at the MFT service, such as `/Incoming`
   - **Default Filename Specification**—Specifies an optional template for creating the name of the received file. See the method Ens.Util.File.CreateTimestamp() for documentation of the various options.
   - **Overwrite**—Specifies whether to overwrite any existing file with the same name as the one currently being sent. Some MFT services may automatically change the name of an incoming file to include a timestamp; in this case, the **Overwrite** may have no effect.

   For information on other settings, see Settings for the File Outbound Adapter.

A

Using the MFT API for InterSystems IRIS

If the MFT business hosts do not meet your needs, you can use the MFT API, which enables you to manage connections, resources (files and directories), and users at each of the supported MFT services. This API consists of the packages %SYS.MFT.Connection and %MFT. The provided methods cover the following activities:

- Connection management
- File management
- User management

The %SYS.MFT.Connection package contains methods for managing network connections to MFT services. The %MFT package contains methods and properties for file and user management, in addition to service-specific functions. Both classes have subclasses for each of the supported MFT services (Box, Dropbox, and Accellion kiteworks). When creating and managing a connection, always use the connection-specific class.

Sample usages for methods are listed below. For more information about all of the items listed here, including detailed parameter specifications and return values for every method and property, see the Class Reference entries for classes in %SYS.MFT.Connection and %MFT.

A.1 Connection Management APIs

This section describes how to create a saved connection programmatically (a one-time setup), obtain a connection object, and then manage connections.

A.1.1 Setup Steps for Creating a Saved Connection

To create a connection to a MFT service, first complete the following one-time setup steps:

1. Create a connection object. To do so, call the %New() method of %SYS.MFT.Connection.Box, %SYS.MFT.Connection.Dropbox, or %SYS.MFT.Connection.Kiteworks, depending on the MFT service you want to use. For example:

   ```
   set connection = ##class(%SYS.MFT.Connection.Dropbox).%New()
   ```

2. Set the following properties of this object:
   - *Name*—Name for this connection for use within the production.
   - *Service*—MFT service used for this connection.

- *SSLConfiguration*—Name of the TLS configuration to use for this connection.
- *Username*—Email address of the administrator of the MFT account.
- *URL*—Root URL of the transfer service page for your organization.
- *ApplicationName*—Name of the app as created on the MFT service.

For example:

```
set connection.URL = "https://companyname.kiteworks.com/"
```

3. Save the connection object by calling its Save() method (not **%Save()**). For example:

```
set status = connection.Save()
```

4. Add the OAuth 2.0 client information. To do so, call the **CreateClient()** class method of the same class with the following properties as its arguments:

   - *name*—The name of the entry for the production, as configured at the MFT service.
   - *sslConfiguration*—The TLS configuration used for this MFT connection.
   - *clientId*—The OAuth 2.0 client ID from the MFT service.
   - *clientSecret*—The OAuth 2.0 client secret from the MFT service.

   For the complete signature including all optional arguments, see the Class Reference.

   For example:

```
set status = ##class(%SYS.MFT.Connection.Dropbox).CreateClient(
        OAuth2AppName,
        "tlsconfig",
        clientId,
        clientSecret)}
```

   This method adds the OAuth 2.0 client information and saves the object again. The method returns an error status if it did not successfully save the record.

These steps are equivalent to the steps described in Creating a Managed File Transfer Connection and Authorizing an MFT Connection.

A.1.2 Obtaining a Connection Object

After you have performed the setup work described in the previous section, use the **GetConnection()** method of the %MFT.Box, %MFT.Dropbox, or %MFT.Kiteworks class to obtain a connection object that you can use to manage connections.

GetConnection()

```
classmethod GetConnection(connectionName As %String,
                          Output sc As %Status) as %SYS.MFT.Connection.Base
```

Returns a connection object for use with the given MFT service. The arguments are as follows:

- *connectionName* is the name of the connection.
- *sc*, which is returned by reference, is the status code that indicates whether the system retrieved the object successfully.

This method is in the class %MFT.API as well as the service-specific subclasses %MFT.Box, %MFT.Dropbox, and %MFT.Kiteworks. However, InterSystems recommends that you call **GetConnection()** from the appropriate service-specific class.

Sample usage:

```
set connection = ##class(%MFT.Box).GetConnection(connectionname,.sc)
```

A.1.3 Managing Connections

Once you have retrieved a connection object via **GetConnection()**, use the following methods to manage and retrieve information about MFT connections. Except where noted, these are instance methods within the connection object.

GetAuthorizationCodeURL()

```
method GetAuthorizationCodeURL(redirect As %String,
                               scope As %String,
                               ByRef properties As %String,
                               Output sc As %Status) as %String
```

Returns the URL to use for obtaining authorization from the MFT service. The arguments are as follows:

- *redirect* is the redirect URL.
- *scope* is the OAuth 2.0 scope.
- *properties* is a multidimensional array of properties.
- *sc*, which is returned by reference, is the status code that indicates whether the system was successful in obtaining the URL.

IsAuthorized()

```
method IsAuthorized(Output errorMessage As %String) as %Boolean
```

Checks the authorization status of the connection. There is one argument:

- *errorMessage*, which is returned as output, contains any error messages returned by the MFT service.

RevokeToken()

```
method RevokeToken() as %Status
```

Revokes the token for the connection and returns a %Status value to indicate success or failure of this operation.

DeleteId()

```
classmethod DeleteId(name As %String) as %Status
```

Deletes the connection object and returns a %Status value to indicate success or failure of this operation. There is one argument:

- *name* is the name of the connection object.

This method is in each of the classes %SYS.MFT.Connection.Box, %SYS.MFT.Connection.Dropbox, and %SYS.MFT.Connection.Kiteworks. Although all these classes inherit from %SYS.MFT.Connection.Base, InterSystems recommends that you call **DeleteId()** from the appropriate service-specific class.

A.2 Directory and File Management APIs

This section lists the methods you can use to manage directories and files at an MFT service. These methods are in the service-specific subclasses %MFT.Box, %MFT.Dropbox, and %MFT.Kiteworks. Unlike the connection management methods, InterSystems recommends that you call the directory and file management methods from the parent class %MFT.API; for example, use **%MFT.API.CreateFolder()** instead of **%MFT.DropBox.CreateFolder()**. InterSystems IRIS automatically processes all calls as the appropriate service-specific version whenever methods are called from the parent class. If a call does not exist for a particular service, then InterSystems IRIS skips that step and continues to the next one.

Important: To manage directories and files on Dropbox, you must use a connection based on a Dropbox User application with Full Dropbox permission.

A.2.1 Managing Folder Access

Use the following methods for creating, deleting, and sharing folders at an MFT service.

CreateFolder()

```
classmethod CreateFolder(connection As %SYS.MFT.Connection.Base,
                         folderPath As %String,
                         Output itemInfo As %MFT.ItemInfo) as %Status
```

Creates a folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderPath* is the full path of the folder to create.
- *itemInfo*, which is returned as output, is an instance of %MFT.ItemInfo that provides a handle for the new folder. See %MFT.ItemInfo Details.

Sample usage:

```
set status = ##class(%MFT.API).CreateFolder(connection,"/NewDir",.itemInfo)
```

DeleteFolder()

```
classmethod DeleteFolder(connection As %SYS.MFT.Connection.Base,
                         path As %String
                         permanent as %Boolean = 0) as %Status
```

Deletes a folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full path of the folder. You can instead specify this argument as a string of the form `"id:id_from_iteminfo"` where *id_from_iteminfo* is an ID from a %MFT.ItemInfo object.
- *permanent* specifies whether the deletion is permanent. If *permanent* is 0, the MFT service moves the folder to the trash folder. If *permanent* is 1, the MFT service deletes the folder permanently. Note that this method does not currently support permanent folder deletion for DropBox.

Sample usage:

```
set status = ##class(%MFT.API).DeleteFolder(connection,"/DirToDelete")
```

ShareFolder()

```
classmethod ShareFolder(connection As %SYS.MFT.Connection.Base,
                        path As %String,
                        accessLevel As %String = "viewer",
                        users As %List) as %Status
```

Shares the given folder with specific users (granting them a specific access level) and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full path of the folder.
- *accessLevel* specifies the access level to give. The default is view-only access. Refer to the documentation from the MFT service for the different levels of access.
- *users* is a %List of user names.

Sample usage:

```
set status = ##class(%MFT.API).ShareFolder(connection,"/SharedDir/","editor",ListOfUsers))
```

UnshareFolder()

```
classmethod UnshareFolder(connection As %SYS.MFT.Connection.Base,
                          path As %String,
                          user As %String) as %Status
```

Removes a given user from those authorized to access a folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full path of the folder.
- *user* is a user name.

Sample usage:

```
set status = ##class(%MFT.API).UnshareFolder(connection,"/FolderToUnshare","mwinters")
```

UnshareFolderAll()

```
classmethod UnshareFolderAll(connection As %SYS.MFT.Connection.Base,
                             path As %String) as %Status
```

Prevents the folder from being shared with any user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full path of the folder.

Sample usage:

```
set status = ##class(%MFT.API).UnshareFolderAll(connection,"/FolderToUnshare")
```

MountFolder()

```
classmethod MountFolder(connection As %SYS.MFT.Connection.Base,
                        folderName As %String) as %Status
```

For Dropbox only: Mounts the folder if it has been shared by the owner and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderName* is the full path of the folder.

Sample usage:

```
set status = ##class(%MFT.DropBox).MountFolder(connection,"MyFolder")
```

UnmountFolder()

```
classmethod UnmountFolder(connection As %SYS.MFT.Connection.Base,
                          folderName As %String) as %Status
```

For Dropbox only: Unmounts the folder *folderName* and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderName* is the full path of the folder.

Sample usage:

```
set status = ##class(%MFT.DropBox).UnmountFolder(connection,"FolderName")
```

SetCurrentFolder()

```
classmethod SetCurrentFolder(connection As %SYS.MFT.Connection.Base,
                             folderPath As %String) as %Status
```

Establishes the given folder as the current working folder in the production and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderName* is the full path of the folder.

Sample usage:

```
set status = ##class(%MFT.API).SetCurrentFolder(connection,"/a/b/c")
```

A.2.2 Getting Folder Information

Use the following methods and properties to obtain information about folders at the MFT service named in *connection*:

GetCurrentFolder()

```
classmethod GetCurrentFolder(connection As %SYS.MFT.Connection.Base,
                             Output folderPath As %String) as %Status
```

Eetrieves the full pathname of the current working folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderName*, which is returned as output, is the full path of the current working folder for the given connection.

Sample usage:

```
set status = ##class(%MFT.API).GetCurrentFolder(connection,.path)
```

GetFolderInfo()

```
classmethod GetFolderInfo(connection As %SYS.MFT.Connection.Base,
                          path As %String,
                          Output itemInfo As %MFT.ItemInfo) as %Status
```

Retrieve a specified folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full path of the folder.
- *itemInfo*, which is returned as output, is an instance of %MFT.ItemInfo that provides a handle to the given folder (if **GetFolderInfo()** returns success). For details, see %MFT.ItemInfo Details.

Sample usage:

```
set status = ##class(%MFT.API).GetFolderInfo(connection,"/dirname",.itemInfo)
```

GetFolderContents()

```
classmethod GetFolderContents(connection As %SYS.MFT.Connection.Base,
                              folderPath As %String,
                              recursive As %Boolean = 0,
                              Output folderContents As %MFT.FolderContents) as %Status
```

Retrieves the contents of a specified folder and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *folderPath* is the full path of the folder.
- *recursive* controls whether to recursively obtain subfolders.
- *folderContents*, which is returned as output, is an instance of %MFT.FolderContents that provides a handle to the folder contents (if **GetFolderContents()** returns success).

Sample usage:

```
set status = ##class(%MFT.API).GetFolderContents(connection,"",IsRecursive,.contents)
```

An instance of the class %MFT.FolderContents has the following properties:

Contents

A list of the contents of the folder, each one in an instance of %MFT.ItemInfo.

Recursive

A Boolean value indicating whether or not %MFT.FolderContents.Contents is a recursive listing.

A.2.3 Managing Files

Use the following methods for managing files at the MFT service named in *connection*:

DeleteFile()

```
classmethod DeleteFile(connection As %SYS.MFT.Connection.Base,
                       path As %String) as %Status
```

Deletes a file and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *path* is the full pathname of the file.

Sample usage:

```
set status = ##class(%MFT.API).DeleteFile(connection,"/dirname/filetodelete.txt")
```

UploadFile()

```
classmethod UploadFile(connection As %SYS.MFT.Connection.Base,
                       localFilePath As %String,
                       filePath As %String,
                       replace As %Boolean = 0,
                       Output itemInfo As %MFT.ItemInfo) as %Status
```

Uploads a file and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *localFilePath* is the full pathname of the file on the local system.
- *filePath* is the full target pathname of the file within the transfer service.
- *replace* specifies whether to replace any existing file that has the same name.
- *itemInfo*, which is returned as output, is an instance of %MFT.ItemInfo that provides a handle to the newly uploaded file (if **UploadFile()** returns success). For details, see %MFT.ItemInfo Details.

Sample usage:

```
set status =
##class(%MFT.API).UploadFile(connection,localdir_"file.txt","/uploadDir/file.txt",,.itemInfo)
```

DownloadFile()

```
classmethod DownloadFile(connection As %SYS.MFT.Connection.Base,
                         filePath As %String,
                         localFilePath As %String) as %Status
```

Downloads a file and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *filePath* is the full pathname of the file within the transfer service.
- *localFilePath* is the full target pathname of the file on the local system.

Sample usage:

```
set status =
##class(%MFT.API).DownloadFile(connection,"/DownloadDir/file.txt",localdir_"file.txt")
```

UploadStream()

```
classmethod UploadStream(connection As %SYS.MFT.Connection.Base,
                         stream As %BinaryStream,
                         filePath As %String,
                         replace As %Boolean = 0,
                         Output itemInfo As %MFT.ItemInfo) as %Status
```

Uploads the contents of the given stream, creating a file within the transfer service, and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *stream* is the stream whose contents are to be uploaded.
- *filePath* is the full target pathname of the file within the transfer service.
- *replace* specifies whether to replace any existing file that has the same name.
- *itemInfo*, which is returned as output, is an instance of %MFT.ItemInfo that provides a handle to the new file (if **UploadStream()** returns success). For details, see %MFT.ItemInfo Details.

Sample usage:

```
set status =
##class(%MFT.API).UploadStream(connection,stream,"/uploadDir/newfile.txt",0,.itemInfo)
```

DownloadStream()

```
classmethod DownloadStream(connection As %SYS.MFT.Connection.Base,
                           filePath As %String,
                           ByRef stream As %BinaryStream) as %Status
```

Downloads a file, creating a stream object, and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *filePath* is the full pathname of the file within the transfer service.
- *stream*, which is returned by reference, is the stream object that contains the file contents (if **DownloadStream()** returns success).

Sample usage:

```
set status = ##class(%MFT.API).DownloadStream(connection,"/DownloadDir/file.txt",.resultstream)
```

A.2.4 Getting File Information

The %MFT.ItemInfo class provides an API for information about files. The **GetFolder()**, **UploadFile()**, and **UploadStream()** methods (listed in previous sections) each return an instance of this class as output. You can also obtain an instance of this class by calling the **GetFileInfo()** method of the %MFT.API, %MFT.Box, %MFT.Dropbox, or %MFT.Kiteworks class:

GetFileInfo()

```
classmethod GetFileInfo(connection As %SYS.MFT.Connection.Base,
                        path As %String,
                        Output itemInfo As %MFT.ItemInfo) as %Status
```

Retrieves information about the given file and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.

- *path* is the full pathname of the file within the transfer service.
- *itemInfo*, which is returned by reference, is an instance of %MFT.ItemInfo that contains information about the file (if **GetFileInfo()** returns success). See %MFT.ItemInfo Details.

Sample usage:

```
set status = ##class(%MFT.API).GetFileInfo(connection,"filename.txt",.itemInfo)
```

A.2.4.1 %MFT.ItemInfo Details

An instance of the class %MFT.ItemInfo has the following method and properties:

%MFT.ItemInfo.GetPath() method

```
method GetPath(trailingSlash As %Boolean = 1) as %String
```

Returns the path of the file, as a string. The argument *trailingSlash* specifies whether the path should include a trailing slash.

Details property

Contains information about the file. This property is an instance of %DynamicObject.

ItemId property

Contains the internal identifier for the item, as a string.

Modified property

Indicates the item's time of creation in UTC, as a %Timestamp.

ModifiedBy property

Contains the internal identifier (UserId) of the user who last modified the item, as a string. Use %MFT.API.GetUsername() on the returned value to get the external username.

Name property

Contains the name of the item, as a string.

Type property

Indicates whether the item is a file (1) or a folder (2).

A.3 User Management APIs

This section lists the methods you can use to manage users at an MFT service. These methods are in the service-specific subclasses %MFT.Box, %MFT.Dropbox, and %MFT.Kiteworks. Unlike the connection management methods, InterSystems recommends that you call the user management methods from the parent class %MFT.API; for example, use **%MFT.API.CreateUser()** instead of **%MFT.DropBox.CreateUser()**. InterSystems IRIS automatically processes all calls as the appropriate service-specific version whenever methods are called from the parent class. If a call does not exist for a particular service, then InterSystems IRIS skips that step and continues to the next one.

Important: To manage users on Dropbox, you must use a connection based on a Dropbox Business application with Team Member Management permission.

A.3.1 Creating and Deleting Users

Use the following methods to create and delete users.

CreateUser()

```
classmethod CreateUser(connection As %SYS.MFT.Connection.Base, userInfo As %MFT.UserInfo) as
%Status
```

Creates a user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *userInfo* is an instance of %MFT.UserInfo that contains information about the user to create. See %MFT.UserInfo Details.

DeleteUser()

```
classmethod DeleteUser(connection As %SYS.MFT.Connection.Base, username As %String) as %Status
```

Given a username, deletes the user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *username* is the name of the user.

DeleteUserById()

```
classmethod DeleteUserById(connection As %SYS.MFT.Connection.Base, userid As %String) as %Status
```

Given a user ID, deletes the user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *userid* is the ID of the user.

A.3.2 Getting a User Object

Use the following methods to get user information from the MFT service named in *connection*:

GetUser()

```
classmethod GetUser(connection As %SYS.MFT.Connection.Base,
                    username As %String,
                    Output userInfo As %MFT.UserInfo) as %Status
```

Given a username, retrieves information about the user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *username* is the name of the user.

- *userInfo*, which is returned as output, is an instance of %MFT.UserInfo that contains information about the user. See %MFT.UserInfo Details.

Sample usage:

```
set status = ##class(%MFT.API).GetUser(connection,"User ToGet",.user)
```

GetUsername()

```
classmethod GetUsername(connection As %SYS.MFT.Connection.Base,
                        internalId As %String,
                        Output username As %String) as %Status
```

Given a user ID, retrieves the username and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *internalId* is the ID of the user.
- *username*, which is returned as output, is the name of the user.

Sample usage:

```
set status = ##class(%MFT.API).GetUsername(connection,itemInfo.UserID,.username)
```

GetUserById()

```
classmethod GetUserById(connection As %SYS.MFT.Connection.Base,
                        userid As %String,
                        Output userInfo As %MFT.UserInfo) as %Status
```

Given a user ID, retrieves information about the user and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *userid* is the ID of the user.
- *userInfo*, which is returned as output, is an instance of %MFT.UserInfo that contains information about the user. See %MFT.UserInfo Details.

Sample usage:

```
set status = ##class(%MFT.API).GetUserById(connection,itemInfo.ModifiedBy,.user)
```

GetUserList()

```
classmethod GetUserList(connection As %SYS.MFT.Connection.Base, Output userList As %MFT.UserList)
 as %Status
```

Retrieves a list of all the users for the account and returns a %Status value to indicate success or failure of this operation. This method has the following arguments:

- *connection* is a connection object for an MFT service, obtained via **GetConnection()**.
- *userList*, which is returned as output, is an instance of %MFT.UserList that contains information about the users. The %MFT.UserList object includes the property Users, which is a list of %MFT.UserInfo objects.

Sample usage:

```
set status = ##class(%MFT.API).GetUserList(connection,.userList)
```

A.3.3 %MFT.UserInfo Details

The **CreateUser()** method uses an instance of the class %MFT.UserInfo as input. Other methods return instances of this class as output.

An instance of the class %MFT.UserInfo has the following properties:

Details

A %DynamicObject that contains information about the user.

Name

The (descriptive) name of the named user, as a string.

UserId

An internal identifier for the named user, as a string.

UserName

The unique name (login) of the user (typically the user's email address), as a string.