X-Stack (H.323)

- API -

API 함수.
헤더파일 : H323UI.h

호 객체 관련 API

> 함수 원형
ICall *HapiCallOpen();
: 호 객체에 할당하고 Call Handle을 할당한 후 객체를 되돌림.

> Return
할당된 호 객체 반환, 실패 시 NULL 반환.

> Parameter
없음.

> Remark
: 단지 호 객체를 할당할 뿐 Signaling을 시작하는 함수는 아니다. 이 함수로 할당된 호에, 호 상세 내역을 설정하여 Command API를 이용하여 호 Signaling을 시도한다.
(HapiCommandMakeCall 함수 참조.)

> 함수 원형
HS_RESULT HapiCallClose(ICall *pCall);
: 호 객체를 메모리 해제한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall 메모리 해제할 호 객체 포인터.

> Remark
: 단지 호 객체를 메모리 해제할 뿐 Singlaing과는 관련이 없는 함수이다. HapiCallOpen으로 호 객체를 만든 후 어떠한 이유로 이 객체 메모리를 해제하고 싶을 경우 사용한다. Signaling이 시작된 호에 대해서는 Programer 가 직접 호 객체에 접근하는 것은 금지되며 HapiCallOpen시 할당된 Call Handle 만을 이용하여 호를 Control 한다는 점에 주의.

Alias 설정 관련 API

> 함수 원형

HS_RESULT HapiAddEndpointAliasDial(IEndpoint *pEndpoint, char *pAlias);

: E.164 Dial Alias를 단말에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint Alias를 설정할 단말 객체의 포인터.

pAlias 설정할 Alias String의 포인터.

> Remark

없음.

> 함수 원형

HS_RESULT HapiAddEndpointAliasH323Id(IEndpoint *pEndpoint, char *pAlias);

: H323 ID Alias를 단말에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint Alias를 설정할 단말의 객체 포인터.

pAlias 설정할 Alias String의 포인터.

> Remark

없음.

> 함수 원형

HS_RESULT HapiAddEndpointAliasH323IdEx(
IEndpoint *pEndpoint,
HS_USHORT *pAlias,
HS_UINT pLen);

: H323 ID Alias를 단말에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint Alias를 설정할 단말의 객체 포인터.
pAlias 설정할 Alias String의 포인터.
pLen 설정할 Alias String의 길이.

> Remark

: H323 ID Alias는 실제로 2바이트의 String으로 ASN Encoding 된다. 따라서 '표시가능한' (VisibleString)의 경우 HapiAddEndpointAliasH323Id 함수를 쓰면 되며, 실제 2바이트의 코드 값을 갖는 ID를 설정하고 싶을 경우 이 함수를 사용한다. 이때 pLen 값은 Byte 길이가 아니고 실제 2Byte String의 길이 값임.

> 함수 원형

HS_RESULT HapiAddEndpointAliasUrlId(IEndpoint *pEndpoint, char *pAlias);

: URI 형의 Alias를 단말에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint Alias를 설정할 단말의 객체 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark

없음.

> 함수 원형
HS_RESULT HapiAddEndpointAliasEmailId(IEndpoint *pEndpoint, char *pAlias);
: E-mail 형의 Alias를 단말에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pEndpoint Alias를 설정할 단말의 객체 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddCallingAliasDial(ICall *pCall, char *pAlias);
: E.164 Dial Alias를 호 발신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
: 호 시도 시에 등록 할때 와는 다른 발신자 Alias List를 가지고 호를 시도해야 할 때가 있다.
이때 호 발신자 Alias List를 새로 구성하고자 할 때, 이하 API를 사용한다.

> 함수 원형

HS_RESULT HapiAddCallingAliasH323Id(ICall *pCall, char *pAlias);

: H.323 ID Alias를 호 발신자에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pCall Alias를 설정할 호 객체의 포인터.

pAlias 설정할 Alias String의 포인터.

> Remark

없음.

> 함수 원형

HS_RESULT HapiAddCallingAliasH323IdEx(ICall *pCall, HS_USHORT *pAlias, HS_UINT pLen);

: H323 ID Alias를 호 발신자에 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pCall Alias를 설정할 단말의 객체 포인터.

pAlias 설정할 Alias String의 포인터.

pLen 설정할 Alias String의 길이.

> Remark

: H323 ID Alias는 실제로 2바이트의 String으로 ASN Encoding 된다. 따라서 '표시가능한' (VisibleString)의 경우 HapiAddCallingAliasH323Id 함수를 쓰면 되며, 실제 2바이트의 코드 값을 갖는 ID를 설정하고 싶을 경우 이 함수를 사용한다. 이때 pLen 값은 Byte 길이가 아니고 실제 2Byte String의 길이 값임.

> 함수 원형
HS_RESULT HapiAddCallingAliasUrlId(ICall *pCall, char *pAlias);
: URI 형태의 Alias를 호 발신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddCallingAliasEmailId(ICall *pCall, char *pAlias);
: E-mail 형태의 Alias를 호 발신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddCalledAliasDial(ICall *pCall, char *pAlias);
: E.164 Dial Alias를 호 수신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
: 호 시도 시에 등록 할때 와는 다른 발신자 Alias List를 가지고 호를 시도해야 할 때가 있다.
이때 호 수신자 Alias List를 새로 구성하고자 할 때, 이하 API를 사용한다.

> 함수 원형
HS_RESULT HapiAddCalledAliasH323Id(ICall *pCall, char *pAlias);
: H.323 ID 형태의 Alias를 호 수신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddCalledAliasH323IdEx(ICall *pCall, HS_USHORT *pAlias, HS_UINT pLen);
: H.323 ID 형태의 Alias를 호 수신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.
pLen 설정할 Alias String의 길이.

> Remark
: H323 ID Alias는 실제로 2바이트의 String으로 ASN Encoding 된다. 따라서 '표시가능한' (VisibleString)의 경우 HapiAddCalledAliasH323Id 함수를 쓰면 되며, 실제 2바이트의 코드 값을 갖는 ID를 설정하고 싶을 경우 이 함수를 사용한다. 이때 pLen 값은 Byte 길이가 아니고 실제 2Byte String의 길이 값임.

> 함수 원형
HS_RESULT HapiAddCalledAliasUrlId(ICall *pCall, char *pAlias);
: URI 형태의 Alias를 호 수신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddCalledAliasEmailId(ICall *pCall, char *pAlias);
: E-mail 형태의 Alias를 호 수신자에 설정한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Alias를 설정할 호 객체의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
NoLockList *HapiMakeAliasList();
: 새로운 Alias List를 구성하기 위한 NoLockList 객체를 생성.

> Return
만들어진 NoLockList 객체 포인터 반환, 실패시 NULL 반환.

> Parameter
없음.

> Remark
: 단말에 Alias List는 하나씩 변경할 수 없으며, Alias List를 새로 구성하여 Command API를 이용하여 바꿔야 한다. 이 때 사용될 새로운 Alias List를 구성할 때, 이 API를 이용한다. 변경 시 사용될 Command API 함수는 HapiCommandChangeEndpointAliases를 참조.

> 함수 원형
HS_RESULT HapiAddAliasDial(NoLockList *pList, char *pAlias);
: E.164 Dial Alias를 Alias List에 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pList Alias를 더할 NoLockList의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddAliasH323Id(NoLockList *pList, char *pAlias);
: H.323 ID Alias를 Alias List에 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pList Alias를 더할 NoLockList의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddAliasH323IdEx(NoLockList *pList, HS_USHORT *pAlias, HS_UINT pLen);
: H.323 ID Alias를 Alias List에 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pList Alias를 더할 NoLockList의 포인터.
pAlias 설정할 Alias String의 포인터.
pLen 설정할 Alias String의 길이.

> Remark
: H323 ID Alias는 실제로 2바이트의 String으로 ASN Encoding 된다. 따라서 '표시가능한' (VisibleString)의 경우 HapiAddAliasH323Id 함수를 쓰면 되며, 실제 2바이트의 코드 값을 갖는 ID를 설정하고 싶을 경우 이 함수를 사용한다. 이때 pLen 값은 Byte 길이가 아니고 실제 2Byte String의 길이 값임.

> 함수 원형
HS_RESULT HapiAddAliasUrlId(NoLockList *pList, char *pAlias);
: URI 형태의 Alias를 Alias List에 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pList Alias를 더할 NoLockList의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddAliasEmailId(NoLockList *pList, char *pAlias);
: E-mail 형태의 Alias를 Alias List에 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pList Alias를 더할 NoLockList의 포인터.
pAlias 설정할 Alias String의 포인터.

> Remark
없음.

Capability Setting 관련 API

> 함수 원형
HS_RESULT HapiAddG711Alaw64k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.711 A-Law 64K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
: 스텍에서는 운용의 융통성을 위해 각 호마다 Capability를 다르게 설정할 수 있다.
pSimul의 Group 번호는 그 값에는 큰 의미가 없으며 같은 Group으로 묶을 Capability 값을 같은 Group 번호로 설정하면 된다. Simultaneous Group은 보통 Audio, Video, Data 로 각각 묶어주면 된다. (H.245 Annex B.2 항 참고)
발신 호에 대해서는 HapiCommandMakeCall 전에 ICall 객체에 Capability를 설정해주면 되며, 수신 호에 대해서는 IEndpoint에 CallbackCallIncoming callback 함수에서 설정해 준다. 이 callback 함수의 두 번째 parameter가 ICall 객체의 포인터이며, 이는 Callback 함수로써 Stack 내부 Thread(Task) 이므로 이때는 Programer가 직접 ICall 객체에 접근해도 된다.

> 함수 원형
HS_RESULT HapiAddG711Alaw56k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.711 A-Law 56K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG711Ulaw64k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.711 U-Law 64K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG711Ulaw56k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.711 U-Law 56K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG72264k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.722 64K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG72256k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.722 56K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG72248k(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.722 48K Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.711의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG7231(ICall *pCall, HS_UINT pSimul, HS_UCHAR pMaxAl, BOOL pSilence);
: 호에 G.723.1 Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pMaxAl G.723.1 Max Al 값.
pSilence G.723.1 Silence Suppression 여부를 나타낼 파라미터.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG728(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.728 Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.728의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG729(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.729 Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.729의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG729AnnexA(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.729 AnnexA Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.729 AnnexA 의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG729wAnnexB(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.729 AnnexB Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.729 AnnexB 의 INTEGER 값.

> Remark
없음.

> 함수 원형
HS_RESULT HapiAddG729AnnexAwAnnexB(ICall *pCall, HS_UINT pSimul, HS_UCHAR pValue);
: 호에 G.729 AnnexAwAnnexB Capability를 더한다.

> Return
성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter
pCall Capability를 더할 호의 객체 포인터.
pSimul Simultaneous Group 번호.
pValue G.729 AnnexAwAnnexB의 INTEGER 값.

> Remark
없음.

TSAP Address 설정 관련 API

> 함수 원형

HS_RESULT HapiSetRasAddress(IEndpoint *pEndpoint, HS_UCHAR *pIp, HS_USHORT pPort);

: 메시지에 들어갈 Local RAS TSAP Address 값을 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint 설정할 단말 객체의 포인터.

pIp TSAP IP 값 unsigned char 4 byte 형의 unsigned char 포인터.

pPort TSAP Port Address 값.

> Remark

: 때로는 실제 Local IP 주소와는 다른 TSAP Address를 사용해야 할 경우 이하의 API를 사용한다.

>함수 원형

```
HS_RESULT HapiSetCallSignalingAddress(
    IEndpoint *pEndpoint,
    HS_UCHAR *pIp,
    HS_USHORT pPort );
```

: 메시지에 들어갈 Local Call Si후aling Address 값을 설정한다.

> Return

성공시 HS_OK 반환, 실패시 에러코드 반환.

> Parameter

pEndpoint 설정할 단말 객체의 포인터.

pIp TSAP IP 값 unsigned char 4 byte 형의 unsigned char 포인터.

pPort TSAP Port Address 값.

> Remark

: 때로는 실제 Local IP 주소와는 다른 TSAP Address를 사용해야 할 경우 이하의 API를 사용한다.

Logical Channel 설정 관련 API

> 함수 원형

HS_RESULT HapiAddOpenLogicalChannel(IEndpoint *pEndpoint, ICall *pCall, ICapability *pCapa);

: Programer가 직접 Logical Channel을 결정하고자 할 때 결정된 Capability를 스텍에 설정한다.

> Return

성공 시 HS_OK 반환, 실패 시 에러코드 반환.

> Parameter

pEndpoint 설정할 단말 객체 포인터.

pCall 설정할 호 객체 포인터.

pCapa Programer가 결정한 LogicalChannel의 Capability.

> Remark

: IEndpoint 구조체에 있는 Callback 함수 중 CallbackCheckTerminalCapabilitySet를 NULL 로 남겨 두면 호 객체에 설정했던 Capability List에 의해 Stack이 자동 협상하며, 이 협상과정을 Programer가 직접 하고자할 때는 이 Callback 함수를 등록해 준 후 이 함수에서 협상한 후 위 함수를 이용하여 직접 LogicalChannel을 더해주면 된다. Callback 함수는 Stack의 Thread(Task) 이므로 이때는 관련 객체들에 Programer가 직접 접근할 수 있다.

(OEHPhone의 CallbackCheckTerminalCapabilitySet callback 함수 부분 참조)

> 함수 원형

HS_RESULT HapiAddOpenLogicalChannelEx(IEndpoint *pEndpoint, ICall *pCall, HS_UINT pIndex);

: Programer가 직접 Logical Channel을 결정하고자 할 때 결정된 Capability를 스텍에 설정한다.

> Return

성공 시 HS_OK 반환, 실패 시 에러코드 반환.

> Parameter

pEndpoint 설정할 단말 객체 포인터.

pCall 설정할 호 객체 포인터.

pIndex Programer가 결정한 LogicalChannel의 Index (나의 Capability List)

> Remark

: IEndpoint 구조체에 있는 Callback 함수 중 CallbackCheckTerminalCapabilitySet를 NULL 로 남겨 두면 호 객체에 설정했던 Capability List에 의해 Stack이 자동 협상하며, 이 협상과정을 Programer가 직접 하고자할 때는 이 Callback 함수를 등록해 준 후 이 함수에서 협상한 후 위 함수를 이용하여 직접 LogicalChannel을 더해주면 된다. Callback 함수는 Stack의

Thread(Task) 이므로 이때는 관련 객체들에 Programer가 직접 접근할 수 있다.
(OEHPhone의 CallbackCheckTerminalCapabilitySet callback 함수 부분 참조)

> 함수 원형
HS_RESULT HapiAddForwardFastStart(ICall *pCall, ASNH245OpenLogicalChannel *olc);
: Programer가 직접 FastStart의 Logical Channel을 결정하고자 할 때 결정된 H245OpenLogicalChannel을 스텍에 설정한다.

> Return
성공 시 HS_OK 반환, 실패 시 에러코드 반환.

> Parameter
pEndpoint 설정할 단말 객체 포인터.
pCall 설정할 호 객체 포인터.
pIndex Programer가 결정한 LogicalChannel의 Index (나의 Capability List)

> Remark
: IEndpoint 구조체에 있는 Callback 함수 중 CallbackCheckFastStart를 NULL 로 남겨두면 호 객체에 설정했던 Capability List에 의해 Stack이 FastStart를 자동 협상하며, 이 협상과정을 Programer가 직접 하고자할 때는 이 Callback 함수를 등록해 준 후 이 함수에서 협상한 후 위 함수를 이용하여 직접 H245OpenLogicalChannel을 더해주면 된다. Callback 함수는 Stack의 Thread(Task) 이므로 이때는 관련 객체들에 Programer가 직접 접근할 수 있다.
(OEHPhone의 CallbackCheckFastStart callback 함수 부분 참조)

> 함수 원형
HS_RESULT HapiAddReverseFastStart(ICall *pCall, ASNH245OpenLogicalChannel *olc);
: Programer가 직접 FastStart의 Logical Channel을 결정하고자 할 때 결정된 H245OpenLogicalChannel을 스텍에 설정한다.

> Return
성공 시 HS_OK 반환, 실패 시 에러코드 반환.

> Parameter
pEndpoint 설정할 단말 객체 포인터.
pCall 설정할 호 객체 포인터.
pIndex Programer가 결정한 LogicalChannel의 Index (나의 Capability List)

> Remark
: IEndpoint 구조체에 있는 Callback 함수 중 CallbackCheckFastStart를 NULL 로 남겨두면 호 객체에 설정했던 Capability List에 의해 Stack이 FastStart를 자동 협상하며, 이 협상과정을 Programer가 직접 하고자할 때는 이 Callback 함수를 등록해 준 후 이 함수에서 협상한 후 위 함수를 이용하여 직접 H245OpenLogicalChannel을 더해주면 된다. Callback 함수는 Stack의 Thread(Task) 이므로 이때는 관련 객체들에 Programer가 직접 접근할 수 있다.
(OEHPhone의 CallbackCheckFastStart callback 함수 부분 참조)

상태 확인 관련 API

> 함수 원형
BOOL HapiStateIsRegisted(HS_STACK_HANDLE pHandle);
: 현재 등록 상태를 읽는다.

> Return
등록된 상태일 경우 TRUE, 미등록 상태일 경우 FALSE를 반환.

> Parameter
pHandle Stack Handle 값. (최초 HapiStartStack 함수에 의해 얻어진 Handle)

> Remark
없음.

> 함수 원형
HS_RESULT HapiCommandMakeCall(HS_STACK_HANDLE pHandle, ICall *pCall);
: 호 Signaling을 시작한다.

> Retrun
성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 호 Signaling에 관한 결과값이 아님.
  호 Signaling에 관한 과정을 IEndpoint 구조체의 callback 함수에 의해 알려진다.)

> Parameter
pHandle Stack handle.
pCall Signaling을 시도할 호 객체 포인터.

> Remark
없음.

> 함수 원형

HS_RESULT HapiCommandChangeGatekeeper(HS_STACK_HANDLE pHandle, char *pTarget);

: Gatekeeper로의 등록/등록해지/재등록을 Stack에 명령한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.

(Command 전송 성공 여부에 대한 결과이며 등록 여부에 관한 결과값이 아님)

> Parameter

pHandle Stack Handle.

pTarget 등록할 Gatekeeper Targer. (IP String, DNS Format 등 가능)

> Remark

: 등록 해지를 하고자 할 때는 pTarget 에 NULL String ("")을 입력하고, Multicast 등록을 원할 때는 pTarget에 “@”를 넘기면 된다. 이미 등록된 상태에서 위 API로 등록 시도를 하면 기존 등록을 해지하고 재등록 시도한다.

> 함수 원형

HS_RESULT HapiCommandAcceptCall(
  HS_STACK_HANDLE hStack,
  HS_CALL_HANDLE pHandle);

: 수신 호를 받아들인다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.

(Command 전송 성공 여부에 대한 결과이며 호 접수 성공 여부에 관한 결과값이 아님)

> Parameter

hStack Stack Handle.

pHandle 접수할 호 객체 핸들.

> Remark

: pHandle 값은 CallbackCallIncoming callback 함수에서 얻어온다. 이 callback 함수의 두 번째 Parameter 가 수신 호 객체의 포인터 값이며 여기서 handle을 얻을 수 있다.

> 함수 원형

HS_RESULT HapiCommandRemoveCall(
HS_STACK_HANDLE hStack,
HS_CALL_HANDLE pHandle,
CallCloseReason pReason);

: 호 Signaling을 종료한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 호 종료 성공 여부에 관한 결과 값이 아님)

> Parameter

hStack Stack Handle.
pHandle 종료할 호의 Handle.
pReason 종료 이유.

> Remark

없음.

> 함수 원형

HS_RESULT HapiCommandUseGRQ(HS_STACK_HANDLE pHandle, BOOL pUse);

: Gatekeeper에 등록 시도 시 GK 발견 과정 (GXX)을 할 것인가를 결정한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 성공 여부에 관한 결과 값이 아님)

> Parameter

pHandle Stack Handle.
pUse GXX 사용여부.

> Remark

: GXX 과정에 대해서는 H.323 7.2.1 절을 참고.

> 함수 원형

```
HS_RESULT HapiCommandChangeBandwidth(
  HS_STACK_HANDLE hStack,
  HS_CALL_HANDLE pHandle,
  HS_UINT pBandwidth );
```

: Bandwidth 할당 과정 (BXX)을 시도한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 BXX 과정 성공 여부에 관한 결과 값이 아님)

> Parameter

hStack Stack Handle.
pHandle BXX 과정을 시도할 호의 Handle.
pBandwidth 요청할 Bandwidth.

> Remark

: BXX 과정에 대해서는 H.323 7.2.4 절 참조.
Vendor에 따라서 Q.931 non Routed mode에서 Connection의 시점을 알기위해 BXX 과정을 사용하기도 한다.

> 함수 원형

```
HS_RESULT HapiCommandH245UserInputIndication(
  HS_STACK_HANDLE hStack,
  HS_CALL_HANDLE pHandle,
  ControlMsg *pMsg );
```

: H245의 UserInputIndication 메시지를 전송한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 메시지 전송 성공 여부에 관한 결과 값이 아님)

> Parameter

hStack Stack Handle.
pHandle 메시지를 전송할 호의 Handle.
pMsg 전송할 H.245 UserInputIndication 메시지 포인터.

> Remark

: 메시지의 융통성을 위해서 Programer가 직접 UserInputIndication 메시지를 만들어서 전송하도록 한다. (OEHPhone의 AppMakeDtmfH245UserInputIndication 함수 참조)

> 함수 원형

HS_RESULT HapiCommandChangeEndpointAliases(
HS_STACK_HANDLE hStack,
NoLockList *pList);

: 단말 객체의 Alias List를 바꾼다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.
(Command 전송 성공 여부에 대한 결과이며 Alias List 변경 성공 여부에 관한 결과 값이 아님)

> Parameter

hStack Stack Handle.
pList 새로 작성한 Alias List의 객체 포인터.

> Remark

: HapiMakeAliasList 함수 참조.

> 함수 원형

HS_STACK_HANDLE HapiStartStack(IEndpoint *pEndpoint);

: Stack을 구동한다.

> Return

Stack Handle 반환, 실패시 HS_INVALID_STACK_HANDLE 반환.

> Parameter

pEndpoint Stack에 사용될 단말 객체.

> Remark

pEndpoint는 Stack 구동전에 본 설정이 완료되어 있어야 한다.
(OEHPhone의 AppH323MakeEndpoint 함수 참조)

> 함수 원형

HS_RESULT HapiStopStack(HS_STACK_HANDLE handle);

: Stack을 종료한다.

> Return

성공시 HS_OK 반환, 실패시 실패 코드 반환.

> Parameter

handle Stack Handle.

> Remark

없음.

Callback 함수.
헤더파일 : IEndpoint.h

메시지 처리 관련 Callback 함수

> 함수 원형

```
BOOL (*CallbackReceiveRasRawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
BOOL (*CallbackReceiveQ931RawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
BOOL (*CallbackReceiveH245RawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
```

: RAS / Q.931 / H.245 Channel을 통해 Octet String의 Raw Data가 수신 되었을 때 발생.

> Return

수신된 Raw Data 가 처리되기를 원하지 않을 경우 FALSE, 그 이외의 경우 TRUE 반환 함.

> Parameter

HS_STACK_HANDLE Stack Handle.
HS_CALL_HANDLE Call Handle.
AsnStream Raw Data를 넣은 AsnStream 구조체 포인터. (AsnStream 구조체 참조)

> Remark

: Raw Data의 처리 Callback 함수는 Raw Data로써만 처리할 수 있는 일들 (전체 메시지 암/복호화 특정 Header 추가 등)을 하기 위해 발생한다.
RAS Channel은 UDP 이므로 Decoding 하기 전의 Raw Data 만으로는 어떤 호의 메시지인지 알 수 없으므로 HS_CALL_HANDLE은 HS_INVALID_CALL_HANDLE 값이 들어온다.

> 함수 원형

BOOL (*CallbackReceiveRasMessage)(HS_STACK_HANDLE,HS_CALL_HANDLE,RasMsg*);
BOOL (*CallbackReceiveQ931Message)(HS_STACK_HANDLE,HS_CALL_HANDLE,Q931Message*);
BOOL (*CallbackReceiveH245Message)(HS_STACK_HANDLE,HS_CALL_HANDLE,ControlMsg*);
: RAS / Q.931 / H.245 Channel을 통해 H.323 메시지가 수신 되었을 때 발생.

> Return

수신된 메시지가 처리되지 않기를 원할 때는 FALSE, 그 이외의 경우 TRUE 반환 함.

> Parameter

HS_STACK_HANDLE Stack Handle.
HS_CALL_HANDLE Call Handle.
RasMsg / Q931Message / ControlMsg 수신된 메시지 객체 포인터.

> Remark

: 이 Callback 함수에서 메시지의 특정 필드 변경이 가능하다.
호와 관련 없는 메시지(GXX, RXX, IRX 등)은 HS_CALL_HANDLE 값이
HS_INVALID_CALL_HANDLE 이다.

> 함수 원형

BOOL (*CallbackSendRasMessage)(HS_STACK_HANDLE,HS_CALL_HANDLE,RasMsg*);
BOOL (*CallbackSendQ931Message)(HS_STACK_HANDLE,HS_CALL_HANDLE,Q931Message*);
BOOL (*CallbackSendH245Message)(HS_STACK_HANDLE,HS_CALL_HANDLE,ControlMsg*);
: RAS / Q.931 / H.245 Channel을 통해 H.323 메시지가 발신 될 때 발생.
(정확히 Raw Data로 Encoding 되기 직전에 발생함)

> Return

메시지가 발신되지 않기를 원할 때는 FALSE, 그 이외의 경우 TRUE 반환 함.

> Parameter

HS_STACK_HANDLE Stack Handle.
HS_CALL_HANDLE Call Handle.
RasMsg / Q931Message / ControlMsg 전송될 메시지 객체 포인터.

> Remark

: 이 Callback 함수에서 메시지의 특정 필드 변경이 가능하다.
호와 관련 없는 메시지(GXX, RXX, IRX 등)은 HS_CALL_HANDLE 값이
HS_INVALID_CALL_HANDLE 이다.

> 함수 원형

BOOL (*CallbackSendRasRawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
BOOL (*CallbackSendQ931RawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
BOOL (*CallbackSendH245RawData)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
: RAS / Q.931 / H.245 Channel을 통해 Octet String의 Raw Data가 전송되기 직전 발생.

> Return

Raw Data 가 전송되기를 원하지 않을 경우 FALSE, 그 이외의 경우 TRUE 반환 함.

> Parameter

HS_STACK_HANDLE Stack Handle.
HS_CALL_HANDLE Call Handle.
AsnStream Raw Data를 넣은 AsnStream 구조체 포인터. (AsnStream 구조체 참조)

> Remark

: Raw Data의 처리 Callback 함수는 Raw Data로써만 처리할 수 있는 일들 (전체 메시지 암/복호화 특정 Header 추가 등)을 하기 위해 발생한다.

예외 처리 관련 Callback 함수

> 함수 원형

void (*CallbackAsnDecodingError)(HS_STACK_HANDLE,HS_CALL_HANDLE,AsnStream*);
: Raw Data를 ASN.1 PER Decoding 중에 에러 발생시 호출 됨.

> Return

없음.

> Parameter

HS_STACK_HANDLE Stack Handle.
HS_CALL_HANDLE Call Handle.
AsnStream ASN.1 PER Decoding 에러가 발생한 Raw Data의 AsnStream 구조체 포인터.

> Remark

없음.

> 함수 원형
  void (*CallbackException)(HSException,HS_RESULT);
  : Signaling 중 예외 상황 발생시 호출 됨.

> Return
  없음.

> Parameter
  HSException 예외 내용.
  HS_RESULT 예외가 발생시의 에러 코드.

> Remark
  : HSException enum 값 참조.

> 함수 원형
  void (*CallbackRasMessageTimeout)(
    HS_STACK_HANDLE,
    HS_CALL_HANDLE,
    ASNH225RasMessageChoice,
    HS_USHORT);
  : RAS Message에 대한 응답이 없어 Timeout이 발생했을 경우 발생.

> Return
  없음.

> Parameter
  HS_STACK_HANDLE Stack Handle.
  HS_CALL_HANDLE Call Handle.
  ASNH225RasMessageChoice Timeout이 발생한 RAS Message의 종류.
  HS_USHORT Timeout이 발생한 RAS Message의 RequestSeqNumber 값.

> Remark
  없음.

> 함수 원형
void (*CallbackRegistrationTimeout)(HS_STACK_HANDLE);
: Gatekeeper 등록 시도가 Timeout 되었을 때 발생.

> Return
없음.

> Parameter
HS_STACK_HANDLE Stack Handle.

> Remark
: HapiCommandChangeGatekeeper 함수 호출 시 Stack은 단 한번의 등록 시도만을 한다.
물론, 등록 이후에 TTL에 의한 재 시도는 Stack에서 동작하지만, 등록 실패 시 에는 다시 시도를 하지 않으므로 지속적인 등록 시도를 하려면 이 Callback 함수에서 HapiCommandChangeGatekeeper 함수를 다시 호출해야 한다.
(OEHPhone AppOnRegistrationTimeout 함수 참조.)

> 함수 원형
HS_RESULT (*CallbackCheckFastStart)(AsnSequenceOf*,NoLockList*,void*);
: FastStart 요청 / 응답이 올 경우 호출 된다. (Remark 참조)

> Return
협상 성공 시 HS_OK 반환, 실패 시 에러코드 반환 함.
(에러코드 반환 시에 Stack은 호를 종료 함.)

> Parameter
AsnSequenceOf 수신한 FastStart List
NoLockList Local 호의 Capability List
void ICall 객체 포인터.

> Remark
: OEHPhone의 AppOnCheckFastStart 함수와 HapiAddForwardFastStart / HapiAddReverseFastStartAPI 참조.
Callback 함수 이므로 호 객체에 직접 접근해도 된다.

> 함수 원형

```
HS_RESULT (*CallbackCheckTerminalCapabilitySet)
(
  HS_STACK_HANDLE,
  void*,
  void*,
  ASNH245TerminalCapabilitySet* );
```

: TermianlCapabilitySet 수신 시 호출된다.

> Return

협상 성공 시 HS_OK 반환, 실패 시 에러코드 반환 함.
(에러코드 반환 시에 Stack은 호를 종료 함.)

> Parameter

HS_STACK_HANDLE Stack Handle.
void 단말 객체 포인터 (IEndpoint)
void 호 객체 포인터 (ICall)
ASNH245TerminalCapabilitySet 수신한 TerminalCapabilitySet

> Remark

: OEHPhone의 AppOnCheckTerminalCapabilitySet 함수와 HapiAddOpenLogicalChannel API 참조.
Callback 함수 이므로 단말 객체 / 호 객체에 직접 접근해도 된다.

> 함수 원형

```
void (*CallbackOpenReceiveMedia)
(
  HS_CALL_HANDLE,
  HS_USHORT,
  HS_USHORT,
  ASNH245DataType* );
```

: H.245 OpenLogicalChannel 또는 FastStart 과정이 정상적으로 완료된 경우 호출 됨.
(수신 LogicalChannel에 관련된 Callback 함수)

> Return

없음.

> Parameter

HS_CALL_HANDLE Call Handle.
HS_USHORT RTP Port
HS_USHORT RTCP Port
ASNH245DataType LogicalChannel의 Data Type (ASN.1 형태)

> Remark
: Stack은 LogicalChannel에 대해서는 관여하지 않는다. 따라서 LogicalChannel이 열렸다는 이 Callback 함수가 발생되면 Programer가 적절한 LogicalChannel을 열어주어야 한다. 위 Callback 함수는 우선 RTP 포맷에 맞추어져 있다.
OEHPhone의 AppOnOpenReceiveMedia 함수 참조. 예제 프로그램에서는 Audio RTP만을 관장하는 프로그램이 함께 실행되며, 이 Callback이 호출되면 그 프로그램으로 메시지를 보내 RTP 통신을 하도록 구성되어 있음.

> 함수 원형

```
void (*CallbackOpenSendMedia)
(
  HS_CALL_HANDLE,
  struct sockaddr_in,
  struct sockaddr_in,
  ASNH245DataType* );
```

: H.245 OpenLogicalChannel 또는 FastStart 과정이 정상적으로 완료된 경우 호출 됨.
(발신 LogicalChannel에 관련된 Callback 함수)

> Return
없음.

> Parameter

HS_CALL_HANDLE	Call Handle.
struct sockaddr_in	RTP TSAP 주소
struct sockaddr_in	RTCP TSAP 주소
ASNH245DataType	LogicalChannel의 Data Type (ASN.1 형태)

> Remark
: Stack은 LogicalChannel에 대해서는 관여하지 않는다. 따라서 LogicalChannel이 열렸다는 이 Callback 함수가 발생되면 Programer가 적절한 LogicalChannel을 열어주어야 한다. 위 Callback 함수는 우선 RTP 포맷에 맞추어져 있다.
OEHPhone의 AppOnOpenSendMedia 함수 참조. 예제 프로그램에서는 Audio RTP만을 관장하는 프로그램이 함께 실행되며, 이 Callback이 호출되면 그 프로그램으로 메시지를 보내 RTP 통신을 하도록 구성되어 있음.

> 함수 원형
void (*CallbackCloseReceiveMedia)(HS_CALL_HANDLE,ASNH245DataType*);
: LogicalChannel이 Close 되어야 하는 시점에서 호출됨.
(수신 LogicalChannel 관련)

> Return
없음.

> Parameter
HS_CALL_HANDLE Call Handle.
ASNH245DataType Close 될 LogicalChannel의 DataType.

> Remark
: Stack은 LogicalChannel에 대해서는 관여하지 않는다. 따라서 LogicalChannel이 닫힌다는 이 Callback 함수가 발생되면 Programer가 적절한 LogicalChannel을 닫아주어야 한다.
OEHPhone의 AppOnCloseReceiveMedia 함수 참조. 예제 프로그램에서는 Audio RTP만을 관장하는 프로그램이 함께 실행되며, 이 Callback이 호출되면 그 프로그램으로 메시지를 보내 RTP 통신을 하도록 구성되어 있음.

> 함수 원형
void (*CallbackCloseSendMedia)(HS_CALL_HANDLE,ASNH245DataType*);
: LogicalChannel이 Close 되어야 하는 시점에서 호출됨.
(발신 LogicalChannel 관련)

> Return
없음.

> Parameter
HS_CALL_HANDLE Call Handle.
ASNH245DataType Close 될 LogicalChannel의 DataType.

> Remark
: Stack은 LogicalChannel에 대해서는 관여하지 않는다. 따라서 LogicalChannel이 닫힌다는 이 Callback 함수가 발생되면 Programer가 적절한 LogicalChannel을 닫아주어야 한다.
OEHPhone의 AppOnCloseSendMedia 함수 참조. 예제 프로그램에서는 Audio RTP만을 관장하는 프로그램이 함께 실행되며, 이 Callback이 호출되면 그 프로그램으로 메시지를 보내 RTP 통신을 하도록 구성되어 있음.

> 함수 원형
  void (*CallbackCallIncoming)(HS_STACK_HANDLE,void*);
  : 수신 호가 있을 경우 호출.

> Return
  없음.

> Parameter
  HS_STACK_HANDLE Stack Handle.
  void 호 수신으로 만들어진 ICall 객체 포인터.

> Remark
  : void 포인터로 넘어온 ICall 객체 포인터에 Capability등의 ICall 정보 설정 작업을 해 주고,
   호 Handle을 얻어내야만 이후에 이 호를 Handling 할 수 있다.
    Callback 함수 이므로 호 객체에 직접 접근할 수 있다.

> 함수 원형
  void (*CallbackCallRemoved)(HS_CALL_HANDLE,CallCloseReason,Q931CauseType);
  : 호가 완전히 종료 되었을 때 호출 됨.

> Retrun
  없음.

> Parameter
  HS_CALL_HANDLE Call Handle.
  CallCloseReason Programer User Level의 호 종료 이유.
  Q931CauseType Q.931 Cause Value.

> Remark
  없음.