THRU Server API 4.0
Programming Reference

Table of Contents

1. General

THE PURPOSE OF THIS DOCUMENT IS TO DESCRIBE WEB SERVICES API VERSION 4.0 (WS4) WITH HTTP EXTENSIONS WHICH IS IMPLEMENTED IN THE CURRENT RELEASE OF THRU SERVICE AND USED BY OUTLOOK CLIENT 2.X. THRU SERVER IS BUILT ON SERVICE-ORIENTED ARCHITECTURE THAT PROVIDES EXTENSIVE API VIA WEB SERVICES.

THIS PROGRAMMING REFERENCE GUIDE DESCRIBES AND SUMMARIZES THE THRU SERVER API OF THE THRU PLATFORM. THIS DOCUMENT HAS BEEN PREPARED SOLELY FOR INFORMATIONAL PURPOSES AND IS BEING FURNISHED FOR USE BY CUSTOMERS. DISTRIBUTION OF THIS DOCUMENT IS LIMITED TO CUSTOMERS.

THE PROGRAMMING REFERENCE GUIDE AND ALL INFORMATION CONTAINED HEREIN ARE OF A CONFIDENTIAL NATURE. EACH RECIPIENT WILL TREAT IT IN A CONFIDENTIAL MANNER. SUCH RECIPIENT WILL, NOT DIRECTLY OR INDIRECTLY, DISCLOSE OR PERMIT ITS AGENTS, REPRESENTATIVES, EMPLOYEES, OFFICERS, DIRECTORS OR AFFILIATES TO DISCLOSE ANY SUCH INFORMATION AND SUCH RECIPIENT WILL USE THE DOCUMENT AND ANY RELATED INFORMATION ONLY TO ACCESS THE THRU SERVICE. IF EITHER THE RECIPIENT OR THE COMPANY ELECTS NOT TO PURSUE OR CONTINUE WITH THE THRU SERVICE, THE RECIPIENT MUST RETURN THIS DOCUMENT AND ANY OTHER MATERIAL RELATING TO THE THRU. SERVICE AND THRU, LLC., WITHOUT RETAINING ANY COPIES THEREOF.

THE INFORMATION CONTAINED HEREIN IS PROVIDED ON A CONFIDENTIAL BASIS AND MAY NOT BE REPRODUCED IN WHOLE OR IN PART.

IF YOU REQUIRE ANY MORE INFORMATION REGARDING THE THRU SERVICE PLEASE CONTACT:

THRU. HELP CENTER
HELPCENTER@THRUINC.NET
800.871. 9316

2. Thru Server Web Service API

2.1 Web Service API addressing

Build-in .NET documentation on implemented web services is available at URL **<servername>.thruinc.net/ws4/fthservice.asmx** on Thru server.
WDSL file for API version 4.0 is located at **<servername>.thruinc.net/ws4/fthservice.asmx?WSDL**

2.2 Retrieve settings for Thru Data Center and Thru Server

2.2.1 Data Structures

FTHServerSettingData

FTHServerSettingData		
Thru site settings for specific site		
Data Type	**Field Name**	**Description**
int	**SiteID**	
int	**SettingsExpirationMinutes**	How long this configuration should be used before being refreshed.
bool	**RetentionEnabled**	Is feature enabled
bool	**RetentionAdmin**	Is user who opened the server is a member of Retention Administrators group
bool	**MirroringOutboundEnabled**	Is outbound mirroring feature enabled
bool	**MirroringInboundEnabled**	Is inbound mirroring feature enabled
bool	**AuditMgrEnabled**	Is Audit Manager feature enabled
bool	**AuditUser**	Is user who opened the server a member of Auditors group
FTHMsgRequestLoginType	**MsgRequestLoginType**	Enumeration, policy setting to request login when recipient clicks on the link in the email: Never=1/Always=2/User Choice=3
bool	**MsgRequestloginUserChoiceDefault**	If Request Login is 'User Choice' this field determines the default value (request login or not).
bool	**MsgMaxDownloadsEnabled**	Not implemented
int	**MsgMaxDownloadsDefault**	Not implemented
bool	**PublishingEnabled**	Is Publishing feature enabled
bool	**PublishingUser**	Is a user who opened the server a member of Publishing Users group
bool	**PublishingAdmin**	Is a user who opened the server a member of Publishing Administrators group
bool	**PublishingTagDictionaryManaged ByUsers**	Indicates whether publishing users can edit the tag dictionary.

2.2.2 Methods

GetDataCenterSettings

GetDataCenterSettings

Retrieves current settings for Thru Datacenter in a form of Name=Value pairs. Datacenter contains multiple Thru Sites.
 	
Example of datacenter values:
 	
`LandingURL sjdev52/FTHLanding75B`
`MirroringChunkSizeMB 25`
`MirroringPublishingRoot C:\Thru\MirroringPublishingRoot`
`MirroringLandingRoot C:\Thru\MirroringLandingRoot`
`MirroringTransferProtocol HTTPS`
`MirroringPublishingAreaLimitGB 20`
`MirroringLandingAreaLimitGB 20`
`NotificationPrimaryContactEmail sergey@thruinc.com`
`NotificationSecondaryContactEmail arutiunov@yahoo.com`
`NotificationSMTPServer 10.0.0.148`
`NotificationSentFromContactEmail Notifications@thruinc.net`
`ReportingServerName sjdev6\FTH75C`
`MobileSiteUrl sjdev7.thruinc.com/sergey75c_mobile`
`MirroringFileChunkMinSizeMB 2`
`MirroringFileChunkMaxSizeMB 50`
`MirroringFileChunkCount 20`
`MirroringFileChunkCountLimitMB 1000`

`FTHNameValueData[] GetDataCenterSettings(Guid sessionID)`

Parameter Name	Description
`sessionID`	Current user session token returned by one of Open methods

GetServerSettings

GetServerSettings
Retrieves settings for Thru site where user is signed in.
`FTHServerSettingData` **`GetServerSettings`**`(Guid sessionID)`

2.3 Open and Close the Server

Thru server can be opened in 3 different ways by Open, OpenImmutable or OpenWithTicket web service calls used by Thru SSO Ticket server or SSO-based applications. Each call returns **`FTHSessionData`** data structure which contains Session ID and User ID of the user who logged in.

2.3.1 Data Structures

FTHSessionData

FTHSessionData		
User Session data		
Data Type	**Field Name**	**Description**
Guid	**SessionID**	ID of a new session that is created in response to the call. Should be supplied as parameter to all subsequent web service calls.
int	**UserID**	ID of Thru user account under which the connection is signed in.

2.3.2 Methods

Open

Open	
Open call opens the server using username and password and creates a session for given ApplicationID. The session is alive until the next sign-in is done using Open or OpenWithTicket for the same user and the same ApplicationID, then new session replaces the previous. Different applications are allowed to sign in concurrently under the same user.	
`FTHSessionData` **`Open`**`(string userName, string password, int applicationID)`	
Name	**Description**
`username`	Username for sign-in.
`password`	Password for sign-in
`applicationID`	ApplicationID parameter should be picked by client developers and communicated to Thru to be reserved for specific client. New ApplicationID value has to be added to the Thru Database table TBL_APPLICATION by Thru server administrator. ApplicationID >=1000, values below 1000 are reserved for applications developed by Thru.

OpenWithTicket

OpenWithTicket
Accepts the ticket that contains user identity encrypted by Thru library on the client side. Encrypting party and Thru server should use the same password since encryption is symmetrical. In order for OpenWithTicket call to succeed, Thru SSO product should be deployed on a customer's LAN and a user whose identity is contained in the ticket had to be previously created by Thru SSO Sync server. Thru SSO sync server synchronizes the contents of specified user group in corporate Active Directory with the user list in Thru server.
FTHSessionData **OpenWithTicket**(string ticket, int applicationID)

Parameter Name	Description
ticket	String with the user identity encrypted by Thru libraries in SSO or other products. Encrypting party and Thru server should use the same password for symmetric encryption/decryption.
applicationID	ApplicationID parameter should be picked by client developers and communicated to Thru to be reserved for specific client. New ApplicationID value has to be added to the Thru Database table TBL_APPLICATION by Thru server administrator. ApplicationID >=1000, values below 1000 are reserved for applications developed by Thru.

OpenImmutable

OpenImmutable
Similar to **Open** call but the session created as immutable meaning that it will not be deleted when another Open or OpenWithTicket function is called for the same user and the same AplicationID. Used when session concurrency is required for the same user. Immutable sessions are cleaned automatically every 48 hours from the time they were created.
FTHSessionData **OpenImmutable**(string userName, string password, int applicationID)

Parameter Name	Description
username	Username for sign-in.
password	Password for sign-in
applicationID	ApplicationID parameter should be picked by client developers and communicated to Thru to be reserved for specific client. New ApplicationID value has to be added to the Thru Database table TBL_APPLICATION by Thru server administrator. ApplicationID >=1000, values below 1000 are reserved for applications developed by Thru.

Close

Close
Close web call deletes existing user session.
Void **Close**(Guid sessionID)

Parameter Name	Description
sessionID	Active user session to be deleted

2.4 File and Folder management in Thru server

2.4.1 Manage Thru folders.

2.4.1.1 Data Structures

FTHFolderData

FTHFolderData		
Folder object data		
Data Type	**Field Name**	**Description**
int	**FolderID**	ID of a newly created folder
int	**ParentFolderID**	ID of a parent folder
string	**Name**	Name of new folder
int	**EffectivePermission**	Effective access permission for new folder for currently signed user Permission enum: None = 0x00, //0 Show = 0x01, //1 Read = 0x02, //2 Create = 0x04, //4 Download = 0x08, //8 Distribute = 0x010, //16 Manage = 0x020, //32 Modify = 0x040, //64 Delete = 0x080, //128 FullControl = Read | Create | Download | Distribute | Manage | Modify | Delete
DateTime	**DateCreated**	Timestamp when the folder was created
DateTime	**DateModified**	Timestamp when the folder was modified
int	**CreatedByUserID**	ID of the user who created the file
int	**ModifiedByUserID**	ID of the user who modified the file
bool	**IsDeleted**	Flag shows if the folder is deleted

int	**FolderType**	One of the following Folder Types:
		`Regular = 0,`
		`Root = 1,`
		`UserHome = 2,`
		`UserDocuments = 3,`
		`UserDropBox = 4,`
		`FolderRoot = 5,`
		`HomeRoot = 6,`

2.4.1.2 Methods

CreateFolder

CreateFolder
Creates a folder in Thru file system
FTHFolderData **CreateFolder**(Guid sessionID, int parentFolderID, string name)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
parentFolderID	ID for the parent folder where new folder will be created
name	Name for a new folder

DeleteFolder

DeleteFolder
Deletes a folder in Thru file system
FTHFolderData **DeleteFolder**(Guid sessionID, int folderID)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID of the folder to be deleted

GetFolder

GetFolder
Retrieves folder data from Thru system by folder ID
FTHFolderData **GetFolder**(Guid sessionID, int folderID)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID of the folder to be retrieved

GetFolders

GetFolders
Retrieves a list of subfolders in the specified parent folder of Thru file system
FTHFolderData [] **GetFolders** (Guid sessionID, int parentFolderID, int index, int length)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
parentFolderID	ID of the parent folder where we look for subfolders
index	Index points to the first subfolder to start retrieval, zero-based
length	Number of subfolders to retrieve.

GetFolderByName

GetFolderByName
Retrieves a single folder by name in the specified parent folder of Thru file system
FTHFolderData **GetFolderByName**(guid SessionID, int parentFolderID, string name)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
parentFolderID	ID of the parent folder where we look for a folder
name	Name string by which a folder will be retrieved. Result is unambiguous since folder/file name is unique in the parent folder

GetRootFolder

GetRootFolder
Retrieves a Root folder object in Thru file system
FTHFolderData **GetRootFolder** (Guid sessionID)

Parameter Name	Description
sessionID	User session returned by one of the Open functions

MoveFolders

MoveFolders
Moves a list of folders to a folder specified by destination folder ID
`void MoveFolders(Guid sessionID, int[] folderIDList, int destinationFolderID)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`folderIDList`	List of folder IDs that will be moved to a destination folder
`destinationFolderID`	ID of destination folder where the folders will be moved

SaveFolder

SaveFolder
Saves changes to a folder object
`FTHFolderData SaveFolder(Guid sessionID, FTHFolderData folder)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`folder`	Folder object

GetFolderCount

GetFolderCount
Retrieves the count of subfolders in the specified parent folder.
`int GetFolderCount (Guid sessionID, int parentFolderID)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`parentFolderID`	ID of the parent folder where subfolders are counted

GetFolderPath

GetFolderPath
The method returns folder path in Thru file system as a string
`string GetFolderPath (Guid sessionID, int parentFolderID)`

Name	Description
`sessionID`	User session returned by one of the Open functions
`folderID`	ID of the folder for which the path is retrieved

GetFolderSize

GetFolderSize
Returns total size of files located in specified folder sub-tree in bytes, long value.
`long` `GetFolderSize (Guid sessionID, int folderID)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`folderID`	ID of the folder for which the size is retrieved

2.4.2 Manage folder metadata.

2.4.2.1 Data Structures

FolderMetadataResult

FolderMetadataResult		
Folder metadata in a form of name = value pairs		
Data Type	**Field Name**	**Description**
int	**FolderMetadataID**	ID of a newly created metadata item for a folder
int	**FolderID**	ID of a folder for which metadata is created
string	**Name**	Metadata item name
int	**DataType**	Type of the data
string	**Data**	Value

2.4.2.2 Methods

CreateFolderMetadata

CreateFolderMetadata
Creates metadata for a specific folder in Thru file system.

```
FolderMetadataResult
CreateFolderMetadata(Guid sessionID, int folderID, string name, string
data)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID for the folder for which metadata is created
name	Name for a new metadata item
data	String with the data that will populate metadata item

GetFolderMetadata

GetFolderMetadata
Retrieves metadata for specific Thru folder

```
FolderMetadataResult[]
GetFolderMetadata (Guid sessionID, int folderID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID for the folder for which metadata is retrieved

2.4.3 Folder permission management

2.4.3.1 Data Structure

FTHPermissionListItemData

FTHPermissionListItemData		
Holds the data on a single item in permission list		
Data Type	**Field Name**	**Description**
int	**FolderPermissionID**	ID of a folder permission item
int	**FolderID**	ID of a folder for which permission is retrieved
int	**Permission**	Permission value
int	**UserID**	ID of a user whose permission is retrieved, equals to (-1) if permission is set for a group
int	**UserGroupID**	ID of a group whose permission is retrieved, equals to (-1) if permission is set for a user

2.4.3.2 Methods

GetFolderPermissionList

GetFolderPermissionList
Retrieves a list of permissions for a Thru folder

```
FTHPermissionListItemData []
GetFolderPermissionList (Guid sessionID, int folderID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID of the folder for which the permission is retrieved

SetFolderPermissionList

SetFolderPermissionList
Set permission list for a Thru folder

```
void
SetFolderPermissionList(Guid sessionID, int folderID,
FTHPermissionListItemData[] newPermissionList)
```

Parameter Name	Description
SessionID	User session returned by one of the Open functions
FolderID	ID of the folder for which the permission is set
newPermissionList	List of permissions items, see specification of data structure FTHPermissionListItemData

2.4.4 Manage Files and File Metadata.

2.4.4.1 Data Structures

FTHFileMetadataItemData

FTHFileMetadataItemData		
File metadata in a form of name = value pairs		
Data Type	**Field Name**	**Description**
int	**FileMetadataID**	ID of a metadata item
int	**FileID**	ID of a file for which metadata is created
string	**Name**	Name of metadata item
int	**DataType**	Metadata type, extendable
object	**Data**	Metadata value

FTHFileData

FTHFileData		
Thru File object data		
Data Type	**Field Name**	**Description**
int	**FileID**	ID of a file in Thru server
int	**ParentFolderID**	ID of a parent folder that contains a file
string	**Name**	Name of a file
long	**Size**	File size
int	**EffectivePermission**	Effective permission of a file (reserved for future use)
DateTime	**DateCreated**	Date when a file was created
DateTime	**DateModified**	Date when a file was modified last time
int	**CreatedByUserID**	ID of the user who created the file
int	**ModifiedByUserID**	ID of the user who modified the file last time
int	**Version**	File version (reserved for future use)
bool	**IsDeleted**	Flag shows if a file is in Deleted state
bool	**IsQuarantined**	Flag shows if a file is in Quarantined state, means that file body was removed by Anti-virus software.
bool	**IsPublished**	Flag shows that a file is published into one of Thru collections.

2.4.4.2 Methods

DeleteFile

DeleteFile	
Deletes file object in Thru system. File body will be deleted later after Purge waiting period is over.	
`void DeleteFile(Guid sessionID, int fileID)`	
Parameter Name	**Description**
`SessionID`	User session returned by one of the Open functions
`fileID`	ID of the file to be deleted

GetFile

GetFile
Retrieves a specific file object in Thru system by file ID. Note: method does not retrieve a file body, it should be downloaded using HTTP GET call to URL ***servername/download.aspx***, see section 2.6

```
FTHFileData
GetFile(Guid sessionID, int fileID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
fileID	ID of the file object to be retrieved

GetFileByName

GetFileByName
Retrieves a specific file object from Thru system by specific file name, which is unique in the parent folder. Note: method does not retrieve a file body, it should be downloaded using HTTP GET call to URL ***servername/download.aspx***, see section 2.6

```
FTHFileData
GetFileByName(Guid sessionID, int parentFolderID, string name)
```

Parameter Name	Description
SessionID	User session returned by one of the Open functions
parentFolderID	ID of the parent folder that contains the file
Name	File name

GetFileCount

GetFileCount
Returns number of files in the specified parent folder.

```
int
GetFileCount(Guid sessionID, int parentFolderID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
parentFolderID	ID of the parent folder that contains the files to be counted

GetFiles

GetFiles
Retrieves an array of file objects in the specified parent folder. Note: method does not retrieve the file bodies, each file should be downloaded using HTTP GET call to URL ***servername/download.aspx*** see section 2.6
`FTHFileData[]` `GetFiles(Guid sessionID, int parentFolderID, int index, int length)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`parentFolderID`	ID of the parent folder that contains the files to be retrieved
`index`	Starting index for retrieval
`length`	Number of elements to retrieve

GetFilePath

GetFilePath
Retrieves file path in Thru server to specific file.
`string` `GetFilePath(Guid sessionID, int fileID)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`fileID`	ID of the file for which the path is retrieved

MoveFiles

MoveFiles
Moves specified files to destination folder.
`void` `MoveFiles(Guid sessionID, int[] fileIDList, int destinationFolderID)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`fileIDList`	List of file IDs that should be moved to destination folder
`destinationFolderID`	Destination folder where the files will be moved

SaveFile

SaveFile
Saves changes to a Thru file object, different from overwriting file body
`FTHFileData` `SaveFile(Guid sessionID, FTHFileData file)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`file`	File object that should be saved

CreateFileMetadata

CreateFileMetadata
Creates metadata for a specific file in a form of name = value pair
`FTHFileMetadataItemData` `CreateFileMetadata (Guid sessionID, int fileID, string name, object data)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`fileID`	ID of the file for which the metadata is created
`name`	Metadata item name
`data`	Metadata value

GetFileMetadata

GetFileMetadata
Retrieves array of metadata items for a specific file.
`FTHFileMetadataItemData[]` `GetFileMetadata(Guid sessionID, int fileID)`

Parameter Name	Description
`SessionID`	User session returned by one of the Open functions
`fileID`	ID of the file for which the metadata is retrieved

SaveFileMetadata

SaveFileMetadata	
Saves metadata for a specific file.	
`FTHFileMetadataItemData SaveFileMetadata(Guid sessionID, int fileID, string name, object data)`	
Parameter Name	**Description**
`sessionID`	User session returned by one of the Open functions
`fileID`	ID of the file for which the metadata is saved
`name`	Metadata item name
`data`	Metadata value

2.4.5 Retrieve Thru Audit records for files and folders

2.4.5.1 Data Structures

FTHFileSystemLogItemData

FTHFileSystemLogItemData		
Contains data from the log record on the actions performed on files and folders		
Data Type	**Field Name**	**Description**
int	**FolderID**	ID of a Folder for which log info is requested. Equals (-1) if information is requested for a File.
int	**FileID**	ID of a File for which log info is requested. Equals (-1) if information is requested for a Folder.
int	**UserID**	ID of a user who performed the action. Equals (-1) if action has been performed by external user who is not a Thru server user (ex. external distribution download). In this case user is identified by email field only.
string	**EmailAddress**	Email address for the user who performed the action. In case of external user this email is the only identifier of a user.

int	**Operation**	Code of logged operation:
		`None = 0,`
		`Created = 1,`
		`Overwritten = 2,`
		`InternalDownload = 4,`
		`ExternalDownload = 8,`
		`DisclaimerDownload = 16,`
		`Deleted = 32,`
		`Purged = 64,`
		`MovedIn = 128,`
		`MovedOut = 256,`
		`Restored = 512,`
		`Versioned = 1024,`
		`MetadataEdit = 2048,`
		`PermissionEdit = 4096,`
		`Distributed = 8192,`
		`InternalUpload = 16384,`
		`ExternalUpload = 32768,`
		`FileRename = 65536,`
		`FolderRename = 131072,`
		`Quarantine = 262144,`
		`DescriptionEdit = 524288,`
		`ChangeFileOwner = 1048576,`
		`All = 2097151`

string	**OperationDescription**	Description of operation:
		`None`
		`Created`
		`Overwritten`
		`InternalDownload`
		`ExternalDownload`
		`DisclaimerDownload`
		`Deleted`
		`Purged`
		`MovedIn`
		`MovedOut`
		`Restored`
		`Versioned`
		`MetadataEdit`
		`PermissionEdit`
		`Distributed`
		`InternalUpload`
		`ExternalUpload`
		`FileRename`
		`FolderRename`
		`Quarantine`
		`DescriptionEdit`
		`ChangeFileOwner`

DateTime	**DateLogged**	Timestamp of the date/time the log was recorded
string	**IPAddress**	IP address that performed the operation
FTHUserData	**UserInfo**	Information on the user who had performed the operation, see data structure FTHUserData

2.4.5.2 Methods

GetFileFilesystemLog

GetFileFilesystemLog
Retrieve Log records for a file
FTHFileSystemLogItemData[] **GetFileFilesystemLog** (Guid SessionID, int FileID)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
fileID	ID of the file for which the log data is retrieved

GetFolderFilesystemLog

GetFolderFilesystemLog
Retrieved Log records for a folder
FTHFileSystemLogItemData[] **GetFolderFilesystemLog** (Guid sessionID, int folderID)

Parameter Name	Description
sessionID	User session returned by one of the Open functions
folderID	ID of the folder for which the log data is retrieved

2.5 User and Group management

2.5.1 Data Structures

FTHUserData

FTHUserData		
Holds the data from user object		
Data Type	**Field Name**	**Description**
int	**UserID**	User ID in Thru
string	**Name**	User name in Thru, used for sign-in
string	**DomainUsername**	Username in the LAN Windows domain in the format domain\username
bool	**DomainUserFlag**	True if user is a domain user created by some Sync server
string	**FirstName**	User first name
string	**LastName**	User last name
string	**Email**	User email
int	**Role**	User role in Thru: Administrator/Manager/Member
bool	**Active**	Is user active
bool	**Deleted**	Is user deleted
string	**UserSyncID**	Sync ID that is created by synchronization services such as Active Directory integration/SSO Thru product.
string	**FullName**	Full user name, a string that contains both first and last name
string	**SortableName**	Name to be used in sorting
int	**HomeFolderID**	ID of the home folder

FTHUserGroupData

FTHUserGroupData		
Holds the data from User Group object		
Data Type	**Field Name**	**Description**
int	**UserGroupID**	User Group ID in Thru
string	**Name**	Group name in Thru

2.5.2 Methods

CreateDomainUser

CreateDomainUser	
Creates user on request from services that synchronize user list in corporate Active Directory with Thru user list.	
`FTHUserData CreateDomainUser (Guid sessionID, string domainUsername, string userSyncID, string firstName, string lastName, string email, bool createHomeFolder, int role, bool active )`	
Parameter Name	**Description**
`sessionID`	User session returned by one of the Open functions
`domainUsername`	Username in the LAN Windows domain in the format “domain\username”
`userSyncID`	Unique synchronization ID provided by external sync service, in case of Thru SSO service synching with Active Directory it is equal to user GUID from Active Directory.
`firstName`	user first name in corporate directory
`lastName`	User last name in corporate directory
`email`	User email in corporate directory. Mandatory parameter. This call will fail if email is not provided
`createHomeFolder`	Flag to create home folder, home folder name is derivative from user email address. ‘@’ sign is replaced by a dot

`role`	User role in Thru, allowed values:
	`Member = 3`
	`Manager = 4`
	`Administrator = 5`

`active`	Flag: if the user should be created active

GetUserByUserSyncID

GetUserByUserSyncID	
Get user by a user SyncID, used by external synchronization service like Thru integration with Active Directory/SSO service.	
`FTHUserData GetUserByUserSyncID (Guid sessionID, string userSyncID)`	
Parameter Name	**Description**
`sessionID`	User session returned by one of the Open functions
`userSyncID`	Unique synchronization ID provided by external sync service, in case of Thru SSO service synching with Active Directory it is equal to user GUID from Active Directory.

CreateUser

CreateUser	
Create a user in a Thru site.	
`FTHUserData` `CreateUser (Guid sessionID, string firstName, string lastName, string email, bool createHomeFolder, int role, bool active )`	
Parameter Name	**Description**
`sessionID`	User session returned by one of the Open functions
`firstName`	User first name in corporate directory
`lastName`	User last name in corporate directory
`email`	User email in corporate directory
`createHomeFolder`	Flag to create home folder
`role`	User role in Thru
`active`	Flag: if the user should be created active

DeleteUser

DeleteUser	
Deletes a user in Thru site by a user ID	
`void` `DeleteUser (Guid sessionID, int userID)`	
Parameter Name	**Description**
`SessionID`	User session returned by one of the Open functions
`UserID`	Thru user ID of the user to delete

FindUsers

FindUsers
Find users by search criteria

```
FTHUserData[]
FindUsers (Guid sessionID, string emailSearch, bool emailAll, string
nameSearch, bool nameAll, string fullNameSearch, bool fullNameAll, string
companyNameSearch, bool companyNameAll, bool includeDeleted)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
emailSearch	Search pattern for emails
emailAll	Flag to include all emails
nameSearch	Search pattern for user name
nameAll	Flag to include all user names
fullNameSearch	Search pattern for full name
fullNameAll	Flag to include all full names
companyNameSearch	Search pattern for company name
companyNameAll	Flag to include all company names
includeDeleted	Include deleted users into search results

GetUser

GetUser
Get user by a user ID

```
FTHUserData
GetUser (Guid sessionID, int userID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
userID	Thru user ID of the user to retrieve

GetUserGroup

GetUserGroup
Get user group data by a group ID, see data structure description

```
FTHUserGroupData
GetUserGroup(Guid sessionID, int userGroupID)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
userGroupID	ID of the user group to retrieve

SaveUser

SaveUser
Save user object data to Thru server

```
FTHUserData
SaveUser(Guid sessionID,  FTHUserData user)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
user	User data that should be saved, see data structure description

UndeleteUser

UndeleteUser
Undelete user is used in order to undelete previously deleted user instead of creating a new user with the same name and therefore preserve audit information. Previously existing home folder cannot be found since it has been unlinked, new home folder can be created.

```
FTHUserData
UndeleteUser (Guid sessionID, int userID, bool createHomeFolder, string
homeFolderName)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
userID	User ID of the user to be undeleted
createHomeFolder	Flag that defines if home folder should be created
homeFolderName	Name for the home folder if flag requires home folder creation

ChangeUserPassword

ChangeUserPassword
Change password for a user with the known session

```
void
ChangeUserPassword(Guid sessionID, int userID, string oldPassword, string
newPassword)
```

Parameter Name	Description
sessionID	User session returned by one of the Open functions
userID	User ID of the user for who the password will be changed
oldPassword	Old password
newPassword	New password

2.6 File and Folder Download workflows

File and folder downloads from Thru server should be implemented via HTTP GET to a URL ***servername/download.aspx***, this URL requires authentication via SessionID

- External application that connects to a Thru server should browse to a Thru folder that contains required files using web services. Once in the right folder, Web call *GetFiles* returns an array of FTHFileData objects, each object contains FileID for the files stored in requested parent folder. WS call *GetFolders* returns an array of FTHFolderData objects, each object contains FolderID for the folders stored in requested parent folder.

- **Single File download:** using HTTP library such as web client library included into .NET framework, external application issue the HTTP GET command to the URL ***servername/download.aspx?fileid=FileID&sessionid=SessionID*** , where FileID is retrieved by previous *GetFiles* WS call and SessionID parameter is returned by one of the Open() web service calls.
  In response Thru server starts writing selected file as a binary stream into HTTP client, the stream has to be saved into the local file system.
  It is highly recommended to open HTTPS connection to provide secure download.

- **Single Folder Download:** using HTTP library such as web client library included into .NET framework, external application issues the HTTP GET command to the URL ***servername/download.aspx?folderid=FolderID&sessionid=SessionID***, where FolderID is retrieved by previous *GetFolders* web call and SessionID parameter is returned by one of the Open() web service calls.

  In response Thru server creates an archive in a ZIP format that includes all subfolders of a selected folder and starts writing an archive file as a binary stream into HTTP client. The stream has to be saved into the local file. Archive file is named following the pattern **foldername.zip.**

  When external application has to download a folder, it is generally recommended to navigate down Thru folder tree and download files separately for better control at the file level, as opposed to downloading the whole folder that comes in a zipped form and has to be unzipped by a client application.

- **Downloading of multiple files and folders:** using HTTP library such as web client library included into .NET framework, external application issue the HTTP GET command to the URL ***servername/download.aspx?fileid=FileID1, …FileDN& folderid=FolderID1, …FolderDN&sessionid=SessionID*** , where FileID1, …FileIDN is a comma-separated list of File IDs returned by a previous call to *GetFiles* and FolderD1, …FolderDN is a comma separated list of folder IDs returned by a previous call to *GetFolders.*
  SessionID parameter is returned by one of the Open() web service calls.
  In response Thru server packages requested files and folders into ZIP archive named using the pattern "**THRU-XXXXXXXXXX.zip**" where XXX represents random combination of hexadecimal letters and digits, and starts writing the ZIP archive as a binary stream into HTTP client. The stream has to be saved into the local file system.
  It is highly recommended to open HTTPS connection to provide secure download.

  The following values of HTTP headers are returned in all cases:
  - Content-Type "application/binary"
  - Content-Disposition: attachment; filename=filename.ext (to save the file under original name)
  - Content-Length: <file size in bytes>

  In case of error the following custom HTTP header is returned that contains an error code and message:
  - "X-FTH-Error", error contents is ErrorCode and Error Message separated by pipe character: "ErrorCode | ErrorMessage"

2.7 File and Folder Upload workflows

2.7.1 Uploading a single file

File upload is implemented by a Thru server via HTTP POST command to a URL ***servername/upload.aspx*** that requires authentication via session ID. It is recommended to use HTTPS connection for security purposes.

HTTP POST command is used to send the parameters and upload files as part of multipart forms of the following type:

<FORM ENCTYPE="multipart/form-data" action=" ***servername/upload.aspx*** " method=post>
<INPUT NAME="file" TYPE="file">
<INPUT NAME="filename0" TYPE="hidden" value="filename.ext">
<INPUT NAME="Description0" TYPE="hidden" value="description_here">
<INPUT NAME=" SessionID" TYPE="hidden" value=sessionID>
<INPUT NAME=" folderID " TYPE="hidden" value= folderID >
<INPUT NAME=" EntityCount " TYPE="hidden" value = 1>
<INPUT NAME=" overwrite" TYPE="hidden" value=[1 or 0 here]>
<INPUT NAME=" NoRedirect " TYPE="hidden" value="any value here">

File/Folder upload process can be implemented using .NET framework web client library or any other file upload library like COM objects from third-party vendors.

Upload via HTTP POST should be used in conjunction with web services to log into the server, browse to the target location and retrieve information on FileIDs and FolderIDs.

Web client library used by external application should issue HTTP POST command to the page ***servername/upload.aspx*** with the following set of fields:

- 'file' contains file body. File size is limited to 2GB due to HTTP POST implementation by the browsers.
- 'filename0' (ends with zero): name that will be given to the file in Thru server.
- 'Description0' (ends with zero): name that will be given to the file in Thru server.
- 'SessionID' should be set to current sessionID, returned by one of the Open() web service calls.
- 'folderID' should be set to id of the target parent folder where a file should be uploaded. The folderID has to be retrieved via prior calls to web services such as GetFolders or GetFolder.
- "EntityCount" should be set to 1 since only one file is uploaded
- "overwrite" flag equals 1 means that existing file with the same name in the same parent folder should be overwritten, if set to 0, the new file name will be versioned to name[i]
- "NoRedirect" parameter has to be defined in the POST request and set to any value for web libraries that call Thru server 3rd party applications. **Please note this is mandatory parameter for external applications.**

The following values of HTTP headers are returned as result of upload:

In case of successful upload:
"X-FileID0" (ends with zero) contains File ID of newly created file in Thru server.

In case of error:

"X-FTH-Error", error contents is ErrorCode and Error Message separated by pipe character: “ErrorCode | ErrorMessage”

2.7.2 Uploading multiple files

To upload multiple files external application should follow the process specified in 2.7.1

2.7.3 Uploading Folder hierarchies

To upload the folder hierarchies to Thru server, external application should use connect to Thru server using one of Open() web service calls and use call CreateFolder() to create Thru folder. Each file should be uploaded according to the process specified in 2.7.1. There is no size limit on the folder upload except for 2GB upload per each file.

2.7.4 External Java upload API

Overview

Any 3rd party application can use Thru Java upload API for uploading files or folders to the Thru server. For that, 3rd party application should use page at URL like this https://thruservername/ExternalJavaUpload.aspx and provide parameters specified below.

Parameters

- ThruSessionID [mandatory] – Unique identifier of the current session
- ThruFolderID [mandatory] – Unique identifier of a folder on Thru server
- ThruSuccessUrl [mandatory] – Redirect URL that will be used on successful upload
- ThruErrorReportingUrl [mandatory] – Redirect URL that will be used if upload fails.
  If upload failed, server will return server error code (see below) using parameter “ThruErrorCode”.
  If an error occurred and ThruErrorReportingURL is not defined then server will throw an exception.
  Note: the value of ThruErrorReportingUrl is not validated. You should make sure that it is a valid URL that does exist when integrating Thru java upload..
- ThruFileFilter [optional] – string for file filtering used in file open dialog.
  String format: <Description> : <extension1>, <extension2> … <extensionN> .
  For example: String “Zip files (*.zip, *.7z, *.tar): zip, 7z, tar” will only match files with zip, 7z, tar extensions. Only these files can be uploaded. If parameter is not passed all files are shown. Frag and drop for the other file types is ignored.

Server error codes

These codes are passed as parameters to the ThruErrorReportingUrl in case any error occurs:

0 – Unknown error, no more information is available.
1 – SessionID is not a valid Guid variable.
2 – FolderID is not a valid integer variable.
3 – Unknown ThruSessionID. Provided session identifier does not exist on server

4 – Folder with specified ID does not exist.
5 – Upload failed. This error code is returned for any error on upload.

Sample

Javascript Sample

(Session and folder ID are fictitious)

```
<html>
<head>
   <script language="javascript" src="http://nccdev.thruinc.net/Scripts/jquery-1.3.2.min.js"
type="text/javascript"></script>
  <script language="javascript" type="text/javascript">
      $(function(){
         var session = '289BEE79-F1B6-4DDE-9A6B-DB5461780FCA';
         var url = "https://nccdev.thruinc.net/ExternalJavaUpload.aspx?";
         url += "ThruFolderID=" + "98";
         url += "&ThruSessionID=" + session;
         url += "&ThruSuccessUrl=" + escape("http://someserver/ExtUpload/success.html");
         url += "&ThruErrorReportingUrl=" + escape("http://someserver/ExtUpload/error.html");
         url += "&ThruFileFilter=" + "Zip files (*.zip, *.7z, *.tar): zip, 7z, tar";
         $("#f").attr("src", url);
      });

   </script>
</head>
<body>
   <iframe id="f" width="800px" height="800px" src=""></iframe>
</body>
</html>
```

URL Sample

(session and folderID are fictitious):
http://nccdev.thruinc.net/ExternalJavaUpload.aspx?ThruFolderID=1074&ThruSessionID=AF258C14-4425-4B30-834D-0EBD8EC55161&ThruSuccessUrl=http://www.thruinc.com"&ThruErrorReportingUrl=http://www.yahoo.com&ThruFileFilter="Zip files (*.zip, *.7z, *.tar): zip, 7z, tar"

2.8 Search

2.8.1 Data Structures

FTHFolderData

FTHFolderData		
Folder object data		
Data Type	**Field Name**	**Description**
int	**FolderID**	ID of a newly created folder

int	**ParentFolderID**	ID of a parent folder
string	**Name**	Name of new folder
int	**EffectivePermission**	Effective access permission for new folder for currently signed user Permission enum: None = 0x00, //0 Show = 0x01, //1 Read = 0x02, //2 Create = 0x04, //4 Download = 0x08, //8 Distribute = 0x010, //16 Manage = 0x020, //32 Modify = 0x040, //64 Delete = 0x080, //128 FullControl = Read | Create | Download | Distribute | Manage | Modify | Delete
DateTime	**DateCreated**	Timestamp when the folder was created
DateTime	**DateModified**	Timestamp when the folder was modified
int	**CreatedByUserID**	ID of the user who created the file
int	**ModifiedByUserID**	ID of the user who modified the file
bool	**IsDeleted**	Flag shows if the folder is deleted

int	**FolderType**	One of the following Folder Types:
		`Regular = 0,`
		`Root = 1,`
		`UserHome = 2,`
		`UserDocuments = 3,`
		`UserDropBox = 4,`
		`FolderRoot = 5,`
		`HomeRoot = 6,`

FTHFolderSearchPagingData

FTHFolderSearchPagingData		
Paging data for search results		
Data Type	**Field Name**	**Description**
string	**SearchString**	Search pattern for folders, use asterisk symbol for wildcard
int	**RequestedStartIndex**	Starting index to return a page of results
int	**RequestedLength**	Number of results on the page
int	**TotalMatchingFolders**	Total count of matched folders
FTHFolderData[]	**PageFolders**	Array of folder descriptors returned by the search

FTHFileData

FTHFileData		
File object data		
Data Type	**Field Name**	**Description**
int	**FileID**	ID of a file in Thru server
int	**ParentFolderID**	ID of a parent folder that contains a file
string	**Name**	Name of a file
long	**Size**	File size
int	**EffectivePermission**	Effective permission of a file (reserved for future use)
DateTime	**DateCreated**	Date when a file was created
DateTime	**DateModified**	Date when a file was modified last time
int	**CreatedByUserID**	ID of the user who created the file
Int	**ModifiedByUserID**	ID of the user who modified the file last time
int	**Version**	File version (reserved for future use)
bool	**IsDeleted**	Flag shows if a file is in Deleted state
bool	**IsQuarantined**	Flag shows if a file is in Quarantined state, means that file body was removed by Anti-virus software.
bool	**IsPublished**	Flag shows that a file is published into one of Thru collections.

FTHFileSearchPagingData

FTHFileSearchPagingData		
Paging data for file searches		
Data Type	**Field Name**	**Description**
string	**SearchString**	Search pattern for folders, use asterisk symbol for wildcard
int	**RequestedStartIndex**	Starting index to return a page of results
int	**RequestedLength**	Number of results on the page
int	**TotalMatchingFiles**	Total count of matched files
FTHFileData []	**PageFiles**	Array of file descriptors returned by the search

2.8.2 Methods

SearchForFiles

SearchForFiles	
Search for files in Thru server by a file name. Uses asterisk as a wildcard character.	
`FTHFileData[] SearchForFiles(Guid sessionID, string searchString)`	
Parameter Name	**Description**
`sessionID`	User session returned by one of the Open functions
`searchString`	Search string, use asterisk symbol for a wildcard.

SearchForFiles2

SearchForFiles2
Search for files in Thru server by a file name. Uses asterisk as a wildcard character. Return results in pages.
`FTHFileSearchPagingData SearchForFiles2(Guid sessionID, string searchString, int index, int length)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`searchString`	Search string, use asterisk symbol for a wildcard.
`index`	Return search results starting with this index
`length`	Number of returned search results

SearchForFolders

SearchForFolders
Search for folders in Thru server by a folder name. Uses asterisk as a wildcard character.
`FTHFileData[]SearchForFolders(Guid sessionID, string searchString)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`searchString`	Search string, use asterisk symbol for a wildcard.

SearchForFolders2

SearchForFolders2
Search for folders in Thru server by a folder name. Uses asterisk as a wildcard character. Return results in pages.
`FTHFileSearchPagingData SearchForFolders2(Guid sessionID, string searchString, int index, int length)`

Parameter Name	Description
`sessionID`	User session returned by one of the Open functions
`searchString`	Search string, use asterisk symbol for a wildcard.
`index`	Return search results starting with this index
`length`	Number of returned search results

AdvancedSearchForFiles

AdvancedSearchForFiles
Advanced search for files in Thru file system using combination of search parameters.

```
FTHFileData[] AdvancedSearchForFiles(Guid sessionID, string keywordSearch,
bool keywordAll, string nameSearch, bool nameAll, string descriptionSearch,
bool descriptionAll, string modifiedBySearch, DateTime? startDate,
DateTime? endDate, int? folderID, bool recursive, int index, int length)
```

Parameter Name	Description
sessionID	User session returned by one of Open functions
keywordSearch	String contains tags (keywords) separated by space
keywordAll	Set the flag to true if all tags are required, otherwise search returns a union of searches for each tag
nameSearch	String contains tokens to search in file name, separated by space.
nameAll	Set the flag to true if all tokens are required in the name, otherwise search returns a union
descriptionSearch	String contains tokens to search in file description, separated by space.
descriptionAll	Set the flag to true if all tokens are required in a description, otherwise search returns a union
modifiedBySearch	Name of a user who was the last to modify the file
startDate	Search the files with modification date later then startDate, optional
endDate	Search the files with modification date earlier then endDate, optional
folderID	Search only in the folder with this ID, optional
recursive	Flag is set to run recursive search on subfolders
index	Search returns results starting with this index
length	Number of found items to return

AdvancedSearchForFolders

AdvancedSearchForFolders
Advanced search for files in Thru file system using combination of search parameters.

```
FTHFolderData[] AdvancedSearchForFolders(Guid sessionID, string
keywordSearch, bool keywordAll, string nameSearch, bool nameAll, string
descriptionSearch, bool descriptionAll, string modifiedBySearch, DateTime?
startDate, DateTime? endDate, int? folderID, bool recursive, int index, int
length)
```

Parameter Name	Description
sessionID	User session returned by one of Open functions
keywordSearch	String contains tags (keywords) separated by space
keywordAll	Set the flag to true if all tags are required, otherwise search returns a union of searches for each tag
nameSearch	String contains tokens to search in file name, separated by space.
nameAll	Set the flag to true if all tokens are required in the name, otherwise search returns a union
descriptionSearch	String contains tokens to search in file description, separated by space.
descriptionAll	Set the flag to true if all tokens are required in a description, otherwise

	search returns a union
`modifiedBySearch`	Name of a user who was the last to modify the file
`startDate`	Search the files with modification date later then `startDate, optional`
`endDate`	Search the files with modification date earlier then `endDate, optional`
`folderID`	Search only in the folder with this ID, optional
`recursive`	Flag is set to run recursive search on subfolders
`index`	Search returns results starting with this index
`length`	Number of found items to return

2.9 Secure Message

2.9.1 Messaging Data Structures

FTHMsgStorageData

FTHMsgStorageData		
Data location in Thru site to tore outgoing messages		
Data Type	**Field Name**	**Description**
int	**OutboxFolderID**	Folder to store local attachments and files from SharePoint (Email Thru action).See more info in review of web method **MsgGetStorage()**
int	**AuxFileSetID**	Optional field to store the ID of the set that link all the auxiliary files associated with the message.
FTHAuxFileData	**MissingAuxFiles**	After checksum check – list of files which do NOT exist on the server and need to be uploaded.

FTHAuxFileData

FTHAuxFileData		
Auxiliary files data: thumbnails and other		
Data Type	**Field Name**	**Description**
FTHAuxFileType	**FileType**	Type of auxiliary file. Currently, the only possible value is “MessageEmbeddedImage”.
String	**FileName**	Auxiliary file name
String	**Checksum**	SHA1 file checksum.

FTHMsgData

FTHMsgData		
Thru email message data		
Data Type	**Field Name**	**Description**
int	**MsgID**	Message unique identifier.
int	**UserID**	Sender.
string	**Subject**	Message subject
DateTime?	**DateSent**	Timestamp when email was sent to last recipient.
DateTime	**DateExpired**	Message expiration timestamp.
bool	**CollectedFlag**	Flag to indicate if any of the recipients read the private message or downloaded files.
bool	**NotifyFlag**	A flag which can be set by the sender to trigger email notification to the sender when a recipient downloads the file(s).
string	**PublicMessage**	Public message body
string	**PrivateMessage**	Private message body
string	**TrackingNo**	Tracking number assigned by Thru server to any email. Enables search of secure emails and related events in Thru Audit Manager.
bool	**SendByClient**	True if client application such as Outlook or Lotus Notes add-on is responsible for sending of the

		message. False if the server sends the message.
bool	**RequestLogin**	Indicates if recipient should go through self-registration process to set the password to access protected emails and files.
Int	**MaxDownloads**	Reserved for future use.
DateTime	**DateCreated**	Timestamp of the creation date for the message in Thru server. Could be earlier then Send timestamp since it takes time for the client to create individually trackable messages in Outbox.
int	**ApplicationID**	ID of the application connector that created the message. Application IDs are stored in the table TBL_APPLICATION
FTHMailPriority	**Priority**	Message priority. High = 1, Low = 2, Normal = 3
bool	**IsExpired**	Indicates if the message is still available for external download. Expired messages are visible in Sent Thru folder in native or web clients.
FTHMsgRecipientData[]	**ToRecipients**	List of message recipient IDs.
FTHMsgRecipientData[]	**CcRecipients**	List of message recipient IDs.
FTHMsgRecipientData[]	**BccRecipients**	List of message recipient IDs for hidden recipients.
FTHFolderData[]	**FolderData**	List of folder IDs.
FTHFileData[]	**FileData**	List of file IDs.
bool	**SeparateEmails**	Flag indicates whether to send separate email per recipient or a single email to all recipients.
bool	**AttachmentsExist**	Flag indicates if any attachments exist.
string	**MsgHtmlTemplate**	Template used when a message is prepared, HTML format. More info in “Message Templates” section.
string	**MsgTextTemplate**	Template used when a message is prepared, plain text format. More info in “in “Message Templates” section.
string	**UserPassword**	For backward compatibility - this is the password created by a sender when a message is created. Thru Clients that support this mode: Outlook 1.x, Lotus Notes 1.x
string	**MsgLevelPublicID**	If a single email is sent to all recipients, use this field as invitation ID that identifies the message for all recipients.
int	**MsgTemplateID**	This is the template Id for the message to send an email to recipient. Template with this ID participates in the merge process when original email merges with template in Outlook or other client.

FTHMsgRecipientData

FTHMsgRecipientData		
Message recipient data		
Data Type	**Field Name**	**Description**
int	**MsgID**	Unique identifier of the message.
int	**RecipientID**	Unique identifier of the recipient for the message.
string	**RecipientEmail**	Recipient email address.
string	**MsgRecipientLevelPublicID**	String to be passed with external download URL to identify the message and the recipient.
int	**Type**	TO, CC, or BCC
bool	**Collected**	Flag indicates the recipient read the message or downloaded the files.

FTHMsgValidationData

FTHMsgValidationData		
Store message in case of failed validation		
Data Type	**Field Name**	**Description**
bool	**Valid**	Set if the message is considered valid under the current site policy.
string	**UserMessage**	In case the message is not valid, the field contains the message to display, explaining why the message violates the site policy.

FTHFileData

FTHFileData		
Thru file object data		
Data Type	**Field Name**	**Description**
int	**FileID**	ID of a file in Thru server
int	**ParentFolderID**	ID of a parent folder that contains a file
string	**Name**	Name of a file
long	**Size**	File size
int	**EffectivePermission**	Effective permission of a file (reserved for future use)
DateTime	**DateCreated**	Date when a file was created
DateTime	**DateModified**	Date when a file was modified last time
int	**CreatedByUserID**	ID of the user who created the file
Int	**ModifiedByUserID**	ID of the user who modified the file last time
int	**Version**	File version (reserved for future use)
bool	**IsDeleted**	Flag shows if a file is in Deleted state
bool	**IsQuarantined**	Flag shows if a file is in Quarantined state, means that file body was removed by Anti-virus software.
bool	**IsPublished**	Flag shows that a file is published into one of Thru collections.

FTHFolderData		
Thru folder object data		
Data Type	**Field Name**	**Description**
int	**FolderID**	ID of a newly created folder
int	**ParentFolderID**	ID of a parent folder
string	**Name**	Name of new folder
int	**EffectivePermission**	Effective access permission for new folder for currently signed user Permission enum: None = 0x00, //0 Show = 0x01, //1 Read = 0x02, //2 Create = 0x04, //4 Download = 0x08, //8 Distribute = 0x010, //16 Manage = 0x020, //32 Modify = 0x040, //64 Delete = 0x080, //128 FullControl = Read | Create | Download | Distribute | Manage | Modify | Delete
DateTime	**DateCreated**	Timestamp when the folder was created
DateTime	**DateModified**	Timestamp when the folder was modified
int	**CreatedByUserID**	ID of the user who created the file
int	**ModifiedByUserID**	ID of the user who modified the file
bool	**IsDeleted**	Flag shows if the folder is deleted

int	**FolderType**	One of the following Folder Types:
		`Regular = 0,`
		`Root = 1,`
		`UserHome = 2,`
		`UserDocuments = 3,`
		`UserDropBox = 4,`
		`FolderRoot = 5,`
		`HomeRoot = 6,`

2.9.2 Messaging methods

MsgGetStorage

GetMsgStorage	
Returns information on the location where the message should be stored in Thru site.	
`FTHMsgStorageData` **`MsgGetStorage`**`(Guid sessionID, string msgSubject, FTHAuxFileData[] auxFiles)`	
Parameter Name	**Description**
`sessionID`	User session returned by one of Open functions
`msgSubject`	If a message contains local attachments Thru server creates a subfolder under

	Outbox root. First 50 characters of subject are used and versioning is applied if needed.
auxFiles	List of auxiliary files to be added to the storage. The method will return a subset of these files in the FTHMsgStorageData result, within the MissingAuxFiles property. This subset will contain only the files that are missing in the server and, therefore, need to be uploaded.

MsgCreate

Create Message

Create the message on the server. For thin clients like web browsers email is sent by the server, for thick email clients a link to secure distribution is returned. Link will replace original attachments.

```
FTHMsgData
MsgCreate (Guid sessionID, bool sendByClient, FTHMailPriority priority,
string subject, string publicMessage, publicMessageIsHtml, string
privateMessage, bool privateMessageIsHtml, bool requestLogin, DateTime
dateExpired, bool notifyFlag, int maxDownloads, string[] toEmailList,
string[] ccEmailList, string[] bccEmailList, int[] folderIDList, int[]
fileIDList, int? auxFileSetID)
```

Parameter Name	Description
sessionID	User session returned by one of Open functions
sendByClient	True if the client is responsible to send the mail. False if the server sends it.
priority	Message priority.
subject	Message subject.
publicMessage	Message body.
publicMessageIsHtml	Tells the server if public message body is in HTML format, reserved for future use
privateMessage	Private message body.
privateMessageIsHtml	Tells the server if private message body is in HTML format, reserved for future use
requestLogin	Whether the recipient will have to go through self password process.
dateExpired	Message expiration date.
notifyFlag	Flag that tells the server to send notifications when the files are downloaded or private message is read, ON by default.
maxDownloads	Maximal number of downloads for each file separately or as a package allowed to recipients. Value (-1) means unlimited downloads are allowed.
toEmailList	Array of To recipient list
ccEmailList	Array of Cc recipient list
bccEmailList	Array of Bcc recipient list
folderIDList	Array of attached folders
fileIDList	Array of attached files
auxFileSetID	Optional field linking message embedded images

MsgValidate

Validate Message
Validate combination of parameters and policies for the email. Will return error and message with explanation if parameters/policies are incompatible.
FTHMsgValidationData **MsgValidate**(Guid sessionID, bool sendByClient, FTHMailPriority priority, bool isPrivateMessage, bool requestLogin, DateTime dateExpired, bool notifyFlag, int maxDownloads, int recipientCount, bool isAttachments)

Parameter Name	Description
sessionID	User session returned by one of Open functions
sendByClient	True if the client is responsible to send the mail. False if the server sends it.
priority	Message priority.
isPrivateMessage	Indicates whether the message has a private message body.
requestLogin	Whether the recipient will have to go through self password process.
dateExpired	Message expiration date.
notifyFlag	Flag that tells the server to send notifications when the files are downloaded or private message is read, ON by default.
maxDownloads	Reserved for future use.
recipientCount	The total number of the message recipients.
isAttachments	Indicates whether the message contains (file or folder) attachments.

Message Discarded

Discard Client Side Msg
Discard message before it is sent. Discard deletes the message and attachments stored in in Thru site. Method should not be called after the message was sent since it invalidates the link to the files.
void **MsgDiscard**(Guid sessionID, int msgID)

Parameter Name	Description
sessionID	User session returned by one of Open functions
msgID	Message identifier.

MsgComplete

Complete Client Side Msg
Mark message as complete after the message is discarded
void Msg**Complete**(Guid sessionID, int msgID)

Parameter Name	Description
sessionID	User session returned by one of Open functions
msgID	Message identifier.

MsgGetSentThruList

Get Msg Sent Thru List	
Returns a list of sent Thru messages	
FTHMsgData[] **MsgGetSentThruList**(Guid sessionID, string searchValue, DateTime? startDate, DateTime? endDate, int? pagingLastMsgID, int pagingMsgLimit)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
searchValue	Filter messages contain either subject or tracking number or private or public message .
startDate	Filter messages sent later than given date.
endDate	Filter messages sent earlier than given date.
pagingLastMsgID	Give null for first page. Give last MsgID to get next page.
pagingMsgLimit	Limit number of returned messages.

MsgGet

Get Msg	
Retrieve the message object from the server	
FTHMsgData[] **MsgGet**(Guid sessionID, int msgID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgID	Message ID.

MsgDelete

Msg Delete	
Delete the message from the server	
MsgDelete(Guid sessionID, int msgID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgID	Message ID.

MsgDeleteFile

Msg Delete File	
Detach the file from the message, file is not deleted from Thru file system	
MsgDeleteFile(Guid sessionID, int msgID, int fileID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgID	Message ID.
fileID	File ID.

MsgDeleteFolder

Msg Delete Folder	
Detaches the folder from the message, folder is not deleted from Thru file system	
MsgDeleteFolder(Guid sessionID, int msgID, int folderID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgID	Message ID.
folderID	Folder ID.

MsgExpire

Msg Expire	
Sets message expiration to certain date. Expired message cannot be accessed by any recipients.	
MsgExpire(Guid sessionID, int[] msgIDList, DateTime utcDateExpired)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgIDList	List of Message ID to set expired date.
utcDateExpired	Expired Date to be set.

MsgExpireNow

Msg Expire Now	
Expires message immediately. Expired message cannot be accessed by any recipients.	
MsgExpireNow(Guid sessionID, int[] msgIDList)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
msgIDList	List of Message ID to set expired date.

MsgDeleteRecipient

Msg Delete Recipient	
Delete a specific recipient from a message.	
MsgDeleteRecipient(Guid sessionID, int recipientID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
recipientID	Recipient ID to delete.

MsgGetCount

Msg Get Count	
Retrieves the count of messages for the current user.	
MsgGetCount(Guid sessionID)	
sessionID	User session returned by one of Open functions

2.10 Publishing

2.10.1 Data Structures

FTHPublishCollectionData

FTHPublishCollectionData		
Collection that contains links to published files, links to other collections and links to external sites.		
Data Type	**Field Name**	**Description**
int	**PublishCollectionID**	Unique ID of collection
int	**SiteID**	ID of the site that contains the collection
string	**CollectionTitle**	collection title
string	**Description**	Text that describes the collection
string	**LinksSubtitle**	Subtitle of the section that displays links to files
string	**RelatedPagesSubtitle**	Subtitle of the section that displays links to other web pages.
string	**HeaderText**	Text of collection web page header
string	**FooterText**	Text of collection web page footer
bool	**Active**	Flag is set when collection is active
DateTime	**DateCreated**	Timestamp of creation
Int	**CreatedByUserID**	User ID of creator
DateTime	**DateModified**	Timestamp of modification
int	**ModifiedByUserID**	ID of the user who modified the collection recently
DateTime	**DateDeleted**	Timestamp shows when collection was deleted. NULL if collection is not deleted.
bool	**RequireRegistration**	Flag is set if file links in the collection require registration.
bool	**CollectionActive**	Flag is set if collection is active, same as Active.
bool	**CollectionVisible**	Flag set if collection is visible via Thru publishing URL
bool	**SearchVisible**	Flag set if search field is visible in collection
bool	**SearchShowResultAtStart**	Flag set if collection shows all links on start when search field is empty.

FTHPublishRequestData

FTHPublishRequestData		
Request to publish the file		
Data Type	**Field Name**	**Description**
int	**PublishRequestID**	Unique ID of request to publish the files

`FTHPublishRequestState`	**PublishRequestState**	State of publishing request:
		`Pending = 1,`
		`Rejected = 2,`
		`Approved = 3`

int	**SiteID**	ID of Thru site that contains the collection.
int	**PublishCollectionID**	ID of the target collection where file(s) will be published if approved
string	**CollectionTitle**	Title of the target collection
int	**PublishLinkID**	ID of the file link object that points to the file
string	**RequestText**	Text description of the request for publishing
bool	**UseDefaultLinkInfo**	Flag set if default link text and description for the file link is used. By default link text is set to file name, link description is empty. Once default values are set for link and description can be reused with each next publishing of the file.
string	**LinkText**	Text of the hyperlink that points to the file
string	**LinkDescription**	Description of the hyperlink that points to the file, displayed below the link
DateTime	**DateCreated**	Creation timestamp
int	**CreatedByUserID**	User ID of request creator
DateTime	**DateModified**	Timestamp of last modification
int	**ModifiedByUserID**	ID of s user who modified the request last
string	**ProcessingNote**	Note for approver

FTHPublishLinkData

FTHPublishLinkData		
Published File Link object		
Data Type	**Field Name**	**Description**
int	**PublishLinkID**	Unique ID of this File Link object. Object is used to store text, URL, description and tags associated with the published file. It is different from Thru File object.
int	**SiteID**	ID of Thru site that contains the object
int	**RelatedFileID**	ID of the file published via link object
string	**InternetUrl**	Url to the object
string	**DefaultLinkText**	Default text for hyperlink
string	**DefaultDescription**	Default description for file link
DateTime	**DateCreated**	Timestamp of date created
int	**CreatedByUserID**	ID of creator
DateTime	**DateModified**	Timestamp of last modification
int	**ModifiedByUserID**	ID of user who modified object last
`FTHPublishTagData[]`	**Tags**	List of tags

FTHPublishCollectionLinkData

FTHPublishCollectionLinkData		
Object: link to publishing collection		
Data Type	**Field Name**	**Description**
int	**PublishCollectionLinkID**	ID of this Link object. Object is used to store all information associated with link to an item inside the collection. Item can be link to file, link to another collection or URL of external page.
int	**SiteID**	ID of Thru site that contains the collection link
int	**PublishCollectionID**	ID of the collection that contains the item
int	**PublishLinkID**	ID of the item, applies to files and collections so far
`FTHPublishLinkType`	**PublishLinkType**	Supported item types File = 1, Collection = 2, Internet = 3
int	**RelatedFileID**	If item link pints to a file, this field contains file ID
int	**RelatedPublishCollectionID**	If item link points to collection, this field contains collection ID
string	**InternetUrl**	If item link points to external URL, it is stored here.
bool	**UseDefaultLinkInfo**	Flag is set of default information should be used for the link.
string	**LinkText**	Default name for the link.
string	**LinkDescription**	Text that describes the link.
string	**CollectionTitle**	Title of collection
bool	**CollectionVisible**	Is collection visible via URL
bool	**CollectionActive**	Is collection active
bool	**CollectionLinkActive**	Is the link to collection active
bool	**RelatedCollectionActive**	Are links to other collections contained in this collection active
bool	**RelatedCollectionVisible**	Are links to other collections contained in this collection visible
DateTime	**DateCreated**	Timestamp when link was created in collection
int	**CreatedByUserID**	User ID who created the link
string	**CreatedByUserName**	User Name who created the link
int	**RequestedByUserID**	User ID who created publishing request
string	**RequestedByUserName**	User Name who created publishing request.
DateTime	**DateModified**	Timestamp of the last link modification
int	**ModifiedByUserID**	User ID who last modicied the link
string	**ModifiedByUserName**	User name who last modified the link.

FTHPublishTagData

FTHPublishTagData		
Data : text tag attached to published file		
Data Type	**Field Name**	**Description**
int	**PublishTagID**	ID of the tag
string	**TagName**	Text of the tag

FTHPublishFileData

FTHPublishFileData		
Publishing data for a file		
Data Type	**Field Name**	**Description**
FTHFileData	**FileInfo**	File descriptor
FTHPublishLinkData	**FileLink**	File link descriptor
FTHPublishCollectionLinkData [ ]	**PublishingCollections**	Array of published links into different collections
FTHPublishRequestData [ ]	**PendingRequests**	Array of pending requests

FTHPublishStorageData

FTHPublishStorageData		
Storage for published files in case if publishing occurs from external application like SharePoint		
Data Type	**Field Name**	**Description**
int	**ApplicationID**	ID of external application that requested publishing
int	**ExternalMajorLocationID**	External ID of external top level location, like SharePoint site
string	**ExternalMajorLocationName**	Name of external top level location like SharePoint site
int	**ExternalMinorLocationID**	ID of external low level location like SharePoint library
string	**ExternalMinorLocationName**	Name of external low level location like Sharepoint library
int	**ExternalFileID**	ID of the file in external system
int	**PublishingFolderID**	ID of the folder to store published files uploaded from external system
int	**ThruFileID**	ID of the file in Thru

2.10.2 Publishing methods

PublishingGetStorage

Get Publishing Storage
Return the storage location on Thru server where file published from external application will be stored.
`FTHPublishStorageData PublishingGetStorage(Guid sessionID, string externalMajorLocationID, string externalMajorLocationName, string externalMinorLocationID, string externalMinorLocationName, string externalFileID)`
This method accepts 3 parameters: - Location Major: General identification of external system (such as Sharepoint site url). - Location Minor: Detailed identification of file library - External File ID If ExternalFileID is found In Thru file system publishing area, this method will return Thru parent folder IDand Thru file ID. If not than the method will return only parent folder ID for uploading the file and returned Thru file ID will be null.

Parameter Name	Description
`sessionID`	User session returned by one of Open functions
`externalMajorLocationID`	For Sharepoint Integration: Sharepoint site Url
`externalMajorLocationName`	For Sharepoint Integration: Sharepoint site name
`externalMinorLocationID`	For Sharepoint Integration: Sharepoint library ID
`externalMinorLocationName`	For Sharepoint Integration: Sharepoint library name
`externalFileID`	For Sharepoint Integration: Sharepoint file ID

CreateFileExternalSourceMapping

Create File External Source Mapping
Binds Id of the file in Thru server with external IDs of 2 levels to identify the target file on Thru site for overwrite operations when the source file in external system is overwritten.
`void CreateFileExternalSourceMapping (Guid sessionID, int thruFileID, string externalMajorLocationID, string externalMajorLocationName, string externalMinorLocationID, string externalMinorLocationName, string externalFileID, string externalFileName)`

Parameter Name	Description
sessionID	User session returned by one of Open functions
thruFileID	Thru File
`externalMajorLocationID`	For Sharepoint Integration: Sharepoint site Url
`externalMajorLocationName`	Name of Sharepoint Application
`externalMinorLocationID`	For Sharepoint Integration: Sharepoint library ID
`externalMinorLocationName`	Name of Document Library

externalFileID	For Sharepoint Integration: Sharepoint file ID
externalFileName	Sharepoint file name

PublishingGetFileInfo

PublishingGetFileInfo	
Retrieve information on the publishing status of the file	
`FTHPublishFileData PublishingGetFileInfo` `(Guid sessionID, int fileID)`	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
fileID	ID of the file in Thru database

PublishingCreateRequest

PublishingCreateRequest	
Create request to publish the file	
`FTHPublishRequestData PublishingCreateRequest` `(Guid sessionID, int publishLinkID, int publishCollectionID, bool useDefaultLinkInfo, string overWriteLinkText, string overWriteDescription, string requestText )`	
Parameter Name	**Description**
`sessionID`	User session returned by one of Open functions
`publishLinkID`	ID of the file in Thru server file system that is requested to be published. In the future could be used for a folder publishing.
`publishCollectionID`	Collection that should contain the published file
`useDefaultLinkInfo`	If set, use default information for the published link: Name, Description
`overWriteLinkText`	Text to overwrite default link name
`overWriteDescription`	Text to overwrite default Description
`requestText`	Text of request.

PublishingDeletePendingRequest

Publishing Delete Pending Request	
`Delete pending publishing request`	
`Void PublishingDeletePendingRequest` `(Guid sessionID, int publishingRequestID)`	
Parameter Name	**Description**
`sessionID`	User session returned by one of Open functions
`publishingRequestID`	ID of the pending publishing request

PublishingSavePendingRequest

Save Changes to Pending Publishing Request	
Saves changed information to a pending publishing request.	
void PublishingSavePendingRequest (Guid sessionID, FTHPublishRequestData requestData)	
Parameter Name	**Description**
sessionID	Session ID for the current user
requestData	Data for the pending request that will be saved

PublishingGetCollections

Publishing Get Collection	
method returns collection list which is used in collection dropdown in new request form	
FTHPublishCollectionData[] **PublishingGetCollections**(Guid sessionID, int publishLinkID)	
Parameter Name	**Description**
sessionID	User session returned by one of Open functions
publishLinkID	ID of the collection link object

PublishingTagDictionaryGetItems

Publishing Tag Dictionary Get Items	
Retrieve tags from publishing Tag dictionary	
FTHPublishTagData[] **PublishingTagDictionaryGetItems**(Guid sessionID)	
sessionID	Session ID for the current user returned by one of the Open functions.

PublishingTagDictionaryAddItems

Publishing Tag Dictionary Add Items	
Add tags to dictionary	
void PublishingTagDictionaryAddItems(Guid sessionID, string commadDelimitedTagList)	
Parameter Name	**Description**
sessionID	Session ID for the current user returned by one of the Open functions.
commadDelimitedTagList	New tags (Duplicates and existing tags are silently ignored)

PublishingTagDictionaryRemoveItems

Publishing Tag Dictionary Remove Items	
Remove items from publishing Tag dictionary	
`void PublishingTagDictionaryRemoveItems(Guid sessionID, string commadDelimitedTagList)`	
Parameter Name	**Description**
`sessionID`	Session ID for the current user returned by one of the Open functions.
`commadDelimitedTagList`	Tags to remove (Non existing tags are silently ignored).

PublishingSaveLink

Publishing Save Link	
Use this method to update the default link, description and tags. Link is a virtual entity that holds file metadata and is pointed to by 'Collection Links' which are links to the file that exist in the context of a collection.	
`void PublishingSaveLink(Guid sessionID, FTHPublishLinkData publishLinkData)`	
Parameter Name	**Description**
`sessionID`	Session ID for the current user returned by one of the Open functions.
`publishLinkData`	Data for the publishing file link

PublishingDeleteCollectionLink

Publishing Delete Collection Link	
Use this method to delete the collection link	
`void PublishingDeleteCollectionLink(Guid sessionID, int publishCollectionLinkID)`	
Parameter Name	**Description**
`sessionID`	Session ID for the current user returned by one of the Open functions.
`publishCollectionLinkID`	ID of the publishing collection link

PublishingSaveCollectionLink

Publishing Save Collection Link
Use this method to save the changed data for ‘collection link’ which is the file link existing in the context of collection

```
void
PublishingSaveCollectionLink (Guid sessionID, FTHPublishCollectionLinkData
collectionLinkData)
```

Parameter Name	Description
`sessionID`	User session
`collectionLinkData`	Collection data to save

2.11 Mirroring (SyncThru)

2.11.1 Data Structures

FTHMirrFolderSyncInfoData

FTHMirrFolderSyncInfoData		
Data on the file in Outbound folder		
Data Type	**Field Name**	**Description**
int	**SourceFileID**	ID of the file that will be mirrored in a source location
string	**FileName**	Name of the file
string	**Description**	Description of the file
long	**Size**	Size of the file
DateTime	**DateModified**	Date last modified, UTC
DateTime	**LocalDateModified**	Date last modified, local time
bool	**IsQuarantined**	Flag is set if file is quarantined by anti-virus

2.11.2 Methods

GetMirroringTargetFolderSyncInfo

GetMirroringTargetFolderSyncInfo	
Retrieves mirroring target folder information for synchronization process.	
`FTHMirrFolderSyncInfoData[] GetMirroringTargetFolderSyncInfo(Guid sessionID, int sourceFolderID)`	
Parameter Name	**Description**
`sessionID`	Session opened for the current user by one of the Open web calls.
`sourceFolderID`	ID of the source folder in the source Thru site.

2.12 Audit Manager report for SharePoint

2.12.1 Data structures

FTHAuditMgrPagingStateData

FTHAuditMgrPagingStateData		
Data structure contains the current state of the paging		
Data Type	**Field Name**	**Description**
int	**KeysToSkip**	Key field is assumed to be integer and the first column in result set.
int	**RowsOnCurrentPage**	Number of rows on the page
int	**MatchingKeys**	Number of matching keys
Int	**PageNumber**	Page number
int	**TotalPages**	Total pages
bool	**IsFirstPageEnabled**	Is first page available in this report
bool	**IsPreviousPageEnabled**	Is previous page available
bool	**IsNextPageEnabled**	Is next page available
bool	**IsLastPageEnabled**	Is last page available
string	**ReportPageResult**	Report in XML format

FTHAuditMgrSharePointReportFilterData

FTHAuditMgrSharePointReportFilterData		
filtering data to retrieve report		
Data Type	**Field Name**	**Description**
string[]	**Operations**	Operations to include into report
DateTime	**DateLoggedFrom**	Start of operation in the log
DateTime	**DateLoggedTo**	End of operation in the log
string	**Operations**	File name
string	**OperationNotes**	Notes on operation
string	**MsgTrackingNo**	Unique Tracking number of the message
string	**MsgSubject**	Message subject
string	**MsgCoverOrPrivateContent**	Public or private content of the message
int	**KeysPerPage**	Report items per page, default is 20
FTHAuditMgrPagingAction	**PagingAction**	Enumeration FirstPage, PreviousPage, NextPage, LastPage
FTHAuditMgrSortOrder	**SortOrder**	Enumeration: Ascending, Descending
FTHAuditMgrPagingStateData	**PagingStateData**	State of the paging, see data structure above

2.12.2 Methods

AuditMgrSharePointReport

AuditMgrSharePointReport	
The method returns a report in XML format from Thru file system transaction log	
FTHAuditMgrPagingStateData **AuditMgrSharePointReport**(Guid sessionID, **FTHAuditMgrSharePointReportFilterData** filterData)	
Parameter Name	**Description**
ApplicationID	Filter only messages from given application.
filterData	Report filter and state of pagination.

2.13 Miscellaneous methods

GetTime

GetTime
Returns UTC time from the server.
`DateTime GetTime()`

ProbeSession

ProbeSession	
Checks if the session exists	
`bool ProbeSession(Guid sessionID)`	
Parameter Name	**Description**
`sessionID`	Session opened for the current user by one of the Open web calls.

2.14 Deprecated methods

FTHServerData

FTHServerData		
Data Type	**Field Name**	**Description**
FTHSecureMessageType	**SecureMessageType**	Type of secure message, enumeration. None=0, Regular = 1
bool	**SecureSite**	Is the site secure: secure site supports SSL connection.

GetServer

GetServer
`FTHServerData GetServer ()`

About Trademarks and Abbreviations

Trademarks

THRU, the THRU logo, are registered trademarks of THRU Inc.
Microsoft, Windows, Vista, and Outlook are trademarks of Microsoft Corporation in the United States and/or other countries.
Other company or product names used in this manual and not included in the list above are trademarks of those companies and may be registered in certain jurisdictions.

909 Lake Carolyn Parkway, Suite 750
Irving, Texas 75039
Phone 214.496.0100 • Fax 972.767.4523