****

|  |  |
| --- | --- |
|  | **openMSYNC Java API**  **설명서** |
|  |  |
|  |  |

**목차**

[개요 4](#_Toc339617047)

[JAVA API 사용을 위한 구성 (Android 기준) 4](#_Toc339617048)

[mSyncClientJNI class 4](#_Toc339617049)

[Flag ( OP\_CODE ) 5](#_Toc339617050)

[기타 5](#_Toc339617051)

[mSync\_InitializeClientSession() 7](#_Toc339617052)

[mSync\_ConnectToSyncServer() 7](#_Toc339617053)

[mSync\_AuthRequest() 7](#_Toc339617054)

[mSync\_SendDSN() 8](#_Toc339617055)

[mSync\_SendTableName() 9](#_Toc339617056)

[mSync\_SendUploadData() 9](#_Toc339617057)

[mSync\_SendUploadData2() 10](#_Toc339617058)

[mSync\_UploadOK() 11](#_Toc339617059)

[mSync\_ReceiveDownloadData () 11](#_Toc339617060)

[mSync\_ReceiveDownloadData2() 12](#_Toc339617061)

[mSync\_Disconnect() 12](#_Toc339617062)

[mSync\_ClientLog() 13](#_Toc339617063)

[mSync\_GetSyncError () 13](#_Toc339617064)

[mSyncClientJNI 사용 방법 14](#_Toc339617065)

*구문설계부터 기타 설계는 스펙/사용자 위주의 설계이고, 내부 모듈, 자료구조 설계는 상세설계에서 기술해주세요.*

# 개요

openMSYNC 동기화 모듈인 JAVA API에 대한 사용 설명서이다.

mSyncClientJNI JAVA class는 아래와 같은 구성을 가지고 있다.

**mSyncClientJNI API**

**mSyncClient**

**Library Core**

**mSyncClientJNI Class**

* mSyncClientJNI API
  + native interface의 C코드 영역
  + mSync 서버와의 통신을 담당하는 실질적인 클라이언트 라이브러리
* mSyncClientJNI class
  + mSyncClientJNI class는 JNI를 제공하기 위한 java class
  + mSyncClientJNI.java에 정의되며 javah를 이용하여 com\_openMSYNC\_mSyncClientJNI.h를 생성
  + com\_openMSYNC\_mSyncClientJNI.c는 com\_openMSYNC\_mSyncClientJNI.h에서 정의된 함수들의 실제 구현 부분
  + mSyncClientJNI API를 이용하여 com\_openMSYNC\_mSyncClientJNI.c를 구현

# JAVA API 사용을 위한 구성 (Android 기준)

* Library
  + libmSyncClientJNI.so
* Java 파일
  + com/openMSYNC/mSyncClientJNI.java
* 기타
  + 사용자의 App. 구성에 필요한 라이브러리 및 Java파일들

# mSyncClientJNI class

본 클래스는 JNI용 JAVA단 코드로 wrapper에 해당하며 자바코드 자체로서의

주요 기능은 가지고 있지 않다.

Final로 정의된 constant 값들이 아래와 같이 정의되어 있다.

이 값들 중 상당 부분은 서버와의 통신에 사용되기 때문에 임의로 변경하지 말아야 한다.

## Flag ( OP\_CODE )

* 통신 프로토콜 상의 op code에 해당하는 값들
* 사용자가 알아야 할 주요 Flag
  + UPDATE\_FLAG : upload하는 데이터가 update할 대상임을 지정
  + DELETE\_FLAG : upload하는 데이터가 delete할 대상임을 지정
  + INSERT\_FLAG : upload하는 데이터가 insert할 대상임을 지정
  + NACK\_FLAG : 오류 발생
  + ACK\_FLAG : 정상
  + XACK\_FLAG : 사용자의 수가 지정된 숫자보다 많은 경우
  + UPGRADE\_FLAG : 버전이 업그레이드 되어 있어 호환 문제가 있음.

|  |
| --- |
| public static final char UPDATE\_FLAG = 'U';  public static final char DELETE\_FLAG = 'D';  public static final char INSERT\_FLAG = 'I';  public static final char DOWNLOAD\_FLAG = 'F';  public static final char DOWNLOAD\_DELETE\_FLAG = 'R';  public static final char END\_FLAG = 'E';  public static final char OK\_FLAG = 'K';  public static final char TABLE\_FLAG = 'M';  public static final char APPLICATION\_FLAG = 'S'  public static final char ACK\_FLAG = 'A'; // ACK  public static final char NACK\_FLAG = 'N'; // Nack  public static final char XACK\_FLAG = 'X'; // Excess  public static final char UPGRADE\_FLAG = 'V';  public static final char END\_OF\_TRANSACTION = 'T'; // EOT  public static final char NO\_SCRIPT\_FLAG = 'B'; /\* No Script \*/ |

## 기타

* 사용자가 알아야 할 주요 설정 값
  + ALL\_SYNC : 동기화의 범주를 전체로 설정한다.
  + MOD\_SYNC : 동기화의 범주를 부분으로 설정한다. 이때 부분이라는 것은 마지막 동기화 한 이후에 입력/삭제/갱신이 발생한 레코드를 의미한다.
  + FIELD\_DEL, RECORD\_DEL : upload/download시 각각 필드와 레코드의 구분자로 사용되는 값이다.
  + NORMAL, ABNORMAL : 네트웍을 종료할 때 각각 정상, 오류에 따른 비정상 종료를 나타낸다.
  + HUGE\_STRING\_LENGTH : 네트웍을 사용할 때 한번에 담을 수 있는 최대 버퍼의 크기이다.

|  |
| --- |
| public static final int ALL\_SYNC = 1;  public static final int MOD\_SYNC = 2;  public static final int DBSYNC = 1;  public static final char FIELD\_DEL = 0x08;  public static final char RECORD\_DEL = 0x7F;  public static final int NORMAL = 1;  public static final int ABNORMAL = 2;  public static final int HUGE\_STRING\_LENGTH = 8156; |

## mSync\_InitializeClientSession()

* 클라이언트 쪽 소켓을 초기화하고 로그 파일을 생성하며 mSync 서버와의 타임아웃 값을 설정하는 함수

|  |
| --- |
| public native static void mSync\_InitializeClientSession (String filename, int timeout); |

* Parameter
  + filename: 클라이언트 쪽 로그 파일의 이름.  
     디렉토리도 포함한 이름이며 파일만 생성하므로 디렉토리는 미리 생성되어 있어야 한다
  + timeout : mSync 서버와 통신할 때의 타임아웃을 지정하는 것으로 단위는 초(second)이다.
* Return
  + None

## mSync\_ConnectToSyncServer()

* mSync 서버에 연결하는 함수

|  |
| --- |
| public native static int mSync\_ConnectToSyncServer (String serverAddr, int port); |

* Parameter
  + serverAddr: mSync 서버의 IP address
  + port : mSync 서버의 접속 port 번호
* Return
  + 0 : success
  + 음수 : fail

## mSync\_AuthRequest()

* 사용자에 대한 인증을 요청 및 버전 체크 등을 수행한다.

|  |
| --- |
| public native static int mSync\_AuthRequest (String id, String password,  String application, int version); |

* Parameter
  + id : Admin관리 도구를 통해 등록한 사용자의 ID
  + password : Admin 관리 도구를 통해 등록된 사용자의 Password
  + application: Admin 관리 도구를 통해 지정된 Application 이름
  + version : application의 버전( 자연수 )
* Return
  + 음수 : fail
  + NACK\_FLAG : id가 등록되어 있지 않거나 password 값이 잘못된 경우로 인증 실패를 나타낸다.
  + XACK\_FLAG : mSync 서버에 허용된 사용자보다 더 많은 사용자가 등록되어 mSync 서버를 이용할 수 없음을 나타낸다.
  + UPGRADE\_FLAG : application의 version이 높은 게 있어서 upgrade해야 할 필요가 있는 경우를 나타낸다
  + ACK\_FLAG : 인증에 문제가 없으며 계속 동기화 작업을 진행한다.  
    ACK\_FLAG를 받지 못한 경우는 계속해서 진행하지 않고 적절한 조처를 취해야 한다.

## mSync\_SendDSN()

* 동기화할 테이블들이 있는 데이터베이스 서버를 지정할 수 있도록 DSN 또는 SID 값을 전달하는 함수로 데이터베이스 서버 DSN 값이 아니라 admin 도구에서 지정한 클라이언트 쪽에서 인식하고 있는 DSN 값을 넘겨준다

(Admin 관리 도구를 통해서 일반적인 DBMS에 접근하는 DSN이 아닌 mSync Server가 관리하는 특정 Application으로 접근하는 DSN을 설정하고 있다.)

|  |
| --- |
| public native static int mSync\_SendDSN (String dsn); |

* Parameter
  + Dsn: client에서 인식하는 dsn값
* Return
  + 0 : success
  + 음수 : fail

## mSync\_SendTableName()

* 동기화할 테이블의 이름을 전달하는 함수로 admin 도구에서 지정한 서버쪽 테이블 이름을 파라미터로 전달하며 전체 데이터의 동기화인지, 변경된 데이터만의 동기화인지를 syncMode라는 파라미터에 전달한다

|  |
| --- |
| public native static int mSync\_SendTableName (int syncmode, String tablename); |

* Parameter
  + syncmode: sync mode 지정  
    ALL\_SYNC : 전체 데이터의 copy인 경우  
    MOD\_SYNC : 변경된 데이터만 sync 할 경우
  + tablename: 서버쪽의 sync 대상 table의 이름
* Return
  + 0 : success
  + 음수 : fail

## mSync\_SendUploadData()

* 클라이언트쪽의 변경 사항들을 mSync 서버에 전달하는 역할을 담당하는 함수로 변경 사항에 대한 flag와 데이터를 파라미터로 지정한다

|  |
| --- |
| public native static int mSync\_SendUploadData (String flag, String data); |

* Parameter
  + flag: upload할 data의 변경 사항을 나타내는 flag  
    INSERT\_FLAG / DELETE\_FLAG / UPDATE\_FLAG
  + data : upload data의 buffer  
    data의 내용은 Admin 관리 툴을 통해 등록한 Sync 처리용 script의 “?”에 해당하는 내용으로 ”?”의 순서와 동일하게 지정해 준다.

각 필드 값의 구분자로는 FIELD\_DEL(0x08), 각 레코드의 구분자로는 RECORD\_DEL(0x07) 을 사용한다.

Buffer의 maximum size는 HUGE\_STRING\_LENGTH이다

|  |
| --- |
| 예) Update에 대한 script가 다음과 같은 경우  update employee set ename = ? where empid=?  Upload data의 값은  ename’0x08empid’0x7Fename’’0x08empid’’0x7Fename’’’0x08empid’’’0x7F…이 된다.  |-----| |---| |----| |----| |-----| |-----| 🡺 데이터 값 |

레코드 구분자의 바로 앞에 있는 필드 끝에는 필드 구분자가 없음에 유의해야 한다.

* Return
  + 0 : success
  + 음수 : fail

## mSync\_SendUploadData2()

* mSync\_SendUploadData()와 내용은 동일하며, 차이점은 byte[] 형태로 데이터를 넘겨준다는데 있다.

|  |
| --- |
| public native static int mSync\_SendUploadData2 (String flag, byte[] data); |

* Parameter
  + flag: upload할 data의 변경 사항을 나타내는 flag  
    INSERT\_FLAG / DELETE\_FLAG / UPDATE\_FLAG
  + data : upload data의 byte buffer  
    mSync\_SendUploadData()함수를 이용할 때와 같이 준비된 String 데이터를 byte[]로 변환해서 넘겨주면 된다.
* Return
  + 0 : success
  + 음수 : fail

## mSync\_UploadOK()

* 업로드 과정이 종료되었음을 mSync 서버에 알리고 정상적으로 반영이 되었는지 확인하는 함수로 업로드 데이터가 없어서 mSync\_SendUploadData()를 호출하지 않더라도 이 함수는 호출해야 하고 다운로드 스크립트에 조건 절(where 문)을 사용한 경우에는 조건 절에서 사용할 데이터 값을 전달파라미터로 전달한다.

|  |
| --- |
| public native static int mSync\_UploadOK (String parameter); |

* Parameter
  + Parameter : 다운로드의 조건 절에 들어갈 파라미터의 값들을 스크립트의 순서대로 업로드 데이터와 같은 형식으로 만들어 준다
* Return
  + 0 : success
  + 음수 : fail

## mSync\_ReceiveDownloadData ()

* 데이터베이스 서버의 변경 사항을 클라이언트 쪽에 전달하는 역할을 하는 함수로서 각 데이터들은 다운로드 스크립트에 지정한 필드의 순서대로 각 필드의 구분자와 레코드의 구분자를 이용하여 만들어져 있으며 mSync 클라이언트에서 이 데이터들을 파싱하여 레코드와 필드의 데이터를 얻어 내는 작업이 필요하다

|  |
| --- |
| public native static String mSync\_ReceiveDownloadData (); |

* Parameter
  + None
* Return
  + NULL : No more result
* String:

|  |
| --- |
| 예) 다운로드에 대한 스크립트가 다음과 같은 경우  select empid, ename, dept from employee  다운로드 데이터 값은  empid’0x08ename’0x08dept’0x7Fempid’’0x08ename’’0x08dept’’0x7F  empid’’’0x08ename’’’0x08dept’’’0x7F…… 이 된다. |

레코드 구분자의 바로 앞에 있는 필드 끝에는 필드 구분자가 없음에 유의해야 한다.

## mSync\_ReceiveDownloadData2()

* mSync\_ReceiveDownloadData() 함수와 동일한 기능을 수행한다.
* 다만, byte Array형태로 리턴한다는 점이 다르다.

|  |
| --- |
| public native static byte[] mSync\_ReceiveDownloadData2(); |

* Parameter
  + None
* Return
  + NULL : No more result
  + Byte[] : byte Array형태로 리턴된 값을 적당한 코드셋의 String으로 변환한 후 사용할 수 있다.

## mSync\_Disconnect()

* mSync 서버와의 접속을 종료하는 함수로 동기화 작업이 끝나면 호출해준다

|  |
| --- |
| public native static int mSync\_Disconnect (int mode); |

* Paramter
  + Mode : 종료 방법 설정  
    NORMAL : 동기화 과정이 정상적으로 종료된 경우

ABNORMAL : 동기화 과정에 에러가 발생한 후 강제로 종료하는 경우

* Return
  + 0 : success
  + 음수 : fail

## mSync\_ClientLog()

* 로그를 출력하는 함수로 mSync\_InitializeClientSession( )에서 지정한 log file에 log가 남는다

|  |
| --- |
| public native static void mSync\_ClientLog (String msg); |

* Parameter
  + Msg : 로그로 남길 message
* Return
  + 0 : success
  + 음수 : fail

## mSync\_GetSyncError ()

* 동기화 과정 중의 에러가 생기는 경우 즉, API 호출 결과 음수가 return 된 경우 에러 메시지를 구하는 함수로 특별한 에러 원인이 없는 경우 “No Error msg”를 출력한다.

|  |
| --- |
| public native static String mSync\_GetSyncError (); |

* Parameter
  + None
* Return
  + String : 에러 메시지 내용  
    "No Error msg” : 특별한 에러 원인이 없는 경우

“Send Error” : 서버로 데이터를 보내다가 통신에 에러가 난 경우

“Receive Error” : 서버로부터 데이터를 받다가 통신에 에러가 난 경우

“Timeout” : 서버와의 통신이 timeout된 경우

“Message is cut off” : 받아야 하는 메시지 사이즈 보다 적게 받은 경우

# mSyncClientJNI 사용 방법

1. mSyncClientJNI.java를 작성할 Example.java 코드의 폴더에 복사
2. Example.java 작성 (코드 작성시 mSyncClientJNI.java에 정의된 함수들을 이용)
3. Example.java 작성시 libmSyncClientJNI.so를 링크 하도록 코드 추가
   1. 예

|  |
| --- |
| public class Example  {  static  {  System.loadLibrary("mSyncClientJNI"); }  }  } |