

# Using the Zumero Framework

---

# Using the Zumero 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 .....	11
3.2.1. dataToSqlGuid .....	11
3.2.2. sqlGuidToData .....	11

---

## List of Figures

2.1. Adding the ZумeroSync xcframework to a workspace .....	2
2.2. Include ZумeroSync in your project .....	3

---

# Chapter 1. Introduction

This documentation explains the Zумero XCFramework, an Objective C wrapper around Zумero's sync functions. It is recommended that you first read `zumero_for_sql_server_client_api.pdf`<sup>1</sup> for a thorough introduction to Zумero's core concepts and abilities.

The XCFramework provides a common Objective-C API for developing your Zумero based application across multiple platforms and architectures; MacOS (arm64 and x86\_64), iOS (arm64), iOS Simulator (arm64 and x86\_64), and MacCatalyst (arm64 and x86\_64). These Objective-C APIs may also be called from Swift with a couple additional setup steps that will be document later in this document.

The classes documented here are the recommended method for developing with Zумero on MacOS and iOS. Though if you intend to develop using languages other than Objective-C or Swift, native libraries are provided in the SDK containing the C APIs which may be called directly; both methods may be used side-by-side.

---

<sup>1</sup> found under `/docs/` in the Zумero SDK distribution

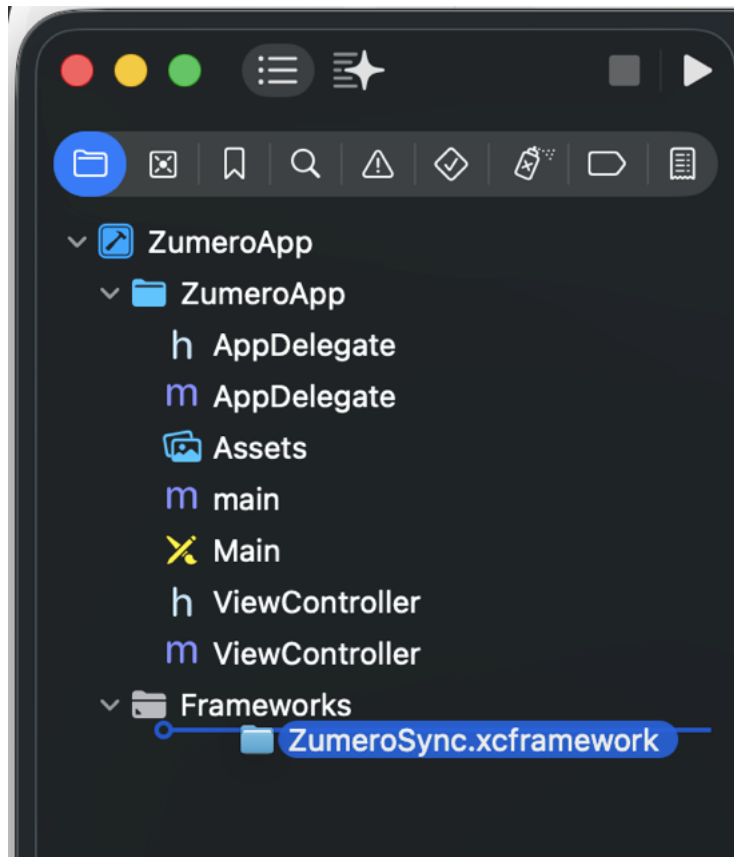
---

## Chapter 2. Getting Started

Locate the `ZumeroSync.xcframework` inside the apple directory of the Zumero SDK. This contains everything you will need to add Zumero to your project.

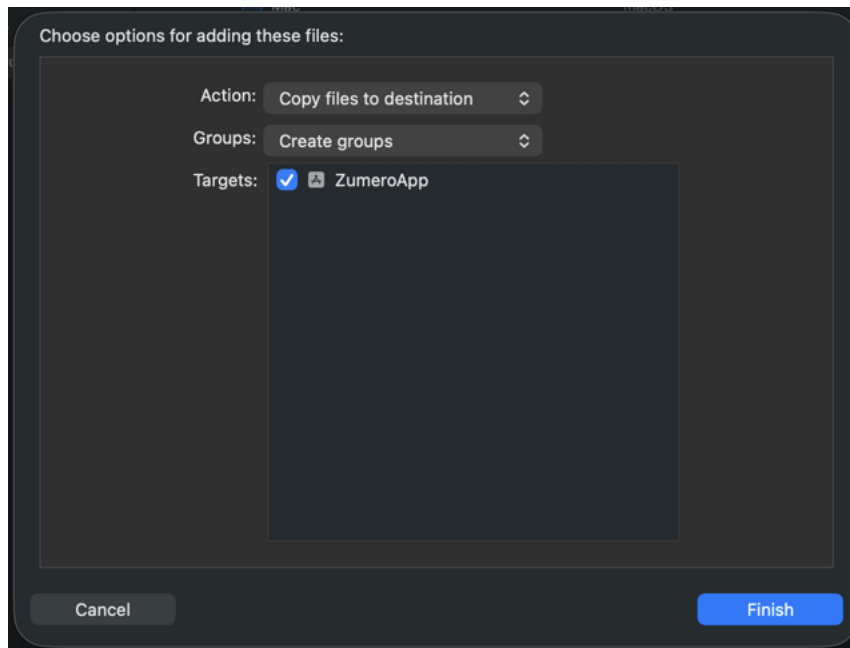
Drag `ZumeroSync.xcframework` into your workspace.

**Figure 2.1. Adding the ZumeroSync xcframework to a workspace**



Alternatively, you can add the xcframework using the "Add Files to <PROJECT NAME>" menu item in the file menu or the contextual menu in the Project Navigator pane

Ensure the desired target in your project references the framework

**Figure 2.2. Include ZumeroSync in your project**

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

```
#import "ZumeroSync.h"
```

Zumero has external dependencies on `SQLite` and `CFNetwork.framework`, but in modern Objective-C and Swift projects these two libraries will usually be linked by the default project settings.

Zumero also depends on `libz`, which will need to be added to the **Frameworks, Libraries, and Embedded Content** list in the General tab of your project's build settings. Click the "+" button and search for "libz", then select `libz.tbd` from the list. If you have issues adding `libz` to this list, alternately try adding it to the list of libraries in the **Build Phases / Link Binary With Libraries** section of your project's build settings.

When including `ZumeroSync.xcframework` in a Swift project, you will need to add the import statement listed above to your project's Bridging Header. If your project doesn't have a bridging header, you can add one to your project via the manner described in Apple's Documentation [<https://developer.apple.com/documentation/swift/importing-objective-c-into-swift>].

---

# 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.<sup>1</sup>

---

<sup>1</sup>For more information on authentication schemes, see the Development with Zumero for SQL Server [[http://zumero.com/docs/zumero\\_for\\_sql\\_server.html](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](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 cancellationTokен, int phase, zumero_int64 bytesSoFar, zumero_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;

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

+ (BOOL) QuarantineSinceLastSync:(NSString *)filename
                        cipherKey:(NSString *)cipherKey
                        optionsJS:(NSString *)optionsJS
                        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.
options	Additional quarantine options as a Dictionary. May be nil.
optionsJS	Additional quarantine options as a JSON string. May be nil.
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;

+ (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
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    options:(NSDictionary *)options
    syncId:(int *)syncId
    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
    callback:(ZumeroProgressCallback)callback
    dataPointer:(void *)dataPointer
    optionsJS:(NSString *)optionsJS
    syncId:(int *)syncId
    partial:(BOOL *)partial
    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 nil.
qid	A quarantine ID returned by QuarantineSinceLastSync.
serverUrl	The base URL of your Zumero server.

Parameter	Description				
authScheme	<p>Authentication scheme to use with sync credentials. May be <code>nil</code> to sync anonymously (if permitted by the server).</p> <p>Zumero'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).				
options	Additional sync options. May be <code>nil</code> .				
optionsJS	Additional sync options as a JSON string. May be <code>nil</code> .				
partial	If YES, SyncQuarantine 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 = [ZumeroSync
        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;

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

+ (BOOL) DeleteQuarantine:(NSString *)filename
               cipherKey:(NSString *)cipherKey
                   qid:(sqlite3_int64)qid
               optionsJS:(NSString *)optionsJS
                 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 <code>nil</code> .
qid	A quarantine ID returned by <code>QuarantineSinceLastSync</code> .
options	Additional quarantine options. May be <code>nil</code> .
optionsJS	Additional quarantine options as a JSON string. May be <code>nil</code> .
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. <code>nil</code> if <code>uuid</code> is <code>nil</code> , or if the data buffer is not exactly 16 bytes long.
uuid	16-byte blob, the mobile representation of a SQL Server <code>uniqueidentifier</code> .

### 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. <code>nil</code> if <code>sqlguid</code> is <code>nil</code> , or if the format is not exactly as listed.
<code>sqlguid</code>	MS SQL-formatted string representation of a <code>uniqueidentifier</code> .