
(주)유진투자증권

SmartChamp OpenAPI 개발가이드

내용

1. 소개	3
1.1 SmartChamp OpenAPI란	3
1.2 개발환경	3
1.3 제공서비스 범위	4
2. OpenAPI 적용	4
2.1 파일 구성	4
2.2 샘플 프로젝트 구성	4
2.3 작성하는 앱의 필요 권한	6
2.4 난독화 예외 적용	6
3. SmartChamp OpenAPI 동작	6
3.1 초기화 및 시작	6
3.2 종료	8
3.3 로그인	8
3.4 서비스 조회	9
3.5 실시간 조회	11
4. SmartChamp OpenAPI 명세서	12
4.1 클래스 명세서	12
4.1.1 SmartChampAPI	12
4.1.2 SmartChampTran	31
4.1.3 SmartChampReal	38
4.1.4 SmartChampDefs	43
4.1.5 SmartChampDev	44
4.2 인터페이스	45
4.2.1 INetInitListener	45

4.2.2	INetLoginListener	49
4.2.3	INetLogoutListener.....	50
4.2.4	IAccountPwdCheckListener	50
4.2.5	ITranDataListener	51
4.2.6	IRealDataListener.....	52
4.2.7	ICertImportListener	54
4.2.8	ICertExportListener.....	55
4.2.9	ICertDeleteListener.....	55

1. 소개

1.1 SmartChamp OpenAPI란

스마트 챔피언 모바일 앱의 시세 조회와 주문 등을 고객이 직접 작성할 수 있도록 제공하는 라이브러리 제품입니다. 안드로이드 스튜디오에서 aar파일을 링크하여 원하는 형태의 어플리케이션을 작성할 수 있습니다.

1.2 개발환경

SmartChamp OpenAPI는 안드로이드 스튜디오에서 Java버전으로 개발된 모듈입니다.

라이브러리 컴파일 환경의 android sdk 정보는 다음과 같습니다.

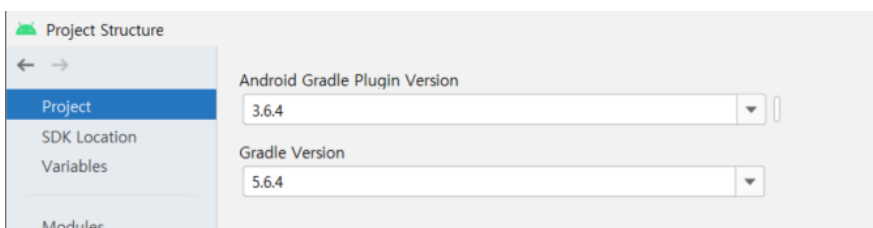
항목	내용
compileSdkVersion	31
minSdkVersion	16
targetSdkVersion	31

SmartChamp OpenAPI 빌드환경은 다음과 같습니다. 이와 동일하게 맞춰 프로젝트를 생성하는 것은 아닙니다.

- 안드로이드 스튜디오



- Gradle 정보



1.3 제공서비스 범위

주식 (주식/ELW/ETF/ETN)

지수 선물/옵션, 주식 선물/옵션, 변동성/섹터

해외주식 (미국, 중국, 홍콩, (일본제외))

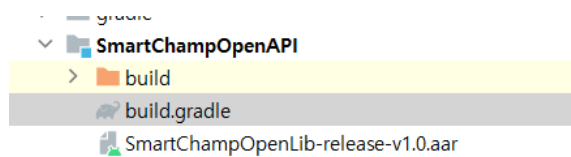
조건검색

공동인증서 관리(내보내기/가져오기/삭제)

2. OpenAPI 적용

2.1 파일 구성

라이브러리는 arr형태로 제공되며, aar모듈을 추가하여 다음과 같이 사용하면 됩니다.



build.gradle 구성은 다음과 같습니다.

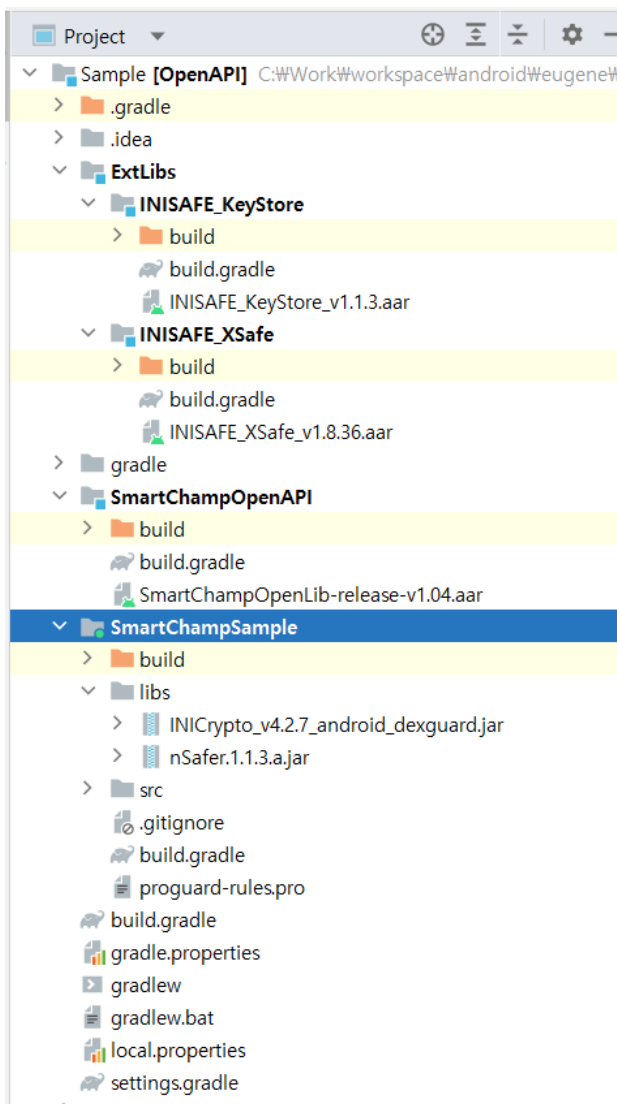
```
1 configurations.maybeCreate("default")
2 artifacts.add("default", file('SmartChampOpenLib-release-v1.0.aar'))
```

SmartChamp OpenAPI는 서버와의 통신을 위한 암호화 라이브러리와 공동인증서 관리/전자서명 작업을 위해 별도의 보안 라이브러리를 사용합니다. 그래서 OpenAPI를 사용하는 경우 보안 라이브러리까지 링크를 해줘야 합니다. 반드시 사용해야 하는 보안 라이브러리 목록은 다음과 같습니다.

라이브러리	형태	설명
INISAFE_KeyStore_v1.1.3	Aar	암호화 라이브러리
INISAFE_XSafe_v1.8.36	Aar	공동인증 라이브러리
INICrypt_v4.2.7_android_dexguard	Jar	암호화 라이브러리

2.2 샘플 프로젝트 구성

샘플 프로젝트의 구성은 다음과 같습니다.



샘플 앱 build.gradle에 dependency는 다음과 같습니다.

```

22 dependencies {
23     api fileTree(include: ['*.jar'], dir: 'libs')
24
25     implementation project(':SmartChampOpenAPI')
26
27     //noinspection GradleCompatible
28     implementation 'com.android.support:appcompat-v7:28.0.0'
29     implementation 'androidx.recyclerview:recyclerview:1.2.1'
30
31     implementation project(path: ':ExtLibs:INISAFE_KeyStore')
32     implementation project(path: ':ExtLibs:INISAFE_XSafe')
33
34     api files('libs/INICrypto_v4.2.7_android_dexguard.jar')
35 }

```

2.3 작성하는 앱의 필요 권한

SmartChamp OpenAPI는 네트워크 상태 체크를 통한 서버와의 통신 상태 유지 기능 및 라이브러리 동작에 필요한 파일을 최신 상태로 유지하는 기능이 필요합니다. 또한 서버에 단말을 인식할 수 있는 정보를 전달해야 해야 해서 단말정보를 취득할 수 있는 기능 또한 필요합니다. 이를 위해 앱 작성 시 다음과 같은 권한이 필요합니다. AndroidManifest.xml에 다음과 같이 권한을 추가한 후

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_NUMBERS"/>
```

앱 초기 시작 시 다음과 같이 권한체크 코드가 필요합니다.

```
ContextCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) != PackageManager.PERMISSION_GRANTED ||
ContextCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_NUMBERS) != PackageManager.PERMISSION_GRANTED ||
ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED ||
ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED
```

자세한 사항은 샘플앱 SampleActivity의 checkPermission() 함수를 참고합니다.

2.4 난독화 예외 적용

SmartChamp OpenAPI를 사용하여 만든 앱을 배포 시 난독화 적용을 위해, 기존 보안 라이브러리는 난독화가 적용된 버전입니다. 해당 라이브러리에서 난독화 예외 적용이 되어야 정상 동작하는 경우가 있습니다. 샘플 프로젝트의 SmartChampSample 모듈의 proguard-rules.pro의 예외적용 rule을 참조하여 예외적용이 필요합니다.

3. SmartChamp OpenAPI 동작

SmartChamp OpenAPI를 사용하는 주요 작업별로 설명합니다.

3.1 초기화 및 시작

SmartChamp OpenAPI를 사용하기 위해 초기화 및 필요한 리스너 등록하고, 서버와의 통신을 위해 접속 작업을 수행합니다. 샘플 앱에 적용된 코드는 다음과 같습니다.

```
// 서버 접속모드 지정, api 초기화 전에 호출해야 함
SmartChampAPI.setDevMode(SmartChampDefs.SERVER_MODE_DEV);

// 초기화
SmartChampAPI.initOpenAPI(this);

//Listener 셋팅
SmartChampAPI.getInstance().setInitListener(new INetInitListener() {
    @Override
    public void onSessionConnecting() {}

    @Override
    public void onSessionConnected(boolean isSuccess, String strErrorMsg) {}

    @Override
    public void onAPIVersionState(int nCheckState, String sUpdateInfo) {}

    @Override
    public void onUpdateDownloadState(int nState, int nIndex, int nCount, String sMessage) {}

    @Override
    public void onMasterLoadState(int nState) {}

    @Override
    public void onInitFinished() {}

    @Override
    public void onRequiredRefresh(int nReason) {}

    @Override
    public void onRequestedTerminate() {}
});

SmartChampAPI.getInstance().startConnect();
```

setDevMode함수는 앱 작성 시 테스트 서버(개발용)에 접속해서 개발할 경우 호출하는 함수입니다. 실제 운영되는 서버에 접속할 것인지 테스트를 위한 개발용 서버에 접속할 것인지 선택합니다.

setInitListener는 startConnect()하기 전에 호출해서 리스너를 등록해줘야 초기 작업에 대한 이벤트를 받을 수 있습니다.

startConnect() 함수를 호출하게 되면 통신 모듈이 동작하여 서버 접속 및 암호화 채널 연결, 최신 파일 업데이트, 필요 정보 로드 작업을 수행합니다. 단계별 작업 및 INetInitListener를 통해 호출되는 이벤트 함수는 다음과 같습니다. 단계별 리턴되는 함수에서 성공/실패에 대한 정보를 주는 경우, 성공하면 다음단계로 넘어가고, 실패하면 그 상태에서 중지됩니다. 각각의 함수에 대한 설명은 INetInitListener 명세서 참고 바랍니다.

초기화 단계	INetInitListener	설명
접속	onSessionConnecting	접속 시작을 알림
	onSessionConnected	접속 결과 알림 성공/실패 알림
업데이트	onAPIVersionState	현재 사용되는 통신API의 버전을 체크하고, 업데이트가 필요한지 판단합니다. 반드시 업데이트가 필요한 경우 상위버전의 라이브러리가 적용된 앱이 필요한 상황이므로 진행이 중지되

		고, 앱 종료가 요구됩니다. onRequestedTerminate()가 호출됩니다.
	onUpdateDownloadState	API에서 사용되는 파일의 업데이트가 필요한 경우 진행상황에 대한 정보를 알립니다. 자세한 사항은 INetInitListener 명세서 참조.
종목코드 로드	onMasterLoadState	API에서 사용하는 파일을 읽는 작업 상태를 표시합니다.
완료	onInitFinished	초기 접속 작업 완료됨
기타	onRequiredRefresh	앱 사용 중에 네트워크 단절이나 앱 비활성상태 지속 등으로 접속이 끊어진 경우, 재접속이 자동으로 이루어지며, 화면 갱신을 해야 현재 상태를 유지할 수 있으므로 필요한 경우 이 함수에서 화면 갱신을 수행해야 합니다.
	onRequestedTerminate	접속이 종료되어야 하는 경우 이 함수가 호출됩니다. 앱 종료 작업을 수행하면 됩니다.

3.2 종료

SmartChamp OpenAPI를 사용하는 앱이 종료되는 시점에 인스턴스 해제 작업을 호출합니다.

```
/**
 * Open API 종료...
 */
SmartChampAPI.releaseOpenAPI();
```

별도의 결과 리턴이나 리스너를 통한 상태 알림을 하지 않습니다.

3.3 로그인

SmartChamp OpenAPI의 서비스 호출이나 계좌 목록등을 얻으려면 로그인 작업 필요합니다. 주요 입력은 아이디, 아이디비밀번호, 공동인증서 DN, 공동인증서 비밀번호, 제휴사 코드입니다. 로그인 유형에 따라 입력값이 달라지며 아래 코드를 참조 바랍니다.

setLoginListener로 로그인 관련 리스너를 설정해야 로그인 결과를 수신할 수 있습니다. 로그인 시작 시 onLoginStarted()가 호출되고, 로그인이 성공하든 실패하든 결과는 onLoginResult()로 호출됩니다.

```

SmartChampAPI api = SmartChampAPI.getInstance();
api.setLoginListener(new INetLoginListener() {
    @Override
    public void onLoginStarted() {
        // 로그인 시작
    }

    @Override
    public void onLoginResult(boolean isSuccess, String strErrorMsg) {
        // 로그인 완료됨
    }
});

//로그인 시작
if(m_isSiseLogin) // 시세전용 로그인
    api.loginUser(getUserId(), getUserPwd(), "", getCorpCode());
if(m_isCertLogin) // 거래용 로그인
    api.loginUser(getCertDN(), getCertPwd(), getCorpCode());
else
    api.loginUser(getUserId(), getUserPwd(), getCertPwd(), getCorpCode());

```

3.4 서비스 조회

서비스(TR)는 앱에서 서비스에 대한 객체(SmartChampTran)를 생성하고, 필요한 서비스 정보를 로드(setQueryFile)하여 SmartChampTran이 서비스 정보를 사용할 수 있게 하고, 요청에 필요한 데이터를 구성(setDataValue)하고 서버에 요청(requestData)하고, ITranDataListener를 통해 요청 결과를 수신하는 방식입니다.

서비스를 사용하기 위해 초기화 작업을 합니다. 다음과 같이 인스턴스 생성하고 initInstance로 Listener를 등록합니다. ITranDataListener에 대한 자세한 내용은 ITranDataListener명세서 부분을 참고 바랍니다.

setTranId()로 별도의 서비스 아이디를 지정할 수 있으며, 지정하지 않으면 setQueryFile에 지정한 이름이 기본값으로 사용됩니다. ITranDataListener의 함수가 호출될 때, sTranID에 서비스 아이디가 들어옵니다. 한 뷰에서 SmartChampTran를 여러 개 지정해서 사용할 경우 ITranDataListener를 함께 사용할 수 있으며, 이벤트 함수 호출될 때 어떤 서비스인지 구분하기 위해 sTranID를 사용할 수 있습니다.

```

SmartChampTran m_tranCurrPrice = null;           // 주식 현재가 조회

m_tranCurrPrice = new SmartChampTran(getContext());
m_tranCurrPrice.initInstance(new ITranDataListener() {

    @Override
    public void onTranBeforeRequest(String sTranID, boolean isNextQuery) {
        // 서비스 요청 전에 호출됨
    }

    @Override
    public void onTranDataReceived(String sTranID, String sMsgCode, String sMsgText) {
        // 요청 결과 수신
    }

    @Override
    public void onTranError(String sTranID, String sErrCode, String sErrText) {
        // 요청 오류 수신
    }

    @Override
    public void onTranTimeout(String sTranID) {
        // 서버 응답 없음
    }

});
m_tranCurrPrice.setQueryFile("stmst2");
m_tranCurrPrice.setTranId("주식현재가");

```

위와 같이 초기화가 이루어진 후, 입력 값 지정 후 서버에 요청한다.

```

String strItemCode = "000660";
String strFullCode = SmartChampAPI.getInstance().getItemFullCode(strItemCode);

// 입력전 데이터 초기화 (반드시 해야 할 필요 없음)
m_tranCurrPrice.clearInputData();
m_tranCurrPrice.clearOutputData();

// 입력 데이터 설정
m_tranCurrPrice.setDataValue( "InRec1", "sCode", 0, strFullCode);

// 서버에 요청
m_tranCurrPrice.requestData();

```

요청에 대한 결과는 ITranDataListener를 통해서 받습니다. 다음은 데이터 수신해서 sTranID를 구분하고, 결과 데이터를 얻는 방식의 예시입니다.

```

@Override
public void onTranDataReceived(String sTranID, String sMsgCode, String sMsgText)
{
    if(sTranID.equals("주식현재가"))
    {
        // 현재가
        String strPrice = m_tranCurrPrice.getDataValue("OutRec1", "LCPrice", 0);
        int nPriceAttr = m_tranCurrPrice.getDataAttr("OutRec1", "LCPrice", 0);
        // 등락
        String strDiff = m_tranCurrPrice.getDataValue("OutRec1", "LDiff", 0);
        int nDiffAttr = m_tranCurrPrice.getDataAttr("OutRec1", "LDiff", 0);
        // 등락기호
        String strSign = m_tranCurrPrice.getDataValue("OutRec1", "cPCheck", 0);
        int nSignAttr = m_tranCurrPrice.getDataAttr("OutRec1", "cPCheck", 0);
    }
}

```

3.5 실시간 조회

실시간 서비스(REAL)를 사용하기 위해 초기화 작업을 합니다. 다음과 같이 인스턴스 생성하고 initInstance로 Listener를 등록합니다. IRealDataListener에 대한 자세한 내용은 IRealDataListener명세서 부분을 참고 바랍니다.

setRealId()로 별도의 서비스 아이디를 지정할 수 있으며, 지정하지 않으면 setQueryFile에 지정한 이름이 기본값으로 사용됩니다. sRealId는 조회 서비스와 동일하게 동작합니다.

```

// 멤버 변수
private SmartChampReal m_realCurrPrice = null; // 주식 체결 실시간

private void initCurrPriceReal(Context context)
{
    m_realCurrPrice = new SmartChampReal(context);
    m_realCurrPrice.initInstance(new IRealDataListener() {
        @Override
        public void onRealBeforeRequest(String sRealId) {
            // 실시간 요청 전에 호출됨
            // 여기서 setDataValue로 원하는 데이터를 지정할 수 있음
        }

        @Override
        public void onRealDataReceived(String sRealId) {
            // 서버에서 실시간 데이터가 수신된 경우
            // 여기서 수신된 데이터 처리
        }
    });
    m_realCurrPrice.setQueryFile("S00");
    m_realCurrPrice.setRealId("주식체결실시간");
}

```

초기화가 되었으면 다음과 같이 서버에 실시간 데이터를 요청합니다.

```

private void requestCurrPriceReal()
{
    m_realCurrPrice.releaseReal();

    String strFullCode = SmartChampAPI.getInstance().getItemFullCode("000660");

    m_realCurrPrice.setDataValue("InBlock1", "sCode", 0, strFullCode);

    m_realCurrPrice.requestReal();
}

```

서버에 요청 후 데이터가 갱신될 때마다 onRealDataReceived가 호출되며 여기서 데이터를 얻을 수 있습니다. 실시간 데이터가 더 이상 필요없는 경우 releaseReal()을 호출하면 이전에 설정한 값으로 실시간 데이터 수신을 해제 할 수 있습니다.

4. SmartChamp OpenAPI 명세서

4.1 클래스 명세서

4.1.1 SmartChampAPI

SmartChamp OpenAPI를 사용하기 위한 기본이 되는 클래스이며, 이 클래스에 정의된 함수는 다음과 같습니다.

분류	반환	함수명	설명
초기화	void	initOpenAPI	API 인스턴스 초기화
	void	releaseOpenAPI	API 인스턴스 해제
	void	setDevMode	운영/개발 접속 서버 선택
리스너 등록	void	setInitListener	접속 처리 결과 리스너 등록
	void	setLoginListener	로그인 결과 리스너 등록
	void	setLogoutListener	로그아웃 결과 리스너 등록
초기화시작	void	startConnect	초기 접속 시작
로그인 로그아웃	void	loginUser	로그인 요청
	void	logoutUser	로그아웃
사용자	String	getLoginUserId	사용자 아이디
	int	getLoginType	현재 로그인 유형 반환
계좌	void	checkAccPwd	계좌비밀번호 체크
	int	getAccountSize	계좌 개수 리턴
	String	getAccountNo	계좌번호 리턴
	String	getAccountName	계좌명 반환
	String	getAccountInfo	계좌정보 반환
인증서	Array<String>	getUserCertList	사용자 인증서 리스트 반환
	String	getUserCertInfo	인증서 정보 리턴
	void	requestCertImportNumber	인증서 가져오기 랜덤번호 생성
	void	importCert	인증서 가져오기 실행

	void	exportCert	인증서 내보내기 실행
	void	deleteCert	인증서 삭제
종목	Array<String>	getKospiCodeList	코스피 종목 리스트 반환
	Array<String>	getKosdaqCodeList	코스닥 종목 리스트 반환
	Array<String>	getStockCodeList	주식(코스피/코스닥) 종목 리스트 반환
	Array<String>	getETFCodeList	ETF 종목 리스트 반환
	Array<String>	getETNCodeList	ETN 종목 리스트 반환
	Array<String>	getFuturesCodeList	선물 종목 리스트 반환
	Array<String>	getOptionsCodeList	옵션 종목 리스트 반환
	Array<String>	getOverseasStockCodeList	해외주식 종목 리스트 반환
	Array<String>	getItemCodeList	종목 코드 리스트 반환(타입별)
	String	getItemCodeType	종목 시장구분 반환
	String	getItemFullCode	종목 표준 코드 반환
	String	getItemShortCode	종목 축약 코드 반환
	String	getItemMarketTypeText	시장 구분명 반환
	String	getItemCodeInfo	종목 정보 반환

각각의 함수의 설명은 다음과 같습니다.

- 초기화

함수명	static void initOpenAPI(Activity act)		
설명	SmartChamp OpenAPI초기화, 처음실행되는 activity에서 한번만 호출하면 됩니다		
리턴값	없음		
파라미터	Activity act	OpenAPI사용을 위한 activity	
내용	API사용 전에 반드시 호출해야 합니다.		
	<pre>// 초기화 SmartChampAPI.initOpenAPI(activity: this);</pre>		

함수명	static void releaseOpenAPI()		
설명	API 해제, 앱 종료 전에 호출하면 됩니다.		
리턴값	없음		
파라미터	없음		
내용	앱 종료 시점에 한번 호출합니다. <pre>/** * Open API 종료... */ SmartChampAPI.releaseOpenAPI();</pre>		

--	--

함수명	static void setDevMode(String strDev)	
설명	접속하는 서버를 선택하는 기능입니다.	
리턴값	없음	
파라미터	String strDev	운영/테스트 서버 선택
내용	<p>반드시 SmartChampAPI 초기화 전에 호출해야 합니다. 이 함수로 선택한 서버로 접속과 내부의 운영/테스트 선택 동작이 이루어 집니다.</p> <p>앱 최초 설치 후 기본값은 운영 서버입니다. 한번 지정이 되면 다음 앱 실행시에도 선택한 사항이 유지가 됩니다.</p> <pre> // 서버 접속모드 지정, api 초기화 전에 호출해야 함 SmartChampAPI.setDevMode(SmartChampDefs.SERVER_MODE_DEV); // 초기화 SmartChampAPI.initOpenAPI(activity: this); </pre> <p>입력값은 SmartChampDefs에 정의되어 있습니다.</p> <pre> // 서버 접속 모드 public static final String SERVER_MODE_REAL = "0"; public static final String SERVER_MODE_DEV = "1"; public static final String SERVER_MODE_DEFAULT = SERVER_MODE_REAL; </pre>	

- 리스너 등록

함수명	void setInitListener(INetInitListener listener)	
설명	세션 초기화, 접속 상태, 초기 업데이트 상태 등에 대한 결과를 통보받기 위한 리스너를 지정합니다.	
리턴값	없음	
파라미터	INetInitListener listener	API 초기 접속 처리 결과를 받기 위한 리스너
내용	SmartChamp OpenAPI 초기화 작업 후 startConnect 호출 전에 리스너를 등록해야 하며, 1개의 인스턴스만 등록해야 합니다.	

	<pre>//Listener 셋팅 SmartChampAPI.getInstance().setInitListener(new INetInitListener() { @Override public void onSessionConnecting() { } @Override public void onSessionConnected(boolean isSuccess, String strErrorMsg) { } @Override public void onAPIVersionState(int nCheckState, String sUpdateInfo) { } @Override public void onUpdateDownloadState(int nState, int nIndex, int nCount, String sMessage) { } @Override public void onMasterLoadState(int nState) { } @Override public void onInitFinished() { } @Override public void onRequiredRefresh(int nReason) { } @Override public void onRequestedTerminate() { } });</pre>	
	리스너 내의 함수 호출에 대한 내용은 인터페이스 INetInitListener 설명 부분 참조 바랍니다.	

함수명	void setLoginListener(INetLoginListener listener)	
설명	로그인 요청에 대한 결과를 리턴 받기 위한 리스너 등록	
리턴값	없음	
파라미터	INetLoginListener listener	로그인 결과를 받기 위한 리스너
내용	요청한 로그인에 대한 시작과 로그인 작업 완료 시 성공여부와 실패 시 실패메시지를 받을 수 있습니다.	

	<pre>SmartChampAPI.getInstance().setLoginListener(new INetLoginListener() { @Override public void onLoginStarted() { // 로그인 시작 } @Override public void onLoginResult(boolean isSuccess, String strErrorMsg) { // 로그인 완료됨 // 성공/실패, 실패 메시지 } });</pre> <p>리스너 내의 함수 호출에 대한 내용은 인터페이스 INetLoginListener 설명 부분 참조 바랍니다.</p>
--	---

함수명	void setLogoutListener(INetLogoutListener listener)	
설명	로그아웃 요청 시, 로그아웃 결과를 리턴 받을 수 있는 리스너를 등록	
리턴값	없음	
파라미터	INetLogoutListener listener	로그아웃 결과를 받을 리스너
내용	<p>로그아웃 요청에 대해 처리 완료를 받을 수 있다.</p> <pre>SmartChampAPI.getInstance().setLogoutListener(new INetLogoutListener() { @Override public void onLogoutFinished() { // 로그아웃 완료 알림 } });</pre> <p>리스너 내의 함수 호출에 대한 내용은 인터페이스 INetLogoutListener 설명 부분 참조 바랍니다.</p>	

- 초기 접속 시작

함수명	void startConnect()	
설명	OpenAPI로 서버에 접속을 요청합니다.	
리턴값	없음 처리 결과는 INetInitListener를 통해서 받습니다.	
파라미터	없음	
기타	<p>초기화 및 리스너 등록이 완료 된 후, 서버에 접속을 시작합니다. 접속 이후, API버전체크 및 최신 업데이트가 필요한 파일 다운로드 및 초기 로그인 작업 등이 처리됩니다. 접속시작부터 완료 시점까지 단계 별로 INetInitListener를 통해서 처리 상태가 이벤트 함수를 통해 리턴됩니다.</p> <pre>SmartChampAPI.getInstance().startConnect();</pre>	

- 로그인/로그아웃

함수명	void loginUser(String sUserId, String sUserPwd, String sCertPwd, String sCorpCode) void loginUser(String sCertDN, String sCertPwd, String sCorpCode)	
설명	로그인 유형에 따라 로그인 요청을 합니다.	
리턴값	없음 처리 결과는 INetLoginListener를 통해서 받습니다.	
파라미터	String sUserId	사용자 아이디 (대문자,숫자)
	String sUserPwd	사용자 아이디비번
	String sCertPwd	공동인증서 패스워드
	String sCertDN	공동인증서 DN (인증서 정보)
	String sCorpCode	제휴사 코드
내용	<p>SmartChamp OpenAPI에서 로그인은 조회(시세전용)용 로그인([사용자아이디+아이디비번])과 거래용 로그인([공동인증서DN+인증서비번] 또는 [아이디+아이디비번+공동인증서비번]) 2가지 방식으로 로그인 합니다.</p> <p>거래용 로그인 [공동인증서DN + 인증서비번] 방식이 일반적으로 사용되고, 공동인증서 1개에 여러 아이디가 생성된 경우, 공동인증서DN만으로 사용자아이디를 특정할 수 없으므로 [아이디+아이디비번+공동인증서비번] 방식으로 로그인해야 합니다.</p> <p>조회용 로그인 login(sUserId, sUserPwd, "", sCorpCode)와 같이 공동인증서 비번을 입력하지 않게 되면 조회용 로그인으로 동작합니다.</p> <pre>//로그인 시작 if(m_isSiseLogin) // 시세전용 로그인 SmartChampAPI.getInstance().loginUser(getUserId(), getUserPwd(), "", getCorpCode()); if(m_isCertLogin) // 거래용 로그인 SmartChampAPI.getInstance().loginUser(getCertDN(), getCertPwd(), getCorpCode()); else SmartChampAPI.getInstance().loginUser(getUserId(), getUserPwd(), getCertPwd(), getCorpCode());</pre>	

함수명	void logoutUser()	
설명	로그인 안한 상태(로그아웃) 상태로 전환합니다.	
리턴값	없음 처리가 완료되면 INetLogoutListener를 통해서 완료가 되었음을 받을 수 있습니다.	
파라미터	없음	
내용	<p>로그인한 상태에서 호출 시 내부에 가지고 있는 로그인 정보를 삭제하고 서버에 로그아웃을 알립니다.</p> <pre>SmartChampAPI.getInstance().logoutUser();</pre>	

- 사용자 정보

함수명	String getLoginUserId()	
설명	현재 로그인한 사용자의 아이디를 얻습니다.	
리턴값	사용자 아이디	
파라미터	없음	
내용	현재 로그인한 사용자의 아이디를 리턴합니다.	

함수명	int getLoginType()																
설명	현재 로그인한 로그인 유형을 얻습니다.																
리턴값	로그인 유형																
파라미터	없음																
내용	<p>로그인 유형은 SmartChampDefs클래스에 정의되어 있습니다.</p> <table border="1"> <thead> <tr> <th>유형정의</th><th>값</th><th>설명</th></tr> </thead> <tbody> <tr> <td>LOGIN_TYPE_NONE</td><td>0</td><td>로그인 안됨</td></tr> <tr> <td>LOGIN_TYPE_AUTO</td><td>1</td><td>자동로그인 로그인 후 로그아웃 하지 않고, 앱을 종료한 경우 후에 앱 실행 시 이전 아이디로 로그인 됨 시세 조회만 가능</td></tr> <tr> <td>LOGIN_TYPE_SISE</td><td>2</td><td>조회용 로그인 주문 서비스 호출은 불가하며, 시세조회 계좌조회만 가능</td></tr> <tr> <td>LOGIN_TYPE_CERT</td><td>3</td><td>주문용 로그인 공동인증서를 통해 로그인 한 경우 주문 서비스 호출 가능</td></tr> </tbody> </table> <p>로그인 유형 판단은 다음과 같습니다.</p> <pre>SmartChampAPI mng = SmartChampAPI.getInstance(); int nLoginType = mng.getLoginType(); String strUserId = mng.getLoginUserID(); String strStatus = nLoginType == SmartChampDefs.LOGIN_TYPE_CERT ? "거래용 로그인" : nLoginType == SmartChampDefs.LOGIN_TYPE_SISE ? "시세 로그인" : nLoginType == SmartChampDefs.LOGIN_TYPE_AUTO ? "자동로그인" : "비로그인";</pre>		유형정의	값	설명	LOGIN_TYPE_NONE	0	로그인 안됨	LOGIN_TYPE_AUTO	1	자동로그인 로그인 후 로그아웃 하지 않고, 앱을 종료한 경우 후에 앱 실행 시 이전 아이디로 로그인 됨 시세 조회만 가능	LOGIN_TYPE_SISE	2	조회용 로그인 주문 서비스 호출은 불가하며, 시세조회 계좌조회만 가능	LOGIN_TYPE_CERT	3	주문용 로그인 공동인증서를 통해 로그인 한 경우 주문 서비스 호출 가능
유형정의	값	설명															
LOGIN_TYPE_NONE	0	로그인 안됨															
LOGIN_TYPE_AUTO	1	자동로그인 로그인 후 로그아웃 하지 않고, 앱을 종료한 경우 후에 앱 실행 시 이전 아이디로 로그인 됨 시세 조회만 가능															
LOGIN_TYPE_SISE	2	조회용 로그인 주문 서비스 호출은 불가하며, 시세조회 계좌조회만 가능															
LOGIN_TYPE_CERT	3	주문용 로그인 공동인증서를 통해 로그인 한 경우 주문 서비스 호출 가능															

- 계좌

함수명	void checkAccPwd(String strAccNo, String strAccPwd, IAccountPwdCheckListener listener)	
설명	계좌번호에 대한 계좌비밀번호가 맞는지 체크합니다.	
리턴값	없음	

	체크 결과는 IAccountPwdCheckListener를 통해서 받습니다.	
파라미터	String strAccNo	계좌번호
	String strAccPwd	계좌비번
	IAccountPwdCheckListener listener	비밀번호 체크 결과를 리턴받을 리스너
내용	<p>OpenAPI를 사용하여 작성하는 앱에서 계좌비번 체크가 필요한 경우 사용합니다.</p> <pre> SmartChampAPI.getInstance().checkAccPwd(strAccNo, strAccPwd, new IAccountPwdCheckListener() { @Override public void onAccountPwdCheckResult(boolean isSuccess, String strAccNo, String strMessage) { if(isSuccess) { Toast.makeText(getContext(), text: "계좌 비밀번호 체크 완료", Toast.LENGTH_LONG).show(); // 여기서 계좌비밀번호 체크 후 해야할 작업 } else { Toast.makeText(getContext(), strMessage, Toast.LENGTH_LONG).show(); // 계좌비밀번호가 실패한 경우 } } }); </pre>	

함수명	Array<String> getAccountList()	
설명	로그인한 사용자의 계좌리스트를 리턴합니다.	
리턴값	계좌번호 리스트	
파라미터	없음	
내용	<p>사용자 계정의 계좌 중에 OpenAPI 사용 등록이 된 계좌의 목록을 가져옵니다.</p> <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrAcc = mngNet.getAccountList(); for(String strAccNo : arrAcc) { String strAccName = mngNet.getAccountName(strAccNo); // 계좌 리스트 처리 } </pre>	

함수명	String getAccountName(String strAccNo)	
설명	계좌번호에 해당하는 계좌의 계좌명을 리턴합니다.	
리턴값	주어진 계좌번호의 계좌명	
파라미터	String strAccNo	계좌번호
내용	<p>계좌명을 검색하여 리턴합니다.</p> <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); String strAccName = mngNet.getAccountName(strAccNo); </pre>	

- 공동 인증서

함수명	ArrayList<String> getUserCertList()	
설명	사용하는 앱에 저장된 공동인증서 리스트를 리턴합니다.	
리턴값	인증서 리스트	
파라미터	없음	
내용	<p>인증서 리스트를 얻습니다. 인증서의 DN값이 배열형태로 리턴됩니다. (인증서 DN : cn=xxxx,ou=xxx,o=xxx,c=kr)</p> <pre> ArrayList<String> arrCertList = SmartChampAPI.getInstance().getUserCertList(); for(String strDN : arrCertList) { // TODO 인증서 목록 처리 작업 } </pre>	

함수명	String getUserCertInfo(String strDN, String strInfo)																
설명	공동인증서DN에 해당하는 인증서의 정보를 리턴합니다. 정보의 종류는 strInfo에 의해 지정됩니다.																
리턴값	인증서 정보																
파라미터	String strDN	공동인증서 DN															
	String strInfo	정보 유형 (아래 내용 참조)															
내용	주어진 공동인증서 DN에 해당하는 공동인증서에서 원하는 정보를 리턴합니다.																
	<p>strInfo에 해당하는 값은 다음과 같습니다.</p> <table><tr><th>strInfo</th><th>내용</th></tr><tr><td>subjectname</td><td>사용자명</td></tr><tr><td>Policy</td><td>용도 구분</td></tr><tr><td>issuename</td><td>발급자명</td></tr><tr><td>issuerdn</td><td>발급자 인증서 DN</td></tr><tr><td>expired</td><td>만료여부</td></tr><tr><td>expiredtime</td><td>만료일</td></tr><tr><td>serialnum</td><td>인증서 일련번호</td></tr></table> <pre>SmartChampAPI mng = SmartChampAPI.getInstance(); ArrayList<String> arrCertList = mng.getUserCertList(); for(String strDN : arrCertList) { String strCertName = mng.getUserCertInfo(strDN, strInfo: "subjectname"); } }</pre>		strInfo	내용	subjectname	사용자명	Policy	용도 구분	issuename	발급자명	issuerdn	발급자 인증서 DN	expired	만료여부	expiredtime	만료일	serialnum
strInfo	내용																
subjectname	사용자명																
Policy	용도 구분																
issuename	발급자명																
issuerdn	발급자 인증서 DN																
expired	만료여부																
expiredtime	만료일																
serialnum	인증서 일련번호																

--	--

함수명	void requestCertImportNumber(ICertImportListener listener)	
설명	공동인증서 가져오기 1단계 - 랜덤번호 생성	
리턴값	없음 처리결과는 ICertImportListener를 통해서 리턴 받음	
파라미터	ICertImportListener listener	인증서 가져오기 결과를 받을 리스너
내용	<p>공동인증서 가져오기 작업은 2단계로 이루어진다. 1단계는 내보내기하는 단말에서 입력할 랜덤번호를 생성한다. 2단계는 내보내기하는 단말에서 랜덤번호를 입력하고 내보내기 요청을 한 후에 가져오기 작업을 할 단말에서 가져오기 요청을 한다.</p> <p>이 함수는 1단계 작업을 위한 랜덤번호 생성하는 함수이다. 2단계 작업은 다음목록의 importCert함수를 사용한다.</p> <pre> SmartChampAPI mngrAPI = SmartChampAPI.getInstance(); mngrAPI.requestCertImportNumber(new ICertImportListener() { @Override public void onCertImportNumber(boolean isSuccess, String strRandom, String strMessage) { // 가져오기를 위한 랜덤번호 생성 if(isSuccess) { // TODO 랜덤번호(strRandom)를 내보내기하는 단말에 입력 } } @Override public void onCertImportResult(boolean isSuccess, String strCertDN, String strMessage) { } }); </pre> <p>onCertImportNumber에 strRandom으로 생성된 랜덤번호를 리턴 받는다. 이 번호를 내보내기하는 단말에서 입력하면 된다.</p> <p>isSuccess가 false면 오류인 상황이며, 오류에 대한 내용은 strMessage로 전달 받는다.</p>	

함수명	void importCert(ICertImportListener listener)	
설명	공동인증서 가져오기 2단계 - 공동인증서 수신 후 로컬에 저장	
리턴값	없음 처리결과는 ICertImportListener를 통해서 리턴 받음	
파라미터	ICertImportListener listener	인증서 가져오기 결과를 받을 수 있는 리스너
내용	<p>공동인증서 가져오기 2단계 작업 중 2단계에 해당하는 함수이다. 인증서 내보내기를 하는 단말에서 랜덤번호를 입력 후 내보내기 요청한 후에, 이 함수를 호출하여 단말에 인증서를 저장한다.</p>	

	<pre> SmartChampAPI.getInstance().requestCertImportNumber(listener: CertImportDialog.this); SmartChampAPI mgrAPI = SmartChampAPI.getInstance(); mgrAPI.importCert(new ICertImportListener() { @Override public void onCertImportNumber(boolean isSuccess, String strRandom, String strMessage) { } @Override public void onCertImportResult(boolean isSuccess, String strCertDN, String strMessage) { // 인증서 가져오기 성공여부 // TODO strCertDN에 해당하는 인증서가 로컬에 저장됨 } }); </pre> <p>가져오기가 완료되면 onCertImportResult가 호출되고, 성공하면 strCertDN에 지정된 공동인증서가 가져오기가 완료된 상태임, 실패(isSuccess가 false)하면, strMessage에 실패 사유가 저장되어 있음.</p>
--	--

함수명	void exportCert(String strCertDN, String strCertPwd, String strSimpleCertNum, ICertExportListener listener)	
설명	인증서 내보내기 작업을 수행한다.	
리턴값	없음 내보내기 결과는 ICertExportListener를 통해서 전달된다.	
파라미터	String strCertDN	내보내기 할 공동인증서 DN
	String strCertPwd	인증서 비번
	String strSimpleCertNum	인증서 가져오기에서 생성된 랜덤번호
	ICertExportListener listener	인증서 내보내기 결과를 수신할 리스너
내용	<p>공동인증서 내보내기는 가져오기를 수행할 단말에서 생성된 랜덤번호를 입력받아, 지정된 인증서를 중계서버로 전송한다.</p> <pre>// 내보내기 랜덤번호는 4글자씩 3개로 나뉘서 입력받을 String strEdit1 = m_edit1.getText().toString(); String strEdit2 = m_edit2.getText().toString(); String strEdit3 = m_edit3.getText().toString(); if(TextUtils.getTrimmedLength(strEdit1) != 4 TextUtils.getTrimmedLength(strEdit2) != 4 TextUtils.getTrimmedLength(strEdit3) != 4) { Toast.makeText(getContext(), text: "승인번호가 입력되지 않았습니다.", Toast.LENGTH_SHORT).show(); return; } SmartChampAPI.getInstance().exportCert(m_strCertDN, strCertPwd, strSimpleCertNum: strEdit1 + strEdit2 + strEdit3, new ICertExportListener() { @Override public void onCertExportResult(boolean isSuccess, String strMessage) { // 내보내기 결과 } });</pre>	

	<p>내보내기가 완료되면 onCertExportResult가 호출되고, isSuccess 플래그로 성공/실패를 판단하면 된다. 실패시 strMessage에 실패사유가 들어있다.</p> <p>인증서 비번이 틀린 경우, 오류 카운트가 증가된다.</p>
--	--

함수명	void deleteCert(String strCertDN, String strCertPwd, ICertDeleteListener listener)	
설명	단말에 저장된 공동인증서를 삭제한다.	
리턴값	없음 결과는 ICertDeleteListener를 통해 리턴 받는다.	
파라미터	String strCertDN	삭제할 공동인증서 DN
	String strCertPwd	삭제할 인증서의 비번
	ICertDeleteListener listener	삭제 결과를 수신할 리스너
내용	<p>로컬에 저장된 공동인증서를 삭제한다. 서버와 주고받는 내용은 인증서 비번에 대한 오류 카운트 외에는 없다.</p> <pre> SmartChampAPI.getInstance().deleteCert(m_strCertDN, strCertPwd, new ICertDeleteListener() { @Override public void onCertDeleteResult(boolean isSuccess, String strMessage) { // TODO 처리 결과 } }); </pre> <p>입력한 DN에 해당하는 인증서의 비밀번호를 검증하고, 통과되면 로컬의 파일을 삭제한다. 삭제 후 onCertDeleteResult로 결과를 받는다.</p>	

● 종목 리스트 / 종목 정보

함수명	ArrayList<String> getKospiCodeList()	
설명	코스피 종목코드 리스트를 얻는다.	
리턴값	이름으로 정렬된 코스피 종목 리스트	
파라미터	없음	
내용	<p>코스피 종목 리스트를 반환합니다.</p> <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getKospiCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>	

--	--

함수명	ArrayList<String> getKosdaqCodeList()	
설명	코스닥 종목코드 리스트를 얻는다.	
리턴값	이름으로 정렬된 코스닥 종목 리스트	
파라미터	없음	
내용	코스닥 종목 리스트를 반환합니다. <pre>SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getKosdaqCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); }</pre>	

함수명	ArrayList<String> getStockCodeList()	
설명	국내주식 종목코드(코스피+코스닥) 리스트를 얻는다.	
리턴값	이름으로 정렬된 주식 종목 리스트	
파라미터	없음	
내용	국내 주식 종목 리스트를 반환합니다. <pre>SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getStockCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); }</pre>	

함수명	ArrayList<String> getETFCodeList()	
설명	ETF 종목코드 리스트를 얻는다.	
리턴값	이름으로 정렬된 ETF 종목 리스트	
파라미터	없음	
내용	ETF 종목 리스트를 반환합니다.	

	<pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getETNCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>
--	---

함수명	ArrayList<String> getETNCodeList()	
설명	ETN 종목코드 리스트를 얻는다.	
리턴값	이름으로 정렬된 ETN 종목 리스트	
파라미터	없음	
내용	ETN 종목 리스트를 반환합니다. <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getETNCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>	

함수명	ArrayList<String> getFuturesCodeList()	
설명	국내지수선물 종목코드 리스트를 얻는다.	
리턴값	지수선물 종목 리스트	
파라미터	없음	
내용	지수선물 종목 리스트를 반환합니다. <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getFuturesCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>	

함수명	ArrayList<String> getOptionsCodeList()	
설명	국내 지수옵션 종목코드 리스트를 얻는다.	
리턴값	지수옵션 종목 리스트	
파라미터	없음	
내용	지수옵션 종목 리스트를 반환합니다.	

	<pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getOptionsCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>
--	---

함수명	ArrayList<String> getOverseasStockCodeList()	
설명	해외주식 종목코드 리스트를 얻는다.	
리턴값	해외주식 종목 리스트	
파라미터	없음	
내용	<p>해외주식 종목 전체(미국,중국,홍콩) 리스트를 반환합니다.</p> <pre> SmartChampAPI mngNet = SmartChampAPI.getInstance(); ArrayList<String> arrItems = mngNet.getOverseasStockCodeList(); for(String strCode : arrItems) { // TODO 종목 리스트 처리 String strItemName = mngNet.getItemCodeInfo(strCode, strType: "name"); } </pre>	

함수명	ArrayList<String> getItemCodeList(String strType)																											
설명	주어진 유형에 맞는 종목코드 리스트를 얻는다.																											
리턴값	종목 리스트																											
파라미터	String strType	종목 유형																										
내용	strType에 따른 종목 리스트를 반환한다. strType에 들어갈 수 있는 값은 다음과 같다. <table><tr><td>strType</td><td>유형 설명</td></tr><tr><td>KOSPI</td><td>주식 - 코스피</td></tr><tr><td>KOSDAQ</td><td>주식 - 코스닥</td></tr><tr><td>STOCK</td><td>주식 - 코스피 + 코스닥</td></tr><tr><td>CFD</td><td>주식 - CFD</td></tr><tr><td>ELW</td><td>ELW</td></tr><tr><td>ETN</td><td>ETN</td></tr><tr><td>ETF</td><td>ETF</td></tr><tr><td>WARRANTS</td><td>신주인수권</td></tr><tr><td>KONEX</td><td>KONEX</td></tr><tr><td>KOTC</td><td>K-OTC</td></tr><tr><td>K200FUT</td><td>지수선물</td></tr><tr><td>K200OPT</td><td>지수옵션</td></tr></table>		strType	유형 설명	KOSPI	주식 - 코스피	KOSDAQ	주식 - 코스닥	STOCK	주식 - 코스피 + 코스닥	CFD	주식 - CFD	ELW	ELW	ETN	ETN	ETF	ETF	WARRANTS	신주인수권	KONEX	KONEX	KOTC	K-OTC	K200FUT	지수선물	K200OPT	지수옵션
strType	유형 설명																											
KOSPI	주식 - 코스피																											
KOSDAQ	주식 - 코스닥																											
STOCK	주식 - 코스피 + 코스닥																											
CFD	주식 - CFD																											
ELW	ELW																											
ETN	ETN																											
ETF	ETF																											
WARRANTS	신주인수권																											
KONEX	KONEX																											
KOTC	K-OTC																											
K200FUT	지수선물																											
K200OPT	지수옵션																											

	MINIFUT	미니선물
	MINIOPT	미니옵션
	WEEKLYOPT	위클리옵션
	KRX300FUT	KRX300선물
	KSD150FUT	코스닥150선물
	KSD150OPT	코스닥150옵션
	STKFUT	주식선물
	VFUTURE	변동성
	TFUTURE	섹터 선물
	CME	야간선물
	BOND	채권
	UPJONG	업종
	WSTOCK	해외주식 - 전체
	WSTOCK_US	해외주식 - 미국 전체
	WSTOCK_NY	해외주식 - 미국 - 뉴욕
	WSTOCK_NS	해외주식 - 미국 - 나스닥
	WSTOCK_AM	해외주식 - 미국 - 아멕스
	WSTOCK_CN	해외주식 - 중국 전체
	WSTOCK_SH	해외주식 - 중국 - 상해
	WSTOCK_SZ	해외주식 - 중국 - 심천
	WSTOCK_HK	해외주식 - 홍콩 전체
	WUPJONG	해외업종

함수명	ArrayList<String> getOverseasStockCodeList(String strNation, String strExchange)																						
설명	해외주식 종목에서 주어진 국가와 거래소에 해당하는 종목 리스트 반환																						
리턴값	해외주식 종목 리스트																						
파라미터	String strNation	국가 구분																					
	String strExchange	거래소 구분																					
내용	<p>원하는 국가와 거래소 구분값으로 종목 리스트 반환 구분값은 다음과 같다.</p> <table border="1"> <thead> <tr> <th>strNation</th><th>strExchange</th><th>종목</th></tr> </thead> <tbody> <tr> <td>US</td><td>NYS</td><td>미국 - 뉴욕</td></tr> <tr> <td>US</td><td>AMX</td><td>미국 - 아멕스</td></tr> <tr> <td>US</td><td>NAS</td><td>미국 - 나스닥</td></tr> <tr> <td>CN</td><td>SHS</td><td>중국 - 상해</td></tr> <tr> <td>CN</td><td>SZS</td><td>중국 - 심천</td></tr> <tr> <td>HK</td><td>HKS</td><td>홍콩</td></tr> </tbody> </table>		strNation	strExchange	종목	US	NYS	미국 - 뉴욕	US	AMX	미국 - 아멕스	US	NAS	미국 - 나스닥	CN	SHS	중국 - 상해	CN	SZS	중국 - 심천	HK	HKS	홍콩
strNation	strExchange	종목																					
US	NYS	미국 - 뉴욕																					
US	AMX	미국 - 아멕스																					
US	NAS	미국 - 나스닥																					
CN	SHS	중국 - 상해																					
CN	SZS	중국 - 심천																					
HK	HKS	홍콩																					

--	--

함수명	String getItemCodeType(String strCode)																																																																											
설명	종목의 종목유형을 반환한다.																																																																											
리턴값	종목유형																																																																											
파라미터	String strCode	종목코드 (단축코드)																																																																										
내용	종목을 분류하기 위한 정보를 리턴한다. 이 값으로 종목유형에 따라 서비스 호출이나 주문로직 등 별도 처리를 하기 위한 정보로 사용된다.																																																																											
	유형별 정의는 다음과 같다.																																																																											
	<table><tr><td>유형</td><td>설명</td><td>유형</td><td>설명</td></tr><tr><td>A</td><td>주식</td><td>R</td><td>채권</td></tr><tr><td>B</td><td>코스피</td><td>S</td><td>코스닥150선물</td></tr><tr><td>C</td><td>코스닥</td><td>T</td><td>국내 - 업종</td></tr><tr><td>D</td><td>ELW</td><td>U</td><td>유로50선물</td></tr><tr><td>E</td><td>ETF</td><td>V</td><td>코스닥150옵션</td></tr><tr><td>F</td><td>ETN</td><td>W</td><td>KRX300선물</td></tr><tr><td>G</td><td>KONEX</td><td>X</td><td>위클리옵션</td></tr><tr><td>H</td><td>K-OTC</td><td>a</td><td>해외주식</td></tr><tr><td>I</td><td>신주인수권</td><td>c</td><td>홍콩주식</td></tr><tr><td>J</td><td>지수선물</td><td>d</td><td>홍콩ETF</td></tr><tr><td>K</td><td>미니선물</td><td>f</td><td>중국 - 상해</td></tr><tr><td>L</td><td>지수옵션</td><td>g</td><td>중국 - 심천</td></tr><tr><td>M</td><td>미니옵션</td><td>i</td><td>미국 - 뉴욕</td></tr><tr><td>N</td><td>야간선물</td><td>j</td><td>미국 - 나스닥</td></tr><tr><td>O</td><td>변동성</td><td>l</td><td>미국 - 아멕스</td></tr><tr><td>P</td><td>섹터</td><td>m</td><td>미국 - 업종</td></tr><tr><td>Q</td><td>주식선물</td><td></td><td></td></tr></table>				유형	설명	유형	설명	A	주식	R	채권	B	코스피	S	코스닥150선물	C	코스닥	T	국내 - 업종	D	ELW	U	유로50선물	E	ETF	V	코스닥150옵션	F	ETN	W	KRX300선물	G	KONEX	X	위클리옵션	H	K-OTC	a	해외주식	I	신주인수권	c	홍콩주식	J	지수선물	d	홍콩ETF	K	미니선물	f	중국 - 상해	L	지수옵션	g	중국 - 심천	M	미니옵션	i	미국 - 뉴욕	N	야간선물	j	미국 - 나스닥	O	변동성	l	미국 - 아멕스	P	섹터	m	미국 - 업종	Q	주식선물		
	유형	설명	유형	설명																																																																								
	A	주식	R	채권																																																																								
	B	코스피	S	코스닥150선물																																																																								
	C	코스닥	T	국내 - 업종																																																																								
	D	ELW	U	유로50선물																																																																								
	E	ETF	V	코스닥150옵션																																																																								
	F	ETN	W	KRX300선물																																																																								
	G	KONEX	X	위클리옵션																																																																								
	H	K-OTC	a	해외주식																																																																								
	I	신주인수권	c	홍콩주식																																																																								
	J	지수선물	d	홍콩ETF																																																																								
	K	미니선물	f	중국 - 상해																																																																								
	L	지수옵션	g	중국 - 심천																																																																								
	M	미니옵션	i	미국 - 뉴욕																																																																								
	N	야간선물	j	미국 - 나스닥																																																																								
	O	변동성	l	미국 - 아멕스																																																																								
	P	섹터	m	미국 - 업종																																																																								
	Q	주식선물																																																																										

함수명	String getItemFullCode(String strCode)		
설명	주어진 종목에 대한 표준코드를 리턴한다.		
리턴값	종목의 표준코드(full code)		
파라미터	String strCode	종목의 단축코드	
내용	단축종목코드에 대한 표준코드를 구한다.		
	<pre>SmartChampAPI mngNet = SmartChampAPI.getInstance(); String strFullCode = mngNet.getItemFullCode(strCode: "000660"); // strFullCode == "KR7000660001"</pre>		

	시세조회, 주문, 실시간 시세 요청 시에 표준코드를 입력으로 해야함. 이 함수를 이용해서 단축코드에서 표준코드 구해서 입력으로 사용함.
--	---

함수명	String getItemShortCode(String strCode)	
설명	주어진 종목에 대한 단축코드를 리턴한다.	
리턴값	종목의 단축코드(short code)	
파라미터	String strCod	종목의 표준코드(full code)
내용	표준코드에 대한 단축코드를 구한다.	

함수명	String getItemMarketTypeText(String strCode)																																																	
설명	주어진 종목코드(단축코드)에 대한 시장유형 이름을 구한다.																																																	
리턴값	시장유형명																																																	
파라미터	String strCode	종목코드(단축코드)																																																
내용	종목에 대한 시장유형 이름을 구한다. 정의는 다음과 같다.																																																	
	<table><tr><th>시장유형명</th><th>유형 설명</th></tr><tr><td>KOSPI</td><td>주식 - 코스피</td></tr><tr><td>KOSDAQ</td><td>주식 - 코스닥</td></tr><tr><td>ELW</td><td>ELW</td></tr><tr><td>ETN</td><td>ETN</td></tr><tr><td>ETF</td><td>ETF</td></tr><tr><td>SRGT</td><td>신주인수권</td></tr><tr><td>KONEX</td><td>KONEX</td></tr><tr><td>KOTC</td><td>K-OTC</td></tr><tr><td>FUTURE</td><td>지수선물</td></tr><tr><td>OPTION</td><td>지수옵션</td></tr><tr><td>MINIFUTURE</td><td>미니선물</td></tr><tr><td>MINIOPTION</td><td>미니옵션</td></tr><tr><td>WEEKOPTION</td><td>위클리옵션</td></tr><tr><td>KRX300F</td><td>KRX300선물</td></tr><tr><td>KDQ150F</td><td>코스닥150선물</td></tr><tr><td>KDQ150_OPT</td><td>코스닥150옵션</td></tr><tr><td>JFUTURE</td><td>주식선물</td></tr><tr><td>VFUTURE</td><td>변동성</td></tr><tr><td>TFUTURE</td><td>섹터 선물</td></tr><tr><td>CMEFUTURE</td><td>야간선물</td></tr><tr><td>BOND</td><td>채권</td></tr><tr><td>UPJONG</td><td>업종</td></tr><tr><td>WSTK_NYS</td><td>해외주식 - 미국 - 뉴욕</td></tr></table>	시장유형명	유형 설명	KOSPI	주식 - 코스피	KOSDAQ	주식 - 코스닥	ELW	ELW	ETN	ETN	ETF	ETF	SRGT	신주인수권	KONEX	KONEX	KOTC	K-OTC	FUTURE	지수선물	OPTION	지수옵션	MINIFUTURE	미니선물	MINIOPTION	미니옵션	WEEKOPTION	위클리옵션	KRX300F	KRX300선물	KDQ150F	코스닥150선물	KDQ150_OPT	코스닥150옵션	JFUTURE	주식선물	VFUTURE	변동성	TFUTURE	섹터 선물	CMEFUTURE	야간선물	BOND	채권	UPJONG	업종	WSTK_NYS	해외주식 - 미국 - 뉴욕	
	시장유형명	유형 설명																																																
	KOSPI	주식 - 코스피																																																
	KOSDAQ	주식 - 코스닥																																																
	ELW	ELW																																																
	ETN	ETN																																																
	ETF	ETF																																																
	SRGT	신주인수권																																																
	KONEX	KONEX																																																
	KOTC	K-OTC																																																
	FUTURE	지수선물																																																
	OPTION	지수옵션																																																
	MINIFUTURE	미니선물																																																
	MINIOPTION	미니옵션																																																
	WEEKOPTION	위클리옵션																																																
	KRX300F	KRX300선물																																																
	KDQ150F	코스닥150선물																																																
	KDQ150_OPT	코스닥150옵션																																																
	JFUTURE	주식선물																																																
	VFUTURE	변동성																																																
	TFUTURE	섹터 선물																																																
	CMEFUTURE	야간선물																																																
	BOND	채권																																																
	UPJONG	업종																																																
	WSTK_NYS	해외주식 - 미국 - 뉴욕																																																

	WSTK_NAS	해외주식 - 미국 - 나스닥
	WSTK_AMX	해외주식 - 미국 - 아멕스
	WSTK_SHS	해외주식 - 중국 - 상해
	WSTK_SZS	해외주식 - 중국 - 심천
	WSTK_HKS	해외주식 - 홍콩 - 주식
	WSTK_HKETF	해외주식 - 홍콩 - ETF
	WUPJONG	해외업종

함수명	String getItemCodeInfo(String strCode, String strType)																																													
설명	주어진 종목의 정보를 반환한다.																																													
리턴값	종목정보																																													
파라미터	String strCode	종목코드(단축코드)																																												
	String strType	원하는 정보유형																																												
내용	<p>종목의 세부 정보를 구할 때 사용한다.</p> <p>종목 유형에 따라 strType이 달라질 수 있다.</p> <table border="1"> <thead> <tr> <th>종목분류</th><th>strType</th><th>설명</th></tr> </thead> <tbody> <tr> <td rowspan="5">공통</td><td>shcode</td><td>단축코드</td></tr> <tr> <td>expcode</td><td>표준코드(풀코드)</td></tr> <tr> <td>name</td><td>종목명(한글)</td></tr> <tr> <td>type</td><td>종목유형 getItemCodeType함수 참조</td></tr> <tr> <td>marketttext</td><td>종목유형명 getItemMarketTypeText함수 참조</td></tr> <tr> <td rowspan="8">선물옵션</td><td>callput</td><td>콜/풋 구분</td></tr> <tr> <td>spread</td><td>스프레드 종목 여부 "1" - true</td></tr> <tr> <td>fullname</td><td>긴종목명</td></tr> <tr> <td>atm</td><td>ATM구분</td></tr> <tr> <td>basemarket</td><td>기초자산 코드</td></tr> <tr> <td>optprice</td><td>옵션 행사가</td></tr> <tr> <td>enddate</td><td>종료일</td></tr> <tr> <td>multiplier</td><td>승수</td></tr> <tr> <td rowspan="6">해외주식</td><td>ename</td><td>종목명(영문)</td></tr> <tr> <td>symbolid</td><td>종목코드(단축코드)</td></tr> <tr> <td>isincode</td><td></td></tr> <tr> <td>upcodes</td><td>업종코드</td></tr> <tr> <td>divcode</td><td></td></tr> <tr> <td>tradeunit</td><td>거래단위</td></tr> </tbody> </table>		종목분류	strType	설명	공통	shcode	단축코드	expcode	표준코드(풀코드)	name	종목명(한글)	type	종목유형 getItemCodeType함수 참조	marketttext	종목유형명 getItemMarketTypeText함수 참조	선물옵션	callput	콜/풋 구분	spread	스프레드 종목 여부 "1" - true	fullname	긴종목명	atm	ATM구분	basemarket	기초자산 코드	optprice	옵션 행사가	enddate	종료일	multiplier	승수	해외주식	ename	종목명(영문)	symbolid	종목코드(단축코드)	isincode		upcodes	업종코드	divcode		tradeunit	거래단위
종목분류	strType	설명																																												
공통	shcode	단축코드																																												
	expcode	표준코드(풀코드)																																												
	name	종목명(한글)																																												
	type	종목유형 getItemCodeType함수 참조																																												
	marketttext	종목유형명 getItemMarketTypeText함수 참조																																												
선물옵션	callput	콜/풋 구분																																												
	spread	스프레드 종목 여부 "1" - true																																												
	fullname	긴종목명																																												
	atm	ATM구분																																												
	basemarket	기초자산 코드																																												
	optprice	옵션 행사가																																												
	enddate	종료일																																												
	multiplier	승수																																												
해외주식	ename	종목명(영문)																																												
	symbolid	종목코드(단축코드)																																												
	isincode																																													
	upcodes	업종코드																																												
	divcode																																													
	tradeunit	거래단위																																												

		floatpoint	소수점 자리
		ticktype	틱 유형
		currency	통화
		bymd	
		exchgcode	거래소 코드
		exchgname	거래소 이름
		exchgcode_eugene	유진 자체 거래소 코드
		nation	국가코드

4.1.2 SmartChampTran

조회(주문)용 서비스를 호출하기 위한 클래스, 호출하고자 하는 TR에 대해 1개체씩 정의하여 사용한다.

분류	반환	함수명	설명
초기화	void	initInstance	인스턴스 초기화
	void	clearInstance	인스턴스 해제
	void	setQueryFile	qry파일 로드
	void	setTranId	TR id 지정
	String	getTranId	TR id 반환
데이터	void	setDataHeader	패킷 헤더에 포함시킬 데이터 지정
	void	setScreenNo	화면 번호 지정
	void	clearInputData	입력 데이터 초기화
	void	clearOutputData	출력 데이터 초기화
	void	setDataValue	TR 입력/출력 필드에 데이터 지정
	String	getDataValue	TR 입력/출력 필드에 존재하는 데이터 리턴
	int	getDataAttr	TR 출력 필드의 데이터 속성값 리턴 (시세 색상표시용)
요청	int	requestData	데이터 요청
	int	requestNextData	넥스트 데이터 요청
	boolean	isMoreNextData	조회 후 연속조회를 위한 데이터가 존재하는지 여부
카운트	int	getDataCount	수신된 데이터 블록의 데이터 개수
메시지	String	getDataMsgText	응답 데이터 메시지
	String	getDataMsgCode	응답 데이터 메시지 코드

각각의 함수 설명은 아래와 같다.

- 초기화

함수명	void initInstance(ITranDataListener listener) void initInstance(String strTrCode, ITranDataListener listener)	
설명	TRAN 서비스 호출을 위해 인스턴스 초기화	
리턴값	없음	
파라미터	ITranDataListener listener	서비스 조회에 대한 결과 리스너
	String strTrCode	서비스명 (서비스가 정의된 파일명)
내용	<p>TRAN 서비스 호출을 위한 객체 초기화</p> <p>strTrCode가 입력된 함수는 초기화와 쿼리파일 로드를 한번에 수행한다. initInstance(listener)를 호출하여 초기화한 경우, setQueryFile(strTrCode)로 쿼리파일 로드를 수행한 후 서비스를 호출해야 한다.</p> <pre> // 인스턴스 정의 private SmartChampTran m_tranCurrPrice = null; // 주식 현재가 조회 // 주식 현재가 서비스 초기화 m_tranCurrPrice = new SmartChampTran(context); m_tranCurrPrice.initInstance(new ITranDataListener() { @Override public void onTranBeforeRequest(String sTranID, boolean isNextQuery) { // 서비스 요청시 서버로 데이터 전송 전에 호출됨 // 여기서 인풋 데이터의 값을 지정할 수 있음 } @Override public void onTranDataReceived(String sTranID, String sMsgCode, String sMsgText) { // 서버에 요청 후 결과 수신된 경우 호출됨 // 여기서 수신된 데이터 처리 } @Override public void onTranError(String sTranID, String sErrCode, String sErrText) { // 요청한 서비스가 오류가 발생한 경우 호출됨 } @Override public void onTranTimeout(String sTranID) { // 서비스 요청에 정해진 시간안에 응답이 없는 경우 호출됨 } }); m_tranCurrPrice.setQueryFile("stmst2"); m_tranCurrPrice.setTranId("주식현재가"); // ITranDataListener의 sTranID </pre> <p>ITranDataListener에 sTranID는 별도로 지정하지 않으면 서비스명과 동일하게 들어온다. setTranId로 별도의 아이디를 지정하면 지정된 아이디로 리스너의 이벤트 함수가 호출된다. 예를 들어 stmst2 서비스에 대해 여러 instance를 생성한 경우(ex: m_tranCurrPrice1, m_tranCurrPrice2), 각각의 인스턴스를 구별하기 위해 setTranId로 구별되는 값을 지정할 수 있다(ex : 주식현재가1, 주식현재가2). ITranDataListener를 여러 인스턴스에 동시에 사용해도 sTranID에 setTranId로 지정한 아이디로 들어와서 어느 인스턴스에 대한 이벤트인지 구분할 수 있다.</p>	

함수명	void clearInstance()
-----	----------------------

설명	TRAN 서비스 호출을 위한 인스턴스를 해제한다.	
리턴값	없음	
파라미터	없음	
내용	TRAN 서비스에 대한 인스턴스를 해제한다. 서비스 사용 후 에 더 이상 필요치 않으면 해제를 호출해야 한다.	

함수명	void setQueryFile(String strTrCode)	
설명	TRAN 서비스 인스턴스에서 주어진 서비스의 정보를 파일에서 읽는다.	
리턴값	없음	
파라미터	String strTrCode	서비스명 (서비스가 정의된 파일명)
내용	초기화된 인스턴스에서 사용할 서비스의 정보를 로드한다. initInstance(strTrCode, listener)를 통해서 초기화 했다면 이 함수를 호출하지 않는다. initInstance(listener)를 통해서 초기화한 경우에만 호출할 수 있다.	

함수명	void setTranId(String strTranId)	
설명	TRAN 서비스명 외에 별도의 TRAN서비스아이디 지정	
리턴값	없음	
파라미터	String strTranId	지정할 TRAN서비스 아이디
내용	TRAN 서비스 인스턴스의 TranId는 별도로 지정하지 않으면 setQueryFile로 지정한 서비스 이름이 기본값으로 지정된다. TRAN서비스를 구분해야 할 필요가 있거나, 별도의 아이디를 지정하고자 할 경우, 이 함수로 서비스 아이디를 지정한다.	

함수명	String getTranId()	
설명	서비스 인스턴스의 서비스 아이디를 구한다.	
리턴값	TRAN 아이디	
파라미터	없음	
내용	TRAN서비스 인스턴스의 아이디가 필요한 경우 호출한다.	

함수명	void setDataHeader(String strKey, String strValue)	
설명	TRAN서비스의 호출 시 요청데이터 헤더 부분에 지정되는 값을 지정할 수 있다.	
리턴값	없음	
파라미터	String strKey	지정할 데이터 이름(키)
	String strValue	지정할 데이터
내용	서비스 호출 전에 특정 예외처리를 위한 내용	
	시장유형명	유형 설명
	DETAIL_CONT	조회 시 다음 데이터 조회 가능하게 설정 "1"
	DETAIL_QRY_CNT	조회 시 데이터 건 수 지정

	<div> <div>"20" 한번에 20건씩만 조회</div> <pre> m_tranChegyul.setDataValue("InRec1", "ACNO", 0, strAccountNo); // 계좌번호 m_tranChegyul.setDataValue("InRec1", "AC_PWD", 0, strAccountPwd); // 계좌비번 m_tranChegyul.setDataValue("InRec1", "BUY_SEL_TR_TCD", 0, "%"); m_tranChegyul.setDataValue("InRec1", "ITEM_COD", 0, "%"); m_tranChegyul.setDataValue("InRec1", "SORT_TURN_I01CD", 0, "2"); m_tranChegyul.setDataValue("InRec1", "SCR_QRY_TCD", 0, "01"); m_tranChegyul.setDataHeader("DETAIL_CONT", "1"); // 연속조회 m_tranChegyul.setDataHeader("DETAIL_QRY_CNT", "10"); // 조회개수 m_tranChegyul.requestData(); </pre> <p>체결내역, 시세추이와 같은 출력이 array로 동작하는 서비스에 대해 한번에 몇건씩 페이지 단위로 조회할 경우 위의 예와 같이 연속조회와 조회개수를 지정하여 사용한다.</p> </div>
--	---

- 데이터

함수명	void clearInputData()	
설명	서비스 입력에 지정된 값을 클리어한다.	
리턴값	없음	
파라미터	없음	
내용	TRAN 서비스의 입력데이터를 클리어한다.	

함수명	void clearOutputData()	
설명	서비스 출력에 있는 데이터를 클리어한다.	
리턴값	없음	
파라미터	없음	
내용	TRAN 서비스의 출력데이터를 클리어한다.	

함수명	void setDataValue(String sBlockName, String sFieldName, int nRecIndex, String sData)	
설명	TRAN 서비스 블록의 데이터 필드에 값을 지정한다.	
리턴값	없음	
파라미터	String sBlockName	데이터 블록 이름
	String sFieldName	데이터 필드 이름
	int nRecIndex	데이터 인덱스
	String sData	지정할 데이터
내용	TRAN 서비스의 InBlock, OutBlock의 필드에 값을 지정한다. 블록이 single인 경우 nRecIndex는 항상 0이고, array형태인 경우 nRecIndex로 인덱스를 지정하여 데이터를 지정할 수 있다.	

	<pre>// 종목 시세 요청 String strItemCode = "000660"; // 조회할 종목 코드 (단축코드) String strFullCode = SmartChampAPI.getInstance().getItemFullCode(strItemCode); m_tranCurrPrice = new SmartChampTran(context); m_tranCurrPrice.initInstance(listener: this); m_tranCurrPrice.setQueryFile("stmst2"); m_tranCurrPrice.setTranId("주식현재가"); m_tranCurrPrice.clearInputData(); m_tranCurrPrice.clearOutputData(); // 입력 데이터 지정 m_tranCurrPrice.setDataValue(sBlockName: "InRec1", sFieldName: "sCode", nRecIndex: 0, strFullCode); // 서비스 요청 m_tranCurrPrice.requestData();</pre>	
--	--	--

함수명	String getDataValue(String sBlockName, String sFieldName, int nRecIndex)	
설명	TRAN 서비스 블록의 데이터 필드에 있는 값을 얻는다.	
리턴값	필드의 데이터	
파라미터	String sBlockName	데이터 블록 이름
	String sFieldName	데이터 필드 이름
	int nRecIndex	데이터 인덱스
내용	<p>TRAN 서비스의 InBlock, OutBlock의 필드의 값을 얻는다. 블록이 single인 경우 nRecIndex는 항상 0이고, array형태인 경우 nRecIndex는 인덱스를 지정하여 원하는 row의 데이터를 얻을 수 있다.</p> <pre>@Override public void onTranDataReceived(String sTranID, String sMsgCode, String sMsgText) { if (sTranID.equals("주식현재가")) { // 요청한 현재가 서비스에서 OutRec1이라는 블록에서 lCPrice(현재가) 필드의 값을 얻는다. String strPrice = m_tranCurrPrice.getDataValue("OutRec1", "lCPrice", 0); } }</pre>	

함수명	int getDataAttr(String sBlockName, String sFieldName, int nRecIndex)	
설명	TRAN 서비스의 블록의 데이터 속성값을 얻는다.	
리턴값	데이터 속성값	
파라미터	String sBlockName	데이터 블록 이름
	String sFieldName	데이터 필드 이름
	int nRecIndex	데이터 인덱스
내용	<p>데이터의 속성값은 모든 필드에 있는 것은 아니며, 필요한 필드에서만 사용된다. 시세 서비스의 현재가, 등락, 등락률 등 상승/하락에 대해 표현색상이 달라지는 경우, 이 속성값으로 이를 결정해서 표현할 수 있다.</p>	

TRAN 서비스의 InBlock, OutBlock의 필드에서 속성값을 얻는다.

블록이 single인 경우 nRecIndex는 항상 0이고, array형태인 경우 nRecIndex는 인덱스를 지정하여 원하는 row의 속성값을 얻을 수 있다.

속성에 대한 정의 SmartChampDefs의 DATA_ATTR_XXX항목으로 판단할 수 있다.

```
@Override
public void onTranDataReceived(String sTranID, String sMsgCode, String sMsgText)
{
    if (sTranID.equals("주식현재가"))
    {
        // 요청한 현재가 서비스에서 OutRec1이라는 블록에서 LCPrice(현재가) 필드의 값을 얻는다.
        String strPrice = m_tranCurrPrice.getDataValue("OutRec1", "LCPrice", 0);
        // 현재가 데이터의 속성값을 얻는다.
        int nPriceAttr = m_tranCurrPrice.getDataAttr("OutRec1", "LCPrice", 0);

        // 현재가 텍스트의 색상 결정
        int nPriceColor =
            nPriceAttr == SmartChampDefs.DATA_ATTR_RED ? Color.RED :
            nPriceAttr == SmartChampDefs.DATA_ATTR_BLUE ? Color.BLUE : Color.GRAY;
    }
}
```

● 요청

함수명	int requestData()
설명	TRAN 서비스 조회를 서버에 요청한다.
리턴값	요청 결과 (리턴이 < 0면 오류)
파라미터	없음
내용	<p>TRAN 서비스에 대해 조회를 서버에 요청한다.</p> <p>요청전에 setDataValue로 InBlock에 데이터를 설정하고 requestData()를 호출하는 순서로 동작한다.</p> <pre>// 종목 시세 요청 String strItemCode = "000660"; // 조회할 종목 코드 (단축코드) String strFullCode = SmartChampAPI.getInstance().getItemFullCode(strItemCode); m_tranCurrPrice = new SmartChampTran(context); m_tranCurrPrice.initInstance(listener: this); m_tranCurrPrice.setQueryFile("stmst2"); m_tranCurrPrice.setTranId("주식현재가"); m_tranCurrPrice.clearInputData(); m_tranCurrPrice.clearOutputData(); // 입력 데이터 지정 m_tranCurrPrice.setDataValue(sBlockName: "InRec1", sFieldName: "sCode", nRecIndex: 0, strFullCode); // 서비스 요청 m_tranCurrPrice.requestData();</pre>

함수명	int requestNextData()	
설명	TRAN 서비스 조회 후 다음 데이터가 존재하는 경우 다음 데이터를 요청한다.	
리턴값	요청 결과 (리턴이 < 0이면 오류)	
파라미터	없음	
내용	<p>TRAN 서비스에 대해 다음 데이터 조회를 서버에 요청한다.</p> <p>요청전에 다음 데이터가 있는지 isMoreNextData()를 호출하여 true인 경우에만 호출해야 한다.</p> <pre>// 체결 리스트 스크롤 마지막 -> 다음 데이터 조회 @Override public void onCheMicheScrollEnd() { if(m_tranChegyul == null) return; if(m_tranChegyul.isMoreNextData()) { m_tranChegyul.requestNextData(); } }</pre> <p>데이터의 연속조회는 조회된 array형태의 데이터를 listview형태의 뷰에 표시 한 후 스크롤이 마지막 항목에 다다랐을 때, 다음 데이터가 있는 체크하여 있으면 다음데이터를 조회하고 조회된 데이터를 listview의 마지막에 추가하여 표시하는 형태로 동작한다.</p>	

함수명	int getDataCount(String sBlockName)	
설명	TRAN 서비스의 블록에 들어있는 데이터 건수를 얻는다.	
리턴값	블록의 데이터 건수	
파라미터	String sBlockName	블록명
내용	<p>데이터 블록이 array인 경우, 블록 내에 몇건의 데이터가 있는지 얻는다.</p> <pre>else if(sTranID.equals("주식잔고조회")) { int nDataCount = m_tranJango.getDataCount("OutRec2"); for(int i = 0; i < nDataCount; i++) { JangoInfo info = new JangoInfo(); info.m_strItemCode = m_tranJango.getDataValue("OutRec2", "ITEM_COD", i); info.m_strItemName = m_tranJango.getDataValue("OutRec2", "ITEM_NM", i); info.m_strAvailQty = m_tranJango.getDataValue("OutRec2", "BNS_ABLE_Q", i); info.m_strHaveQty = m_tranJango.getDataValue("OutRec2", "BNS_BAL_Q", i); info.m_strUnitPrice = m_tranJango.getDataValue("OutRec2", "BUY_UPR", i); info.m_strAvgPrice = m_tranJango.getDataValue("OutRec2", "STK_EA", i); info.m_strProfit = m_tranJango.getDataValue("OutRec2", "EV_PL_A", i); m_viewJango.addList(info); } }</pre>	

- 처리 결과 메시지

함수명	String getDataMsgText()
설명	TRAN 서비스 조회 후 조회데이터와 함께 수신한 메시지 정보를 얻는다.

리턴값	메시지 텍스트	
파라미터	없음	
내용	메시지 정보는 ITranDataListener의 onTranDataReceived나 onTranError에서 파라미터로 전달되는 sMsgText값을 통해서 얻을 수 있으며, getDataMsgText를 통해서도 얻을 수 있다.	

함수명	String getDataMsgCode()	
설명	TRAN 서비스 조회 후 조회데이터와 함께 수신한 메시지 코드를 얻는다.	
리턴값	메시지 코드	
파라미터	없음	
내용	메시지 코드는 ITranDataListener의 onTranDataReceived나 onTranError에서 파라미터로 전달되는 sMsgCode값을 통해서 얻을 수 있으며, getDataMsgCode를 통해서도 얻을 수 있다.	

4.1.3 SmartChampReal

실시간 시세 정보를 받기 위한 클래스, 실시간 서비스 1개에 각 1개의 클래스를 선언하여 사용한다.

분류	반환	함수명	설명
초기화	void	initInstance	인스턴스 초기화
	void	clearInstance	인스턴스 해제
	void	setQueryFile	qry파일 로드
	void	setRealId	실시간 id 지정
	String	getRealId	실시간 id 반환
데이터	void	setDataValue	실시간 입력/출력 필드에 데이터 지정
	String	getDataValue	실시간 입력/출력 필드에 존재하는 데이터 리턴
	int	getDataAttr	실시간 출력 필드의 데이터 속성값 리턴(시세 색상표시용)
요청/해제	void	requestReal	실시간 시세 등록 요청
	void	releaseReal	실시간 시세 해제 요청
카운트	int	getDataCount	수신된 데이터 블록의 데이터 개수

- 초기화

함수명	void initInstance(IRealDataListener listener)
설명	REAL 서비스 호출을 위해 인스턴스 초기화
리턴값	없음

파라미터	IRealDataListener listener	실시간 데이터 요청/수신 이벤트 리스너
내용	<p>REAL 서비스 호출을 위한 객체 초기화</p> <p>initInstance(listener)를 호출하여 초기화한 후, setQueryFile(strRealCode)로 쿼리파일 로드를 수행한 후 실시간 요청 작업을 해야 한다.</p> <pre> // 멤버 변수 private SmartChampReal m_realCurrPrice = null; // 주식 체결 실시간 private void initCurrPriceReal(Context context) { m_realCurrPrice = new SmartChampReal(context); m_realCurrPrice.initInstance(new IRealDataListener() { @Override public void onRealBeforeRequest(String sRealId) { // 실시간 요청 전에 호출됨 // 여기서 setDataValue로 원하는 데이터를 지정할 수 있음 } @Override public void onRealDataReceived(String sRealId) { // 서버에서 실시간 데이터가 수신된 경우 // 여기서 수신된 데이터 처리 } }); m_realCurrPrice.setQueryFile("S00"); m_realCurrPrice.setRealId("주식체결실시간"); } </pre> <p>IRealDataListener에 sRealID는 별도로 지정하지 않으면 서비스명과 동일하게 들어온다. setRealId로 별도의 아이디를 지정하면 지정된 아이디로 리스너의 이벤트 함수가 호출된다.</p>	

함수명	void clearInstance()	
설명	REAL 서비스 요청을 위한 인스턴스를 해제한다.	
리턴값	없음	
파라미터	없음	
내용	REAL 서비스에 대한 인스턴스를 해제한다. 서비스 사용 후 에 더 이상 필요치 않으면 해제를 호출해야 한다.	

함수명	void setQueryFile(String strRealCode)	
설명	REAL 서비스 인스턴스에서 주어진 서비스의 정보를 파일에서 읽는다.	
리턴값	없음	
파라미터	String strRealCode	서비스명 (서비스가 정의된 파일명)
내용	초기화된 인스턴스에서 사용할 서비스의 정보를 로드한다. initInstance(listener)를 통해서 초기화한 경우에만 호출할 수 있다.	

함수명	void setRealId(String strRealId)	
설명	REAL 서비스명 외에 별도의 REAL서비스아이디 지정	
리턴값	없음	
파라미터	String strRealId	지정할 REAL서비스 아이디
내용	REAL 서비스 인스턴스의 RealId는 별도로 지정하지 않으면 setQueryFile로 지정한 서비스 이름이 기본값으로 지정된다. REAL서비스를 구분해야 할 필요가 있거나, 별도의 아이디를 지정하고자 할 경우, 이 함수로 서비스 아이디를 지정한다.	

함수명	String getRealId()	
설명	서비스 인스턴스의 서비스 아이디를 구한다.	
리턴값	REAL 아이디	
파라미터	없음	
내용	REAL서비스 인스턴스의 아이디가 필요한 경우 호출한다.	

- 데이터

함수명	void setDataValue(String sBlockName, String sFieldName, int nRecIndex, String sData)	
설명	REAL 서비스 블록의 데이터 필드에 값을 지정한다.	
리턴값	없음	
파라미터	String sBlockName	데이터 블록 이름
	String sFieldName	데이터 필드 이름
	int nRecIndex	데이터 인덱스
	String sData	지정할 데이터
내용	<p>REAL 서비스의 InBlock, OutBlock의 필드에 값을 지정한다.</p> <p>블록이 single인 경우 nRecIndex는 항상 0이고, array형태인 경우 nRecIndex로 인덱스를 지정하여 데이터를 지정할 수 있다.</p> <pre> private void requestCurrPriceReal() { m_realCurrPrice.releaseReal(); String strFullCode = SmartChampAPI.getInstance().getItemFullCode("000660"); m_realCurrPrice.setDataValue("InBlock1", "sCode", 0, strFullCode); m_realCurrPrice.requestReal(); } </pre>	

함수명	String getDataValue(String sBlockName, String sFieldName, int nRecIndex)	
설명	REAL 서비스 블록의 데이터 필드에 있는 값을 얻는다.	
리턴값	필드의 데이터	
파라미터	String sBlockName	데이터 블록 이름

	String sFieldName	데이터 필드 이름
	int nReclIndex	데이터 인덱스
내용	<p>REAL 서비스의 InBlock, OutBlock의 필드의 값을 얻는다. 블록이 single인 경우 nReclIndex는 항상 0이고, array형태인 경우 nReclIndex는 인덱스를 지정하여 원하는 row의 데이터를 얻을 수 있다.</p> <pre> @Override public void onRealDataReceived(String strRealId) { if(strRealId.equals("주식제결실시간")) { // 현재가 String strPrice = m_realCurrPrice.getDataValue("OutBlock1", "LCPrice", 0); } } </pre>	

함수명	int getDataAttr(String sBlockName, String sFieldName, int nReclIndex)	
설명	REAL 서비스의 블록의 데이터 속성값을 얻는다.	
리턴값	데이터 속성값	
파라미터	String sBlockName	데이터 블록 이름
	String sFieldName	데이터 필드 이름
	int nReclIndex	데이터 인덱스
내용	<p>데이터의 속성값은 모든 필드에 있는 것은 아니며, 필요한 필드에서만 사용된다. 시세 서비스의 현재가, 등락, 등락률 등 상승/하락에 대해 표현색상이 달라지는 경우, 이 속성값으로 이를 결정해서 표현할 수 있다.</p> <p>REAL 서비스의 InBlock, OutBlock의 필드에서 속성값을 얻는다. 블록이 single인 경우 nReclIndex는 항상 0이고, array형태인 경우 nReclIndex는 인덱스를 지정하여 원하는 row의 속성값을 얻을 수 있다.</p> <p>속성에 대한 정의 SmartChampDefs의 DATA_ATTR_XXX항목으로 판단할 수 있다.</p> <pre> @Override public void onRealDataReceived(String strRealId) { if(strRealId.equals("주식제결실시간")) { // 현재가 String strPrice = m_realCurrPrice.getDataValue("OutBlock1", "LCPrice", 0); // 현재가 속성값 int nPriceAttr = m_realCurrPrice.getDataAttr("OutBlock1", "LCPrice", 0); // 현재가 텍스트 색상 결정 int nPriceColor = nPriceAttr == SmartChampDefs.DATA_ATTR_RED ? Color.RED : nPriceAttr == SmartChampDefs.DATA_ATTR_BLUE ? Color.BLUE : Color.GRAY; } } </pre>	

- 요청

함수명	int requestReal()
-----	-------------------

설명	REAL 서비스 등록을 서버에 요청한다.	
리턴값	없음	
파라미터	없음	
내용	<p>REAL 서비스에 대해 서버에 등록을 요청한다.</p> <p>요청전에 setDataValue로 InBlock에 데이터를 설정하고 requestReal()를 호출하는 순서로 동작한다.</p> <pre> private void requestCurrPriceReal() { m_realCurrPrice.releaseReal(); String strFullCode = SmartChampAPI.getInstance().getItemFullCode("000660"); m_realCurrPrice.setDataValue("InBlock1", "sCode", 0, strFullCode); m_realCurrPrice.requestReal(); } </pre>	

함수명	int releaseReal()	
설명	REAL 서비스 등록을 해제하기 위해 서버에 요청한다.	
리턴값	없음	
파라미터	없음	
내용	<p>REAL 서비스에 대해 서버에 등록했던 요청을 해제한다.</p> <p>요청시세 사용 했던 setDataValue로 지정된 값이 변경되지 않았다면 releaseReal만 해제해도 된다.</p> <p>지정되어 있지 않다면 다음과 같이 등록할 때 같이 처리한다.</p> <pre> private void releaseCurrPriceReal() { String strFullCode = SmartChampAPI.getInstance().getItemFullCode("000660"); m_realCurrPrice.setDataValue("InBlock1", "sCode", 0, strFullCode); m_realCurrPrice.releaseReal(); } </pre>	

- 데이터 카운트

함수명	int getDataCount(String sBlockName)	
설명	REAL 서비스의 블록에 들어있는 데이터 건수를 얻는다.	
리턴값	블록의 데이터 건수	
파라미터	String sBlockName	블록명
내용	데이터 블록이 array인 경우, 블록 내에 몇건의 데이터가 있는지 얻는다.	

```

@Override
public void onRealDataReceived(String strRealId)
{
    if(strRealId.equals("주식제결실시간"))
    {
        int nDataCount = m_realCurrPrice.getDataCount("OutBlock1");

        for(int i = 0; i < nDataCount; i++) {

            String strTime = m_realCurrPrice.getDataValue("OutBlock1", "lTime", i);
            // 현재가
            String strPrice = m_realCurrPrice.getDataValue("OutBlock1", "lCPrice", i);
            int nPriceAttr = m_realCurrPrice.getDataAttr("OutBlock1", "lCPrice", i);
            // 등락
            String strDiff = m_realCurrPrice.getDataValue("OutBlock1", "lDiff", i);
            int nDiffAttr = m_realCurrPrice.getDataAttr("OutBlock1", "lDiff", i);
            ...
        }
    }
}

```

4.1.4 SmartChampDefs

SmartChamp Open API에 사용되는 상수 정의

- 로그인 유형

SmartChampAPI의 getLoginType()으로 얻는 값으로 현재 로그인 상태에 대한 값이다.

타입	정의	값	설명
int	LOGIN_TYPE_NONE	0	로그인 안됨
int	LOGIN_TYPE_AUTO	1	자동로그인 이전에 로그인한 경우, 다음 앱 실행시 저장된 아이디로 자동로그인 상태로 유지됨
int	LOGIN_TYPE_SISE	2	시세 조회용 로그인 공동인증서 없이 아이디/비번만으로 로그인한 경우
int	LOGIN_TYPE_CERT	3	거래용 로그인 공동인증서를 이용하여 로그인한 경우, 주문을 위해서는 거래용 로그인이 필수

- 화면 재조회 요청 사유

INetInitListener의 onRequiredRefresh로 화면 재조회 요청 시, 재조회가 일어난 이유를 함께 받는데 그 이유를 정의.

타입	정의	값	설명
int	REFRESH_REASON_NORMAL	0	앱이 비활성화되는 경우, 실시간 데이터를 수신 하지 않기 위해 서버에 등록 해제를 요청함. 앱이 다시 활성화되면 데이터를 최신으로 유지 하고 실시간 데이터를 다시 요청할 필요가 있 음. 이 사유로 onRequiredRefresh가 호출되면

			데이터 요청 및 실시간 등록을 수행해야 함
int	REFRESH_REASON_RECONNECT	1	앱 통신 단절 시 또는 장시간 비활성화 상태에 의한 단절 시 재접속이 발생함. 재접속이 완료 시 화면의 데이터 갱신 작업이 필요함. 처리 방식은 REFRESH_REASON_NORMAL 경우와 동일함
int	REFRESH_REASON_CHANGED_OSSISE	2	해외주식 실시간 서비스는 1개의 단말에서만 사용 가능함. 앱 사용중 실시간 사용 상태가 변경된 경우 호출됨. 현재 화면이 해외주식 실시간 화면인 경우 데이터 재조회 수행이 필요함

- 서버 접속 모드

SmartChampAPI의 setDevMode로 서버 접속 모드를 지정할 수 있다. 서버 접속 모드 지정 시 입력하는 값을 정의.

타입	정의	값	설명
String	SERVER_MODE_REAL	"0"	운영서버, 앱 시작 시 운영서버로 접속할 것인지, 개발/테스트 서버로 접속할 것인지 지정할 경우 사용되는 값
String	SERVER_MODE_DEV	"1"	개발/테스트용 서버

4.1.5 SmartChampDev

서비스 테스트를 위한 테스트 뷰 생성용 클래스, OpenAPI에서 호출할 수 있는 모든 서비스에 대해 입력값을 직접 입력하고 호출하여 OpenAPI사용하여 앱 작성시 참고할 수 있도록 한다.

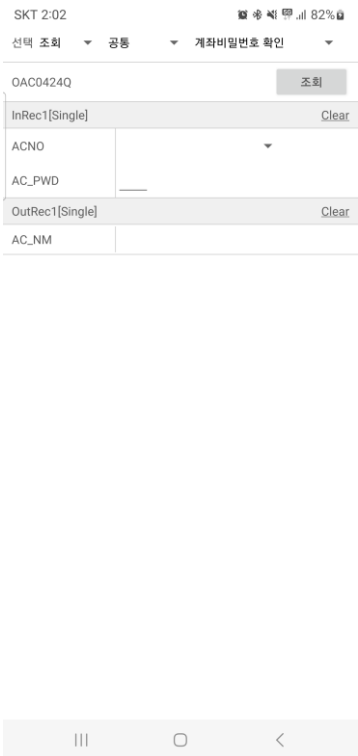
분류	반환	함수명	설명
초기화	void	initInstance	인스턴스 초기화
	void	releaseInstance	인스턴스 해제
뷰생성	ViewGroup	makeTestView	TR테스트를 위한 뷰 생성

- 초기화

함수명	void initInstance(Context context)		
설명	서비스 테스트 인스턴스를 생성한다.		
리턴값	없음		
파라미터	Context	뷰를 사용할 activity instance	
내용	서비스 테스트 뷰를 구성하기 위한 준비 작업		

함수명	void releaseInstance()
-----	------------------------

설명	서비스 테스트 인스턴스 해제	
리턴값	없음	
파라미터	없음	
내용	서비스 테스트를 사용후에 인스턴스 해제가 필요한 시점에 호출한다.	

함수명	ViewGroup makeTestView(Context context)	
설명	서비스 테스트를 위한 뷰를 생성한다.	
리턴값	테스트를 위한 뷰	
파라미터	Context context	view 생성을 위한 context
내용	<p>SmartChamp OpenAPI에서 사용할 수 있는 서비스를 api를 이용하여 프로그래밍 하지 않고, 제공된 테스트 화면에서 직접 입력값과 출력값을 확인 할 수 있는 기능을 제공한다.</p> <p>뷰의 구성은 아래와 같이 테스트할 서비스를 선택할 수 있는 상단 영역과 입력값을 입력하는 인블럭과 출력값을 표시하는 아웃블럭 영역으로 구성된다.</p> 	

4.2 인터페이스

4.2.1 INetInitListener

앱 실행 후 초기 접속, 버전처리, 종목정보 로드 등 초기 실행 관련한 상태를 리턴하는 리스너입니다.

반환	함수명	설명
void	onSessionConnecting	초기 접속 시작

void	onSessionConnected	접속 결과 리턴 (성공여부, 실패 메시지)
void	onAPIVersionState	"라이브러리 버전 체크 결과
void	onUpdateDownloadState	"버전처리 진행 상황
void	onMasterLoadState	마스터 로드 상태 (0 : 시작, 1 : 완료)
void	onInitFinished	초기화 완료

함수명	void onSessionConnecting()	
설명	SmartChampAPI의 startConnect() 호출시 접속이 시작됨을 알린다.	
리턴값	없음	
파라미터	없음	
내용	<p>서버 접속이 시작되었음을 알리는 이벤트</p> <pre> @Override public void onSessionConnecting() { //Toast.makeText(this, "서버 접속 시작.", Toast.LENGTH_SHORT).show(); if(m_viewIntro != null) m_viewIntro.setInfoText("서버에 접속합니다."); } </pre>	

함수명	void onSessionConnected(boolean isSuccess, String strErrMsg)	
설명	서버에 접속이 되거나, 접속이 실패한 경우 호출된다.	
리턴값	없음	
파라미터	boolean isSuccess	성공여부
	String strErrMsg	실패시 실패 내용
내용	<p>접속 작업이 완료된 후 접속에 대한 결과를 알리기 위해 호출된다. isSuccess가 true이면 자동으로 다음작업이 수행되고, false면 startConnect로 시작된 접속 처리가 중지된다.</p> <pre> @Override public void onSessionConnected(boolean isSuccess, String strErrMsg) { // 서버 성공 if(isSuccess == true) { if(m_viewIntro != null) m_viewIntro.setInfoText("서버 접속 성공"); } else// 서버 실패 { Toast.makeText(this, strErrMsg, Toast.LENGTH_SHORT).show(); if(m_viewIntro != null) m_viewIntro.setInfoText("서버 접속 실패"); } } </pre>	

함수명	void onAPIVersionState(int nCheckState, String strMessage)
-----	--

설명	API 버전체크에 대한 결과 알림.	
리턴값	없음	
파라미터	int nCheckState	버전체크 결과 < 0 : 버전체크 실패 (진행 중지됨) 0 : 정상 (다음 작업 자동 진행) 1 : 업그레이드 필요(다음 작업 자동 진행) 2 : 업그레이드 필수 상황 (진행 중지됨) 사용자에게 안내하고 앱 업데이트 유도
	String strMessage	체크 결과 메시지
내용	<p>API 버전에 대한 비교 작업이 이루어 지고, 허용 범위 내의 버전은 통과되고, 오류가 발생했거나 반드시 업데이트가 필요한 상황이면, 접속 프로세스가 종료되고 사용자에게 알려야 한다.</p> <pre> @Override public void onAPIVersionState(int nCheckState, String strMessage) { //Toast.makeText(this, "라이브러리 버전체크 완료.", Toast.LENGTH_SHORT).show(); if(nCheckState < 0) { // 버전체크 실패 (서비스 오류) (앱 중지) } else if(nCheckState == 0) { // 정상 } else if(nCheckState == 1) { // 버전 업그레이드 필요 (정상) } else if(nCheckState == 2) { // 버전 업그레이드 필수 (앱 중지) AlertDialog.Builder builder = new AlertDialog.Builder(this); builder.setTitle("안내"); builder.setMessage("유진 SmartChamp OpenAPI 업그레이드가 필요합니다.\n\n" + strMessage); builder.setNeutralButton("확인 ", new DialogInterface.OnClickListener() { @Override public void onClick(DialogInterface dialog, int which) { dialog.dismiss(); } }); builder.setOnDismissListener(new DialogInterface.OnDismissListener() { @Override public void onDismiss(DialogInterface dialog) { SampleActivity.this.terminateApp(); } }); builder.show(); } } </pre>	

함수명	void onUpdateDownloadState(int nState, int nIndex, int nCount, String sMessage)	
설명	OpenAPI에서 필요한 파일(종목정보, 쿼리정보)을 다운로드 하는 경우에 진행 상황을 아리기 위해 호출됨	
리턴값	없음	
파라미터	int nState	업데이트 진행 상태 0 - 업데이트 작업 중(체크,다운로드) 1 - 진행상황 알림

		2 – 완료됨
	int nIndex	다운로드 중인 파일의 순번
	int nCount	총 다운로드 파일 수
	String sMessage	1 – 진행상황 알림의 경우 내용
내용	<p>파일 다운로드에 대한 알림 제공</p> <pre> @Override public void onUpdateDownloadState(int nState, int nIndex, int nCount, String sMessage) { if(nState == 0) { // 진행 상태 // nIndex = 0, nCount = 0 : 준비작업 // nCount > 0 ? 다운로드 진행 } else if(nState == 1) { // 안내메시지 } else if(nState == 2) { // 작업 완료 } if(m_viewIntro != null) m_viewIntro.updateProgress(nState, nIndex, nCount, sMessage); } </pre>	

함수명	void onMasterLoadState(int nState)	
설명	OpenAPI에서 사용할 종목정보 로드 작업 완료를 알림	
리턴값	없음	
파라미터	int nState	의미없음
내용	<p>종목정보 파일을 로드하는 작업을 수행하고 작업이 완료되면 알리는 이벤트</p> <pre> @Override public void onMasterLoadState(int nState) { //Toast.makeText(this, "Master 파일 Loading...", Toast.LENGTH_SHORT).show(); if(m_viewIntro != null) m_viewIntro.setInfoText("필요한 파일 로드 완료"); } </pre>	

함수명	void onInitFinished()	
설명	초기 접속 작업이 완료되었음을 알림	
리턴값	없음	
파라미터	없음	
내용	<p>앱 시작 시 초기 접속 작업 요청하고 접속작업이 완료되고, OpenAPI를 사용하여 조회/실시간 요청이 가능한 상태가 된 것을 알리는 이벤트.</p>	

```

@Override
public void onInitFinished()
{
    // 초기 접속 및 OpenAPI 초기화가 완료됨

    // 로그인이나, 초기화면 등을 전환 필요

    // 메뉴 선택창
    showMenuView();
}

```

4.2.2 INetLoginListener

로그인 작업에 대한 시작과 결과 정보를 리턴하기 위한 리스너입니다.

반환	함수명	설명
void	onLoginStarted	로그인 작업 시작 알림
void	onLoginResult	로그인 결과 리턴 (성공여부, 실패 메시지)

함수명	void onLoginStarted()	
설명	로그인 요청 시 로그인 작업 시작을 알림	
리턴값	없음	
파라미터	없음	
내용	loginUser 함수 호출 시 로그인 작업 시작된 경우 알림 <pre> @Override public void onLoginStarted() { if(m_viewFrame != null) m_viewFrame.showProgress("로그인 중 입니다."); } </pre>	

함수명	void onLoginResult(boolean isSuccess, String strErrMsg)	
설명	로그인 작업 완료 후 결과 알림	
리턴값	없음	
파라미터	Boolean isSuccess	로그인 성공 여부
	String strErrMsg	로그인 오류 메시지
내용	주어진 login정보로 로그인 작업 수행하고 그 결과를 리턴	

	<pre>@Override public void onLoginResult(boolean isSuccess, String strErrorMsg) { if(m_viewFrame != null) m_viewFrame.hideProgress(); if(isSuccess) { Toast.makeText(getBaseContext(), "로그인 성공", Toast.LENGTH_SHORT).show(); // 로그인 완료 후 해야 할 작업 수행 // 로그인 성공한 아이디 저장 // 필요하면 다음 로그인 창 표시할 때 저장된 아이디 사용 SampleUtil.setConfig(this, "login.userid", SmartChampAPI.getInstance().getLoginUserID()); } else { Toast.makeText(getBaseContext(), strErrorMsg, Toast.LENGTH_SHORT).show(); // 로그인 실패 시 작업 } }</pre>	
--	---	--

4.2.3 INetLogoutListener

로그아웃 요청시 로그아웃 작업 완료를 알리기 위한 리스너.

반환	함수명	설명
void	onLogoutFinished	로그아웃 완료

함수명	void onLogoutFinished()	
설명	로그아웃 완료 알림	
리턴값	없음	
파라미터	없음	
내용	<p>logoutUser() 호출 시 로그아웃 작업이 완료되면 호출된다.</p> <pre>@Override public void onLogoutFinished() { Toast.makeText(getBaseContext(), "로그아웃 완료", Toast.LENGTH_SHORT).show(); // 로그아웃 후 해야할 작업 if(m_viewFrame != null) { m_viewFrame.onRefreshView(); m_viewFrame.showLoginDialog(); } }</pre>	

4.2.4 IAccountPwdCheckListener

계좌 비밀번호 체크에 대한 결과 리턴을 위한 리스너.

반환	함수명	설명
void	onAccountPwdCheckResult	계좌 비밀번호 체크 결과 리턴

함수명	void onAccountPwdCheckResult(boolean isSuccess, String strAccNo, String strMessage)	
설명	계좌번호와 계좌비번으로 검증 요청 한 후 받는 이벤트	
리턴값	없음	
파라미터	boolean isSuccess	비번 체크 성공 여부
	String strAccNo	검증요청한 계좌번호
	String strMessage	오류 메시지
내용	<p>계좌번호와 키패드에서 입력받은 계좌비번으로 checkAccPwd함수 호출 시 결과를 리턴받는다.</p> <pre> SmartChampAPI.getInstance().checkAccPwd(m_spinnerAccount.getAccountNo(), strPwd, new IAccountPwdCheckListener() { @Override public void onAccountPwdCheckResult(boolean isSuccess, String strAccNo, String strMessage) { if(isSuccess) { Toast.makeText(getContext(), "계좌 비밀번호 체크 완료", Toast.LENGTH_LONG).show(); requestJango(); requestChegyul(); } else { Toast.makeText(getContext(), strMessage, Toast.LENGTH_LONG).show(); m_editAccountPwd.setText(""); } } })); </pre>	

4.2.5 ITranDataListener

조회 서비스 호출 시 호출되기전 시점과 결과 수신 후 데이터 처리를 위한 리스너.

반환	함수명	설명
void	onTranBeforeRequest	데이터 요청 전 호출되는 이벤트, 호출전 필요한 작업을 여기서 수행
void	onTranDataReceived	데이터 수신 호출되는 이벤트, 결과 데이터를 처리 하는 작업을 여기서 수행
void	onTranError	호출한 서비스의 오류 발생 시 호출되는 이벤트
void	onTranTimeout	호출한 서비스가 정해진 시간안에 응답이 오지 않는 경우 발생하는 이벤트

함수명	void onTranBeforeRequest(String sTranID, boolean isNextQuery)	
설명	TRAN 서비스 호출 시 서버로 요청 데이터 전송되기 전에 호출됨	
리턴값	없음	

파라미터	String sTranID	TRAN 서비스 아이디
	boolean isNextQuery	연속조회 여부
내용	<p>TRAN 서비스를 호출하면 인블럭(입력데이터)에 설정된 값으로 요청정보를 생성하게 되는데, 데이터 생성 전에 호출된다.</p> <p>requestData 호출 전에 setDataValue로 입력 데이터 설정하거나, 이 이벤트 함수에서 설정하거나 상관없이 동작한다.</p> <pre> @Override public void onTranBeforeRequest(String sTranID, boolean isNextQuery) { // 계좌비번, 사용자비번 등은 서버에 요청 후 삭제됨 // 요청시마다 다시 설정 필요... if(sTranID.equals("주식체결내역")) { // 비번 조회 String strAccountPwd = m_editAccountPwd.getTextEnc(); m_tranChegyul.setDataValue("InRec1", "AC_PWD", 0, strAccountPwd); // 계좌비번 } } </pre>	

함수명	void onTranError(String sTranID, String sErrCode, String sErrMsg)	
설명	TRAN 서비스 요청 후 오류가 발생한 경우 호출됨	
리턴값	없음	
파라미터	String sTranID	TRAN 서비스 아이디
	String sErrCode	오류 메시지 코드
	String sErrMsg	오류 메시지 내용
내용	<p>TRAN 서비스 데이터를 서버에 요청한 후 오류가 발생한 경우 호출됨</p> <pre> @Override public void onTranError(String sTranID, String sErrCode, String sErrMsg) { //if(sTranID.equals("주식매수주문")) { Toast.makeText(getContext(), sErrMsg, Toast.LENGTH_SHORT).show(); } } </pre>	

함수명	void onTranTimeout(String sTranID)	
설명	TRAN 서비스 요청 후 결과 데이터가 정해진 시간안에 도착하지 않는 경우	
리턴값	없음	
파라미터	String sTranID	TRAN 서비스 아이디
내용	서비스 조회 요청에 응답이 없는 경우 호출됨	

4.2.6 IRealDataListener

실시간 서비스 등록 시 등록 요청 전 시점과 실시간 데이터 수신 시 데이터 처리를 위한 리스너.

반환	함수명	설명
----	-----	----

void	onRealBeforeRequest	실시간 등록 호출 시, 등록 요청 서버에 전송하기 전에 호출됨
void	onRealDataReceived	서버에서 실시간 데이터 도착 시 호출되는 이벤트

함수명	void onRealBeforeRequest(String sRealId)	
설명	실시간 서비스 등록 요청 시 서버에 요청 올리기 전에 호출됨	
리턴값	없음	
파라미터	String sRealId	실시간 서비스 아이디
내용	<p>실시간 서비스 등록 요청 시 서버에 요청이 올라가기 전에 데이터를 구성하는데 그 전에 호출됨. 여기서 원하는 종목을 설정하거나, requestReal호출하기 전에 설정하거나 상관없이 동작한다.</p> <pre> @Override public void onRealBeforeRequest(String sRealId) { // 실시간 데이터 등록 요청전 } </pre>	

함수명	void onRealDataReceived(String sRealId)	
설명	실시간 데이터 수신 시 호출됨	
리턴값	없음	
파라미터	String sRealId	실시간 서비스 아이디
내용	<p>실시간 데이터 수신 시 호출됨</p> <p>실시간 데이터는 array 형태의 데이터이며, 종목체결 같은 경우 한번에 1개 이상의 데이터가 들어올 수 있다. getDataCount로 OutBlock에 들어있는 데이터 개수를 얻은 후 필요한 경우 개수 만큼 처리가 필요하다.</p> <pre> @Override public void onRealDataReceived(String strRealId) { if(strRealId.equals("주식체결실시간")) { int nDataCount = m_realCurrPrice.getDataCount("OutBlock1"); for(int i = 0; i < nDataCount; i++) { String strTime = m_realCurrPrice.getDataValue("OutBlock1", "lTime", i); // 현재가 String strPrice = m_realCurrPrice.getDataValue("OutBlock1", "lCPrice", i); int nPriceAttr = m_realCurrPrice.getDataAttr("OutBlock1", "lCPrice", i); } } } </pre>	

4.2.7 ICertImportListener

공동인증서 가져오기 수행 시 가져오기를 위한 랜덤 번호 생성 정보와 가져오기 결과를 리턴 하기 위한 리스너.

반환	함수명	설명
void	onCertImportNumber	공동인증서 가져오기 단계에서 랜덤번호 생성에 대한 결과 리턴
void	onCertImportResult	공동인증서 가져오기 결과 리턴

함수명	void onCertImportNumber(boolean isSuccess, String strRandom, String strMessage)	
설명	인증서 가져오기를 위해 생성된 랜덤번호를 받는다.	
리턴값	없음	
파라미터	boolean isSuccess	성공여부
	String strRandom	생성된 랜덤번호
	String strMessage	실패 시 실패메시지
내용	<p>인증서 가져오기를 위해 requestCertImportNumber를 호출하면 인증서 관리 모듈에서 랜덤 번호를 생성하여 돌려준다.</p> <pre> @Override public void onCertImportNumber(boolean isSuccess, String strRandom, String strMessage) { if(isSuccess) { // 사용자에게 랜덤 번호를 보여주고, 보내는(export) 단말에서 입력할 수 있게 한다. // 글자씩 잘라서 보여준다. m_viewRandom.setText(strRandom.substring(0, 4) + "-" + strRandom.substring(4, 8) + "-" + strRandom.substring(8)); } else { // 실패 메시지 Toast.makeText(getContext(), strMessage, Toast.LENGTH_SHORT).show(); } } </pre>	

함수명	void onCertImportResult(boolean isSuccess, String strCertDN, String strMessage)	
설명	인증서 가져오기가 완료된 후 호출된다.	
리턴값	없음	
파라미터	boolean isSuccess	성공 여부
	String strCertDN	가져오기가 완료된 인증서 DN
	String strMessage	실패시 메시지
내용	<p>인증서 내보내기 하는 단말에서 랜덤 번호 입력 후 내보내기를 수행한 경우, importCert를 호출하여 단말에 저장할 수 있다. 내려받아 저장하기가 완료되면 이벤트가 호출된다. 수신된 공동인증서는 strCertDN 변수에 DN값이 들어있다.</p>	

```

@Override
public void onCertImportResult(boolean isSuccess, String strCertDN, String strMessage) {

    Toast.makeText(getContext(), strMessage, Toast.LENGTH_SHORT).show();

    if(isSuccess)
    {
        m_isSuccess = true;
        SampleUtil.setConfig(getContext(), "cert_dn", strCertDN);
        dismiss();
    }
}

```

4.2.8 ICertExportListener

공동인증서 내보내기 수행 시 내보내기에 대한 결과 리턴하기 위한 리스너.

반환	함수명	설명
void	onCertExportResult	공동인증서 내보내기 결과 리턴

함수명	void onCertExportResult(boolean isSuccess, String strMessage)	
설명	인증서 내보내기 결과 수신	
리턴값	없음	
파라미터	boolean isSuccess	성공 여부
	String strMessage	실패 시 메시지
내용	exportCert함수로 인증서 DN, 인증서 비번, 가져오기 하는 단말에서 생성된 랜덤번호를 입력하면 내보내기 작업이 시작되고 완료되면 이벤트가 호출된다.	

4.2.9 ICertDeleteListener

공동인증서 삭제 수행 시 삭제에 대한 결과 리턴하기 위한 리스너.

반환	함수명	설명
void	onCertDeleteResult	공동인증서 삭제 결과 리턴

함수명	void onCertDeleteResult(boolean isSuccess, String strMessage)	
설명	인증서 삭제 요청 후 완료 이벤트	
리턴값	없음	
파라미터	boolean isSuccess	성공여부
	String strMessage	실패 시 오류 메시지
내용	인증서 삭제에 대한 결과를 알려준다.	