

Using the Zumero iOS Framework

Using the Zumero iOS Framework

Copyright © 2013-2026 Zumero LLC

Table of Contents

1. Introduction	1
2. Getting Started	2
3. Classes	4
3.1. ZumeroSync	4
3.1.1. Sync	4
3.1.1.1. Example - Sync Anonymously	6
3.1.1.2. Example - Sync with Authentication	7
3.1.2. Cancel	7
3.1.3. QuarantineSinceLastSync	8
3.1.4. SyncQuarantine	8
3.1.5. DeleteQuarantine	10
3.2. ZumeroUtil	10
3.2.1. dataToSqlGuid	10
3.2.2. sqlGuidToData	10

List of Figures

2.1. Adding the ZumeroSync framework to a workspace	2
2.2. Include ZumeroSync in your project	2

Chapter 1. Introduction

This document explains the Zumero iOS Framework, an Objective C-friendly wrapper around Zumero's sync functions. We recommend that you first read `zumero_for_sql_server_client_api.pdf`¹ for a thorough introduction to Zumero's core concepts and abilities.

The classes documented here are not strictly necessary for Zumero development on iOS - the C API may be called directly; both methods may be used side-by-side. The iOS framework allows common Zumero usage to occur in a world of familiar, managed NSObjects and automatic threading.

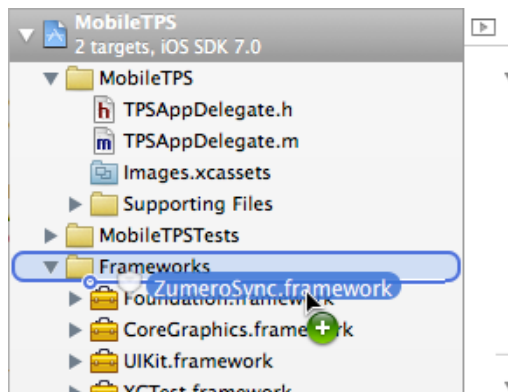
¹ found under `/docs/` in the Zumero SDK distribution

Chapter 2. Getting Started

The `ZumeroSync.Framework` folder in the SDK zip file contains all the code and headers needed to use Zumero in your project.

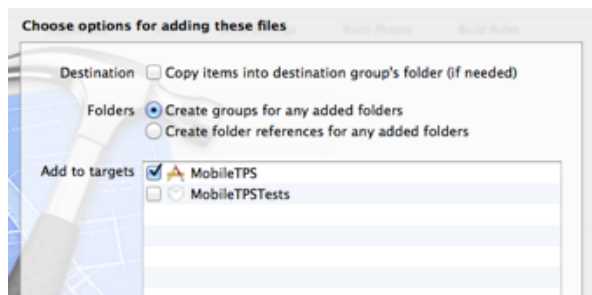
Drag it into your workspace...

Figure 2.1. Adding the ZumeroSync framework to a workspace



... and make sure your project references it.

Figure 2.2. Include ZumeroSync in your project



Import `ZumeroSync.h` in any class file that needs access to Zumero classes:

```
#import <ZumeroSync/ZumeroSync.h>
```

You'll also need to include SQLite¹ (version 3.7.11 or higher) in your project.

Note

Under iOS 6.x and later, including the system `libsqlite3.dylib` will suffice.

Finally, the Zumero library relies on `CFNetwork.framework` and `libz.dylib` (both of which will already be available in XCode's list of libraries) to be included in the **Build Phases / Link Binary With Libraries** list.

To use Zumero's pure-C interface (documented in `zumero_for_sql_server_client_api.pdf`), just include `ZumeroSync/ZumeroSync.h` and you're ready to go.

¹<http://www.sqlite.org/>

Read on to let the library take care of connection management, background synchronization, etc.

Chapter 3. Classes

3.1. ZumeroSync

`ZumeroSync` provides static methods implementing the Zumero Sync and Quarantine APIs.

3.1.1. Sync

Synchronize with a Zumero server.

`Sync` synchronizes a dbfile between the local and remote (server-side) copies. If a schema change has occurred on the server, `Sync` will migrate that change to the client, as well.

Note

`Sync` should *not* be called directly from the main (UI) thread. As with any network activity, `Sync` should occur on a background thread, or via a dispatch queue [<https://developer.apple.com/library/ios/DOCUMENTATION/General/Conceptual/ConcurrencyProgrammingGuide/OperationQueues/OperationQueues.html>].

There are two variations on the `Sync` methods: one takes an `NSDictionary` * defining its authentication scheme (see the example below) and sync options; the other takes a JSON string representing those values. In either case, `nil` may be passed to sync anonymously or sync with default options.¹

¹For more information on authentication schemes, see the Development with Zumero for SQL Server [http://zumero.com/docs/zumero_for_sql_server.html] guide.

For more information on sync options, see the ZSS Client API reference [http://staging.zumero.com/docs/zumero_for_sql_server_client_api.html#sync_details]


```
+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authSchemeJS:(NSString *)authSchemeJS
    user:(NSString *)user
    password:(NSString *)password
    error:(NSError **)error;

+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authScheme:(NSDictionary *)authScheme
    user:(NSString *)user
    password:(NSString *)password
    error:(NSError **)error;

+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authScheme:(NSDictionary *)authScheme
    user:(NSString *)user
    password:(NSString *)password
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    error:(NSError **)error;

+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authSchemeJS:(NSString *)authSchemeJS
    user:(NSString *)user
    password:(NSString *)password
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    error:(NSError **)error;

+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authScheme:(NSDictionary *)authScheme
    user:(NSString *)user
    password:(NSString *)password
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    options:(NSDictionary *)options
    syncId:(int *)syncId
    error:(NSError **)error;

+ (BOOL) Sync:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    serverUrl:(NSString *)serverUrl
    remote:(NSString *)remote
    authSchemeJS:(NSString *)authSchemeJS
    user:(NSString *)user
    password:(NSString *)password
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    optionsJS:(NSString *)optionsJS
    syncId:(int *)syncId
    error:(NSError **)error;
```

Parameter	Description				
(returns)	YES if synchronization completed successfully.				
filename	The full path of the local SQLite dbfile to be synced. If the file does not yet exist, it will be created.				
cipherKey	If SQLCipher encryption is being used on the local dbfile, the cipher key should be passed here. Otherwise pass <code>nil</code> .				
serverUrl	The base URL of your Zумero server. If your server is hosted using an internationalized domain name (IDN), this URL should be the Punycode encoded version on the domain.				
authScheme	<p>Authentication scheme to use with Sync credentials. May be <code>nil</code> to sync anonymously (if permitted by the server).</p> <p>Zумero's "table authentication" scheme requires two values:</p> <table> <tr> <td>scheme_type</td><td>Required in any scheme. When validating against a SQL Server table, it should be "table".</td></tr> <tr> <td>table</td><td>The name of the remote table in which authentication credentials are stored.</td></tr> </table>	scheme_type	Required in any scheme. When validating against a SQL Server table, it should be "table".	table	The name of the remote table in which authentication credentials are stored.
scheme_type	Required in any scheme. When validating against a SQL Server table, it should be "table".				
table	The name of the remote table in which authentication credentials are stored.				
authSchemeJS	JSON string representation of the authentication scheme. May be <code>nil</code> to sync anonymously (if permitted by the server).				
user	Username under which to sync. May be <code>nil</code> to sync anonymously (if permitted).				
password	Password under which to sync. May be <code>nil</code> to sync anonymously (if permitted).				
callback	A callback block that will be called to provide progress information during sync. May be <code>nil</code> . Example: <code>^(int cancellation_token, int phase, zумero_int64 bytesSoFar, zумero_int64 bytesTotal, void * object) { }</code>				
dataPointer	An opaque data pointer that will be passed to the callback block. May be <code>nil</code> .				
options	Dictionary requesting special sync behaviors (e.g. <code>{"sync_details":true}</code>)				
optionsJS	JSON representation of the options object.				
syncId	If <code>sync_details</code> are requested, receives a sync ID with which to look them up.				
error	If we failed to complete synchronization, more information will be allocated here.				

3.1.1.1. Example - Sync Anonymously

In this example, we'll look at a simple use case - anonymous synchronization of a database.

```
// error handling omitted for brevity
NSError *err = nil;

NSString *dbfileName = @"myDbFolder/my.db";
NSString *remoteName = @"mydb";

BOOL ok = [ZumeroSync
           Sync:dbfilename
           cipherKey:nil
           serverUrl:@"http://myserver.example.com"
           remote:remoteName
           authScheme:nil
           user:nil
           password:nil
           error:&err];
```

3.1.1.2. Example - Sync with Authentication

In this example, we'll look at authenticated synchronization against a remote database.

```
// error handling omitted for brevity
NSError *err = nil;

NSString *dbfileName = @"myDbFolder/my.db";
NSString *remoteName = @"mydb";
NSDictionary *scheme = @{
    @"scheme_type": @"table",
    @"table": @"remoteAuthTable"
};

BOOL ok = [ZumeroSync
           Sync:dbfilename
           cipherKey:nil
           serverUrl:@"http://myserver.example.com"
           remote:remoteName
           authScheme:scheme
           user:@"barney"
           password:@"bambam"
           error:&err];

// OR

BOOL ok = [ZumeroSync
           Sync:dbfilename
           cipherKey:nil
           serverUrl:@"http://myserver.example.com"
           remote:remoteName
           authSchemeJS:@"{\"scheme_type\":\"table\",\"table\":\"remoteAuthTable\"}"
           user:@"barney"
           password:@"bambam"
           error:&err];
```

3.1.2. Cancel

Cancel a sync that is in progress

You must use the callback block parameter to the Sync method to get a cancellation token for the ongoing sync.

```
+ (void) Cancel:(int)cancellationToken
```

Parameter	Description
(returns)	This method has no return value
cancellationToken	The cancellation token that was passed into the callback for the sync.

3.1.3. QuarantineSinceLastSync

Move un-synced local changes into an isolated holding area. Typically, the reason to do so is because the local changes conflict with other changes already on the server.

The quarantined changes are stored locally, so no network activity is required.

```
+ (BOOL) QuarantineSinceLastSync:(NSString *)filename
                        cipherKey:(NSString *)cipherKey
                        pqid:(sqlite3_int64 *)pqid
                        error:(NSError **)error;
```

Parameter	Description
(returns)	YES if we successfully quarantined our local updates.
cipherKey	If SQLCipher encryption is being used on the local dbfile, the cipher key should be passed here. Otherwise pass nil.
pqid	On success, a "quarantine ID" will be filled in to the variable pointed to by pqid. This ID must be used later when re-syncing the changes via SyncQuarantine.
error	If we failed to quarantine, more information will be allocated here.

3.1.4. SyncQuarantine

Sync with the remote, including changes previously quarantined by QuarantineSinceLastSync.

SyncQuarantine performs network activity, and should not be called on the UI thread.

```
+ (BOOL) SyncQuarantine:(NSString *)filename
                        cipherKey:(NSString *)cipherKey
                        qid:(sqlite3_int64 *)qid
                        serverUrl:(NSString *)serverUrl
                        remote:(NSString *)remote
                        authScheme:(NSDictionary *)authScheme
                        user:(NSString *)user
                        password:(NSString *)password
                        partial:(BOOL *)partial
                        error:(NSError **)error;

+ (BOOL) SyncQuarantine:(NSString *)filename
                        cipherKey:(NSString *)cipherKey
                        qid:(sqlite3_int64 *)qid
                        serverUrl:(NSString *)serverUrl
                        remote:(NSString *)remote
                        authSchemeJS:(NSString *)authSchemeJS
                        user:(NSString *)user
                        password:(NSString *)password
                        partial:(BOOL *)partial
                        error:(NSError **)error;
```

Parameter	Description
(returns)	YES if synchronization completed successfully.

Parameter	Description				
filename	The full path of the local SQLite dbfile to be synced. If the file does not yet exist, it will be created.				
cipherKey	If SQLCipher encryption is being used on the local dbfile, the cipher key should be passed here. Otherwise pass <code>nil</code> .				
qid	A quarantine ID returned by <code>QuarantineSinceLastSync</code> .				
serverUrl	The base URL of your Zумero server.				
authScheme	<p>Authentication scheme to use with sync credentials. May be <code>nil</code> to sync anonymously (if permitted by the server).</p> <p>Zумero's "table authentication" scheme requires two values:</p> <table> <tr> <td>scheme_type</td><td>Required in any scheme. When validating against a SQL Server table, it should be "table".</td></tr> <tr> <td>table</td><td>The name of the remote table in which authentication credentials are stored.</td></tr> </table>	scheme_type	Required in any scheme. When validating against a SQL Server table, it should be "table".	table	The name of the remote table in which authentication credentials are stored.
scheme_type	Required in any scheme. When validating against a SQL Server table, it should be "table".				
table	The name of the remote table in which authentication credentials are stored.				
authSchemeJS	JSON string representation of the authentication scheme. May be <code>nil</code> to sync anonymously (if permitted by the server).				
user	Username under which to sync. May be <code>nil</code> to sync anonymously (if permitted).				
password	Password under which to sync. May be <code>nil</code> to sync anonymously (if permitted).				
partial	If YES, <code>SyncQuarantine</code> should be called again with the same parameters to complete the resynchronization process.				
error	If we failed to complete synchronization, more information will be allocated here.				

Example:

```

BOOL partial = NO;
BOOL ok = NO;
NSError *err = nil;
sqlite3_int64 qid = 0;

// ...
// a call to QuarantineSinceLastSync sets qid
// ...

do
{
    ok = [ZумeroSync
        SyncQuarantine:dbfilename
            cipherKey:nil
            qid:qid
            serverUrl:@"http://myserver.example.com"
            remote:remoteName
            authScheme:nil
            user:nil
            password:nil
            partial:&partial
            error:&err];
} while (ok && partial);

```

3.1.5. DeleteQuarantine

Permanently delete changes quarantined by QuarantineSinceLastSync.

```
+ (BOOL) DeleteQuarantine:(NSString *)filename
    cipherKey:(NSString *)cipherKey
    qid:(sqlite3_int64)qid
    error:(NSError **)error;
```

Parameter	Description
(returns)	YES if changes were discarded successfully.
filename	The full path of the local SQLite dbfile to be synced. If the file does not yet exist, it will be created.
cipherKey	If SQLCipher encryption is being used on the local dbfile, the cipher key should be passed here. Otherwise pass nil.
qid	A quarantine ID returned by QuarantineSinceLastSync.
error	If we failed to delete the quarantined data, more information will be allocated here.

3.2. ZumeroUtil

Miscellaneous methods to manage Zumero's representation of MS SQL data.

3.2.1. dataToSqlGuid

Given a 16-byte blob (in NSData form), return an MSSQL-style GUID string ("aaba1234-5678-9090-9876-aabbccdd1234")

SQL uniqueidentifier columns are stored as blobs in the mobile database. Use this method and sqlGuidToData to convert between mobile data and the MS SQL string format.

```
+ (NSString *) dataToSqlGuid:(NSData *)uuid
```

Parameter	Description
(returns)	SQL Server-formatted GUID string representation of the passed-in data. nil if uuid is nil, or if the data buffer is not exactly 16 bytes long.
uuid	16-byte blob, the mobile representation of a SQL Server uniqueidentifier.

3.2.2. sqlGuidToData

Given an MSSQL-style GUID string ("aaba1234-5678-9090-9876-aabbccdd1234"), return the corresponding 16-byte blob

```
+ (NSData *) sqlGuidToData:(NSString *)sqlguid
```

Parameter	Description
(returns)	16-byte blob representation of the passed-in SQL Server UUID. nil if sqlguid is nil, or if the format is not exactly as listed.
sqlguid	MS SQL-formatted string representation of a uniqueidentifier.