xingAPI DLL 함수 Reference

함수

서버 연결

이름	설명
ETK_Connect	서버와 연결합니다.
ETK_IsConnected	서버와의 연결 여부를 취득합니다.
ETK_Disconnect	서버와의 연결을 종료합니다.

로그인

이름	설명
ETK_Login	서버에 로그인합니다.

조회TR

이름	설명
ETK_Request	TR을 서버에 요청합니다.
ETK_ReleaseRequestData	수신 데이터를 삭제하고 Request ID를 해제합니다.
ETK_ReleaseMessageData	수신 메시지를 삭제합니다.

실시간TR

이름	설명
ETK_AdviseRealData	실시간 TR을 등록합니다.
ETK_UnadviseRealData	등록된 실시간 TR을 해제합니다.
ETK_UnadviseWindow	윈도우에 등록된 모든 실시간 TR을 해제합니다.

계좌

이름	설명
ETK_GetAccountListCount	계좌의 개수를 취득합니다.
ETK_GetAccountList	계좌를 취득합니다.
ETK_GetAccountName	계좌명을 취득합니다.
ETK_GetAcctDetailName	계좌 상세명을 취득합니다.
ETK_GetAcctNickname	계좌 별명을 취득합니다.

정보

이름	설명
ETK_GetServerName	접속한 서버의 이름을 취득합니다.
ETK_GetLastError	마지막에 발생한 Error Code를 취득합니다.
ETK_GetErrorMessage	Error Code에 대한 메시지를 취득합니다.
ETK_GetTRCountPerSec	TR의 초당 전송 가능 횟수를 취득합니다.
ETK_GetTRCountBaseSec	TR의 Vase시간(초단위)를 취득합니다.
ETK_GetTRCountRequest	TR의 10분내 요청한 총 횟수를 취득합니다.
ETK_GetTRCountLimit	TR의 10분당 제한 건수를 취득합니다.

부가기능

이름	설명
ETK_RequestService	부가 서비스 TR을 서버에 요청합니다.
ETK_RemoveService	부가 서비스 TR을 해제합니다.
ETK_RequestLinkToHTS	API에서 HTS로의 연동을 원할 때, 요청합니다.
ETK_AdviseLinkFromHTS	HTS에서 API로의 연동을 등록합니다.
ETK_UnAdviseLinkFromHTS	HTS에서 API로의 연동을 해제합니다.
ETK_Decompress	t8411 TR 처럼 압축데이터 수신이 가능한 TR에 압축 해제용으로 사용합니다.

서버에 연결합니다.

```
BOOL __stdcall ETK_Connect (
    HWND        hWnd,
    const char* pszSvr,
    int         nPort,
    int         nStartMsgID,
    int         nTimeOut,
    int         nSendMaxPacketSize
)
```

매개변수

변수명	설명
hWnd	Window Handle
pszSvr	연결할 서버 IP
nPort	연결할 서버 Port
nStartMsgID	시작 MessageID
nTimeOut	연결시도 시간 - millisecond단위(1/1000초), -1 은 기본값(10초)으로 설정
nSendMaxPacketSize	전송시 최대 Packet Size, -1은 기본값으로 설정

반환값

0(FALSE) 이면 실패

1(TRUE) 이면 성공

실패시 ETK_GetLastError() 로 실패코드를 얻을 수 있습니다.

설명

hWnd 는 XM_DISCONNECT를 보낼 때 여기에 등록한 윈도우로 보내게 됩니다.

nPort 는 0으로 입력할 경우 자동으로 Port를 설정합니다.

nStartMsgID 는 Client 로 보낼 Message 의 시작 ID를 지정합니다.

Microsoft Windows는 사용자 Message를 사용할 수 있으며
0x400(=WM_USER) 이상의 Message를 사용할 수 있습니다.
그러므로 nStartMsgID는 0x400(=1024) 이상의 값이 들어와야 합니다.
Client 로 정의된 Message를 보낼 때는 nStartMsgID + 정의된 MessageID 로 보내게 됩니다.
예) XM_RECEIVE_DATA를 보낼 때는 nStartMsgID + XM_RECEIVE_DATA 로 보내게 됩니다.

nTimeOut 은 연결시도시간이며 이 시간이 초과할때까지 연결이 이루어지지 않으면 실패하게 됩니다.
기본값을 사용하되 인터넷이 느리거나 불안해서 Timeout 이 발생할 경우 사용하시기 바랍니다.

nSendPacketSize는 전송시 한번에 보낼 수 있는 최대 Data Size 입니다.
이 값을 설정하게 되면 설정한 크기 이상의 데이터를 전송할 경우 여러 번에 걸쳐서 전송하게 됩니다.
보통은 기본값을 사용하면 되나, 인터넷 공유기 등에서 최대 전송 Data Size를 제한하는 경우가 있으므로 전송 실패가 나는 경우에 최대 전송 Data Size 만큼 설정하시면 전송이 가능해질 수 있습니다.

```
BOOL bConnect = ETK_Connect( GetSafeHwnd(), "hts.ebestsec.co.kr", 20001, WM_USER, 10, -1 );
if( bConnect == FALSE )
{
    MessageBox( "서버연결에 실패하였습니다.", "연결실패", MB_ICONSTOP );
}
else
{
    MessageBox( "서버와 연결되었습니다.", "연결성공", MB_ICONINFORMATION );
}
```

서버와의 연결여부를 취득합니다.

```
BOOL __stdcall ETK_IsConnected ( )
```

매개변수

반환값

0(FALSE) 이면 연결중 아님

1(TRUE) 이면 연결중

설명

서버와의 연결을 종료합니다.

```
BOOL __stdcall ETK_Disconnect ( )
```

매개변수

반환값

무조건 1(TRUE)

설명

서버에 로그인합니다.

```
BOOL __stdcall ETK_Login (
    HWND        hWnd,
    const char* pszID,
    const char* pszPwd,
    const char* pszCertPwd,
    int         nType,
    BOOL        bShowCertErrDlg
)
```

매개변수

변수명	설명
hWnd	Window Handle
pszID	로그인 ID
pszPwd	로그인 ID에 대한 비밀번호
pszCertPwd	공인인증 비밀번호
nType	무조건 0
bShowCertErrDlg	공인인증 시 발생한 에러에 대해 미리 정의된 Dialog를 표시할 지 여부

반환값

0 이 아니면 성공(로그인 성공이 아니라 서버로 로그인요청 전송성공을 의미)

0 이면 실패

설명

해당 함수 호출 시 반환값으로 로그인 성공/실패를 받을 수 있는 것이 아니라
로그인 결과는 hWnd에 등록한 윈도우로 Message(XM_LOGIN)가 옵니다.

bShowCertErrDlg 는 미리 정의된 Dialog를 사용하지 않고 따로 정의한 Dialog를 사용하거나
혹은 미리 정의된 Dialog를 표시하지 않기를 원할 때 사용합니다.

Login 이 성공한 후에 프로그램 종료할 경우엔 ETK_Logout() 을 호출하여야 합니다.

서버로 조회TR을 전송합니다.

```
int __stdcall ETK_Request (
    HWND        hParentWnd,
    const char* pszTrCode,
    void*       lpData,
    int         nDataSize,
    BOOL        bNext,
    const char* pszContinueKey,
    int         nTimeOut
)
```

매개변수

변수명	설명
hParentWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszTrCode	요청할 TR Code
lpData	요청할 Data
nDataSize	Data 의 메모리 크기
bNext	연속조회 여부. 연속조회일 경우에 설정합니다
pszContinueKey	연속조회 키
nTimeOut	설정한 시간(초)내에 수신 데이터가 오지 않을 경우 XM_TIMEOUT 이 발생합니다.

반환값

0보다 작을 경우엔 실패이며

0 또는 0보다 클 경우엔 Request ID를 반환합니다.

설명

조회TR을 서버에 전송하기 위해 사용합니다.

조회TR에 대한 수신데이터는 XM_RECEIVE_DATA 메시지로 전송됩니다.

수신 데이터를 삭제하고 RequestID를 해제합니다.

```
void __stdcall ETK_ReleaseRequestData(
    int             nRequestID
)
```

매개변수

변수명	설명
nRequestID	해제할 Request ID

반환값

설명

Request ID와 수신 데이터의 메모리를 해제하기 위해서 사용합니다.

수신 메시지를 삭제합니다.

```
void __stdcall ETK_ReleaseMessageData(
    LPARAM      lp
)
```

매개변수

변수명	설명
lp	XM_RECEIVE_DATA로 받은 LPARAM 데이터 MSG_PACKET 의 Memory Pointer 입니다.

반환값

설명

MSG_PACKET의 메모리를 해제합니다.

MSG_PACKET이 System Error 이면 Request ID도 같이 해제합니다.

실시간 TR을 등록합니다.

```
BOOL __stdcall ETK_AdviseRealData(
    HWND        hWnd,
    const char* pszTrCode,
    const char* pszData,
    int         nDataUnitLen
)
```

매개변수

변수명	설명
hWnd	Window Handle XM_RECEIVE_REAL_DATA 메시지가 전송됩니다
pszTrCode	등록할 TR Code
pszData	등록할 데이터
nDataUnitLen	등록할 데이터의 Unit 크기

반환값

0(FALSE)이면 실패
1(TRUE)이면 성공

설명

실시간 데이터를 요청합니다.
수신데이터는 XM_RECEIVE_REAL_DATA 메시지로 전송됩니다.
ETK_UnadviseRealData() 를 요청하기 전까지 실시간 데이터가 수신됩니다.
한번 요청시에 여러 데이터를 요청할 수 있습니다.
예를 들면 078020 종목과 005930 종목을 요청할 경우 등록 데이터는 "078020005930" 이며 데이터 사이즈는 6 입니다.

※ 실시간 데이터는 TR에 Attribute가 설정되어 있어도, 전송시엔 Attribute를 적용하지 않으며 수신시에만 적용됩니다.

등록된 실시간 TR을 해제합니다.

```
BOOL __stdcall ETK_UnadviseRealData(
    HWND        hWnd,
    const char* pszTrCode,
    const char* pszData,
    int         nDataUnitLen
)
```

매개변수

변수명	설명
hWnd	Window Handle. 실시간이 등록된 윈도우
pszTrCode	등록 해제할 TR Code
pszData	등록 해제할 데이터
nDataUnitLen	등록 해제할 데이터의 Unit 크기

반환값

0(FALSE)이면 실패
1(TRUE)이면 성공

설명

등록된 실시간 데이터를 해제합니다.
한번 해제 시에 여러 데이터를 해제할 수 있습니다.
예를 들면 078020 종목과 005930 종목을 해제할 경우 입력데이터는 "078020005930" 이며 데이터 사이즈는 6 입니다.
등록할 때 A 종목과 B 종목을 한번에 등록하고 C 종목과 D 종목을 한번에 등록하였어도 해제할 때는 A, C 종목을 한번에 해제 가능하며 각각 해제도 가능합니다.

※ 실시간 데이터는 TR에 Attribute가 설정되어 있어도, 전송시엔 Attribute를 적용하지 않으며 수신시에만 적용됩니다.

윈도우에 등록된 모든 실시간 TR을 해제합니다.

```
BOOL __stdcall ETK_UnadviseWindow(
    HWND        hWnd
)
```

매개변수

변수명	설명
hWnd	Window Handle. 실시간이 등록된 윈도우

반환값

0(FALSE)이면 실패

1(TRUE)이면 성공

설명

윈도우에 등록된 실시간 데이터를 모두 해제합니다.

계좌리스트의 개수를 취득합니다.

```
int __stdcall ETK_GetAccountListCount()
```

매개변수

반환값

계좌 갯수

설명

계좌의 갯수를 취득합니다.

계좌를 취득합니다.

```
BOOL __stdcall ETK_GetAccountList(
    int         nIndex,
    char*       pszAcc,
    int         nAccSize
)
```

매개변수

변수명	설명
nIndex	받아올 계좌의 Index. 0 <= Index < ETK_GetAccountListCount().
pszAcc	계좌를 받을 Buffer. 최소 12 바이트는 할당되어 있어야 합니다.
nAccSize	Buffer의 메모리 크기

반환값

0(FALSE)이면 실패

1(TRUE)이면 성공

설명

계좌를 취득합니다.

Index는 0 부터 시작하며 ETK_GetAccountListCount() 보다 하나 적은 수까지 사용합니다.

계좌명을 취득합니다.

```
void __stdcall ETK_GetAccountName (
    LPCTSTR pszAcc,
    LPSTR     pszAccName,
    int          nAccNameSize
)
```

매개변수

변수명	설명
pszAcct	계좌번호
pszAccName	데이터를 받을 Buffer, 최소 41바이트가 할당되어 있어야 합니다
nAccNameSize	pszAccName의 사이즈

반환값

설명

계좌명을 취득합니다.

계좌 상세명을 취득합니다.

```
void __stdcall ETK_GetAcctDetailName (
    LPCTSTR pszAcc,
    LPSTR     pszAccName,
    int          nAccNameSize
)
```

매개변수

변수명	설명
pszAcct	계좌번호
pszAccName	데이터를 받을 Buffer, 최소 41바이트가 할당되어 있어야 합니다
nAccNameSize	pszAccName의 사이즈

반환값

설명

계좌 상세명을 취득합니다.

계좌 별명을 취득합니다.

```
void __stdcall ETK_GetAcctNickname (
    LPCTSTR pszAcc,
    LPSTR     pszAccName,
    int          nAccNameSize
)
```

매개변수

변수명	설명
pszAcct	계좌번호
pszAccName	데이터를 받을 Buffer, 최소 41바이트가 할당되어 있어야 합니다
nAccNameSize	pszAccName의 사이즈

반환값

설명

계좌 별명을 취득합니다.

한 서버의 이름을 취득합니다.

```
void __stdcall ETK_GetServerName(
    char*           pszName
)
```

매개변수

변수명	설명
pszName	데이터를 받을 Buffer, 최소 51바이트가 할당되어 있어야 합니다.

반환값

설명

접속한 서버의 서버명을 취득합니다.

마지막에 발생한 Error Code를 취득합니다.

```
int __stdcall ETK_GetLastError()
```

매개변수

반환값

Error Code

설명

Error Code에 대한 메시지를 취득합니다.

```
int __stdcall ETK_GetErrorMessage(
    int         nErrorCode,
    char*       pszMsg,
    int         nMsgSize
)
```

매개변수

변수명	설명
nErrorCode	Error Code
pszMsg	Message를 받을 Buffer, 충분한 메모리가 확보되어야 합니다
nMsgSize	Buffer 크기

반환값

Message 길이

설명

Error Code에 대한 Message를 취득합니다.

TR의 초당 전송 가능 횟수를 취득합니다.

```
int __stdcall ETK_GetTRCountPerSec (
    LPCTSTR      pszCode
)
```

매개변수

변수명	설명
pszCode	TR Code

반환값

TR의 초당 전송 가능 횟수

설명

TR의 초당 전송 가능 횟수를 취득합니다.

TR의 초당 전송 가능 횟수(Base)를 취득합니다.

```
int __stdcall ETK_GetTRCountBaseSec (
    LPCTSTR      pszCode
)
```

매개변수

변수명	설명
pszCode	TR Code

반환값

Base 시간(초단위)

설명

1초당 1건 전송 가능시 1, 5초당 1건 전송 가능시 5를 취득합니다.

TR의 초당 전송 가능 횟수를 취득합니다.

```
int __stdcall ETK_GetTRCountRequest (
    LPCTSTR      pszCode
)
```

매개변수

변수명	설명
pszCode	TR Code

반환값

10분내 요청한 해당 TR의 총 횟수

설명

10분내 요청한 해당 TR의 총 횟수를 취득합니다.

TR의 초당 전송 가능 횟수를 취득합니다.

```
int __stdcall ETK_GetTRCountLimit (
    LPCTSTR      pszCode
)
```

매개변수

변수명	설명
pszCode	TR Code

반환값

TR의 10분당 제한 건수.

설명

TR의 10분당 제한 건수를 취득합니다. 제한이 없는 경우 0 을 반환합니다.

부가 서비스용 TR을 서버에 요청합니다.

```
int __stdcall ETK_RequestService(
    HWND        hWnd,
    LPCTSTR     pszCode,
    LPCTSTR     pszData
)
```

매개변수

[1] 종목 검색

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"t1807" (HTS '[1807] 종목검색' 에서'API로 내보내기' 저장한 종목을 검색하는 TR)
pszData	'API로 내보내기' 한 파일의 전체 경로 지정

[2] 차트 지표데이터 조회

1) 차트 지표데이터 조회 (HTS '[4201]xing차트1'의 수식관리자 내 지표 기능 제공)

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"ChartIndex" (차트 지표데이터 조회용 TR)
pszData	"ChartIndex" TR내 Inblock의 데이터 구조체

2) 차트 엑셀데이터 조회 (HTS '[4201]xing차트1'의 수식관리자 내 지표 기능 제공)

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"ChartExcel" (차트 지표데이터 조회용 TR)
pszData	"ChartExcel" TR내 Inblock의 데이터 구조체

[3] 조건검색

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"t1857" (HTS '[1892] 조건검색' 에서'API로 내보내기' 혹은 '전략관리->서버저장' 한 종목을 검색하는 TR)
pszData	"t1857" TR내 InBlock의 데이터 구조체

반환값

ETK_Request() 함수와 동일합니다.
0보다 작을 경우엔 실패이며, 0 또는 0보다 클 경우엔 Request ID를 반환합니다.

설명

일반적인 Request방식이 아닌 별도의 처리가 필요한 경우, 부가 서비스용 TR을 통해 서버에 데이터를 요청할 때 사용합니다.

[1] 종목 검색

```
// HTS '[1807] 종목검색' 에서 'API로 내보내기' 저장한 파일이 "D:₩test.adf"
pszCode= "t1833", pszData= "D:₩test.adf"
int nReqID= ETK_RequestService("t1833, "D:₩test.adf")
```

[2] 차트 지표데이터 조회

차트 기초데이터를 이용하여 지표데이터를 제공합니다.

※ 지표데이터는 API내부에서 차트 기초데이터를 가공하여 제공하는 것으로, 조회 및 실시간에 다소 시간이 걸릴 수 있습니다.

1) 차트 지표데이터 조회 (HTS '[4201]xing차트1'의 수식관리자 내 지표 기능 제공)

```
// "MACD" 지표데이터 조회
ChartIndexInBlock sInBlock;
sInBlock.indexname = "MACD"
...생략..
ETK_RemoveService( hWnd, "ChartIndex", sOutBlock.indexed );
int nReqID = ETK_RequestService( hWnd, "ChartIndex", &sInBlock );
```

2) 차트 엑셀데이터 조회 (HTS '[4201]xing차트1'의 수식관리자 내 지표 기능 제공)
직접 저장한 차트 기초데이터를 엑셀 포맷으로 변경한 후, RequestService() 호출 시 지표데이터로 가공
("xingAPI 설치폴더/Excel/ChartExcelData.xls" 참고)

```
// 직접 쌓은 시고저종 데이터를 엑셀 포맷으로 변환하여 저장한 파일이 "D:₩test.xls"
ChartExcelInBlock sInBlock;
sInBlock.indexname      = "MACD"
sInBlock.excelfilename  = "D:₩test.xls"
...생략..
ETK_RemoveService( hWnd, " ChartExcel", sOutBlock.indexed );
int nReqID = ETK_RequestService( hWnd, "ChartExcel", &sInBlock );
```

[3] 조건검색

t1857을 이용하여 조회/실시간조건검색을 제공합니다.

```
t1857InBlock    pckInBlock;
TCHAR           szTrNo[]        = "t1857";
char            szNextKey[]     = "";
...생략..
SetPacketData( pckInBlock.sRealFlag, sizeof( pckInBlock.sRealFlag), str_Real,DATA_TYPE_STRING );
// 실시간 여부 1:등록 0:조회만
SetPacketData( pckInBlock.sSearchFlag, sizeof( pckInBlock.sSearchFlag),
str_Flag ,DATA_TYPE_STRING ); // 조회구분값 S:서버 F:파일
SetPacketData( pckInBlock.query_index, sizeof( pckInBlock.query_index ),
str_Index ,DATA_TYPE_STRING ); // 종목검색입력값
int nReqID = RequestService( hWnd, szTrNo, (LPCTSTR)&pckInBlock );
```

부가 서비스용 TR을 해제합니다.

```
int __stdcall ETK_RemoveService(
    HWND        hWnd,
    LPCTSTR     pszCode,
    LPCTSTR     pszData
)
```

매개변수

[1] 종목 검색

해당 없음

[2] 차트데이터 조회

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"ChartIndex" or "ChartExcel"
pszData	각 TR의 OutBlock의 indexed

[3] 조건검색

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_DATA 메시지로 옵니다.
pszCode	"t1857"
pszData	t1857 TR의 OutBlock의 AlertNum

반환값

[1] 종목 검색

해당 없음

[2] 차트데이터 조회

없음

[3] 조건검색

RequestID 반환

설명

부가서비스 이용 후, 해당 서비스 해제가 필요할 때 사용합니다.

[2] 차트데이터 조회

※ 지표데이터는 API내부에서 차트 기초데이터를 가공하여 제공하는 것으로, 많이 조회할수록 API에 부하가 갈 수 있으니, 사용하지 않는 지표는 해제하는 것이 좋습니다.

```
ChartIndexOutBlock sOutBlock;
ETK_RemoveService("ChartIndex", sOutBlock.indexid)
```

[3] 조건검색

※ 실시간 조건검색은 등록 갯수에 제한이 있기 때문에, 사용하시지 않는 경우 해제[하는 것이 좋습니다.

```
t1857OutBlock sOutBlock;
ETK_RemoveService("t1857", sOutBlock. AlertNum)
```

API에서 HTS로의 연동을 원할 때, 요청합니다.

```
int __stdcall ETK_RequestLinkToHTS(
    HWND      hWnd,
    LPCTSTR   pszLinkKey,
    LPCTSTR   pszData,
    LPCTSTR   pszFiller
)
```

매개변수

[1] 종목연동

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_LINK_DATA 메시지로 옵니다.
pszLinkKey	&STOCK_CODE : 주식 종목코드 &ETF_CODE : ETF 종목코드 &ELW_CODE : ELW 종목코드 &KONEX_CODE : 코넥스 종목코드 &FREEBOARD_CODE : 프리보드 종목코드 &KSPI_CODE : 코스피 업종 코드 &KSQI_CODE : 코스닥 업종 코드 &FUTURE_CODE : 선물종목코드 &OPTION_CODE : 옵션종목코드 &FUTOPT_CODE : 선물/옵션 종목코드 &FUTSP_CODE : 선물스프레드 종목코드 &STOCK_FUTURE_CODE : 주식 선물 종목코드 &STOCK_OPTION_CODE : 주식 옵션 종목코드 &STOCK_FUTOPT_CODE : 주식 선물옵션 종목코드 &STOCK_FUTSP_CODE : 주식 선물스프레드 종목코드 &FUTOPT_STOCK_FUTOPT_CODE : 선물옵션 & 주식 선물옵션 종목코드 &US_CODE : 해외종목코드 &COMMODITY_FUTOPT_CODE : 상품선물/선물옵션 &COMMODITY_FUTURE_CODE : 상품선물 &COMMODITY_STAR_CODE : 스타선물 &CME_FUTURE_CODE : CME야간선물 &EUREX_OPTION_CODE : EUREX야간옵션 &NIGHT_FUTOPT_CODE : 야간선물옵션
pszData	상품별 종목코드
pszFiller	사용 안 함

[2] HTS 화면열기

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_LINK_DATA 메시지로 옵니다.
pszLinkKey	&OPEN_SCREEN : 화면열기
pszData	HTS에서 열고자 원하는 화면번호
pszFiller	사용 안 함

반환값

TRUE면 연동이 성공이며, FALSE면 연동이 실패한 것입니다.

설명

API -> HTS로의 연동을 원할 때 요청하며, 한 번 요청하면 한 번 연동됩니다.
10번 연동해야 할 경우, 10번 요청합니다.

HTS에서 API로의 연동을 등록합니다.

```
void__stdcall ETK_AdviseLinkFromHTS(
    HWND       hWnd,
)
```

매개변수

변수명	설명
hWnd	Window Handle. 수신 데이터를 받을 윈도우입니다. 수신데이터는 XM_RECEIVE_LINK_DATA 메시지로 옵니다.

반환값

없음

설명

HTS -> API로의 종목 또는 계좌 등의 연동이 필요한 경우, 등록합니다.

연동을 등록한 시점부터 HTS상에서 연동정보가 발생할 때마다 데이터 수신이 가능합니다

HTS에서 API로의 연동을 해제합니다.

```
void__stdcall ETK_UnadviseLinkFromHTS()
```

매개변수

없음

반환값

없음

설명

HTS -> API로의 종목 또는 계좌 등의 연동을 해제합니다.

t8411 TR 처럼 압축데이터 수신이 가능한 TR에 압축 해제용으로 사용합니다.

```
int __stdcall ETK_Decompress(
    LPCTSTR     pszSrc,
    LPCTSTR     pszDest,
    int         nSrcLen
)
```

매개변수

변수명	설명
pszSrc	압축상태 데이터
pszDest	압축을 해제한 데이터를 저장할 메모리 (Outblock 구조체 사이즈 최대 2000건)
nSrcLen	pszSrc 데이터의 길이

반환값

압축을 해제한 데이터(pszDest)의 길이

설명

t8411 TR 처럼 압축데이터 수신이 가능한 TR에 압축 해제용으로 사용합니다.

```
// t8411 TR 이용시, InBlock의 comp_yn(압축여부) 필드에 "Y" 입력 후 조회
// ReceiveData() 에서 Occurs 블럭(t8411OutBlock1)이 압축되어 수신되므로, 해당 블럭 압축을 해
// 제

LRESULT t8411_Wnd::OnXMReceiveData( WPARAM wParam, LPARAM lParam )
{
    // Data를 받음
    if( wParam == REQUEST_DATA )
    {
        LPRECV_PACKET pRpData = (LPRECV_PACKET)lParam;
        if( strcmp( pRpData->szBlockName, NAME_t8411OutBlock ) == 0 ) { }
       else if( strcmp( pRpData->szBlockName, NAME_t8411OutBlock1 ) == 0 )
       {
            LPt8411OutBlock1 pOutBlock1 = (LPt8411OutBlock1)pRpData->lpData;
```

```
            t8411OutBlock1 szOutBlock1[2000];    // 압축 해제시 최대 2000건 수신
            int nDestSize = g_iXingAPI.Decompress((char *)pOutBlock1, (char *)&szOutBlock1[0],
pRpData->nDataLength);

            // Occurs 일 경우
            // Header가 'A' 이면 전체길이에서 OutBlock의 길이를 나눠서 갯수를 구한다.
            if (nDestSize > 0)
            {
                int nCount = nDestSize / sizeof( t8411OutBlock1 );
                for( int i=0; i<nCount; i++ )
                {
                    // 데이터 표시
                }
            }
        }
    }
}
```

메시지

메시지는 Window User Message로 제공하게 됩니다.

ETK_Connect() 를 호출할때 4번째 인자로 넣은 StartMsgID 와 MessageID가 합해져서 Message가 됩니다.

이름	설명
XM_DISCONNECT	서버와의 연결이 끊어졌을때 호출
XM_RECEIVE_DATA	조회 TR에 대한 응답을 받았을 때 호출
XM_RECEIVE_REAL_DATA	실시간 TR에 대한 응답을 받았을 때 호출
XM_RECEIVE_REAL_DATA_CHART	차트 지표 실시간 데이터에 대한 응답을 받았을 때 호출 (차트 지표데이터 조회 시, 실시간 자동 등록을 "1"로 했을 경우)
XM_RECEIVE_REAL_DATA_SEARCH	종목검색(신버전API용) 데이터에 대한 응답을 받았을 때 호출 (종목검색(신버전API용) 조회 시, 실시간 구분을 "1"로 했을 경우)
XM_RECEIVE_LINK_DATA	HTS -> API로 연동을 등록하면, HTS에서 연동 정보가 발생시에 호출
XM_LOGIN	Login 과정이 완료되었을 때 호출
XM_TIMEOUT	조회 TR에 대한 응답이 Timeout 되었을 때 호출

XM_DISCONNECT

서버와의 연결이 끊어졌을 때 호출

Message ID : 1

Message Window

ETK_Connect() 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

사용안함

LPARAM

사용안함

설명

ETK_Disconnect() 함수 호출할 때 발생하는 메시지가 아니며, 서버에서 강제로 연결을 끊는다는 통지의 의미도 아닙니다.

XM_DISCONNECT 메시지는 OS의 소켓 끊김 이벤트 발생 시, 해당 이벤트를 그대로 받아 전달합니다.
그러나, 소켓이 끊기면 OS는 소켓 끊김을 자동으로 아는 것이 아니라 소켓에 어떤 action이 발생했을 때 (TR을 조회하는 등) 연결상태를 알게 되므로 **XM_DISCONNECT** 메시지가 바로 발생하지 않을 수도 있습니다.
따라서, 24시간 연결을 해야만 하는 경우에는 유의하여야 합니다.

XM_RECEIVE_DATA

조회 TR에 대한 응답을 받았을 때 호출

Message ID : 3

Message Window

ETK_Request() 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

정수이며 각 값에 따라 처리방식이 다릅니다.

WPARAM	내용	설명
1	Data	- 요청한 Data를 전송 - Data가 없거나 에러가 발생한 경우 수신되지 않을 수 있음
2	Message	- Data 처리에 대한 Message - Data 처리에 성공을 해도 실패를 해도 Message는 발생한다. - Message Data를 처리하고 더 이상 사용하지 않을 경우 ETK_ReleaseMessageData()를 호출하여 Message를 해제하여야 한다.
3	System Error	- Data 처리 이전에 System 문제로 인한 Error가 발생할 경우 - Message Data를 처리하고 더 이상 사용하지 않을 경우 ETK_ReleaseMessageData()를 호출하여 Message를 해제하여야 한다. - System Error이 발생할 경우 Release가 전송되지 않으며 ETK_ReleaseMessageData()에서 자동으로 Request ID를 해제해준다.
4	Release	- Data 처리가 완료된 경우에 전송 - Request ID를 해제하기 위해 ETK_ReleaseRequestData()를 호출하여야 한다.

LPARAM

WPARAM의 값에 따라 다릅니다.

WPARAM	DATA FORMAT	내용	설명
1	RECV_PACKET 의 Memory 주소	Data	TR의 Data를 받았을 때 발생
2	MSG_PACKET 의 Memory 주소	Message	Message를 받았을 때 발생
3	MSG_PACKET 의 Memory 주소	System Error	Error가 발생
4	정수로 Request ID를 의미	Release	TR이 끝났을 때 발생

설명

XM_RECEIVE_REAL_DATA

실시간 TR에 대한 응답을 받았을 때 호출

Message ID : 4

Message Window

ETK_AdviseRealData () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

사용안함

LPARAM

REAL_RECV_PACKET의 메모리 주소

설명

XM_RECEIVE_REAL_DATA_CHART

차트 지표 실시간 데이터에 대한 응답을 받았을 때 호출
(차트 지표데이터 조회 시, 실시간 자동 등록을 "1"로 했을 경우)

Message ID : 10

Message Window

ETK_RequestService () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

지표데이터("ChartIndex") TR 조회 요청 시, 조회 결과 데이터의 **ChartIndexOutBlock의 indexed(고유키)**

LPARAM

REAL_RECV_PACKET 의 메모리 주소
지표데이터("ChartIndex") TR 조회 요청 시, 조회 결과 데이터의 **ChartIndexOutBlock1**

설명

```
// "MACD" 지표데이터 조회
ChartIndexInBlock sInBlock;
sInBlock.indexname = "MACD"   // 지표명
sInBlock. IsReal = "1"              // 실시간 데이터 자동 등록 여부 (0:조회만, 1:실시간 자동 등록)
          ...생략..
ETK_RemoveService( hWnd, "ChartIndex", sOutBlock.indexed );
int nReqID = ETK_RequestService( hWnd, "ChartIndex", &sInBlock );
// 차트 지표데이터 실시간 결과를 수신
LRESULT CChartIndexDlg::OnXMReceiveRealData( WPARAM wParam, LPARAM lParam )
{
    // 조회시 OutBlock의 indexid(indexid가 같으면 같은 지표이다 - 지표구분 key로 판단가능)
    UINT nIndexID = (UINT)wParam;
    // 지표 실시간 데이터는 RECV_REAL_PACKET의 pszData내에 OutBlock1 의 포맷으로 수신
    LPRECV_REAL_PACKET pRcvData = (LPRECV_REAL_PACKET)lParam;
    ChartIndexOutBlock1 *pBlock = (ChartIndexOutBlock1 *)(pRcvData->pszData);
                .. 지표 실시간 데이터 처리 ..
    return 0L;
}
```

XM_RECEIVE_REAL_DATA_SEARCH

종목검색(신버전API용) 실시간 데이터에 대한 응답을 받았을 때 호출
(종목검색(신버전API용)조회 시, 실시간 구분을 "1"로 했을 경우)

Message ID : 11

Message Window

ETK_RequestService () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

사용안함

LPARAM

REAL_RECV_PACKET 의 메모리 주소
종목검색(신버전API용)("t1857") TR 조회 요청 시, 조회 결과 데이터의 **t1857OutBlock1**

설명

```
// 종목검색(신버전API용) 조회
t1857InBlock    sInBlock;
sInBlock.query_index = "XXXXXXXX"   // 종목검색입력값
sInBlock.sRealFlag = "1"            // 실시간 데이터 자동 등록 여부 (0:조회만, 1:실시간 등록)
sInBlock.sRealFlag = "S"            // 조회구분값(S:서버 F:파일)

        ...생략..
ETK_RemoveService( hWnd, "t1857", sOutBlock.AlertNum );
int nReqID = ETK_RequestService( hWnd, "t1857", &sInBlock );
// 종목검색 (신버전API용) 실시간 결과를 수신
LRESULT CChartIndexDlg::OnXMReceiveRealData( WPARAM wParam, LPARAM lParam )
{
    // 실시간 데이터는 RECV_REAL_PACKET의 pszData내에 OutBlock1 의 포맷으로 수신
    LPRECV_REAL_PACKET pRcvData = (LPRECV_REAL_PACKET)lParam;
    t1857OutBlock1 *pBlock = (t1857OutBlock1*)(pRcvData->pszData);
            .. 실시간 데이터 처리 ..
    return 0L;
}
```

XM_RECEIVE_LINK_DATA

HTS -> API로 연동을 등록하면, HTS에서 연동 정보가 발생시에 호출
사용방식은 XM_RECEIVE_REAL_DATA 수신과 동일

Message ID : 8

Message Window

ETK_AdviseLinkData () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

LINK_DATA

LPARAM

LINKDATRRA_RECV_MSG 구조체 데이터

설명

XM_LOGIN

ETK_Login() 함수가 호출 된 후 Login 과정이 완료되었을 때 호출

Message ID : 5

Message Window

ETK_Login () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

Message Code
문자열 형태
"0000" 이면 성공, 그 외에는 실패

LPARAM

Message Text
문자열 형태

설명

XM_TIMEOUT

조회 TR에 대한 응답이 Timeout 되었을 때 호출

Message ID : 7

Message Window

ETK_Request () 함수를 호출하였을때 입력한 Window로 Message 전송

WPARAM

사용안함

LPARAM

Request ID

설명

구조체

이름	설명
RECV_PACKET	서버로부터 조회 데이터를 받았을 때 해당 Structure로 데이터가 들어온다.
RECV_REAL_PACKET	서버로부터 실시간 데이터를 받았을 때 해당 Structure로 데이터가 들어온다.
MSG_PACKET	서버로부터 데이터를 받았을 때 해당 Structure로 데이터가 들어온다.
LINKDATA_RECV_MSG	HTS-> API로 연동데이터를 받았을 때 해당 Structure로 데이터가 들어온다.

RECV_PACKET

서버로부터 조회 데이터를 받았을 때 해당 Structure로 데이터가 들어온다.

필드	Data형	크기	설명
RequestID	정수	4	ETK_Request() 함수를 호출했을 때 Return 값
Data Length	정수	4	Data 의 메모리 크기
Total Data Buffer Size	정수	4	Data 의 메모리에 할당된 크기
Elapsed Time	정수	4	데이터 요청에서부터 데이터를 수신했을때까지 걸린 시간 (1/1000초 단위)
DataMode	정수	4	1 : Block Mode 2 : Non Block Mode
TR Code	문자열	11	요청한 TR Code (마지막 1 Byte는 NULL)
Continue	문자열	1	'0','N' : 연속조회 없음 '1','Y' : 연속조회 있음
Continue Key	문자열	19	연속조회가 있을 경우 Data Header가 B 일 경우 이 값이 다음 조회 시 Key 가 된다. (마지막 1 Byte는 NULL)
User Data	문자열	31	사용안함
Block Name	문자열	17	Block 명, Block Mode 일 경우에 의미 있다. (마지막 1 Byte는 NULL)
Data	문자열 포인터	4	수신된 TR 데이터

RECV_REAL_PACKET

서버로부터 실시간 데이터를 받았을 때 해당 Structure로 데이터가 들어온다.

필드	Data형	크기	설명
TR Code	문자열	4	TR Code (마지막 1 Byte는 NULL)
Key Length	정수	4	
Key	문자열	33	(마지막 1 Byte는 NULL)
RegKey	문자열	33	(마지막 1 Byte는 NULL)
Data Length	정수	4	
Data	문자열 포인터	4	실시간 데이터

MSG_PACKET

서버로부터 메시지를 받았을 때 해당 Structure로 데이터가 들어온다.

필드	Data형	크기	설명
Request ID	정수	4	Request ID
System Error	정수	4	0 : Message 1 : System Error
Message Code	문자열	6	Message Code (마지막 1 Byte는 NULL)
Message Data Length	정수	4	Message Data 길이
Message Data	문자열 포인터	4	Message Data

LINKDATA_RECV_MSG

HTS-> API로 연동데이터를 받았을 때 해당 Structure로 데이터가 들어온다.

필드	Data형	크기	설명
LinkName	문자열	32	연동 키 ex) 주식종목코드 연동 시, &STOCK_CODE
LinkData	문자열	32	연동 값 ex) 주식종목코드 연동 시, 종목코드
Filler	문자열	64	사용 안함

감사합니다

이베스트투자증권