

# OpenAPI FAQ (주요 문의 사항)

- ▶ 주문 TR호출 후 **정상접수여부**는 반환되는 **OutBlock**의 **주문코드(6자리)**로 판단 할 수 있습니다.  
6자리란에 유효한 숫자가 반환되면 정상. OutBlock이 반환되지 않거나 주문코드에 무효한 값이 들어있다면 실패.
- ▶ CA\_RECEIVEMESSAGE 메시지로 반환되는 자료 중 **msg\_cd필드**는 당사 기간시스템에서 사용되는 값으로 오류 발생시 원인 분석용으로 사용하며 언제든지 변경될 수 있습니다. 따라서, 사용자가 이를 기반으로 프로그램을 작성할 경우 오동작할 가능성이 있으므로 직접 사용하지 않도록 주의하시기 바랍니다. 대신 같이 반환되는 ‘문자열’ 형태 메시지를 참고하시기 바랍니다.
- ▶ 별도 테스트 환경은 제공되지 않으니 주의하시기 바랍니다. (모의 투자 서비스는 2023년 5월 31일 서비스 중단됨)
- ▶ 일부 TR을 제외하면 당사에서 제공하는 대부분의 TR의 I/O 구조체에는 '**속성바이트**'라고 부르는 매우 중요한 1byte 추가 필드가 모든 필드마다 덧붙입니다. 이 속성바이트를 감안하지 않고 프로그램을 작성할 경우 TR호출에 실패하게됩니다.  
(구조체를 이용하지 않고 문자열을 덧붙이는 식으로 InBlock을 채워넣을 경우 빈번하게 발생함)  
  
‘속성’이란 주로 색상을 의미하며 등락폭 필드를 예로 들면 전일종가 대비 하락할 경우 파란색을 상승할 경우 붉은 색을 변동이 없을 경우 녹색을 나타내는 용도로 사용됩니다.(주문이나 계좌조회시 내려오는 값은 이와 또 다른 체계가 사용되고 있습니다)
- 이 속성바이트는 HTS용으로 설계되어 있어 OpenAPI에서는 사용을 권장하지 않습니다.  
하지만 TR을 호출할 때는 반드시 이 속성 바이트에 값을 채워 넣어주어야하며 일반적으로는 간단히 공백문자(0x20)을 채워주면 됩니다.  
  
예제에서 보이는 구조체에는 속성바이트가 모든 필드에 포함되어있으므로 사용자가 굳이 크게 의식하지 않아도 자동으로 입력이 됩니다.  
입력값(InBlock)을 문자열 형태로 덧붙여 넣을 경우 속성 바이트를 빠뜨리는 경우 잦으며 TR호출이 실패하거나 엉뚱하게 동작하게되는 원인이 됩니다.  
  
예를 들어 입력 필드가 한영 구분 1 byte와 종목코드 6 byte로 구성되어있는 현재가(c1101) TR을 호출할 경우 아래 첫 번째 예시처럼 공백문자없이 입력값만 연속으로 넣게 되면 TR호출이 실패하게됩니다. 두 번째 예시처럼 각 입력값 뒤에 1바이트씩 공백문자를 추가로 덧붙여서 호출하여야합니다.

# OpenAPI FAQ (주요 문의 사항)

[k000660]	(x)	속성값없이 입력값만 연속으로 넣었음
[k 000660 ]	(o)	속성 바이트가 1바이트씩 정상 입력되었음

## ▶ 출력 자료가 배열(반복) 형태로 수신되는 경우

예제에서 볼 수 있듯이 배열 형태로 결과(OutBlock)가 수신되는 경우 그 배열의 개수는 가변적일 수 있습니다. 즉, 한 건도 수신되지 않을 수도 있고 1건 또는 20건, 1000건이 수신될 수도 있습니다.

이 때 자료 건수를 확인하는 방법은 비교적 단순합니다. 수신된 자료의 총 크기(byte)를 해당 레코드(구조체) 크기로 나누면 됩니다. 예를 들어 레코드 크기가 20 byte인 OutBlock이 있고 수신된 자료의 크기가 1000 byte라면 50건(=1000/20)이 수신된 것입니다. (자세한 내용은 예제 참조)

## ▶ 모든 TR은 30초 이내에 송수신이 이루어져야합니다.

30초를 초과할 경우 TR호출이 실패하게됩니다. 대부분의 조회는 1~2초 이내에 이루어지므로 거의 문제되지 않지만 주문/체결조회와 같은 일부 TR은 자료량이 많을 경우 30초를 초과할 수도 있습니다. 이 경우 조회 결과를 미쳐 모두 수신하기도 전에 30초 제한으로 인해 조회가 강제 취소됩니다. (물론 이런 상황은 주문량이 매우 많은 경우에만 해당됩니다)

특정 TR이 과다하게 시스템 리소스를 사용할 경우 시스템 전체가 느려지거나 현상하지 않도록 하기 위함입니다.

조회할 자료가 많아 30초 이내에 모두 수신할 수 없는 경우라면 일정 개수만큼 나누어서 조회하는 대안을 생각해볼 수 있습니다.

## ▶ 짧은 시간내 과도한 TR호출은 언제든지 제한될 수 있습니다.

당사 서버는 TR종류와 상관없이 초당 약 100건의 요청이 올라올 경우 제한하는 정책을 갖고 있습니다. 동일 서버를 이용하는 다른 사용자에게 영향이 없도록 하기 위함이며 기본적인 경고 메시지 이후 통신 연결이 임의 단절될 수 있습니다.

## ▶ 계좌 번호 인덱스를 요구하는 Query() 함수 호출시 인덱스 번호는 숫자형 '1'부터 시작되며 '0'은 계좌번호를 사용하지 않는다는 의미입니다. 계좌번호 인덱스란 로그인시 수신한 계좌번호의 순서를 의미합니다.

Query()함수를 이용할 경우 계좌번호 인덱스를 요구하는 함수와 그렇지 않은 함수의 명확한 구분이 필요합니다.

# OpenAPI FAQ (주요 문의 사항)

(이 부분에 대한 사용자 실수가 빈번하므로 주의하시기 바랍니다)

주문, 잔고조회등 계좌번호를 요구하는 곳에는 계좌번호를 지정해야하는 반면 현재가 조회 등의 투자정보성 조회에서는 계좌번호를 지정하지 않아야하며 잘못 지정할 경우에는 TR호출이 실패하게됩니다.

```
wmcaQuery(..., 주문TRCODE,..., 계좌번호인덱스) //주문TR이므로 계좌번호를 지정
wmcaQuery(..., 잔고TRCODE,..., 계좌번호인덱스) //잔고조회TR이므로 계좌번호를 지정
wmcaQuery(..., 현재가조회TRCODE) //계좌와 관련없는 현재가 조회이므로 계좌번호를 지정하지 않음
```

▶ 주문TR 등에서 요구하는 입력 비밀번호는 1)계좌비밀번호, 2)거래비밀번호 두 가지 종류가 있습니다. 비밀번호가 틀리다는 거부 발생할 경우 각각의 비밀번호가 정확하게 입력되어 있는지 확인하시기 바랍니다.

다른 종류의 비밀번호로는 ID비밀번호와 공인인증 비밀번호가 있으나 접속시에만 사용하므로 혼동하는 경우가 드물지만 TR 호출시 입력하는 계좌비밀번호와 거래비밀번호는 혼동하는 경우가 종종있습니다. 특히, 당사에만 존재하는 '거래비밀번호'라는 체계에 익숙하지 않은 고객께서는 주의가 필요합니다.

당사 HTS와 달리 OpenAPI는 계좌비밀번호와 거래비밀번호 모두를 요구합니다. (경우에 따라서는 두 가지 중 한 가지만 요청) '거래비밀번호1'과 '거래비밀번호2'에는 거래비밀번호를 동일하게 입력하시기 바랍니다.

계좌비밀번호 지정시 **SetAccountIndexPwd()**함수를 거래비밀번호 지정시 **SetOrderPwd()**함수를 사용하여야하며 만약 평문을 직접 입력할 경우 비밀번호가 틀렸거나 입력 구조가 맞지 않는다는 거부 메시지를 보게됩니다. (예제참조)

▶ 입력값(InBlock)은 반드시 공백문자(0x20)로 초기화하여 주시기 바랍니다.

공백문자로 초기화할 경우 입력값에 영동한 값이 포함되어 프로그램이 오동작하는 현상을 피할 수 있습니다. 한편, null(0x00)문자로 초기화할 경우 서버에서 정상 처리가 안되는 경우가 간혹 있으므로 주의가 필요합니다.

▶ 수량/단가와 같은 숫자형 입력 필드에는 반드시 규정된 형식으로 자료가 입력되어야합니다.

# OpenAPI FAQ (주요 문의 사항)

예를 들어 8바이트 크기의 '수량' 필드에 숫자 '120'을 입력하는 경우를 예를 들면 아래와 같습니다.

[00000120] (o) 빈자리는 반드시 '0'으로 채워져있어야함

[120       ] (x) 공백이 포함되어있으므로 허용되지 않음

[       120] (x) 공백이 포함되어있으므로 허용되지 않음

[120 0000] (x) 공백이 포함되어있으므로 허용되지 않음

[120.0000] (x) 수량필드는 소수점이나 쉼표, 공백문자등 숫자 이외의 문자를 허용하지 않음

이 규격에 맞지 않을 경우 거부 메시지가 반환됩니다.

## “서비스 입력데이터 전문변환 오류(전문 -> 구조체)”

▶ `wmcaQuery()` 함수 호출시 사용자가 지정한 `nTRID`값은 `TR` 식별자로 사용할 수 있습니다.

(결과 값 수신시 `TrIndex`라는 이름으로 반환)

`OnWmReceivedata`, `OnWmReceivemessage`, `OnWmReceivecomplete` 이벤트 핸들러에서 사용자가 입력한 `nTRID`값이 `TrIndex`로 전달되며 이를 이용하여 `TR`을 구분할 수 있습니다.

예를 들어 서로 다른 3개 종목에 대한 현재가 조회 `TR(c1101)`을 동시에 요청할 경우 조회시 사용자가 지정한 `nTRID(ex: 1,2,3)`값을 각기 달리 지정하면 `TR`조회 결과 이벤트 핸들러에서 `TrIndex(ex: 1,2,3)`가 각기 달리 전달되어 개별 요청을 식별할 수 있습니다. 이를 통해 서로 다른 `TR`뿐만 아니라 동일 `TR`도 동시 다수 호출이 가능합니다.

짧은 시간내 동일한 `TR`을 1,2,3 순서대로 호출할 경우 조회 결과가 반드시 1,2,3 순으로 반환되는 것이 아니므로 이 `TrIndex`는 이 들 호출결과를 순서와 상관없이 식별할 수 있는 `key`값으로 활용할 수 있습니다.

▶ 당사 기반 시스템은 AnsiCode(확장 ASCII) 기반으로 모든 자료를 주고 받도록 설계 되어 있습니다. 따라서, 배포되는 자료 역시 `AnsiCode` 기반으로 자료를 주고 받도록 제공되며 `Unicode` 기반으로는 제공되지 않습니다.

# OpenAPI FAQ (주요 문의 사항)

사용자가 AnsiCode코드 자료를 Unicode로 변환 사용하는 것은 제한하지 않으나 당사가 그 신뢰성이나 안정성을 보증하지는 않습니다.

AnsiCode에서는 ‘한글’은 2byte로 구성되어있으나 문자열 뒤 쪽 1byte가 잘려서 자료가 수신될 경우 이를 Unicode로 변환하면 의도치 않게 변환이 되어 프로그램 오동작을 일으킬 수 있습니다. ‘속성’바이트(1byte) 역시 0x80 이상의 값이 올 경우 UniCode로 변환하게되면 같은 오동작을 일으키게 됩니다. 따라서, Ansi-UniCode 변환에 대해서 정확한 이해를 갖춘 상태에서 Unicode 변환을 사용하시기를 바랍니다.

- ▶ 배포되는 OpenAPI(wmca) 라이브러리 및 예제 코드는 Windows 32bit 프로그램 기반으로만 제공됩니다. 64bit 프로그램에서는 사용될 수 없으며 향후 제공될 계획 역시 없습니다. (64bit 프로그램에서 사용하시고자 할 경우 중계 역할을 하는 32bit host 프로그램을 작성하는 방안을 권장드리며 이를 위해서는 소켓 프로그래밍에 대한 일정 수준이상의 기술력이 요구됩니다)