Cloud Storage SDK UsePage

CSCloudServer

Member Variable

Variable Name	Variable Type	Description
serverName	NSString	Cloud Service Name
isWebLogin	BOOL	Determine whether it is logged by the web
isLogin	BOOL	Is user logged in?
serverType	CSCloudServerType	Enum value for corresponding cloud services
userName	NSString	Username that logged in cloud service
password	NSString	User password used to login cloud service. Note that the password can be retrieved when the login method is not OAuth2.
hostName	NSString	cloud (FTP and Webdave) host name
portID	NSString	FTP port number
fileExtension	NSString	Access file via Filename extension
fileExtensions	NSArray	File extension array
delegate	id<CSCloudServerDelegate>	Cloud object delegation
hasDownloadProgress	BOOL	Is there a download
isAutoUpload	BOOL	auto upload or not

Usage

Cloud service login：

Initialize a cloud object: initialize a cloud object of autorealease type via cloud coresspondence enumeration values. As lots of callback being produced while operating the cloud, set the delegate while initialization. Set the delegate as nil while the object is released.

```
CSCloudServer *cloudServer = [CSCloudServer
cloudServerForType:serverType];
    cloudServer.delegate = self;
```

Security login cloud. Chech if it is log in on web through the isweblogin of the server, then to confirm the parameter passing of login type. If YES, pass a pushed-in parent view object. If NO, pass nil. Please assign the parameter of server login before callinglogin types （more detailes please see the following table）. Return callback of logon success/failure while landing. If logon success, cloud drive will remember the user information for next logon. Multi-account is supported in the following cloudservice: Box 、DropBox、googleDrive、GoogleDocs、FTP、Mydisk、WebDav、SugarSync; Single-accout is supported in EverNote and skyDrive. As for those cloud drives support single-accout, if log on for the second time, logon success callback will be returned directly.

```
if (cloudServer.isWebLogin) {
    [cloudServer loginFromController:self];
}else{
    //access to parameter to cloud login
    [cloudServer loginFromController:nil];
}
```

serverType	cloudServer parameter setting			
	userName	password	hostName	portID
Box	weblogin without parameter			
DropBox	weblogin without parameter			
GoogleDrive	weblogin without parameter			
EverNote	weblogin without parameter			
skyDrive	weblogin without parameter			
MyDisk	√	√		
WebDav	√	√	√	
GoogleDocs	√	√		
sugarSync	√	√		
FTP	√	√	√	√

Cloud Log out：

logout cloud account and return callback of logout

success/failure. if logout success, user account and login ID setted by system would be deleted to stay in logout safe status; if logout failure, user account and login ID setted by system would not be deleted.

[cloudServer logout];

Access to File List ：

Access to Root Table：

Access to all files and folders under cloud root table. If accessible, return accessed file array and folder array (two cloud file object arrays); is non-accessible, return cloud object and wrong information.

[cloudServer getRootListAndFolder];

Access to File Table：

Access to all files and folders under specified folder. If success/failure, return callback information as the same in "access to root table".

[cloudServer getListAndFolderByFoler:folder];

Downloading Files:

Download cloud files to specific track and access to file download procedure and status through agency as [CSDownloadManager sharedManager]. Meanwhile, call the functions of addtodownloadlist to add the task to to download list. if download success, return callback of download success; else, return cloud type and wrong information. As for FTP and Evernote, there is no download procedure but download status displayed.

```
[CSDownloadManager sharedManager].delegate = self;
[cloudServer downloadFile:fileObject
                toPath:savePath];
```

Add to Download List：

Add tasks to be downloaded in cloud to download list and show download status. download procedure of the added-to-list taskcould be controlled.

Attention: GoogleDrive、FTP、Evernote are not being added into download list.

```
[cloudServer addToDownloadList:file        withSavePath:savePath];
```

Uploading Files:

Upload local files(including single files and mutipied files) to the specific table of specific cloud and return callback of upload success/failure. Attention: in skyDrive、SugarSync and FTP, the new file will cover the old one if hte two files are of the same name; in DropBox, files of the same name will add a suffix automatically; in GoogleDrive and GoogleDocs, files of the same name are allowed; in Box、MyDisk、WebDav, files of the same name will fails to upload; in EverNote, though of the same name, files of the exact same content will fails to upload.

```
[cloudServer uploadFilePath:localFilePath
        toFolder:destinationFolder];
[cloudServer uploadFiles:localFilePathArr
    toFolder:destinationFolder];
```

Creating New Folders:

Creat a new folder under the current cloud table. Return callback of creat success/failure. In SugarSync, creating new folder is not allowed; in Evernote, note folder is only allowed to create under the root table instead of note; in skyDrive、WebDav、MyDisk、FTP、Box、DropBox, folders of the

same name are not allowed to create; in EverNote、GoogleDrive、GoogleDocs, folders of the same name are allowed to create.

[cloudServer createNewFolder:folderName

inFolder:destinationFolder];

Deleting Files:

Delete specific file object and return callback of delete success/failure.

Attention:

Box、DropBox、GoogleDrive、GoogleDocs、skyDrive support to delete both of files and folders; FTP can only delete files but not folders; Evernote can noly delete folders but not files;

MyDisk、WebDav、SugarSync support no delete.

[cloudServer deleteFile:fileObject];

Renaming Files:

Rename the specific file object. The new file name need no file suffix. Return callback of rename success/failure. Attention, Box、DropBox and skyDrive allow no renaming the same name;

GoogleDrive、EverNote allow renaming the same name; GoogleDocs、FTP、MyDisk、WebDav and SugarSync support no rename.

```
[cloudServer renameFile:fileObject
        withName:newName];
```

Online Editing:

This is the exclusive features of GoogleDirve; Users can online edit and save certain types of files.

```
[cloudServer getOnlineEditorURL:  file];
```

Check administrators who have file permission:

The function of checking administrators who have file permission is only open to GoogleDrive Files to check which administrators have file permission .

```
[cloudServer getPermissionList:file];
```

Add visitors with file permission

Add other users to this file permission (reader or writer)

```
[cloudServer insertPermission:userName wihtRole:@"reader"
toFile:file];
```

Remove this user's file permission

Remove this user's file permission. Note: you can't remove the file owner.

```
[cloudServer deletePermission:file atPermissionID:permission];
```

CloudFile

Member Variable

Variable Name	Variable Type	Description
fileName	NSString	File Name
fileObject	id	File Name
serverType	CSCloudServerType	files belong to which cloud
canOnline	BOOL	if file on-line edit avaiable
fileState	CSCloudFileState	if file on-line edit avaiable

Usage：

Enumerator of File status type：

```
typedef enum{
    CSFileNone = 0, // No Operation
    CSFileDelete, // Deleting File...
    CSFileDownloading, // Downloading File...
    CSFileRenam   // Renaming File...
}CSCloudFileState;
```

CSDownloadTask

Member Variable

Variable Name	Variable Type	Description
mAddress	NSString	Source Address
mFileName	NSString	Name of Downloading File
mReceivedSize	SInt64	Received block size while downloading
mFileSize	SInt64	File Size
mDownloadState	CSDOWNLOAD_STATE	Download Status
serverType	CSCloudServerType	The cloud service that
delegate	Id<CSTaskDelegate>	Delegate
token	NSString	The unique identifier for
savepath	NSString	file save path
loadData	NSMutableData	Save Downloaded data

Delegate Method

@protocol CSTaskDelegate <NSObject>

@optional

Callback for Downloading Process

Call if the task downloading status changed. Download status include: downloading, pause, download finished, download failure and to be downloaded.

- *(void)taskStateChange:(CSDownloadTask *) task;*

Task download procedure callback

call when downloading task and data changing. No callback of download procedure in Evernote and FTP.

- *(void)taskDownloadProgress:(CSDownloadTask *) task;*

Callback for Download Succussful

- *(void)taskDownloadSuccess:(NSString *)name;*

Callback for Download Fail

- *(void)taskDownloadFaile:(NSString *)name withError:(NSError *)error;*

@end

Usage：

Start Downloading Tasks

Start or continue to download. Download the unfinished data, instead re-downloading, when continue downloading the paused tasks.

```
[task startRunning];
```

Pause Downloading

While pausing the current task, it will cacha for subsequent download. the cloud of Evernote, FTP, GoogleDrive can't control the download procedure.

```
[task stopRunning];
```

Deleting Tasks

While deleteing current task, including the downloading and failed task, the download cacha data would also be cleared.

［task deleteTask］;

Determine whether they are the same task

*-(BOOL)isEqualToTask:(CSDownloadTask *)task;*

CSDownloadManager

Member Variable

Variable Name	Variable Type	Description
taskArray	NSMutableArray	save download task array

emunation of task download status

```
typedef enum {
    kCSDownloadStateNone = 0, // no task status
    kCSDownloadStateRunning ,    // Downloading
    kCSDownloadStateStopped ,    // Pause
    kCSDownloadStateFinished ,    // Download Complete
    kCSDownloadStateFailed    // Download Fail
} CSDOWNLOAD_STATE;
```

Delete Downloading Tasks

Delete tasks in any download status, including ongoing and completed download tasks.

*-(void) deleteTask:(CSDownloadTask *)task;*

Delete All Complete Tasks

Delete All Complete Downloading Tasks

-(void)delateFinishTask;

Start Downloading

Start downloading all tasks, with exception of completed, failed and ongoing download tasks.

-(void)startDownload;

Change download status

*-(void)changeTaskState:(CSDownloadTask *)task;*

Save Downloaded List

-(void)saveDownloadTask;

CSPermission

Member Variable

Variable Name	Variable Type	Description
permissionName	NSString	Adminisrtator's name
permissionRole	NSString	Role to this file
permissionnObject	id	File permission object.