Cloud Storage SDK Instructions

CSCloudServer
Property

Variable name	Variable type	Description
serverName	NSString	Server Name
isWebLogin	BOOL	Determine whether it id logged by web.
isLogin	BOOL	Is user Logged in?
serverType	CSCloudServerType	Enum value for corresponding cloud service.
userName	NSString	User Name
password	NSString	Password for cloud log in. Only the no weblogin cloud server get the password.
hostName	NSString	Cloud (FTP & webdave) host Name
portID	NSString	FTP port ID
fileExtension	NSString	Access to document via file Extension
fileExtensions	NSArray	File extensions array
delegate	Id<CSCloudServerDelegate>	Cloud Server Delegate
hasDownloadProgress	BOOL	Has any Download Progress?
isAutoUpload	BOOL	Is Auto Upload?

Usage
Login
Initialize a cloud server: initialize an autorelease cloud server via cloud correspondence enumeration values. Since lots of callbacks when operating the cloud, it is supposed to lower delegate when initiating. Set the delegate as nil while the sever is released.

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

Secure Login

Determine if it is logged in via web through the server isWeblogin to ensure the login parameter has passed. If YES, pass a push-in parent view object. If NO, pass nil.
Before calling logintype, parameters for serverlogin shall be assigned (the following picture has more details). It returns callback of login success/failure. If login succeeds, server will remember account information for next login.
Box, DropBox, GoogleDrive, GoogleDocs, FTP, Mydisk, WebDav, SugarSync: support multi-accounts.
EverNote and SkyDrive: support single-accounts.
For servers that support single-accounts, the second login shall return callback of loginsuccess once login succeeds.

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

ServerType	cloudServer parameter setting			
	UserName	Password	HostName	PortID
Box	Weblogin with no parameter			
DropBox	Weblogin with no parameter			
GoogleDrive	Weblogin with no parameter			
EverNote	Weblogin with no parameter			
skyDrive	Weblogin with no parameter			
MyDisk	√	√		
WebDav	√	√	√	
GoogleDocs	√	√		
sugarSync	√	√		
FTP	√	√	√	√

Logout
Secure cloud server logout and returns callback of logout success /failure. The account information saved by system will be deleted when logging out to stay a secure state. If logout fails, account information saved by system and login sign will not be deleted.

```
[cloudServer logout];
```

Access to Folder
Access to root list

Access all folders and file information under the cloud root list. If successful, return accessedfolderandfilearray (namely two cloudFile object arrays). If failed, return cloudobject and error information.
[cloudServer getRootListAndFolder];

Access to Folder List
Access all folders and files under the specific files. Both successful and failed callbacks will return to the root list.
[cloudServer
[cloudServer getListAndFolderByFoler:folder];

Download Files:
Download cloud files to specific path and access to file download speed and progress via delegate[CSDownloadManagersharedManager]. Downloading cloud files will call the addtodownloadlist function and add the task to download list. If successful, return the downloadsuccess callback; else, return downloadfilecloudtype and download the detailed failure information. Files in FTP and EverNote show no download progress but only the download state.
[CSDownloadManager sharedManager].delegate = self;
[cloudServer downloadFile:fileObject
toPath:savePath];

Add to Download List
Add the cloud download task to download list and download state will be visible. The download process of tasks being added into the download list can be controlled.
Attention:
GoogleDrive、FTP、Evernote: not being added into the download list.
[cloudServer addToDownloadList:file
withSavePath:savePath];

Upload Files
Upload local files((including single and multil-files)
) into specific tables under specific clouds and return callback of upload success/failure
Attention:
SkyDrive, SugarSync and FTP: The existing files will overwrite files with the same name in SkyDrive, SugarSync and FTP.
DropBox: Files with the same name in DropBox will add extensions automatically.
GoogleDrive, GoogleDocs: supports files with the same name.
Box, MyDisk, WebDav: fails to upload the files with same name.

EverNote: allow no file with the same content though, even if they have a different name.
[cloudServer uploadFilePath:localFilePath
toFolder:destinationFolder];
[cloudServer uploadFiles:localFilePathArr
toFolder:destinationFolder];

New Folder
Create new folders under the local folder list of the local cloud, return callback of createnewfolder success/failure.
sugarSync: do not create new folder
Evernote: only create notefolder under root list; cannot create note folder under note
skyDrive, WebDav, MyDisk, FTP, Box, DropBox; cannot create new folders with the same name as already existing folders
EverNote, GoogleDrive, GoogleDocs: supports creating new folders with the same name as already existing folders
[cloudServer createNewFolder:folderName
inFolder:destinationFolder];

Delete Files
Delete specific file object, return callback of delete success/failure.
Attention:
Box, DropBox, GoogleDrive, GoogleDocs, skyDrive: can delete both folders and files
FTP: can delete files but not folders
Evernote: can delete folders but not files
MyDisk, WebDav, SugarSync: unable to delete files or folders
[cloudServer deleteFile:fileObject];

Folder Renaming
Rename the specific file object. File suffix is not required. Return callback of rename success/failure.
Attention:
Box, DropBox, skyDrive: does not allow renaming of existing folders
GoogleDrive, EverNote : allows renaming of existing folders
GoogleDocs, FTP, MyDisk, WebDav, SugarSync: does not allow renaming of files.
[cloudServer renameFile:fileObject
withName:newName];

Online Editing
GoogleDrive exclusively supports the function of online editing. It edits and saves various types of files online.
[cloudServer getOnlineEditorURL：file]；

Check File Permission
Only GoogleDrive files supports checking file permissions (who can gain access to the file).
[cloudServer getPermissionList:file];

Add File Permissions
Grant other users access to the file (ability to read/edit)
[cloudServer insertPermission:userName
wihtRole:@"reader" toFile:file];

Remove User Permissions
Remove permission for certain users. You can not remove the file owner.
[cloudServer deletePermission:file
atPermissionID:permission];

CloudFile
Property

Variable names	Variable type	Memos
fileName	NSString	File name
fileObject	id	File object
serverType	CSCloudServerType	File belong to which cloud
canOnline	BOOL	If thefile could be edited online
fileState	CSCloudFileState	File statues

Usage:
File statues enumeration type
typedef enum{
CSFileNone = 0, //no operation
CSFileDelete, //file deleting
CSFileDownloading, //file downloading
CSFileRenam //file renaming
}CSCloudFileState;
CSDownloadTask

Property

Variable names	Variable type	Description
mAddress	NSString	Downloaded files source address
mFileName	NSString	Downloaded files name
mReceivedSize	SInt64	Received data size when downloading
mFileSize	SInt64	Received file size
mDownloadState	CSDOWNLOAD_STATE	File Download State
serverType	CSCloudServerType	Downloaded file belong to which cloud server
delegate	Id<CSTaskDelegate>	delegate
token	NSString	The only token for cloud login
savepath	NSString	File save path
loadData	NSMutableData	Save download data

Task Delegate
@protocol CSTaskDelegate <NSObject>
@optional

Callback of taskdownloadstate
Call if the task download state changes.
Download states include: downloading, paused, download completed, download failed, to be operated.
- (void)taskStateChange:(CSDownloadTask *) task;

Callback of taskdownloadprogress
Call if the task is downloading and the data changes.
EverNote and FTP will not return callback of download progress.
- (void)taskDownloadProgress:(CSDownloadTask *)
task;

Callback of taskdownloadsuccess
- (void)taskDownloadSuccess:(NSString *)name;

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

@end

Usage

Start Running

Start or continue to download.
If continuing from a paused state, resume downloading the unfinished task instead of restarting the whole task.

```
［task
startRunning］；
```

Stop Running

Stop the current download task. Once stopped, the download state will be cached for subsequent download.
Evernote,FTP,GoogleDrive: does not control the download progress.

```
［task
stopRunning］；
```

Delete Task

Delete download tasks, including current download tasks and failed download tasks. This will also clear the cached data.

```
［task
deleteTask］；
```

Is Qual to Task?

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

Property

Variable name	Variable type	description
taskArray	NSMutableArray	Save download task array

Enumeration of Download Task State

```
typedef enum {
kCSDownloadStateNone = 0, //no state
kCSDownloadStateRunning , //downloading
kCSDownloadStateStopped , //download stopped
kCSDownloadStateFinished , //download completed
```

Delete Specific Download Task

Delete download tasks in any state, including currently downloading tasks, completed tasks etc.

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

Delete All Finished Task
Deletes all completed download tasks
-(void)delateFinishTask;

Start Downloading
Starts downloading all tasks except for completed and failed download tasks, and tasks that are already downloading.
-(void)startDownload;

Change Task Download State
-(void)changeTaskState:(CSDownloadTask *)task;

Save Download Task
-(void)saveDownloadTask;

CSPermission
Property

Variable name	Variable type	Description
permissionName	NSString	Permission Name
permissionRole	NSString	Permissions
permissionnObject	id	Permissionn Object