Windows NT용**
eNetwork 통신 서버 버전 6.0 및
Windows 95/Windows NT용**
eNetwork 퍼스널 통신 버전 4.2

IBM

시스템 관리 프로그래밍

SA30-0537-01

Windows NT용**
eNetwork 통신 서버 버전 6.0 및
Windows 95/Windows NT용**
eNetwork 퍼스널 통신 버전 4.2

IBM

시스템 관리 프로그래밍

SA30-0537-01

주의사항

이 책과 이 책이 지원하는 제품을 사용하기 전에, 715페이지의 『부록B. 주의사항』에 있는 정보를 읽으십시오.

제 2판(1998년 7월)

개정판에서 달리 언급되지 않는 한, 이 책은 Windows NT용 IBM eNetwork 통신 서버 버전 6.0, Windows 95 및 Windows NT용 퍼스널 통신 버전 4.2, 그리고 모든 후속 릴리스와 수정판에 적용됩니다.

목차

표

이 책에 관하여

이 책에서는 Windows NT용 IBM eNetwork 통신 서버와 Windows 95 및 Windows NT용 IBM eNetwork 퍼스널 통신을 사용하는 프로그램의 개발 방법에 대해 설명합니다.

Windows NT용 IBM eNetwork 통신 서버(이 책에서는 *통신 서버*라고 함)는 통신 서비스 플랫폼입니다. 이 플랫폼에서는 호스트 컴퓨터 및 다른 워크스테이션과 통신하는 Windows NT 워크스테이션에 광범위한 서비스를 제공합니다. 통신 서버 사용자는 다양한 원격 연결 옵션을 선택할 수 있습니다.

Windows 95 및 Windows NT용 IBM eNetwork 퍼스널 통신(이 책에서는 *퍼스널 통신*이라고 함)은 전기능 에뮬레이터입니다. 이것은 호스트 터미널 에뮬레이션과 함께 다음과 같은 유용한 기능을 제공합니다.

- 파일 전송
- 동적 구성
- 사용이 간편한 그래픽 인터페이스
- SNA 기본 클라이언트 응용프로그램용 API
- TCP/IP 기본 응용프로그램이 SNA 기본 네트워크를 통해 통신할 수 있도록 하는 API.

대부분의 경우, 퍼스널 통신이나 통신 서버용 프로그램 개발은 각 프로그램이 여러 개의 동일한 명령을 지원한다는 점에서는 매우 유사하지만 몇 가지 차이점이 있습니다. 이러한 차이점들은 아이콘을 사용하여 표시합니다. 구체적인 세부사항은 x페이지의 『아이콘』을 참조하십시오. 이 책에서 *프로그램*은 퍼스널 통신이나 통신 서버 둘다를 의미합니다. 퍼스널 통신 프로그램이나 통신 서버 프로그램만이 적용될 경우에는 구체적인 프로그램명이 사용됩니다.

이 책에서 Windows**는 Windows 95와 Windows NT**를 의미합니다. 이 책 전반에 걸쳐 *워크스테이션*은 지원되는 모든 퍼스널 컴퓨터를 의미합니다. 퍼스널 컴퓨터의 한 모델이나 구조를 언급할 때는 해당 유형만을 지정합니다.

이 책의 사용자

이 책은 노드 연산자 기능(NOF) API 메세지를 사용하여 퍼스널 통신이나 통신 서버 작업을 관리하고 조회하거나 ASCII 구성 파일을 사용하려는 프로그래머 및 개발자들을 위한 것입니다.

이 책은 또한 퍼스널 통신이나 통신 서버에서 제공되는 기본 관리 서비스 지원을 사용하여 원격(호스트 중재점) 네트워크 관리 응용프로그램과 통신하는 네트워크 관리 응용프로그램을 작성 중인 개발자들을 위한 것입니다.

이 책의 사용법

이 책은 두 부로 구성되어 있습니다. 1페이지의 『제1부 퍼스널 통신이나 통신 서버 노드 연산자 기능』에는 다음 장이 포함됩니다.

- 5페이지의 『제1장 소개』, 이 책의 목적을 설명합니다.
- 9페이지의 『제2장 이 책의 명령 개요』, 노드 연산자 기능 API 구조와 이 구조가 지원하는 명령에 대해 설명합니다. 이 장에서는 퍼스널 통신이나 통신 서버에서 구현된 명령의 범주와 추가 신호를 요약합니다.
- 19페이지의 『제3장 노드 연산자 기능 진입점』, 진입점 확장에 대해 설명합니다.
- 4-12장에서는 각 명령의 구문에 대해 설명합니다. 각 명령에 대한 정보를 보유하는 구조의 사본이 들어 있으며, 각 입력항목을 설명하고 가능한 리턴 코드를 나열합니다.

677페이지의 『제2부 퍼스널 통신이나 통신 서버 관리 서비스 API』에는 다음 장이 포함됩니다.

- 679페이지의 『제13장 관리 서비스 API 소개』, 관리 서비스 API에 대해 설명합니다.
- 683페이지의 『제14장 관리 서비스 진입점』, 관리 서비스 명령의 진입점에 대해 설명합니다.
- 697페이지의 『제15장 관리 서비스 명령』, 각 명령의 구문에 대해 설명합니다. 각 명령에 대한 정보를 보유하는 구조의 사본이 들어 있으며, 각 입력항목을 설명하고 가능한 리턴 코드를 나열합니다.

아이콘

이 책에서 특별한 정보를 전달해야 할 경우에는 다음과 같은 아이콘이 표시됩니다.

이 아이콘은 주의사항, 즉 퍼스널 통신이나 통신 서버의 운영 또는 타스크의 완료에 영향을 미칠 수 있는 중요한 정보를 나타냅니다.

이 아이콘은 정보가 퍼스널 통신 프로그램에만 적용되는 경우에 나타납니다.

이 아이콘은 정보가 통신 서버 프로그램에만 적용되는 경우에 나타납니다.

이 책에서 사용되는 규칙

퍼스널 통신이나 통신 서버 라이브러리 전반에 걸쳐 다음 규칙이 사용됩니다. 나열된 규칙 중 몇몇은 이 책에서 사용되지 않을 수도 있습니다.

텍스트 규칙

굵은체	굵은체는 프로그램이나 명령 프롬프트에서 사용할 수 있는 명령, 함수 및 매개변수를 지시합니다. 값은 대소문자가 구별되며 텍스트에 표시되는 그대로 입력해야 합니다.
이탤릭체	이탤릭체는 다음을 지시합니다. • 사용자가 값을 입력하는 변수. • 리스트, 체크 박스, 입력 필드, 누름 버튼 및 메뉴 선택항목과 같은 창 제어 이름. 창에 표시되는 그대로 텍스트에 표시됩니다. • 책 제목. • 문자는 문자로서 단어는 단어로서 사용됩니다. **예:** *a*가 *an*으로 사용되지 않는지 확인하십시오.
굵은 이탤릭체	굵은 이탤릭체는 단어를 강조하는 데 사용됩니다.
대문자	대문자는 프로그램이나 명령 프롬프트에서 사용할 수 있는 상수, 파일명, 키워드 및 옵션을 지시합니다. 대문자나 소문자로 값을 입력할 수 있습니다.
큰따옴표	큰따옴표는 창에 표시되는 메세지를 지시합니다. 이러한 예는 에뮬레이터 세션의 운영요원 정보 영역(OIA)에 나타나는 메세지입니다.
예제 유형	예제 유형은 사용자가 명령 프롬프트나 창에서 입력해야 할 정보를 지시합니다.

수 규칙

2진수	텍스트로 표시되는 특정한 경우(『2진수 xxxx xxxx의 값은...』)를 제외하고 BX'xxxx xxxx' 또는 BX'x'로 표시합니다.
비트 위치	가장 오른쪽 위치에서 0으로 시작합니다(최소 유효 비트).
십진수	4자리 이상의 십진수는 메트릭 양식으로 표시합니다. 쉼표 대신 공백을 사용하여 세 자리씩 구분합니다. 예를 들어, 16147은 16 147과 같이 표시합니다.
16진수	텍스트에서 hex xxxx 또는 X'xxxx'로 표시합니다(『인접 노드의 주소는 hex 5D이며, X'5d'로 지정합니다』).

추가 정보 위치

자세한 내용은 *빠른 시작*을 참조하십시오. 여기에는 통신 서버 라이브러리와 관련 서적에 관한 전반적인 설명이 들어 있습니다.

통신 서버를 설치한 후에 특정 서적을 보려면, 데스크탑에서 다음 경로를 사용하십시오.

1. 프로그램
2. IBM 통신 서버
3. 문서
4. 서적 리스트에서 선택

통신 서버 서적은 Adobe Acrobat Reader로 볼 수 있는 포터블 문서양식(PDF)으로 되어 있습니다. 사용자 시스템에 이 프로그램의 사본이 없는 경우에는 문서 리스트에서 이를 설치할 수 있습니다.

인터넷 상의 통신 서버 홈 페이지에는 일반 제품 정보와 APAR 및 수정판에 관한 서비스 정보가 들어 있습니다. 이 홈 페이지를 방문하려면, IBM Web Explorer와 같은 인터넷 브라우저를 사용하여 다음 URL로 가십시오.

http://www.software.ibm.com/enetwork/commserver/about/csnt.html

자세한 내용은 *빠른 시작*을 참조하십시오. 여기에는 퍼스널 통신 라이브러리와 관련 서적에 관한 전반적인 설명이 들어 있습니다.

퍼스널 통신을 설치한 후에 특정 서적을 보려면, 데스크탑에서 다음 경로를 사용하십시오.

1. 프로그램
2. IBM 통신 서버
3. 문서
4. 서적 리스트에서 선택

퍼스널 통신 서적은 IBM Library Reader로 볼 수 있는 북매니저 형식(BOO)으로 되어 있습니다. 사용자 시스템에 이 프로그램의 사본이 없는 경우에는 eNetwork 퍼스널 통신CD-ROM에서 이를 설치할 수 있습니다.

인터넷 상의 퍼스널 통신 홈 페이지에는 일반 제품 정보와 APAR 및 수정판에 관한 서비스 정보가 들어 있습니다. 이 홈 페이지를 방문하려면, IBM Web Explorer와 같은 인터넷 브라우저를 사용하여 다음 URL로 가십시오.

http://www.software.ibm.com/enetwork/pcomm/

제1부 퍼스널 통신이나 통신 서버 노드 연산자 기능

제1장 소개

이 부에서는 퍼스널 통신이나 통신 서버에서 제공되는 노드 연산자 기능(NOF)에 대해 설명합니다.

이 책의 목적

이 책의 목적은 다음과 같습니다.

- 노드 연산자 기능 API의 구조에 대한 간략한 개요 제공
- 인터페이스를 통한 신호의 구문 정의.

퍼스널 통신이나 통신 서버 노드 연산자 기능

퍼스널 통신이나 통신 서버 노드 연산자 기능은 노드 운영요원과 제어점(CP) 및 논리 장치(LU) 간의 통신을 가능하게 합니다. 노드 연산자 기능은 운영요원에게서 노드 구성 정보를 수신하여, 노드가 시작될 때 이 정보 로 제어점(CP)을 초기화합니다. 노드 연산자 기능은 또한 노드 구성 정보를 조회하고 표시하기 위한 요청을 수신합니다. 노드 운영요원은 다음을 수행할 수 있습니다.

- LU, DLC, 포트 및 링크 정의 및 삭제
- 링크 및 세션 활성화 및 활성종료
- 데이터베이스의 제어점(CP) 및 LU와 상태 정보 조회

노드 운영요원은 대화방식의 화면 표시, 파일 인터페이스가 액세스하는 명령 파일이나 트랜잭션 프로그램을 조작하는 사람입니다. 노드 연산자 기능은 명령 인터페이스를 사용하여 노드 운영요원과 통신합니다.

진입점

퍼스널 통신이나 통신 서버는 노드 연산자 기능 명령을 처리하는 라이브러리 파일을 제공합니다.

노드 연산자 기능 명령은 단순 언어 인터페이스를 가집니다. 프로그램은 *명령 제어 블럭*이라고 하는 메모리 블럭의 필드를 채웁니다. 그런 후 프로그램 은 진입점을 호출하여 명령 제어 블럭으로 포인터를 전달합니다. 프로그램 작업이 완료되면, 노드 연산자 기능이 명령 제어 블럭 내의 필드를 사용하고 수정한 후 반환됩니다. 이로써 프로그램은 명령 제어 블럭에서 반환된 매개변수를 읽을 수 있습니다.

다음은 노드 연산자 기능 명령의 진입점 리스트입니다.

- WinAsyncNOF()
- WinAsyncNOFEx()

- WinNOFCancelAsyncRequest()
- WinNOFCleanup()
- WinNOFStartup()
- WinNOFRegisterIndicationSink()
- WinNOFUnregisterIndicationSink()
- WinNOFGetIndication()

진입점에 대한 자세한 설명은 제3장 노드 연산자 기능 진입점을 참조하십시오.

명령 제어 블럭(VCB)

프로그래밍 주의: 기본 운영 체제(BOS)는 호출하는 응용프로그램의 주소 영역에서 일부 부속시스템을 실행함으로써 성능을 최적화합니다. 이는 응용프로그램 프로그램이 국지 설명자 테이블(LDT) 선택자를 잘못 사용할 경우 운영이 적절하지 않거나 시스템이 실패할 가능성이 있음을 의미합니다. 따라서, 응용프로그램 프로그램은 포인터의 LDT 선택자 필드를 변경하게 되는 포인터 산술 연산을 수행해서는 안됩니다.

명령 제어 블럭(VCB)에 사용되는 세그멘트는 읽기/쓰기 데이터 세그먼트여야 합니다. 프로그램은 VCB를 변수로 선언하거나, 할당하거나, 보다 큰 세그먼트에서 서브할당할 수 있습니다. VCB는 프로그램이 발행하는 명령의 모든 필드를 포함할 수 있을 만큼 충분히 커야 합니다.

응용프로그램 프로그램은 명령이 발행된 후 완료될 때까지 명령 제어 블럭의 어떠한 부분도 변경해서는 안됩니다. 노드 연산자 기능은 명령 실행을 완료한 후 수정된 전체 VCB를 원래의 블럭으로 다시 복사합니다. 따라서, 프로그램이 명령 제어 블럭을 변수로 선언할 경우에는 내부 프로시듀어의 스택이 아닌 정적 기억영역에서 이를 선언하십시오.

각 VCB에 있는 모든 예약 필드 및 비사용 필드를 0(X'00')으로 채우십시오. 사실상, 프로그램이 매개변수에 값을 지정하기 전에 전체 명령 제어 블록을 0으로 설정함으로써 시간을 좀더 절약할 수도 있습니다. 예약 필드를 0으로 설정하는 것이 특히 중요합니다.

주: VCB가 읽기/쓰기가 아니거나 10바이트 이상이 아닌 경우 (즉, 노드 연산자 기능 1차 및 2차 리턴 코드를 보유할 수 있을 만큼 충분히 크지 않을 경우), 노드 연산자 기능은 여기에 액세스할 수 없으며 기본 운영 체제(BOS)가 비정상적으로 프로세스를 종료합니다. 이러한 종료는 *일반 보호 오류*, 프로세서 예외 트랩 D로 인식됩니다.

노드 연산자 기능은 VCB가 너무 짧거나 틀린 유형

노드 연산자 기능(NOF) 프로그램 작성

퍼스널 통신이나 통신 서버는 NOF 명령을 처리하는 동적 링크 라이브러리(DLL) 파일을 제공합니다.

DLL은 재진입이 가능합니다. 즉, 복수 응용프로그램 프로세스 및 스레드가 동시에 DLL을 호출할 수 있습니다.

NOF 명령은 단순 언어 인터페이스를 가집니다. 프로그램이 *명령 제어 블럭*(VCB)이라고 하는 메모리 블럭의 필드를 채웁니다. 그런 후 프로그램은 NOF DLL을 호출하여 명령 제어 블럭으로 포인터를 전달합니다. 프로그램 작업이 완료되면, NOF가 VCB 내의 필드를 사용하고 수정한 후 반환됩니다. 이로써 프로그램은 명령 제어 블럭에서 반환된 매개변수를 읽을 수 있습니다.

7페이지의 표 1은 NOF 프로그램을 컴파일하고 링크하는 데 필요한 제공되는 헤더 파일 및 라이브러리의 소스 모듈 사용법을 보여줍니다. 일부 헤더 파일에는 요구되는 다른 헤더 파일이 포함될 수도 있습니다.

표 1. NOF용 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL 명
WINNT & WIN95	WINNOF.H	WINNOF32.LIB	WINNOF32.DLL
WIN3.1	WINNOF.H	WINNOF.LIB	WINNOF.DLL
OS/2	APPC_C.H	APPC.LIB	APPC.DLL

통신 서버 SNA API 클라이언트 지원

다음 정보는 통신 서버에만 적용됩니다.

통신 서버에는 Windows 95, Windows NT, Windows 3.1 및 OS/2 운영체제용 클라이언트 세트가 포함되어 있습니다. 이들 클라이언트(이 책에서는 SNA API 클라이언트라고 함)는 전체 노드 연산자 기능의 부속세트만을 지원합니다. 구체적으로, **WINNOF**가 Windows 클라이언트(95, NT, 3.1)에서 지원되는 유일한 API입니다. 다음은 지원되는 NOF 명령 리스트입니다.

- QUERY_LOCAL_LU
- QUERY_LU_0_TO_3
- QUERY_LU_POOL
- QUERY_MODE
- QUERY_MODE_DEFINITION
- QUERY_PARTNER_LU
- QUERY_PARTNER_LU_DEFINITION
- QUERY_PU

- QUERY_SESSION
- QUERY_TP
- QUERY_TP_DEFINITION

통신 서버에서는 지원되고 퍼스널 통신에서는 지원되지 않는 명령

다음 정보는 통신 서버에만 적용됩니다.

다음은 통신 서버에서는 지원되지만 퍼스널 통신에서는 지원되지 않는 명령 리스트입니다.

- DEFINE_DOWNSTREAM_LU
- DEFINE_DOWNSTREAM_LU_RANGE
- DEFINE_DSPU_TEMPLATE
- DELETE_DOWNSTREAM_LU
- DELETE_DOWNSTREAM_LU_RANGE
- DELETE_DSPU_TEMPLATE
- QUERY_ADJACENT_NN
- QUERY_DIRECTORY_STATS
- QUERY_DOWNSTREAM_LU
- QUERY_DOWNSTREAM_PU
- QUERY_DSPU_TEMPLATE
- QUERY_HPR_STATS
- QUERY_ISR_SESSION
- QUERY_NN_TOPOLOGY_NODE
- QUERY_NN_TOPOLOGY_STATS
- QUERY_NN_TOPOLOGY_TG
- DOWNSTREAM_LU_INDICATION
- DOWNSTREAM_PU_INDICATION
- ISR_INDICATION
- NN_TOPOLOGY_NODE_INDICATION
- NN_TOPOLOGY_TG_INDICATION

제2장 이 책의 명령 개요

이 책에서 설명하는 명령 인터페이스는 사용하는 프로그램이 퍼스널 통신이나 통신 서버 네트워크 환경과 연관된 대부분의 구성, 시스템 관리 및 노드 정의 기능을 수행할 수 있도록 합니다. 이 장에서는 이러한 기능과 연관된 명령의 각각에 대한 개요를 제공합니다.

명령 설명을 읽는 방법

4장-12장은 구성, 시스템 관리 및 접속 관리자 명령에 대해 설명합니다.

제공되는 매개변수

각 명령을 설명하는 부분에는 프로그램에서 제공되는 매개변수 및 연관된 매개변수 값에 대해 자세하게 설명하는 절이 있습니다.

몇몇 경우에는 사용자가 매개변수의 변수 값을 지정해야 합니다.

반환되는 매개변수

각 명령을 설명하는 부분에는 프로그램으로 반환되는 매개변수나 연관된 매개변수 값에 대해 자세하게 설명하는 절이 있습니다.

리턴 코드

이 책에서 설명하는 구성, 시스템 관리 및 접속 관리자 명령은 명령 실행 성공 여부에 관한 정보를 제공하거나 오류 정보를 제공하는 연관된 리턴 코드를 가집니다. 이러한 리턴 코드는 각 명령의 『반환되는 매개변수』 절에 나열됩니다.

추가 정보

명령 설명에는 또한 『추가 정보』라는 제목이 붙은 절이 포함될 수도 있습니다. 이 절에서는 명령에 관한 유용한 추가 정보를 제공합니다.

공용 VCB 필드

이 장에서는 노드 연산자 기능 API를 통해 전달되는 각 명령의 구문을 문서화합니다. 여기서는 또한 각 명령에서 전달되거나 각 명령에 반환되는 매개변수에 대해서도 설명합니다.

```
typedef struct nof_hdr
{
        unsigned short  opcode;
        unsigned char   reserv2;            /* reserved */
```

```
        unsigned char   format;
        unsigned short  primary_rc;
        unsigned long   secondary_rc;
} NOF_HDR;
```

각 VCB에는 다수의 공용 필드가 있습니다. 이러한 공용 필드 리스트와 그에 대한 설명은 다음과 같습니다.

opcode
명령 운영 코드. 이 필드는 명령을 식별합니다.

format
VCB 형식을 식별합니다. VCB의 현재 버전을 지정하기 위해 설정해야 하는 값은 각 명령 아래에 개별적으로 문서화됩니다.

primary_rc
1차 리턴 코드. 각 명령에 가능한 값이 각 명령 절에 나열되어 있습니다.

secondary_rc
2차 리턴 코드. 1차 리턴 코드가 제공하는 정보를 보충합니다.

명령 요약

노드 연산자 기능 API는 다음을 수행하는 데 사용할 수 있는 명령으로 구성됩니다.

- 노드 자원 구성
- 링크 및 세션 활성화 및 활성종료
- 노드에서 보유하는 정보 조회
- 세션 수 변경
- 요청되지 않은 표시 처리
- 암호 지원 제공
- 원격 LU 『핑(ping)』
- CPI-C쪽의 정보 정의, 조회 및 삭제

노드 구성

다음 명령을 사용하여 자원을 정의할 수 있습니다.

- DEFINE_ADJACENT_NODE
- DEFINE_CN
- DEFINE_COS
- DEFINE_DEFAULT_PU
- DEFINE_DLC
- DEFINE_DLUR_DEFAULTS
- DEFINE_DOWNSTREAM_LU

DEFINE_DOWNSTREAM_LU는 통신 서버 전용입니다.

- DEFINE_DOWNSTREAM_LU_RANGE

DEFINE_DOWNSTREAM_LU_RANGE는 통신 서버 전용입니다.

- DEFINE_DSPU_TEMPLATE
- DEFINE_FOCAL_POINT
- DEFINE_INTERNAL_PU
- DEFINE_LOCAL_LU
- DEFINE_LS
- DEFINE_LU62_TIMEOUT
- DEFINE_LU_0_TO_3
- DEFINE_LU_0_TO_3_RANGE
- DEFINE_LU_POOL
- DEFINE_MODE
- DEFINE_PARTNER_LU
- DEFINE_PORT
- DEFINE_TP

다음 명령을 사용하여 자원을 삭제할 수 있습니다.

- DELETE_ADJACENT_NODE
- DELETE_CN
- DELETE_COS
- DELETE_DLC
- DELETE_DOWNSTREAM_LU

DELETE_DOWNSTREAM_LU는 통신 서버 전용입니다.

- DELETE_DOWNSTREAM_LU_RANGE

DELETE_DOWNSTREAM_LU_RANGE는 통신 서버 전용입니다.

- DELETE_DSPU_TEMPLATE
- DELETE_FOCAL_POINT
- DELETE_INTERNAL_PU
- DELETE_LOCAL_LU
- DELETE_LS
- DELETE_LU62_TIMEOUT
- DELETE_LU_0_TO_3

- DELETE_LU_0_TO_3_RANGE
- DELETE_LU_POOL
- DELETE_MODE
- DELETE_PARTNER_LU
- DELETE_PORT
- DELETE_TP

활성화 및 활성종료

링크 레벨에서는 다음 명령을 사용할 수 있습니다.

- START_DLC
- START_LS
- START_PORT
- STOP_DLC
- STOP_LS
- STOP_PORT

종속 LU 리퀘스터 함수에는 다음 명령을 사용할 수 있습니다.

- START_INTERNAL_PU
- STOP_INTERNAL_PU

세션 레벨에서는 다음 명령을 사용할 수 있습니다.

- ACTIVATE_SESSION
- DEACTIVATE_CONV_GROUP
- DEACTIVATE_SESSION

다음 명령을 사용하여 고성능 경로지정(HPR) RTP 연결이 경로를 전환하도록 할 수 있습니다.

PATH_SWITCH

노드 조회

다음 명령은 명명된 필드에서 노드 정보를 반환합니다.

- QUERY_DEFAULT_PU
- QUERY_DLUR_DEFAULTS
- QUERY_MDS_STATISTICS
- QUERY_NN_TOPOLOGY_STATS

QUERY_NN_TOPOLOGY_STATS는 통신 서버 전용입니다.

- QUERY_NODE
- QUERY_STATISTICS

다음 명령은 하나 이상의 정보 단위를 반환할 수 있습니다.

- QUERY_ADJACENT_NN
- QUERY_ADJACENT_NODE
- QUERY_CN
- QUERY_CN_PORT
- QUERY_COS
- QUERY_DEFAULTS
- QUERY_DLUS
- QUERY_DOWNSTREAM_PU

QUERY_DOWNSTREAM_PU는 통신 서버 전용입니다.

- QUERY_DSPU_TEMPLATE
- QUERY_FOCAL_POINT
- QUERY_LU_POOL
- QUERY_LU62_TIMEOUT
- QUERY_MDS_APPLICATION
- QUERY_MODE_TO_COS_MAPPING
- QUERY_NMVT_APPLICATION
- QUERY_PU
- QUERY_TP

이러한 정보는 리스트 양식으로 저장되는 것으로 생각할 수 있습니다. 명령은 리스트의 명명된 입력항목을 지정할 수 있으면 이 입력항목이 리스트의 위치 지정자(또는 색인 값)로 간주됩니다. 명령에 있는 **list_options** 필드는 리스트에서 정보가 반환되는 포인트를 지정합니다.

- **list_options**을 AP_FIRST_IN_LIST로 설정하면, 색인 값을 지정하는 필드가 무시되고 반환된 리스트가 맨 처음부터 시작됩니다.
- **list_options**을 AP_LIST_INCLUSIVE로 설정하면, 반환된 리스트는 지정한 색인 값에서 시작됩니다.
- **list_options**을 AP_LIST_FROM_NEXT로 설정하면, 반환된 리스트는 지정한 색인 값 다음의 입력항목부터 시작됩니다.

색인 값은 반환된 정보의 시작점을 지정합니다. 일단 시작 포인트가 판별되면, 일부 조회 명령 역시 반환된 리스트에 대한 추가 필터링 옵션을 제공합니다. 이들은 색인 값과는 별도로 지정됩니다. 달리 지정하지 않는 한, 반환된 리스트의 순서는 IBM의 6611 APPN MIB에 따라 지정됩니다. (명령 매개변수가 MIB 표 입력항목으로 매핑되는 방법에 대해서는 711페이지의 『부록A. IBM APPN MIB 표』를 참조하십시오.)

반환되는 입력항목 수나 채워지는 버퍼 크기가 설정됩니다. (이들이 둘다 설정되어 있으면, 지정된 두 정보 값 중 낮은 값이 명령에 반환됩니다.) 일반적으로 응용프로그램 버퍼 크기는 반환할 수 있는 정보의 양을 제한하므로,

노드 연산자 기능은 요청된 정보를 반환하는 데 필요한 총 버퍼 공간의 양과 이것이 나타내는 총 입력항목 수를 지시하는 추가 정보를 반환합니다.

하나 이상의 정보 단위를 반환하는 것 외에도, 다음 명령은 다른 레벨의 정보를 반환할 수도 있습니다. **list_options** 필드에 포함된 AP_DETAIL 또는 AP_SUMMARY는 요약 정보가 반환되는지 또는 자세한 정보가 반환되는지를 지정합니다. 이러한 옵션은 하나의 **list_options**을 **OR**로 연결하여 지정할 수 있습니다(예를 들면, AP_DETAIL | AP_FIRST_IN_LIST).

- QUERY_DIRECTORY_LU
- QUERY_DLC
- QUERY_DLUR_LU
- QUERY_DLUR_PU
- QUERY_DOWNSTREAM_LU

QUERY_DOWNSTREAM_LU는 통신 서버 전용입니다.

- QUERY_ISR_SESSION

QUERY_ISR_SESSION은 통신 서버 전용입니다.

- QUERY_LOCAL_LU
- QUERY_LOCAL_TOPOLOGY
- QUERY_LS
- QUERY_LU_0_TO_3
- QUERY_MODE
- QUERY_MODE_DEFINITION
- QUERY_NN_TOPOLOGY_NODE

QUERY_NN_TOPOLOGY_NODE는 통신 서버 전용입니다.

- QUERY_NN_TOPOLOGY_TG

QUERY_NN_TOPOLOGY_TG는 통신 서버 전용입니다.

- QUERY_PARTNER_LU
- QUERY_PARTNER_LU_DEFINITION
- QUERY_PORT
- QUERY_RTP_CONNECTION
- QUERY_SESSION
- QUERY_TP_DEFINITION

세션 한계 명령

- CHANGE_SESSION_LIMIT
- INITIALIZE_SESSION_LIMIT
- RESET_SESSION_LIMIT

요청되지 않은 표시

노드 정보를 화면에 나타내는 응용프로그램은 다음과 같은 표시(변경이 발생되는 경우 발행되며 요약 정보만을 반환합니다)를 사용하여 조회 명령(자세한 정보를 반환하는)을 트리거할 수 있습니다. 정보를 수신하도록 등록된 응용프로그램이 있을 경우, 노드는 명명된 이벤트의 요청되지 않은 표시로 아래에 나열된 신호만을 생성합니다. 따라서 응용프로그램은 더이상 정보가 필요하지 않을 경우 등록해제해야 합니다.

- DLC_INDICATION
- DLUR_LU_INDICATION
- DLUS_INDICATION
- DOWNSTREAM_LU_INDICATION

DOWNSTREAM_LU_INDICATION은 통신 서버 전용입니다.

- DOWNSTREAM_PU_INDICATION

DOWNSTREAM_PU_INDICATION은 통신 서버 전용입니다.

- FOCAL_POINT_INDICATION
- ISR_INDICATION

ISR_INDICATION은 통신 서버 전용입니다.

- LOCAL_LU_INDICATION
- LOCAL_TOPOLOGY_INDICATION
- LS_INDICATION
- LU_0_TO_3_INDICATION
- MODE_INDICATION
- NN_TOPOLOGY_NODE_INDICATION

NN_TOPOLOGY_NODE_INDICATION은 통신 서버 전용입니다.

- NN_TOPOLOGY_TG_INDICATION

NN_TOPOLOGY_TG_INDICATION은 통신 서버 전용입니다.

- PLU_INDICATION
- PORT_INDICATION
- PU_INDICATION
- REGISTRATION_FAILURE
- RTP_INDICATION
- SESSION_INDICATION
- SESSION_FAILURE_INDICATION

표시에 사용되는 진입점은 다음과 같습니다.

WinNOFRegisterIndicationSink
표시를 수신하도록 등록

WinNOFUnregisterIndicationSink
표시를 수신하지 않도록 등록해제

WinNOFGetIndication
표시 수신

표시는 노드 연산자 기능에 등록된 표시 싱크로 전달됩니다. 표시를 생성하는 구성요소가 표시를 전송할 수 없는 경우, 이 구성요소는 발행하는 그 다음 표시에 **data_lost** 표시기를 설정합니다. 표시에서 **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 이 플래그는 변경이 발생하여 세부사항이 유실되었음을 응용프로그램에 통지하는 데 사용되며, 응용프로그램이 적절한 조회 명령을 발행하여 응답할 것을 지시합니다.

LULU_EVENT 신호는 REGISTER_LULU_EVENT 및 UNREGISTER_LULU_EVENT로 등록된 프로세스에 노드의 요청 없이 전송되므로, 이 신호 역시 하나의 표시로 분류됨에 주의하십시오. 이 신호가 위의 리스트에 들어 있지 않는 것은 동작에 있어서 상당한 차이가 있기 때문입니다: 등록은 LU-상대방 LU 쌍을 이루기 위한 것이며 **data_lost**와 같은 것이 없습니다 — 이러한 LULU 이벤트 표시는 반드시 생성됩니다.

보안 명령

다음 보안 명령은 LU-LU 검증이나 대화 보안을 위한 암호 관리를 가능하게 합니다.

- DEFINE_LU_LU_PASSWORD
- DEFINE_USERID_PASSWORD
- DELETE_LU_LU_PASSWORD
- DELETE_USERID_PASSWORD

APING 명령

다음 명령은 관리 응용프로그램이 네트워크에 있는 원격 LU를 『핑(ping)』할 수 있도록 합니다.

APING

CPI-C 명령

다음 명령은 CPI-C쪽의 정보를 정의, 조회 및 삭제할 수 있도록 합니다.

- DEFINE_CPIC_SIDE_INFO
- DELETE_CPIC_SIDE_INFO
- QUERY_CPIC_SIDE_INFO

Windows 95 및 Windows NT용 퍼스널 통신이나 통신 서버에서 제공되는 CPI-C 지원에 대해서는 *CPI-C 참조 사항*을 참조하십시오.

접속 관리자 명령

다음 명령은 접속 관리자를 제어하는 데 사용할 수 있습니다.

- DISABLE_ATTACH_MANAGER
- ENABLE_ATTACH_MANAGER
- QUERY_ATTACH_MANAGER

DLC 프로세스, 포트 및 링크 스테이션

DLC 프로세스

퍼스널 통신이나 통신 서버는 복수의 DLC 프로세스를 작성할 수 있습니다. 각 DLC 프로세스는 퍼스널 통신이나 통신 서버가 노드 연산자 기능 API에서 발행된 START_DLC 명령에 대한 응답으로 작성됩니다. 각 DLC는 특정 데이터 링크 프로토콜(예를 들면, SDLC 또는 토큰링)을 사용하는 링크나 링크 집합을 통한 통신을 담당합니다.

각 DLC는 하나 이상의 포트를 관리할 수 있습니다. 다음은 포트에 대한 설명입니다.

포트

포트는 국지 시스템에서의 고유 액세스점(예를 들면, MAC/SAP 주소 쌍)을 나타내며 DLC 프로세스와 연관됩니다. 각 DLC는 하나 이상의 포트를 가질 수 있습니다. 포트는 다음 유형 중 하나가 될 수 있습니다.

교환 D 포트
: 특정 시점에서 활동중인 하나 이상의 인접 링크 스테이션을 가질 수 있습니다. (이는 *SNA APPN 구조 참조 사항*에 기술된 정의와 차이가 있음에 주의하십시오.)

비교환 D 포트
: 포인트간 링크 연결과 멀티포인트 링크 연결 둘다를 가질 수 있습니다. 노드 연산자 기능 구성요소가 비교환식 링크 연결 상의 인접 링

크 스테이션을 정의해야 합니다. 멀티포인트 비교환식 링크는 예상치 못한 결과를 피하기 위해 모든 노드에 1차/2차 관계를 적절히 정의해야 합니다.

SATF 포트
토큰링과 같은 공유 액세스 전송 기능(SATF)을 사용합니다. 이 포트는 공유 액세스 전송 기능(SATF)에 접속하는 링크 스테이션 쌍간의 연결을 가능하게 합니다. 링크 스테이션을 통해 링크 활성화를 시작할 수 있도록 하려면, 토큰링에서 활성화되는 모든 링크 스테이션의 초기 역할을 조정가능으로 정의해야 합니다.

주: SATF 포트는 또한 연결 네트워크와 연관될 수도 있습니다. 이 경우에는 위상(topology) 갱신을 사용하여 고유 액세스점의 주소를 브로드캐스트합니다.

링크 스테이션

링크 스테이션은 포트와 연관되며 인접 노드와의 연결을 나타냅니다. 포트는 복수의 링크 스테이션을 가질 수 있습니다. 링크 스테이션은 다음과 같이 범주화할 수 있습니다.

정의된 링크 :테이션
명시적으로(DEFINE_LS 명령을 사용하여) 정의된 링크 스테이션.

동적 링크 :테이션
연결 네트워크를 통해 동적 연결을 활성화한 결과로 작성된 링크 스테이션(가상 경로지정 노드(VRN)라고도 함).

암C적 링크 :테이션
교환식이나 SATF 포트에서 알 수 없는 상대방 노드에서 수신된 호출의 결과로 작성된 링크 스테이션(이 유형의 포트는 *SNA APPN 구조 참조 사항*에 정의되어 있지 않습니다.).

임C 링크 :테이션
교환식 또는 SATF 포트에서 CONNECT_IN이 DLC 인터페이스를 통해 수신될 때 작성되는 링크 스테이션. 원격 노드 ID가 판별될 때, 이 링크 스테이션은 삭제되거나 동적 또는 암시적 링크 스테이션이 됩니다.

제3장 노드 연산자 기능 진입점

이 장에서는 노드 연산자 기능 명령의 진입점에 대해 설명합니다.

WinNOF()

이 함수는 모든 노드 연산자 기능 명령의 동기 진입점을 제공합니다.

구문

```
void WINAPI WinNOF(long vcb,
                           unsigned short vcb_size)
```

매개변수

설명

vcb 명령 제어 블럭 포인터.

vcb_size
명령 제어 블럭의 바이트 수.

리턴 값

리턴 값이 없음. 명령 제어 블럭 내의 **primary_rc** 및 **secondary_rc** 필드는 오류를 지시합니다.

사용시 주의사항

이 함수는 노드 연산자 기능 API의 주 동기 진입점입니다. 명령이 완료될 때까지 블럭을 호출합니다.

WinAsyncNOF()

이 함수는 모든 노드 연산자 기능 명령의 비동기 진입점을 제공합니다.

구문

```
HANDLE WINAPI WinAsyncNOF(HWND hwnd,
                          long vcb,
                        unsigned short vcb_size)
```

매개변수

설명

hwnd 완료 메세지를 수신하는 창 핸들.

vcb 명령 제어 블럭 포인터.

vcb_size
명령 제어 블럭의 바이트 수.

리턴 값

리턴 값은 비동기 요청이 성공적인지의 여부를 지정합니다. 이 함수가 성공적일 경우, 실제 리턴 값은 핸들입니다. 이 함수가 실패할 경우에는 0이 반환됩니다.

사용시 주의사항

이 진입점을 사용할 때, 각 응용프로그램 스레드는 한 번에 한 개의 미결 요청만을 가질 수 있습니다.

비동기 작업이 완료되면, 응용프로그램 창 *hWnd*는 반환된 메세지 **RegisterWindowMessage**와 입력 문자열 **"WinAsyncNOF"**를 수신합니다. *wParam* 인수에는 원래의 함수 호출에 의해 반환된 비동기 타스크 핸들이 포함됩니다.

이 함수가 성공적으로 반환될 경우, 작업이 완료되거나 대화가 취소될 때 **WinAsyncNOF()** 메세지가 응용프로그램에 전달됩니다.

주: **WinNOFCancelAsyncRequest()**도 참조하십시오.

WinAsyncNOFEx()

이 함수는 모든 노드 연산자 기능 명령의 비동기 진입점을 제공합니다. 동일 스레드에서 복수의 명령을 처리하려면 방해 호출 대신 이 진입점을 사용하십시오.

구문

```
HANDLE WINAPI WinAsyncNOFEx(HANDLE handle,
                            long vcb,
                            unsigned short vcb_size);
```

매개변수

설명

handle
응용프로그램이 서비스를 제공하는 이벤트의 핸들.

vcb 명령 제어 블럭 포인터.

vcb_size
명령 제어 블럭의 바이트 수.

리턴 값

리턴 값은 비동기 요청이 성공적인지의 여부를 지정합니다. 이 함수가 성공적일 경우, 실제 리턴 값은 핸들입니다.

사용시 주의사항

이 진입점은 Win32** API의 WaitForMultipleObjects에서 사용하기 위한 것입니다. 이 함수에 대한 자세한 내용은 Win32 API에 대한 프로그래밍 문서를 참조하십시오.

비동기 작업이 완료되면, 이벤트 신호를 전달하여 이를 응용프로그램에 통지합니다. 이벤트 신호 전달시, 1차 리턴 코드나 2차 리턴 코드를 검토하여 오류 조건이 있는지 확인하십시오.

주: **WinNOFCancelAsyncRequest()**도 참조하십시오.

WinNOFCancelAsyncRequest()

이 함수는 미결 **WinAsyncNOF** 기본 요청을 취소합니다.

구문

```
int WINAPI WinNOFCancelAsyncRequest(HANDLE handle);
```

매개변수

설명

handle

제공되는 매개변수: 취소할 요청의 핸들을 지정합니다.

리턴 값

리턴 값은 비동기 요청이 취소되었는지의 여부를 지정합니다. 리턴 값이 0이면 요청이 취소된 것입니다. 리턴 값이 0이 아닐 경우, 가능한 값은 다음과 같습니다.

WNOFALREADY

취소되는 비동기 요청이 이미 완료되었거나 핸들이 유효하지 않음을 지시하는 오류 코드.

사용시 주의사항

WinAsyncNOF 함수들 중 하나로 이전에 발행한 비동기 요청은 초기 함수에 의해 반환되는 핸들을 *handle*에 지정하여 **WinNOFCancelAsyncRequest()** 호출을 발행함으로써 완료 전에 취소할 수 있습니다.

비동기 요청을 취소하면, 응용프로그램 명령 제어 블럭 갱신이 종료되고 명령이 완료되었음을 통지하는(창 메세지나 이벤트를 통해) 응용프로그램이 종료됩니다. 그러나 기본 요청이 취소되는 것은 아닙니다. 기본 요청을 실제로 취소하려면, 응용프로그램이 적절한 NOF 명령(즉, START_LS를 취소하는 STOP_LS)를 발행해야 합니다.

기존의 비동기 **WinAsyncNOF** 루틴을 취소하려는 시도가 오류 코드 WNOFALREADY로 실패하면, 다음 두 상황 중 하나가 발생한 것입니다. 원래의 루틴이 이미 완료되고 응용프로그램이 완료 통지를 처리하였거나, 원래의 루틴이 이미 완료되었으나 응용프로그램이 아직 완료 통지를 처리하지 않았습니다.

주: **WinAsyncNOF()**도 참조하십시오.

WinNOFCleanup()

이 함수는 노드 연산자 기능 API에서 응용프로그램을 종료하거나 등록취소합니다.

구문

```
BOOL WINAPI WinNOFCleanup(void);
```

리턴 값

리턴 값은 등록이 취소되었는지의 여부를 지정합니다. 리턴 값이 0이 아니면, 응용프로그램이 등록취소된 것입니다. 0 값이 반환되면 응용프로그램이 등록취소되지 않은 것입니다.

사용시 주의사항

노드 연산자 기능 API에서의 노드 연산자 기능 응용프로그램 등록취소를 지시하려면 **WinNOFCleanup()**을 사용하십시오.

WinNOFCleanup은 **WinNOFGetIndication**에서 대기중인 스레드를 블럭해제합니다. 이들은 WNOFNOTREG와 함께 반환됩니다.(응용프로그램이 표시를 수신하도록 등록되어 있지 않습니 다.) **WinNOFCleanup**은 모든 표시에 대해 응용프로그램을 등록해제합니다. **WinNOFCleanup**은 미결 명령(동기 또는 비동기)과 함께 AP_CANCELLED 오류를 반환합니다. 그러나 명령은 노드 내에서 완료됩니다.

반드시 **WinNOFStartup** 및 **WinNOFCleanup**을 사용해야 할 필요는 없습니다. 단, 응용프로그램은 일관성 있게 이러한 호출을 사용해야 합니다. 사용자는 이들 둘다를 사용하거나 혹은 둘다를 사용하지 말아야 합니다.

주: **WinNOFStartup()**도 참조하십시오.

WinNOFStartup()

이 함수는 응용프로그램이 필요한 노드 연산자 기능 API의 버전을 지정하고 제품이 지원하는 API 버전을 검색할 수 있도록 합니다. 이 함수는 응용프로그램이 자신을 등록하기 위해 다른 노드 연산자 기능 API 호출을 발행하기 전에 호출할 수 있습니다.

구문

```
int WINAPI WinNOFStartup(WORD wVersionRequired,
                         LPWNOFDATA nofdata);
```

매개변수

설명

wVersionRequired

필요한 노드 연산자 기능 API 지원의 버전을 지정합니다. 상위(high-order) 바이트는 부 버전(개정) 번호를 지정하며, 하위(low-order) 바이트는 주 버전 번호를 지정합니다.

nofdata

노드 연산자 기능 API의 버전과 API 구현의 설명을 반환합니다.

리턴 값

리턴 값은 응용프로그램이 등록되었는지의 여부와 노드 연산자 기능 API 구현이 지정된 버전 번호를 지원할 수 있는지의 여부를 지정합니다. 리턴 값이 0이면, 응용프로그램이 등록되었으며 지정된 버전을 지원할 수 있습니다. 리턴 값이 0이 아닐 경우, 가능한 값은 다음과 같습니다.

WNOFSYSERROR

기본 네트워크 부속시스템이 네트워크 통신을 위한 준비가 되어 있지 않습니다.

WNOFVERNOTSUPPORTED

이러한 특정한 구현이 필요한 노드 연산자 기능 API 지원 버전을 제공하지 않습니다.

WNOFBADPOINTER

틀린 nofdata 매개변수.

사용시 주의사항

이 호출은 앞으로 나오게 될 노드 연산자 기능 API 릴리스의 호환성을 돕기 위한 것입니다. 현재 버전은 1.0입니다.

반드시 **WinNOFStartup** 및 **WinNOFCleanup**을 사용해야 할 필요는 없습니다. 단, 응용프로그램은 일관성 있게 이러한 호출을 사용해야 합니다. 사용자는 이들 둘다를 사용하거나 혹은 둘다를 사용하지 말아야 합니다.

WinNOFStartup()

주: **WinNOFCleanup()**도 참조하십시오.

WinNOFRegisterIndicationSink()

이 함수에 의해 응용프로그램은 요청되지 않은 표시를 수신하도록 등록할 수 있습니다.

구문

```
BOOL WINAPI WinNOFRegisterIndicationSink(unsigned short indication_opcode,
                                         unsigned short queue_size,
                                          unsigned short *primary_rc,
                                          unsigned long *secondary_rc);
```

매개변수

설명

indication_opcode

등록할 표시.

queue_size

대기행렬화할 수신되지 않은 표시 수. 0은 현재 값의 사용을 의미합니다.(초기 생략시 값은 10으로 설정되어 있습니다.) 응용프로그램이 등록한 모든 표시에 대해 한 개의 대기행렬만이 사용됩니다.

primary_rc

반환됨: 1차 리턴 코드

secondary_rc

반환됨: 2차 리턴 코드

리턴 값

이 함수는 등록되었는지의 여부를 지시하는 값을 반환합니다. 값이 0이 아니면 등록된 것입니다. 값이 0이면, 등록에 실패한 것입니다.

사용시 주의사항

indication_opcode 유형의 요청되지 않은 표시를 수신하도록 등록하려면 **WinNOFRegisterIndicationSink**를 사용하십시오.

응용프로그램은 수신하려는 각 표시 유형에 대해 **WinNOFRegisterIndicationSink**를 발행해야 합니다.

주: **WinNOFUnregisterIndicationSink** 및 **WinNOFGetIndication**도 참조하십시오.

WinNOFUnregisterIndicationSink()

이 함수는 응용프로그램이 요청되지 않은 표시의 수신을 중단할 수 있도록 합니다.

구문

```
BOOL WINAPI WinNOFUnregisterIndicationSink(unsigned short indication_opcode,
                                           unsigned short *primary_rc,
                                           unsigned long *secondary_rc);
```

매개변수

설명

indication_opcode
등록해제할 표시.

primary_rc
반환됨: 1차 리턴 코드.

secondary_rc
반환됨: 2차 리턴 코드.

리턴 값

이 함수는 등록이 해제되었는지의 여부를 지시하는 값을 제공합니다. 값이 0이 아니면 등록이 해제된 것입니다. 값이 0이면 등록이 해제되지 않은 것입니다.

사용시 주의사항

indication_opcode 유형의 요청되지 않은 표시를 더이상 수신하지 않으려면 **WinNOFUnregisterIndicationSink**를 사용하십시오.

응용프로그램은 수신을 중단하려는 각 표시 유형에 대해 **WinNOFUnregisterIndicationSink**를 발행해야 합니다.

주: **WinNOFRegisterIndicationSink** 및 **WinNOFGetIndication**도 참조하십시오.

WinNOFGetIndication()

이 함수는 응용프로그램이 요청되지 않은 표시를 수신할 수 있도록 합니다.

구문

```
int WINAPI WinNOFGetIndication(long buffer,
                               unsigned short *buffer_size,
                               unsigned long timeout);
```

매개변수

설명

buffer 표시를 수신하는 버퍼의 포인터.

buffer_size
버퍼 크기. 반환됨: 표시 크기.

timeout
밀리초 단위의 표시 대기 시간.

리턴 값

이 함수는 표시가 수신되었는지의 여부를 지시하는 값을 반환합니다.

0 표시가 반환되었습니다.

WNOFTIMEOUT
표시 대기 시간종료.

WNOFSYSNOTREADY
기본 네트워크 부속시스템이 네트워크 통신을 위한 준비가 되어 있지 않습니다.

WNOFNOTREG
응용프로그램이 표시를 수신하도록 등록되어 있지 않습니다.

WNOFBADSIZE
버퍼가 너무 작아 표시를 수신할 수 없습니다. 충분한 크기의 버퍼를 지정하여 **WinNOFGetIndication** 호출을 다시 발행하십시오. 표시 크기는 **buffer_size** 매개변수에서 반환됩니다.

WNOFBADPOINTER
버퍼 또는 **buffer_size** 매개변수가 유효하지 않습니다.

WNOFSYSERROR
예기치 못한 시스템 오류가 발생했습니다.

사용시 주의사항

이것이 방해 호출일 경우에는 다음 중 한가지 조건에서 반환됩니다.

- 표시가 반환됩니다
- 시간종료가 만기됩니다

WinNOFGetIndication()

- 응용프로그램이 WinNOFCleanup 호출을 발행합니다
- 제품이 종료됩니다
- 시스템 오류가 발생합니다

주: **WinNOFRegisterIndicationSink** 및 **WinNOFUnregisterIndicationSink**도 참조하십시오.

제4장 노드 구성 명령

다음 명령은 노드 구성 정보를 정의하고 삭제하는 데 사용됩니다.

DEFINE_ADJACENT_NODE

DEFINE_ADJACENT_NODE는 노드 디렉토리 데이터베이스의 인접 노드에 있는 자원에 대한 입력항목을 추가합니다.

주: 이 명령은 필수 명령이 아니며, CP-CP 세션을 사용하는 인접 노드에 대한 활동중인 경로가 있을 경우에는 발행해서는 안됩니다.

이 명령은 끝 노드에서 발행할 수 있으며, 이 경우 노드의 제어점(CP)이 디렉토리의 루트에 추가됩니다.

노드의 제어점(CP) LU를 정의하려면 다음 필드를 설정하십시오.

- **cp_name** 필드에 노드의 제어점(CP) 명을 지정하십시오.
- **fqlu_name** 필드에 제어점(CP) 명을 지정하여 ADJACENT_NODE_LU 구조를 추가하십시오.

노드 상의 추가 LU가 노드 제어점(CP)의 자(child)로 디렉토리에 추가됩니다. DEFINE_ADJACENT_NODE를 사용하여 기존의 노드 정의에 LU 정의를 추가할 수도 있습니다. DELETE_ADJACENT_NODE 명령을 발행하여 동일한 방법으로 LU를 제거할 수 있습니다. 이 명령이 처리 도중에 실패하면, 모든 새로운 디렉토리 입력항목이 제거되며 디렉토리는 명령이 발행되기 이전의 상태로 됩니다.

VCB 구조

DEFINE_ADJACENT_NODE 명령에는 ADJACENT_NODE_LU 중복 가변 수가 포함됩니다. ADJACENT_NODE_LU 구조는 DEFINE_ADJACENT_NODE 구조 끝에 연결됩니다.

```
typedef struct define_adjacent_node
{
        unsigned short  opcode;              /* verb operation code     */
        unsigned char   reserv2;             /* reserved                */
        unsigned char   format;              /* format                  */
        unsigned short  primary_rc;          /* primary return code     */
        unsigned long   secondary_rc;        /* secondary return code   */
        unsigned char   cp_name[17];         /* CP name                 */
        unsigned char   description[RD_LEN]; /* resource description    */
        unsigned char   reserv3[19];         /* reserved                */
        unsigned short  num_of_lus;          /* number of LUs           */
} DEFINE_ADJACENT_NODE;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;         /* wildcard LU name        */
                                             /* indicator               */
        unsigned char   fqlu_name[17];       /* fully qualified LU name */
        unsigned char   reserv1[6];          /* reserved                */
} ADJACENT_NODE_LU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_ADJACENT_NODE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

cp_name

인접 끝 노드에 있는 제어점(CP)의 전체 정식명. 노드가 자체 XID(이를 지원할 경우)로 전송하는 이름과 일치해야 하며, 또한 노드로의 링크를 위해 DEFINE_LS에 지정한 인접 제어점(CP) 명과도 일치해야 합니다. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

description

자원 설명(QUERY_DIRECTORY_LU에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트(nonzero) 문자열입니다. 16바이트 모두 유효 바이트입니다.

num_of_lus

DEFINE_ADJACENT_NODE VCB 뒤에 오는 인접 LU 오버레이 수.

adjacent_node_lu.wildcard_lu

지정한 LU 명이 총칭 문자인지의 여부를 지시합니다(AP_YES 또는 AP_NO).

adjacent_node_lu.fqlu_name

정의할 LU 명. 이 이름이 전체 정식명이 아닐 경우, CP 명의 네트워크 ID가 사용됩니다. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 하나의 A 유형 EBCDIC 문자열로 구성되거나 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다).

wildcard_lu가 TRUE일 때 ″.″ 뒤에 EBCDIC 공백이 오면, 이는 전체 총칭 문자(무엇과도 일치하는)를 의미합니다. 모든 EBCDIC 공백은 CP 명의 Net id로 시작하는 모든 것과 일치합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME
AP_INVALID_WILDCARD_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

secondary_rc
AP_MEMORY_SHORTAGE

AP_DIRECTORY_FULL

DEFINE_CN

DEFINE_CN은 연결 네트워크(가상 경로지정 노드 또는 VRN이라고도 함)를 정의합니다. 이 명령은 연결 네트워크의 네트워크 정식명과 전송 그룹(TG) 특성을 제공합니다. 또한 이 연결 네트워크에 액세스할 수 있는 국지 포트의 이름 리스트도 제공합니다.

DEFINE_CN은 기존의 연결 네트워크를 다시 정의하는 데 사용할 수 있습니다. 특히, 다른 DEFINE_CN을 발행하여 연결 네트워크에 액세스하는 포트 리스트에 새로운 포트를 추가할 수 있습니다.(DELETE_CN 명령을 발행하여 동일한 방법으로 포트를 제거할 수 있습니다.)

VCB 구조

```
typedef struct define_cn
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   fqcn_name[17];  /* name of connection network */
        CN_DEF_DATA     def_data;       /* CN defined data         */
        unsigned char   port_name[8] [8];
                                        /* port names              */
} DEFINE_CN;

typedef struct cn_def_data
{
        unsigned char   description[RD_LEN];
                                         /* resource description       */
        unsigned char   num_ports;       /* number of ports on CN      */
        unsigned char   reserv1[16];     /* reserved                   */
        TG_DEFINED_CHARS tg_chars;       /* TG characteristics         */
} CN_DEF_DATA;

typedef struct tg_defined_chars
{
        unsigned char   effect_cap;         /* effective capacity             */
        unsigned char   reserve1[5];        /* reserved                       */
        unsigned char   connect_cost;       /* connection cost                */
        unsigned char   byte_cost;          /* byte cost                      */
        unsigned char   reserve2;           /* reserved                       */
        unsigned char   security;           /* security                       */
        unsigned char   prop_delay;         /* propagation delay              */
        unsigned char   modem_class;        /* modem class                    */
        unsigned char   user_def_parm_1;    /* user-defined parameter 1       */
        unsigned char   user_def_parm_2;    /* user-defined parameter 2       */
        unsigned char   user_def_parm_3;    /* user-defined parameter 3       */
} TG_DEFINED_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_CN

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

fqcn_name
정의할 연결 네트워크의 전체 정식명(17바이트 길이). 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

def_data.description
자원 설명(QUERY_CN에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.num_ports
이 연결 네트워크와 연관된 포트 수. 각 DEFINE_CN 명령당 최대 8개의 포트를 사용할 수 있으며, 각 CN에 총 239개의 포트를 사용할 수 있습니다.

def_data.tg_chars.effect_cap
효율성 실제 단위. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 각 효율성 단위는 초당 300비트와 동일합니다.

def_data.tg_chars.connect_cost
연결 시간당 비용. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 연결 시간당 최저 비용이고 255는 최고 비용입니다.

def_data.tg_chars.byte_cost
바이트당 비용. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 바이트당 최저 비용이고 255는 최고 비용입니다.

def_data.tg_chars.security
아래 리스트에 설명된 보안 값.

AP_SEC_NONSECURE
보안이 없습니다.

AP_SEC_PUBLIC_SWITCHED_NETWORK
이 연결 네트워크를 통해 전송되는 데이터가 공중 교환 통신망(PSN)을 통해 흐릅니다.

AP_SEC_UNDERGROUND_CABLE
데이터가 안전한 지하 케이블을 통해 전송됩니다.

AP_SEC_SECURE_CONDUIT
회선이 보호되지 않는 안전한 배관(conduit)입니다.

AP_SEC_GUARDED_CONDUIT
배관이 물리적 탭핑으로부터 보호됩니다.

AP_SEC_ENCRYPTED
회선 상에서 암호화.

AP_SEC_GUARDED_RADIATION
회선이 물리적 또는 복사 탭핑으로부터 보호됩니다.

def_data.tg_chars.prop_delay
신호가 링크를 지나는 데 걸리는 시간을 나타내는 마이크로초 단위의 전달 지연. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 생략시 값은 다음과 같습니다.

AP_PROP_DELAY_MINIMUM
전달 지연이 없음.

AP_PROP_DELAY_LAN
480 마이크로초 미만 지연.

AP_PROP_DELAY_TELEPHONE
480과 49 512 마이크로초 사이의 지연.

AP_PROP_DELAY_PKT_SWITCHED_NET
49 512와 245 760 마이크로초 사이의 지연.

AP_PROP_DELAY_SATELLITE
245 760 마이크로초 초과 지연.

AP_PROP_DELAY_MAXIMUM
최대 전달 지연.

def_data.tg_chars.modem_class
예약됨. 이 필드는 항상 0으로 설정해야 합니다.

def_data.tg_chars.user_def_parm_1
0—255 범위 내의 사용자 정의 매개변수.

def_data.tg_chars.user_def_parm_2
0—255 범위 내의 사용자 정의 매개변수.

def_data.tg_chars.user_def_parm_3
0—255 범위 내의 사용자 정의 매개변수.

port_name
연결 네트워크에 정의되는 최대 8개의 포트명 배열. 명명된 각 포트는 DEFINE_PORT 명령으로 이미 정의되어 있어야 합니다. 각 포트명은 국지로 표시가능한 문자 세트로 된 8바이트 문자열로, 연관된 DEFINE_PORT 명령에 정의된 것과 일치해야 합니다. 새로운 포트명을 지정하여 다른 DEFINE_CN을 발행함으로써 연결 네트워크에 추가 포트를 정의할 수 있습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CN_NAME

AP_INVALID_NUM_PORTS_SPECIFIED
AP_INVALID_PORT_NAME
AP_INVALID_PORT_TYPE
AP_DEF_LINK_INVALID_SECURITY
AP_EXCEEDS_MAX_ALLOWED

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PORT_ACTIVE

AP_CANT_MODIFY_VISIBILITY

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_COS

DEFINE_COS는 서비스 클래스(COS) 정의를 추가합니다. DEFINE_COS는 또한 이전에 정의된 COS의 필드를 수정하는 데에도 사용할 수 있습니다.

정의는 노드 및 TG 『행』을 제공합니다. 이들 행은 특정 범위의 노드 및 TG 특성을 경로 계산에 사용되는 가중치와 연관시킵니다. 가중치가 낮을수록 보다 나은 경로를 얻을 수 있습니다.

VCB 구조

DEFINE_COS 명령에는 **cos_tg_row** 및 **cos_node_row** 오버레이의 가변 수가 포함됩니다. **cos_tg_row** 구조는 DEFINE_COS의 끝에 연결되며(가중치를 기준으로 오름차순으로), 그 다음에 **cos_node_row** 구조가 연결됩니다(역시 가중치를 기준으로 오름차순으로).

```
typedef struct define_cos
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   cos_name[8];    /* class-of-service name   */
        unsigned char   description[RD_LEN];
                                        /* resource description        */
        unsigned char   transmission_priority;
                                        /* transmission priority       */
        unsigned char   reserv3[9];      /* reserved                   */
        unsigned char   num_of_node_rows; /* number of node rows       */
        unsigned char   num_of_tg_rows;   /* number of TG rows         */
} DEFINE_COS;

typedef struct cos_node_row
{
        COS_NODE_STATUS minimum;          /* minimum                   */
        COS_NODE_STATUS maximum;          /* max                       */
        unsigned char    weight;          /* weight                    */
        unsigned char   reserv1;          /* reserved                  */
} COS_NODE_ROW;

typedef struct cos_node_status
{
        unsigned char   rar;              /* route additional resistance */
        unsigned char   status;           /* node status.                */
        unsigned char   reserv1[2];       /* reserved                    */
} COS_NODE_STATUS;

typedef struct cos_tg_row
{
        TG_DEFINED_CHARS minimum;         /* minimum                   */
        TG_DEFINED_CHARS maximum;         /* maximum                   */
        unsigned char    weight;          /* weight                    */
        unsigned char   reserv1;          /* reserved                  */
} COS_TG_ROW;

typedef struct tg_defined_chars
{
        unsigned char   effect_cap;         /* effective capacity            */
        unsigned char   reserve1[5];    /* reserved                          */
        unsigned char   connect_cost;       /* cost per connect time         */
        unsigned char   byte_cost;          /* cost per byte                 */
        unsigned char   reserve2;           /* reserved                      */
        unsigned char   security;           /* security                      */
```

```
        unsigned char   prop_delay;         /* propagation delay             */
        unsigned char   modem_class;        /* modem class                   */
        unsigned char   user_def_parm_1;    /* user-defined parameter 1      */
        unsigned char   user_def_parm_2;    /* user-defined parameter 2      */
        unsigned char   user_def_parm_3;    /* user-defined parameter 3      */
} TG_DEFINED_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_COS

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

cos_name

서비스 클래스(COS) 명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

description

자원 설명(QUERY_COS에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

transmission_priority

전송 우선순위. 다음 값 중 하나로 설정됩니다.

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

num_of_node_rows

DEFINE_COS VCB 뒤에 오는 노드 행 오버레이 수. 최대값은 8입니다. 각 노드 행에는 최소 노드 특성 집합, 최대 노드 특성 집합이나 가중치가 들어 있습니다. 노드의 가중치를 계산할 때, 노드의 특성이 각 노드 행에 정의된 최소 및 최대 특성과 비교됩니다. 그런 후 첫번째 노드 행의 가중치가 노드에 지정되며, 이것이 노드의 모든 특성을 지정된 한계값 이내로 제한합니다. 노드 특성이 나열된 노드 행을 하나도 충족시키지 않을 경우에는, 노드가 이 COS에 부적합한 것으로 간주되며 무한 가중치가 지정됩니다. 노드 행은 가중치를 기준으로 하여 오름차순으로 연결해야 함에 주의하십시오.

num_of_tg_rows

노드 행 중복 뒤에 오는 TG 행 오버레이 수. 최대값은 8입니다. 각 TG 행에는 최소 TG 특성 집합, 최대 TG 특성 집합이나 가중치가 들어 있습니다. TG의 가중치를 계산할 때, TG의 특성이 각 TG 행에 정의된 최소 및 최대 특성과 비교됩니다. 그런 후 첫번째 TG 행의 가중치가 TG에 지정되며, 이것이 TG의 모든 특성을 지정된 한

계값 이내로 제한합니다. TG 특성이 나열된 TG 행을 하나도 충족시키지 않을 경우에는, TG가 이 COS에 부적합한 것으로 간주되며 무한 가중치가 지정됩니다. TG 행은 가중치를 기준으로 하여 오름차순으로 연결해야 함에 주의하십시오.

cos_node_row.minimum.rar
경로 추가 저항 최소값. 값 범위는 0—255이어야 합니다.

cos_node_row.minimum.status
노드의 최소 과잉 상태를 지정합니다. 다음 값 중 하나가 될 수 있습니다.

AP_UNCONGESTED
노드 과잉 상태가 아닙니다.

AP_CONGESTED
ISR 세션 수가 **isr_sessions_upper_threshold**보다 큽니다.

cos_node_row.maximum.rar
경로 추가 저항 최대값. 값 범위는 0—255이어야 합니다.

cos_node_row.maximum.status
노드의 최대 과잉 상태를 지정합니다. 다음 값 중 하나가 될 수 있습니다.

AP_UNCONGESTED
노드 과잉 상태가 아닙니다.

AP_CONGESTED
ISR 세션 수가 **isr_sessions_upper_threshold**보다 큽니다.

cos_node_row.weight
노드 행과 연관된 가중치. 값 범위는 0—255이어야 합니다.

cos_tg_row.minimum.effect_cap
효율성 실제 단위의 최소 한계. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 각 효율성 단위는 초당 300비트와 동일합니다.

cos_tg_row.minimum.connect_cost
연결 시간당 비용의 최소 한계. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 연결 시간당 최저 비용이고 255는 최고 비용입니다.

cos_tg_row.minimum.byte_cost
연결 시간당 비용의 최대 한계. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 바이트당 최저 비용이고 255는 최고 비용입니다.

cos_tg_row.minimum.security
다음 리스트에 설명된 보안 값의 최소 한계.

AP_SEC_NONSECURE
보안이 없습니다.

AP_SEC_PUBLIC_SWITCHED_NETWORK
이 연결 네트워크를 통해 전송되는 데이터가 공중 교환 통신망(PSN)을 통해 흐릅니다.

AP_SEC_UNDERGROUND_CABLE
데이터가 안전한 지하 케이블을 통해 전송됩니다.

AP_SEC_SECURE_CONDUIT
회선이 보호되지 않는 안전한 배관(conduit)입니다.

AP_SEC_GUARDED_CONDUIT
배관(conduit)이 물리적 탭핑으로부터 보호됩니다.

AP_SEC_ENCRYPTED
회선 상에서 암호화.

AP_SEC_GUARDED_RADIATION
회선이 물리적 또는 복사 탭핑으로부터 보호됩니다.

cos_tg_row.minimum.prop_delay
신호가 링크를 지나는 데 걸리는 시간을 나타내는 마이크로초 단위의 전달 지연 최소 한계. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 생략시 값은 다음과 같습니다.

AP_PROP_DELAY_MINIMUM
전달 지연이 없음.

AP_PROP_DELAY_LAN
480 마이크로초 미만 지연.

AP_PROP_DELAY_TELEPHONE
480과 49 512 마이크로초 사이의 지연.

AP_PROP_DELAY_PKT_SWITCHED_NET
49 512와 245 760 마이크로초 사이의 지연.

AP_PROP_DELAY_SATELLITE
245 760 마이크로초 초과 지연.

AP_PROP_DELAY_MAXIMUM
최대 전달 지연.

cos_tg_row.minimum.modem_class
예약됨. 이 필드는 항상 0으로 설정해야 합니다.

cos_tg_row.minimum.user_def_parm_1
0—255 범위 내의 사용자 정의 매개변수 최소 한계.

cos_tg_row.minimum.user_def_parm_2
0—255 범위 내의 사용자 정의 매개변수 최소 한계.

cos_tg_row.minimum.user_def_parm_3
0—255 범위 내의 사용자 정의 매개변수 최소 한계.

cos_tg_row.maximum.effect_cap
효율성 실제 단위의 최대 한계. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 각 효율성 단위는 초당 300비트와 동일합니다.

cos_tg_row.maximum.connect_cost
연결 시간당 비용의 최대 한계. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 연결 시간당 최저 비용이고 255는 최고 비용입니다.

cos_tg_row.maximum.byte_cost
바이트당 비용의 최대 한계. 유효한 값은 0—255 범위 내의 정수 값이며, 여기서 0은 바이트당 최저 비용이고 255는 최고 비용입니다.

cos_tg_row.maximum.security
다음 리스트에 설명된 보안 값의 최대 한계.

AP_SEC_NONSECURE
보안이 없습니다.

AP_SEC_PUBLIC_SWITCHED_NETWORK
이 연결 네트워크를 통해 전송되는 데이터가 공중 교환 통신망(PSN)을 통해 흐릅니다.

AP_SEC_UNDERGROUND_CABLE
데이터가 안전한 지하 케이블을 통해 전송됩니다.

AP_SEC_SECURE_CONDUIT
회선이 보호되지 않는 안전한 배관(conduit)입니다.

AP_SEC_GUARDED_CONDUIT
배관이 물리적 탭핑으로부터 보호됩니다.

AP_SEC_ENCRYPTED
회선 상에서 암호화.

AP_SEC_GUARDED_RADIATION
회선이 물리적 또는 복사 탭핑으로부터 보호됩니다.

cos_tg_row.maximum.prop_delay
신호가 링크를 지나는 데 걸리는 시간을 나타내는 마이크로초 단위의 전달 지연 최대 한계. 이 값은 0.1mmm * 2 eeeee 공식으로 표시되는 1바이트 부동 소수점 숫자로 인코딩되며, 여기서 바이트의 비트 표시는 eeeeemmm입니다. 생략시 값은 다음과 같습니다.

AP_PROP_DELAY_MINIMUM
전달 지연이 없음.

AP_PROP_DELAY_LAN
480 마이크로초 미만 지연.

AP_PROP_DELAY_TELEPHONE
480과 49 512 마이크로초 사이의 지연.

AP_PROP_DELAY_PKT_SWITCHED_NET
49 512와 245 760 마이크로초 사이의 지연.

AP_PROP_DELAY_SATELLITE
245 760 마이크로초 초과 지연.

AP_PROP_DELAY_MAXIMUM
최대 전달 지연.

cos_tg_row.maximum.modem_class
예약됨. 이 필드는 항상 0으로 설정해야 합니다.

cos_tg_row.maximum.user_def_parm_1
0—255 범위 내의 사용자 정의 매개변수 최대 한계.

cos_tg_row.maximum.user_def_parm_2
0—255 범위 내의 사용자 정의 매개변수 최대 한계.

cos_tg_row.maximum.user_def_parm_3
0—255 범위 내의 사용자 정의 매개변수 최대 한계.

cos_tg_row.weight
TG 행과 연관된 가중치.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_COS_NAME

AP_INVALID_NUMBER_OF_NODE_ROWS
AP_INVALID_NUMBER_OF_TG_ROWS
AP_NODE_ROW_WGT_LESS_THAN_LAST
AP_TG_ROW_WGT_LESS_THAN_LAST

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_COS_TABLE_FULL

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DEFAULTS

DEFINE_DEFAULTS는 사용자가 노드의 생략시 조치를 정의하거나 재정의할 수 있도록 합니다.

VCB 구조

```
typedef struct define_defaults
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        DEFAULT_CHARS   default_chars;  /* default information     */
} DEFINE_DEFAULTS;

typedef struct default_chars
{
        unsigned char   description[RD_LEN];
                                        /* resource description  */
        unsigned char   mode_name[8];   /* default mode name     */
        unsigned char   implicit_plu_forbidden;
                                        /* disallow implicit     */
                                        /*  PLUs?                */
        unsigned char   specific_security_codes;
                                        /* generiuc security     */
                                        /* sense codes           */
        unsigned short  limited_timeout;/* timeout for limited   */
                                        /* sessions              */
        unsigned char   reserv[244];    /* reserved              */
} DEFAULT_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DEFAULTS

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

default_chars.description

자원 설명(QUERY_DEFAULTS에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

default_chars.mode_name

생략시로 기능하는 모드의 이름. 이것은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

default_chars.implicit_plu_forbidden
프로그램이 알 수 없는 상대방 LU에 대한 암시적 정의를 제위치에 놓는지의 여부를 제어합니다(AP_YES 또는 AP_NO).

default_chars.specific_security_codes
프로그램이 보안 인증 또는 인증 장애시 특정 감지 코드를 사용하는지의 여부를 제어합니다(AP_YES 또는 AP_NO). 특정한 감지 코드는 해당 세션에서 그에 대한 지원을 보고한 상대방 LU에만 반환됩니다.

default_chars.limited_timeout
free limited-resource conwinner 세션이 활성종료될 때까지의 시간종료를 지정합니다. 범위는 0-65535초입니다.

반환되는 매개변수

명령이 실행되면 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

이 명령이 유효하지 않거나(예를 들면, EBCDIC A가 아님) 또는 유효하기는 하나 정의되어 있지 않은 생략시 모드를 지정할 경우, 프로그램 은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MODE_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

각 필드를 재정의했을 때의 결과는 다음과 같습니다.

description
재정의가 즉시 효력을 가집니다. 갱신된 설명은 뒤에 오는 QUERY_DEFAULT 명령에서 반환됩니다.

mode_name

재정의 결과는 뒤에 오는 모든 암시적 모드 정의에 적용됩니다. 예를 들면, 갱신된 모드가 생략시 모드로 기능합니다. 이전의 알 수 없는 모드(예를 들면, 이전의 생략시 모드 특성을 계승했던 모드)에 대한 재정의 결과는 알 수 없는 모드의 재정의와 일치합니다. 예를 들어, 생략시 모드가 #INTER이고 프로그램이(알 수 없는) MODE1에 대해 bIND를 수신할 경우, 차후에 #BATCH로 재정의되는 생략시 모드의 MODE1에 대한 결과는 #BATCH의 모드 특성을 지정하는 DEFINE_MODE(MODE1)의 결과와 일치합니다.

implicit_plu_forbidden

재정의가 즉시 효력을 가집니다. 이 필드의 갱신된 값에 따라 차후의 모든 암시적 PLU 정의가 성공하거나 실패합니다.

specific_security_codes

재정의가 즉시 효력을 가집니다.

limited_timeout

갱신된 값이 재정의 이후에 설정된 모든 새로운 세션에 사용됩니다. 이전 값은 기존 세션에 사용됩니다.

DEFINE_DEFAULT_PU

DEFINE_DEFAULT_PU는 사용자가 생략시 PU의 필드를 정의, 재정의하거나 수정할 수 있도록 합니다. 또한 사용자가 널(null) PU 명을 지정하여 디폴트 PU를 삭제할 수 있도록 합니다. TRANSFER_MS_DATA 명령에 PU 명을 명시적으로 지정하지 않으면, TRANSFER_MS_DATA에서 전송되는 관리 서비스 정보가 생략시 PU와 호스트 SSCP간의 세션에서 전송됩니다. 자세한 내용은 697페이지의 『제15장 관리 서비스 명령』을 참조하십시오.

VCB 구조

```
typedef struct define_default_pu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   pu_name[8];     /* PU name                 */
        unsigned char   description[RD_LEN];
                                        /* resource description    */
        unsigned char   reserv3[16];    /* reserved                */
} DEFINE_DEFAULT_PU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DEFAULT_PU

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pu_name

생략시로 기능하는 국지 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

description

자원 설명(QUERY_DEFAULT_PU에서 반환). 이 이름은 국지로 표시 가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DLC

DEFINE_DLC는 새로운 DLC를 정의하거나 기존의 DLC를 수정합니다. 이 명령은 노드 전반에 걸쳐 고유한 DLC명과 기본 구조에 연결되는 일부 DLC 고유 데이터를 정의합니다. 이 데이터는 DLC 초기화시에 사용되며, 형식은 DLC 유형(예: 토큰링)에 고유한 형식입니다. DEFINE_DLC 명령으로는 이 명령에 추가되는 DLC 고유 데이터만을 수정할 수 있습니다. 이를 수행하려면, DLC가 재설정 상태가 되도록 먼저 STOP_DLC 명령을 발행해야 합니다.

DLC와 포트, 링크 스테이션간의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

VCB 구조

```
typedef struct define_dlc
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   dlc_name[8];    /* name of DLC             */
        DLC_DEF_DATA    def_data;       /* DLC defined data        */
} DEFINE_DLC;

typedef struct dlc_def_data
{
        DESCRIPTION     description;       /* resource description        */
        unsigned char   dlc_type;          /* DLC type                    */
        unsigned char   neg_ls_supp;       /* negotiable LS support       */
        unsigned char   port_types;        /* allowable port types        */
        unsigned char   hpr_only;          /* DLC only supports HPR links:*/
        unsigned char   reserv3;           /* reserved                    */
        unsigned char   retry_flags;       /* conditions for automatic    */
                                           /* retries                     */
        unsigned short  max_activation_attempts;
                                           /* how many automatic retries?   */
        unsigned short  activation_delay_timer;
                                           /* delay between automatic       */
                                           /* retries                       */
        unsigned char   reserv4[4];        /* reserved                      */
        unsigned short  dlc_spec_data_len; /* Length of DLC specific data */
} DLC_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DLC

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dlc_name
DLC의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. OEM 디바이스인 경우, 이 이름은 생산업체에서 지정하는 고유 명입니다. 유효한 값은 LAN, SDLC, AnyNet, X25 또는 TWINAX입니다.(8자가 되도록 공백으로 채워집니다.)

def_data.description
자원 설명(QUERY_DLC에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.dlc_type
DLC 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

def_data.neg_ls_supp
DLC가 조정가능한 링크 스테이션을 지원하는지의 여부를 지정합니다(AP_YES 또는 AP_NO). **dlc_type**이 AP_TWINAX이면, AP_NO만 지원됩니다. **dlc_type**이 AP_ANYNET이면, AP_YES만 지원됩니다.

def_data.port_types
제공되는 **dlc_type**에 허용되는 포트 유형을 지정합니다. 이 값은 다음 값 중 하나 이상이 될 수 있습니다(둘 이상일 경우, OR로 연결).

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

해당하는 DLC 유형의 필드를 설정하려면 다음 표를 사용하십시오.

표 2. DLC 유형별 포트 유형

DLC 유형	포트 유형
AP_ANYNET	AP_PORT_SATF
AP_LLC2	AP_PORT_SATF
AP_OEM_DLC	AP_PORT_SWITCHED 또는 AP_PORT_NONSWITCHED
AP_SDLC	AP_PORT_SWITCHED 또는 AP_PORT_NONSWITCHED
AP_TWINAX	AP_PORT_NONSWITCHED
AP_X25	AP_PORT_SWITCHED 또는 AP_PORT_NONSWITCHED

def_data.max_activation_attempts

이 필드는 DLC가 HPR 링크만을 지원하는지의 여부를 지정합니다. IP 링크 상의 HPR에 대해서는 이 필드를 AP_YES로 설정해야 합니다.

AP_YES
AP_NO

def_data.retry_flags

이 필드는 링크 스테이션이 자동으로 재시도되는 조건을 지정합니다. 비트 필드이며 다음 값을 취할 수 있습니다(OR에 의해 비트와이즈로 연결).

AP_RETRY_ON_START

활성화를 시도할 때 원격 노드로부터 응답이 수신되지 않을 경우 링크 활성화를 재시도합니다. 활성화를 시도할 때 기본 포트가 활동중이 아닌 경우에는 프로그램이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_FAILURE

링크가 활동중에 또는 활성화 미정 상태에서 실패할 경우, 링크 활성화가 재시도됩니다. 활성화를 시도할 때 기본 포트가 실패한 경우, 프로그램이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_DISCONNECT

링크가 원격 노드에 의해 정상적으로 종료될 경우 링크 활성화를 재시도합니다.

AP_DELAY_APPLICATION_RETRIES

응용프로그램에 의해 시작되는(START_LS 또는 요구가 있을 때 링크 활성화에 의해) 링크 활성화 재시도는 **activation_delay_timer**로 페이싱됩니다.

AP_INHERIT_RETRY

이 플래그는 아무런 영향을 미치지 않습니다.

def_data.max_activation_attempts

DEFINE_LS의 **def_data.retry_flags**에 하나 이상의 플래그를 설정하지

않고 DEFINE_LS의 **def_data.max_activation_attempts**를 AP_USE_DEFAULTS로 설정하지 않으며 DEFINE_PORT의 **def_data.max_activation_attempts**를 AP_USE_DEFAULTS로 설정하지 않는 한, 이 필드는 아무런 영향을 미치지 않습니다.

이 필드는 원격 노드가 응답하지 않거나 기본 포트가 활동중이 아닐 경우 프로그램이 허용하는 재시도 수를 지정합니다. 여기에는 자동 재시도 수 및 응용프로그램에 의한 활성화 시도 수 둘다 포함됩니다.

이 한계값에 달하면, 자동으로 재시도하려는 시도가 더이상 행해지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는 OPEN_LU_SSCP_SEC_RQ는 활성화 시도가 한 번만 행해지도록 하며, 활성화에 실패하더라도 재시도가 행해지지 않습니다.

0은 '제한 없음'을 의미합 님 님 님 님 님

AP_INVALID_DLC_TYPE
AP_INVALID_RETRY_FLAGS
AP_INVALID_PORT_TYPE
AP_HPR_NOT_SUPPORTED

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_DLC_ACTIVE

AP_INVALID_DLC_TYPE
AP_CANT_MODIFY_VISIBILITY

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DLUR_DEFAULTS

DEFINE_DLUR_DEFAULTS는 사용자가 생략시 종속 LU 서버(DLUS) 및 백업 디폴트 DLUS를 정의, 재정의하거나 취소할 수 있도록 합니다. 디폴트 DLUS 명은 DLUR이 명시적으로 지정된 연관된 DLUS가 없는 PU에 대해 SSCP-PU 활성화를 시작할 때 사용합니다. DEFINE_DLUR_DEFAULTS 명령에 DLUS 명을 명시적으로 지정하지 않으면, 현재 생략시(또는 백업 DLUS)가 취소됩니다.

VCB 구조

```
typedef struct define_dlur_defaults
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   description[RD_LEN];
                                        /* resource description    */
        unsigned char   dlus_name[17];      /* DLUS name               */
        unsigned char   bkup_dlus_name[17]; /* Backup DLUS name        */
        unsigned char   reserv3;            /* reserved                */
        unsigned short  dlus_retry_timeout; /* DLUS Retry Timeout      */
        unsigned short  dlus_retry_limit;   /* DLUS Retry Limit        */
        unsigned char   reserv4[16];        /* reserved                */
} DEFINE_DLUR_DEFAULTS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DLUR_DEFAULTS

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

description

자원 설명. 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

dlus_name

생략시로서 기능하는 DLUS 노드의 이름. 모두 0으로 설정하거나 EBCDIC 점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 문자열의 오른쪽은 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다). 이 필드를 모두 0으로 설정하면, 현재 생략시 DLUS가 취소됩니다.

bkup_dlus_name

백업 생략시로 기능하는 DLUS 노드의 이름. 모두 0으로 설정하거나

혹은 EBCDIC 점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 문자열의 오른쪽은 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다). 이 필드를 모두 0으로 설정하면, 현재 백업 생략시 DLUS가 취소됩니다.

dlus_retry_timeout
DLUS에 접속하기 위한 두 번째 이후 시도들 간의 간격(초 단위). 초기 시도와 첫번째 재시도 사이의 간격은 항상 1초입니다. 0을 지정하면 생략시 값 5초가 사용됩니다.

dlus_retry_limit
DLUS 접속 초기 실패 후의 재시도 최대 수. 0을 지정하면 디폴트 값 3이 사용됩니다. X'FFFF'를 지정하면 퍼스널 통신이나 통신 서버가 무한정 재시도합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLUS_NAME

AP_INVALID_BKUP_DLUS_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DOWNSTREAM_LU

이 명령은 통신 서버에만 적용됩니다.

DEFINE_DOWNSTREAM_LU 명령은 PU 집중(concentration)에 사용됩니다. PU 집중을 사용하면, 다운스트림 LU가 업스트림 호스트와 통신할 수 있습니다. 이를 위해, 통신 서버는 각 다운스트림 LU를 *호스트 LU*라고 하는 종속 국지 LU로 매핑합니다.

DEFINE_DOWNSTREAM_LU는 새로운 다운스트림 LU를 정의하며, 기존 정의를 수정하는 데에는 사용할 수 없습니다. 다운스트림 LU는 지정된 호스트 LU(DEFINE_LU_0_TO_3 명령으로 정의)로 매핑됩니다. 호스트 LU는 또한 LU 풀이란 용어로도 지정할 수 있습니다.

기존의 다운스트림 LU 정의에 대해 DEFINE_DOWNSTREAM_LU를 발행할 때는 정의가 일치해야 합니다. 다운스트림 링크가 활동중이고 다운스트림 LU가 활동중이 아닐 경우, 이 명령은 성공적인 것으로 반환되며 재활성화 시도가 행해집니다.(이 시도는 실패할 수도 있습니다.) 다운스트림이 활동중이 아니거나 다운스트림 LU가 이미 활동중일 경우에는, 이 명령이 실패합니다. 재활성화 시도 처리는 지정된 호스트 LU의 상태에 따라 달라집니다.

- 호스트 LU가 활동중이면, ACTLU가 즉시 다운스트림 LU로 재송신됩니다.
- 호스트 LU가 활동중이 아니면, 노드는 호스트 LU가 활동 상태가 될 때까지 기다린 후 ACTLU를 다운스트림 LU로 송신합니다. 호스트가 활동중이 아닌 경우, 노드는 호스트에 대한 링크를 활성화하려고 시도합니다.(호스트 링크를 자동으로 활성화할 수 없으면 이 시도가 실패합니다.)

VCB 구조

```
typedef struct define_downstream_lu
{
        unsigned short  opcode;           /* verb operation code     */
        unsigned char   attributes;       /* verb attributes         */
        unsigned char   reserv2;          /* reserved                */
        unsigned char   format;           /* format                  */
        unsigned short  primary_rc;       /* primary return code     */
        unsigned long   secondary_rc;     /* secondary return code   */
        unsigned char   dslu_name[8];     /* Downstream LU name      */
        DOWNSTREAM_LU_DEF_DATA def_data;  /* defined data            */
} DEFINE_DOWNSTREAM_LU;

typedef struct downstream_lu_def_data
{
        unsigned char   description[RD_LEN];
                                          /* resource description       */
        unsigned char   nau_address;      /* Downstream LU NAU address  */
        unsigned char   dspu_name[8];     /* Downstream PU name         */
        unsigned char   host_lu_name[8];  /* Host LU or Pool name       */
        unsigned char   allow_timeout;    /* Allow timeout of host LU?  */
```

```
        unsigned char   delayed_logon;    /* Allow delayed logon to     */
                                          /* host LU                    */
        unsigned char   reserv2[6];       /* reserved                   */
} DOWNSTREAM_LU_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DOWNSTREAM_LU

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dslu_name

정의할 다운스트림 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.description

자원 설명(QUERY_DOWNSTREAM_LU에서 반환). 이 필드의 길이는 4의 배수(바이트)이어야 하며 0이 될 수 없습니다.

def_data.nau_address

DOWNSTREAM LU의 네트워크 지정 장치(NAU) 주소. 값 범위는 1–255이어야 합니다.

def_data.dspu_name

DOWNSTREAM PU의 이름(DEFINE_LS에 지정한). 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.host_lu_name

다운스트림 LU가 매핑되는 호스트 LU 또는 호스트 LU 풀의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.allow_timeout

호스트 LU 정의에 지정한 **timeout** 기간 동안 세션이 활동중이 아닌 상태로 있을 경우 프로그램이 이 다운스트림 LU가 사용하는 호스트 LU를 시간종료할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.delayed_logon

프로그램이 다운스트림 LU에서 첫번째 데이터가 수신될 때까지 다

운스트림 LU와 호스트 LU의 연결을 지연하는지의 여부를 지정합니다. 대신, 시뮬레이트된 로그온 화면이 다운스트림 LU에 송신됩니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DNST_LU_NAME

AP_INVALID_NAU_ADDRESS

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_INVALID_HOST_LU_NAME
AP_LU_NAME_POOL_NAME_CLASH
AP_PU_NOT_ACTIVE
AP_LU_ALREADY_ACTIVATING
AP_LU_DEACTIVATING
AP_LU_ALREADY_ACTIVE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON
AP_DELAYED_VERB_PENDING

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DOWNSTREAM_LU_RANGE

이 명령은 통신 서버에만 적용됩니다.

DEFINE_DOWNSTREAM_LU_RANGE 명령은 PU 집중에 사용됩니다. PU 집중을 사용하면, 다운스트림 LU가 업스트림 호스트와 통신할 수 있습니다. 이를 위해, 통신 서버는 각 다운스트림 LU를 *호스트 LU*라고 하는 종속 국지 LU로 매핑합니다.

DEFINE_DOWNSTREAM_LU_RANGE는 지정한 NAU 범위 내의 복수 다운스트림 LU를 정의할 수 있도록 합니다.(단, 기존의 정의를 수정하는 데에는 사용할 수 없습니다.) 노드 운영요원이 기본명과 NAU 범위를 제공합니다. LU 명은 기본명과 NAU 주소의 조합으로 생성됩니다.

예를 들어, 기본명 LUNME와 NAU 범위 1-4를 지정하면, LU가 LUNME001, LUNME002, LUNME003 및 LUNME004로 정의됩니다. non-pad 문자가 5개 미만인 기본명은 non-pad 문자가 8개 미만인 LU 명을 생성합니다. 이 경우, 통신 서버가 오른쪽을 채워서 8자가 되도록 합니다.

각 다운스트림 LU는 지정한 호스트 LU(DEFINE_LU_0_TO_3 명령으로 정의)로 매핑됩니다.

VCB 구조

```
typedef struct define_downstream_lu_range
{
        unsigned short  opcode;           /* verb operation code        */
        unsigned char   attributes;       /* verb attributes            */
        unsigned char   reserv2;          /* reserved                   */
        unsigned char   format;           /* format                     */
        unsigned short  primary_rc;       /* primary return code        */
        unsigned long   secondary_rc;     /* secondary return code      */
        unsigned char   dslu_base_name[5];/* Downstream LU base name    */
        unsigned char   description[RD_LEN];
                                          /* resource description       */
        unsigned char   min_nau;          /* min NAU address in range   */
        unsigned char   max_nau;          /* max NAU address in range   */
        unsigned char   dspu_name[8];     /* Downstream PU name         */
        unsigned char   host_lu_name[8];  /* Host LU or pool name       */
        unsigned char   allow_timeout;    /* Allow timeout of host LU?  */
        unsigned char   delayed_logon;    /* Allow delayed logon to the */
                                          /* host LU                    */
        unsigned char   reserv4[6];       /* reserved                   */
} DEFINE_DOWNSTREAM_LU_RANGE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DOWNSTREAM_LU_RANGE

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dslu_base_name

다운스트림 LU 명 범위의 기본명. 5바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. NAU 범위 내에 있는 각 LU의 경우, 이 기본명에 세 개의 A 유형 EBCDIC 숫자가 추가되어 NAU 주소의 십진값을 나타냅니다.

description

자원 설명(QUERY_DOWNSTREAM_LU에서 반환). 이 필드의 길이는 4의 배수(바이트)이어야 하며 0이 될 수 없습니다.

min_nau

범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

max_nau

범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

dspu_name

DOWNSTREAM PU의 이름(DEFINE_LS에 지정한). 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

host_lu_name

범위 내의 모든 다운스트림 LU가 매핑되는 호스트 LU 또는 호스트 LU 풀의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

allow_timeout

호스트 LU 정의에 지정한 **timeout** 기간 동안 세션이 활동중이 아닌 상태로 있을 경우 프로그램이 이 다운스트림 LU가 사용하는 호스트 LU를 시간종료할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

delayed_logon

프로그램이 다운스트림 LU에서 첫번째 데이터가 수신될 때까지 다운스트림 LU와 호스트 LU의 연결을 지연하는지의 여부를 지정합니다. 대신, 시뮬레이트된 로그온 화면이 다운스트림 LU에 송신됩니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DNST_LU_NAME

AP_INVALID_NAU_ADDRESS
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_LU_NAME_POOL_NAME_CLASH

AP_LU_ALREADY_DEFINED
AP_INVALID_HOST_LU_NAME
AP_PU_NOT_DEFINED
AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_CANT_MODIFY_VISIBILITY
AP_DELAYED_VERB_PENDING

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_DSPU_TEMPLATE

이 명령은 통신 서버에만 적용됩니다.

이 명령은 PU 집중(concentration)에 사용됩니다. PU 집중을 사용하면, 다운스트림 LU가 업스트림 호스트와 통신할 수 있습니다. 이를 위해 통신 서버는 각 다운스트림 LU를 호스트 LU라고 하는 종속 국지 LU로 맵핑합니다. DEFINE_DSPU_TEMPLATE는 다운스트림 워크스테이션 그룹에 있는 다운스트림 LU에 대한 템플리트를 정의합니다. 이 템플리트는 워크스테이션이 암시적 링크(이전에 정의하지 않은)를 통해 통신 서버에 연결할 때 다운스트림 LU의 정의를 제위치에 놓는 데 사용됩니다. 이들 템플리트는 DEFINE_PORT 명령의 **implicit_dspu_template** 필드에서 참조됩니다. DEFINE_DSPU_TEMPLATE를 사용하여 새로운 템플리트를 정의하거나 기존의 템플리트를 수정할 수 있습니다.(수정된 템플리트의 기존 인스턴스는 영향을 받지 않습니다.)

VCB 구조

```
typedef struct define_dspu_template
{
        unsigned short  opcode;            /* verb operation code       */
        unsigned char   attributes;        /* verb attributes           */
        unsigned char   format;            /* format                    */
        unsigned short  primary_rc;        /* primary return code       */
        unsigned long   secondary_rc;      /* secondary return code     */
        unsigned char   template_name[8];  /* name of template          */
        unsigned char   description;       /* resource description      */
        unsigned char   modify_template;   /* Modify existing template? */
        unsigned char   reserv1[11];       /* reserved                  */
        unsigned short  max_instance;      /* Max active template       */
                                           /* instances                 */
        unsigned short  num_of_dslu_templates;
                                           /* number of DSLU templates  */
} DEFINE_DSPU_TEMPLATE;

typedef struct dslu_template
{
        unsigned char   min_nau;            /* min NAU address in range  */
        unsigned char   max_nau;            /* max NAU address in range  */
        unsigned char   allow_timeout;      /* Allow timeout of host LU? */
        unsigned char   delayed_logon;      /* Allow delayed logon to    */
                                            /* host LU                   */
        unsigned char   reserv1[8];         /* reserved                  */
        unsigned char   host_lu[8];         /* host LU or pool name      */
} DSLU_TEMPLATE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DSPU_TEMPLATE

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

template_name

DSPU 템플리트의 이름(PORT_DEF_DATA의 **implicit_dspu_template** 필드에 지정하는 이름과 일치합니다.). 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

description

자원 설명(QUERY_DSPU_TEMPLATE에서 반환). 이 필드의 길이는 4의 배수(바이트)이어야 하며 0이 될 수 없습니다.

modify_template

이 명령이 기존의 DSPU 템플리트에 추가 DSLU 템플리트를 추가하는지 또는 기존의 DSPU 템플리트를 대체하는지를 지정합니다(AP_MODIFY_DSPU_TEMPLATE 또는 AP_REPLACE_DSPU_TEMPLATE).

modify_template를 AP_MODIFY_DSPU_TEMPLATE로 설정했을 때 명명된 DSPU 템플리트가 없으면, 이 템플리트가 작성됩니다.

modify_template를 AP_MODIFY_DSPU_TEMPLATE로 설정했을 때 명명된 DSPU 템플리트가 없으면, 추가된 DSLU 템플리트가 기존의 DSPU 템플리트에 추가됩니다.

modify_template를 AP_REPLACE_DSPU_TEMPLATE로 설정하면 새로운 템플리트가 작성됩니다. 0-65535 사이의 한 값이 될 수 있으며, 0은 제한 없음을 의미합니다.

max_instance

동시에 활동할 수 있는 템플리트 인스턴스의 최대 수입니다. 이 한계에 달하면 새로운 인스턴스를 작성할 수 없습니다. 0-65535 사이의 한 값이 될 수 있으며, 0은 제한 없음을 의미합니다.

num_of_dslu_templates

DEFINE_DSPU_TEMPLATE VCB 뒤에 오는 DSLU 템플리트 오버레이 수. 0-255 사이의 한 값이 될 수 있습니다.

dslu_template.min_nau

범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

dslu_template.max_nau
범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

def_data.allow_timeout
호스트 LU 정의에 지정한 **timeout** 기간 동안 세션이 활동중이 아닌 상태로 있을 경우 프로그램이 이 다운스트림 LU가 사용하는 호스트 LU를 시간종료할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.delayed_logon
프로그램이 다운스트림 LU에서 첫번째 데이터가 수신될 때까지 다운스트림 LU와 호스트 LU의 연결을 지연하는지의 여부를 지정합니다. 대신, 시뮬레이트된 로그온 화면이 다운스트림 LU에 송신됩니다(AP_YES 또는 AP_NO).

dslu_template.host_lu
범위 내의 모든 다운스트림 LU가 매핑되는 호스트 LU 또는 호스트 LU 풀의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TEMPLATE_NAME

AP_INVALID_NAU_ADDRESS
AP_INVALID_NAU_RANGE
AP_CLASHING_NAU_RANGE
AP_INVALID_NUM_DSPU_TEMPLATES
AP_INVALID_ALLOW_TIMEOUT
AP_INVALID_DELAYED_LOGON
AP_INVALID_MODIFY_TEMPLATE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_INVALID_HOST_LU_NAME

AP_CANT_MODIFY_VISIBILITY

관련 START_NODE 매개변수가 설정되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_FUNCTION_NOT_SUPPORTED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_FOCAL_POINT

퍼스널 통신이나 통신 서버는 다양한 중재점과 다양한 관계를 가질 수 있습니다. DEFINE_FOCAL_POINT 명령은 퍼스널 통신이나 통신 서버가 암시적 관계(1차 또는 백업 유형일 수 있습니다)를 가지는 중재점을 정의합니다. 이러한 관계와 관계를 설정하는 방법이 아래에 설명되어 있습니다. 주어진 범주에서 관리 서비스 중재점과 관리 서비스 진입점 (EP)의 관계는 이들이 관리 서비스 기능 메세지를 교환할 때 설정됩니다. 다음 유형의 FP-EP 관계를 설정할 수 있습니다.

- 명시적

  이 관계는 중재점에 있는 운영요원이 제어 영역에 진입점을 지정함으로써 설정됩니다. 중재점이 관리 서비스 기능 교환을 시작합니다.

- 암시적(1차)

  진입점에 있는 운영요원이 지정된 중재점에 진입점을 지정할 때(예를 들면, 운영요원이 DEFINE_FOCAL_POINT 명령을 발행할 때) 관계가 설정됩니다. 진입점이 관리 서비스 기능 교환을 시작합니다.

- 암시적(백업)

  이 관계는 진입점이 명시적 또는 암시적 1차 중재점을 유실할 때 설정됩니다. 진입점이 관리 서비스 기능 교환을 시작합니다. 백업 중재점의 ID를 정의하거나(DEFINE_FOCAL_POINT 명령을 사용하여) 관리 서비스 기능 교환을 통해 이 ID를 얻을 수 있습니다.

- 생략시

  이 관계는 FP가 운영요원의 개입없이 EP를 얻을 때 설정됩니다. FP가 MS 기능 교환을 시작합니다. 이 관계는 NN인 EP에만 적용됩니다.

- 도메인

  이 관계는 네트워크 노드(NN)가 끝 노드 진입점에 중재점의 ID를 알릴 때 설정됩니다. 도메인 관계는 끝 노드에서만 유효합니다.

- 호스트

  이 관계는 관리 서비스 기능 교환이 수행되지 않으며, SSCP-PU 세션 구성에 의해 진입점 노드에서 호스트까지 설정됩니다. 우선순위가 가장 낮은 중재점 관계입니다.

각 DEFINE_FOCAL_POINT 명령은 암시적 중재점(1차 또는 백업 유형일 수 있습니다)을 정의하는 데에만 사용할 수 있습니다. 각 DEFINE_FOCAL_POINT 명령은 특정 관리 서비스 범주에 대해 발행합니다. 이러한 범주 내에서 DEFINE_FOCAL_POINT 명령은 다음을 수행하는 데 사용할 수 있습니다.

- 중재점 정의
- 중재점(또는 백업 중재점) 대체
- 현재 활동중인 중재점 취소.

DEFINE_FOCAL_POINT 명령 상의 필드는 다음과 같이 사용됩니다.

ms_category는 항상 채워져 있어야 합니다. **fp_fqcp_name**과 **ms_appl_name fields**의 조합은 지정한 범주의 중재점(**backup** 필드를 AP_YES로 설정한 경우에는 백업 중재점)을 지정합니다.

이 명령을 발행하여 현재 활동중인 중재점을 취소하고 새로운 중재점을 제공하지 않으려면, **fp_fqcp_name**과 **ms_appl_name** 필드를 모두 0으로 설정해야 합니다. 중재점을 정의하거나 대체하는 DEFINE_FOCAL_POINT 명령이 수신되면, 퍼스널 통신이나 통신 서버는 관리 서비스 기능 요청을 전송하여 지정된 중재점과의 암시적 1차 중재점 관계를 설정하려고 시도합니다. 퍼스널 통신이나 통신 서버가 현재 활동중인 중재점을 취소하는 DEFINE_FOCAL_POINT 명령을 수신하면, 퍼스널 통신이나 통신 서버는 중재점에 관리 서비스 기능 취소 메세지를 송신합니다.
DELETE_FOCAL_POINT 명령(AP_ACTIVE를 지정하여)를 사용하여 현재 활동중인 중재점을 취소하는 것이 좋습니다.

VCB 구조

```
typedef struct define_focal_point
{
        unsigned short  opcode;           /* verb operation code          */
        unsigned char   reserv2;          /* reserved                     */
        unsigned char   format;           /* format                       */
        unsigned short  primary_rc;       /* primary return code          */
        unsigned long   secondary_rc;     /* secondary return code        */
        unsigned char   reserved;         /* reserved                     */
        unsigned char   ms_category[8];   /* management services category */
        unsigned char   fp_fqcp_name[17]; /* Fully qualified focal        */
                                          /* point CP name                */
        unsigned char   ms_appl_name[8];  /* Focal point application name */
        unsigned char   description[RD_LEN];
                                          /* resource description         */
        unsigned char   backup;           /* is focal point a backup      */
        unsigned char   reserv3[16];      /* reserved                     */
} DEFINE_FOCAL_POINT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_FOCAL_POINT

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

ms_category

관리 서비스 범주. SNA 관리 서비스에 설명된 것과 같이 관리 서비스 범주에 대해 구조적으로 정의한 4바이트 값(오른쪽이 EBCDIC 공백으로 채워진) 중 하나이거나 8바이트 1134 유형 EBCDIC 설치 정의 이름일 수 있습니다.

fp_fqcp_name

중재점의 전체 정식 제어점(CP)명. 모두 0으로 설정하거나 EBCDIC

점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.) 중재점을 취소할 경우에는 이 필드를 모두 0으로 설정해야 합니다.

ms_appl_name
중재점 응용프로그램명. SNA 관리 서비스에 설명된 것과 같이 관리 서비스 응용프로그램에 대해 구조적으로 정의한 4바이트 값(오른쪽이 EBCDIC 공백으로 채워진) 중 하나이거나 8바이트 1134 유형 EBCDIC 설치 정의 이름일 수 있습니다. 중재점을 취소할 경우에는 이 필드를 모두 0으로 설정해야 합니다.

description
자원 설명(QUERY_FOCAL_POINT에서 반환). 이 이름은 국지로 표시 가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

backup
백업 중재점이 정의되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 현재 활동중인 중재점을 취소할 경우에는 이 필드가 예약됩니다. DELETE_FOCAL_POINT 명령(AP_ACTIVE를 지정하여)를 사용하여 현재 활동중인 중재점을 취소하는 것이 좋습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_FP_NAME

AP_INVALID_CATEGORY_NAME

명령이 정상적으로 실행되지 않으면, 프로그램는 다음 매개변수를 반환합니다.

primary_rc
AP_REPLACED

AP_UNSUCCESSFUL

secondary_rc
AP_IMPLICIT_REQUEST_REJECTED

AP_IMPLICIT_REQUEST_FAILED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 또는 프로그램이 중재점 접속에 실패함으로써 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_INTERNAL_PU

DEFINE_INTERNAL_PU 명령은 DLUR이 서비스를

DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. **def_data.dspu_services**를 AP_DLUR로 설정하지 않은 경우에는 이 필드가 무시됩니다.

def_data.dlus_retry_limit
def_data.dlus_name 및 **def_data.bkup_dlus_name** 필드에 지정한 DLUS에 접속하려는 초기 시도가 실패한 후의 재시도 최대 수. 0을 지정하면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. X'FFFF'를 지정하면, 프로그램이 무한정 재시도합니다. **def_data.dspu_services**를 AP_DLUR로 설정하지 않은 경우에는 이 필드가 무시됩니다.

def_data.conventional_lu_compression
이 PU에 종속적인 상용 LU 세션에 대해 데이터 압축이 요청되는지의 여부를 지정합니다.

AP_NO
국지 노드가 이 PU를 사용하는 상용 LU 세션에서 흐르는 데이터를 압축하거나 압축해제하지 않습니다.

AP_YES
호스트가 압축을 요청할 경우, 이 PU에 종속적인 상용 LU 세션에 대해 데이터 압축이 사용가능하게 됩니다. 이 값을 설정해도 노드가 압축을 지원하지 않으면(START_NODE 명령에 정의), INTERNAL_PU가 압축 지원없이 정의됩니다.

def_data.conventional_lu_cryptography
이 PU에 종속적인 상용 LU 세션에 대해 세션 레벨 암호화가 요구되는지의 여부를 지정합니다.

AP_NONE
국지 노드가 이 PU를 사용하는 상용 LU 세션에서 흐르는 데이터를 압축하거나 압축해제하지 않습니다.

AP_MANDATORY
LU에서 반입 키의 사용이 가능할 경우, APPN이 의무 세션 레벨 암호화를 수행합니다. 그렇지 않을 경우에는, LU를 사용하는 응용프로그램이 의무 세션 레벨 암호화를 수행해야 합니다.(이것이 PU 집중일 경우에는 다운스트림 LU에 의해 수행됩니다.)

AP_OPTIONAL
이 값은 호스트 응용프로그램이 세션별로 암호화를 수행할 수 있도록 합니다. 세션에 대한 호스트 요청 암호화가 이 PU에 종속될 경우, 프로그램의 작동은 AP_MANDATORY의 경우와 동일합니다. 호스트가 암호화를 요청하지 않으면, 프로그램의 작동은 AP_NONE과 동일합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_ID
AP_INVALID_DLUS_NAME
AP_INVALID_BKUP_DLUS_NAME
AP_INVALID_CLU_CRYPTOGRAPHY

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PU_ALREADY_DEFINED

AP_CANT_MODIFY_VISIBILITY

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LOCAL_LU

DEFINE_LOCAL_LU 명령은 지정한 특성을 가지는 국지 LU를 정의하도록 요청하거나, 그러한 LU가 이미 있을 경우 그 LU의 **attach_routing_data** 특성을 수정하도록 요청합니다. DEFINE_LOCAL_LU를 기존 정의를 수정하는 데 사용할 경우, **attach_routing_data** 필드 이외의 매개변수는 무시됩니다.

VCB 구조

형 D 1

```
typedef struct define_local_lu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* local LU name           */
        LOCAL_LU_DEF_DATA
                        def_data;       /* defined data            */
} DEFINE_LOCAL_LU;

typedef struct local_lu_def_data
{
        unsigned char   description;      /* resource description       */
        unsigned char   lu_alias[8];      /* local LU alias             */
        unsigned char   nau_address;      /* NAU address                */
        unsigned char   syncpt_support;   /* is sync-point supported?   */
        unsigned short  lu_session_limit; /* LU session limit           */
        unsigned char   default_pool;     /* member of default_lu_pool  */
        unsigned char   reserv2;          /* reserved                   */
        unsigned char   pu_name[8];       /* PU name                    */
        unsigned char   lu_attributes;    /* LU attributes              */
        unsigned char   sscp_id[6];       /* SSCP ID                    */
        unsigned char   disable;          /* disable or enable LOCAL LU */
        unsigned char   attach_routing_data;
                                          /* routing data for           */
                                          /* incoming attaches          */
        unsigned char   lu_model;         /* LU model for SDDLU         */
        unsigned char   model_name[7];    /* LU model name              */
                                          /* for SDDLU                  */
        unsigned char   reserv4[16];      /* reserved                   */
    } LOCAL_LU_DEF_DATA;
```

VCB 구조

형 D 0

```
typedef struct define_local_lu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
```

```
        unsigned char   lu_name[8];     /* local LU name           */
        LOCAL_LU_DEF_DATA
                        def_data;       /* defined data            */
} DEFINE_LOCAL_LU;

typedef struct local_lu_def_data
{
        unsigned char   description;      /* resource description       */
        unsigned char   lu_alias[8];      /* local LU alias             */
        unsigned char   nau_address;      /* NAU address                */
        unsigned char   syncpt_support;   /* is sync-point supported?   */
        unsigned short  lu_session_limit; /* LU session limit           */
        unsigned char   default_pool;     /* member of default_lu_pool  */
        unsigned char   reserv2;          /* reserved                   */
        unsigned char   pu_name[8];       /* PU name                    */
        unsigned char   lu_attributes;    /* LU attributes              */
        unsigned char   sscp_id[6];       /* SSCP ID                    */
        unsigned char   disable;          /* disable or enable LOCAL LU */
        unsigned char   attach_routing_data;
                                          /* routing data for           */
                                          /* incoming attaches          */
    } LOCAL_LU_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LOCAL_LU

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 형식 0이나 형식 1을 지정하려면, 이 필드를 0이나 1로 설정하십시오.

lu_name

정의할 국지 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.description

자원 설명(QUERY_LOCAL_LU에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.lu_alias

정의할 국지 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

def_data.nau_address

LU의 네트워크 지정 장치(NAU) 주소로, 범위는 0—255이어야 합니다. 0 이외의 값은 LU가 종속 LU임을 의미합니다. 0은 LU가 독립 LU임을 의미합니다.

def_data.syncpt_support

이 LU에 동기점 관리자를 사용할 수 없는 경우에는 이 필드를 항상 AP_NO로 설정해야 합니다.

def_data.lu_session_limit

LU가 지원하는 최대 세션 수. 0은 제한 없음을 의미합니다. LU가 독립적이면 이 매개변수를 아무 값으로나 설정할 수 있습니다. LU가 종속적이면 이 매개변수를 1로 설정해야 합니다.

def_data.default_pool

LU가 종속 LU6.2 생략시 풀의 구성원이면 AP_YES로 설정하십시오.

def_data.pu_name

이 LU가 사용하는 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. 이 필드는 종속 LU에서만 사용되며, 독립 LU의 경우에는 모두 2진 0으로 설정됩니다.

def_data.lu_attributes

LU에 관한 추가 정보를 지정합니다. 이 필드는 AP_NONE 값이나 다음 옵션 중 하나 이상을(OR로 연결) 가집니다.

AP_DISABLE_PWSUB

국지 LU에 대한 암호 대체 지원 사용불가능화.

def_data.sscp_id

이 매개변수는 이 LU를 활성화할 수 있는 SSCP의 ID를 지정합니다. 6바이트 2진 필드입니다. 이 필드는 종속 LU에서만 사용되며, 독립 LU에서나 LU가 SSCP에 의해 활성화될 경우에는 모두 2진 0으로 설정됩니다.

def_data.disable

국지 LU가 사용불가능하게 되는지 또는 사용가능하게 되는지를 지시합니다. 이 매개변수를 적절히 설정하여(AP_YES 또는 AP_NO) DEFINE_LOCAL_LU를 재발행함으로써 LU를 동적으로 사용가능화 또는 사용불능화할 수 있습니다. 사용불가능한 LU가 사용가능하게 되면, 프로그램이 NOTIFY(온라인)를 발행합니다. 사용가능 LU가 사용불가능하게 되면, 프로그램이 NOTIFY(오프라인)를 발행합니다. LU가 사용불가능하게 될 때 바인드되면, 프로그램은 UNBIND를 발행한 후 NOTIFY(오프라인)를 발행합니다.

def_data.attach_routing_data

경로지정 연결 데이터 유형.

AP_REGISTERED_OR_DEFAULT_ATTACH_MGR

이 국지 LU에서 트랜잭션 프로그램(TP)에 대한 수신 확인 연결 결과로 생성된 DYNAMIC_LOAD_INDICATION이 이 LU에 대해 DLI를 수신하도록 등록된 접속 관리자로 송신되거나, 이 LU에 대해 등록된 접속 관리자가 없을 경우 생략시 접속 관리자로 송신되도록 지정합니다.

AP_REGISTERED_ATTACH_MGR_ONLY

이 국지 LU에서 트랜잭션 프로그램(TP)에 대한 수신 확인 연결 결과로 생성된 DYNAMIC_LOAD_INDICATION이 이 LU에

대해 DLI를 수신하도록 등록된 접속 관리자로만 송신되도록 지정합니다. 이 LU에 대해 등록된 접속 관리자가 없을 경우, 접속이 거부됩니다.

def_data.lu_model

LU의 모델 유형 및 번호. 이 필드는 종속 LU에서만 사용되며, 독립 LU의 경우 AP_UNKNOWN으로 설정됩니다. 종속 LU의 경우, 이것은 다음 값 중 하나로 설정됩니다.

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

종속 LU의 경우, **model_name**을 모두 2진 0으로 설정하지 않으면 이 필드가 무시됩니다. 호스트 시스템이 SDDLU(Self-Defining 종속 LU)를 지원하는 상태에서 AP_UNKNOWN 이외의 값을 지정하면, 노드가 호스트에서 국지 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. PSID 부속벡터는 이 필드의 값과 일치하는 시스템 유형 및 모델 번호를 포함합니다. 이 필드는 이 명령을 다시 발행함으로써 동적으로 변경할 수 있습니다. 변경사항은 LU를 닫고 활성종료할 때까지 효력을 가지지 않습니다.

def_data.model_name

LU의 모델명. 이 필드는 종속 LU에서만 사용되며, 독립 LU의 경우 2진 0으로 설정됩니다. APPN은 이 필드가 EBCDIC 문자 A-Z, 0–9 및 @, # 및 $로 구성되는지 점검합니다.

호스트 시스템이 SDDLU를 지원하는 상태에서 이 필드를 2진 0으로 설정하지 않으면, 노드가 호스트에서 국지 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. PSID 부속벡터는 이 필드에 지정한 이름을 포함합니다. 이 명령을 다시 발행함으로써 **def_data.model_name**을 동적으로 변경할 수 있습니다. 변경사항은 LU를 닫고 활성종료할 때까지 효력을 가지지 않습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_MODEL

AP_INVALID_LU_NAME
AP_INVALID_NAU_ADDRESS
AP_INVALID_SESSION_LIMIT
AP_INVALID_DISABLE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PU_NOT_DEFINED

AP_INVALID_LU_NAME
AP_LU_ALREADY_DEFINED
AP_ALLOCATE_NOT_PENDING
AP_LU_ALIAS_ALREADY_USED
AP_PLU_ALIAS_ALREADY_USED
AP_PLU_ALIAS_CANT_BE_CHANGED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

secondary_rc
AP_MEMORY_SHORTAGE

DEFINE_LS

DEFINE_LS는 새로운 링크 스테이션(LS)을 정의하거나 기존의 링크 스테이션을 수정하는 데 사용됩니다. 이 명령은 노드 전반에 걸쳐 고유한 LS 명과 이 LS가 사용하는 포트의 이름을 제공합니다. 이 포트는 DEFINE_PORT 명령으로 이미 정의되어 있어야 합니다. 링크 고유 데이터는 기본 구조에 연결됩니다. 링크 스테이션이 재설정 상태이며(STOP_LS를 발행한 후) LS의 이전 정의 이후로 DEFINE_LS에 지정한 **port_name** 이 변경되지 않은 경우, DEFINE_LS은 기존 링크 스테이션의 하나 이상의 필드를 수정하는 데에만 사용할 수 있습니다.

DLC와 포트, 링크 스테이션간의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

LS_DEF_DATA에 있는 여러 개의 필드 설정은 **adj_cp_type** 필드의 값에 따라 달라집니다. **adj_cp_type**은 8개의 값을 취할 수 있으며(이에 대한 자세한 내용은 **def_data.adj_cp_type** 참조), 이들 중 4개는 인접한 유형 2.1(APPN) 노드와의 링크에 사용됩니다.

- AP_NETWORK_NODE
- AP_END_NODE
- AP_APPN_NODE
- AP_BACK_LEVEL_LEN_NODE

그리고 4개는 PU 유형 2.0 트래픽을 전송하는 링크에 사용됩니다.

- AP_HOST_XID3
- AP_HOST_XID0
- AP_DSPU_XID
- AP_DSPU_NOXID.

4가지 유형의 APPN 노드가 있으며, 이들은 다음과 같이 구분됩니다.

- APPN 네트워크 노드는 XID3에 네트워크명 제어 벡터(CV)를 포함하고, 병렬 TG를 지원하며, XID3에 네트워킹 효능 비트를 설정하고, 링크 상의 CP-CP 세션을 지원할 수 있습니다.
- APPN 끝 노드는 XID3에 네트워크명 CV를 포함하고, 병렬 TG를 지원하며, XID3에 네트워킹 효능 비트를 설정하지 않고, 링크 상의 CP-CP 세션을 지원할 수 있습니다.
- 업 레벨 노드는 XID3에 네트워크명 CV를 포함하고, 병렬 TG를 지원하며, XID3에 네트워킹 효능 비트를 설정하지 않고, CP-CP 세션을 지원하지 않습니다.
- 백 레벨 노드는 XID3에 네트워크명 제어 벡터(CV)를 포함하지 않고, 병렬 TG를 지원하지 않으며, XID3에 네트워킹 효능 비트를 설정하지 않고, CP-CP 세션을 지원하지 않습니다.

모든 링크에 대해 다음 필드를 설정해야 합니다.

port_name

adj_cp_type
dest_address
auto_act_supp
disable_remote_act
limited_resource
link_deact_timer
ls_attributes
adj_node_id
local_node_id
target_pacing_count
max_send_btu_size
link_spec_data_len
ls_role

다른 필드는 다음과 같이 설정해야 합니다.

- **adj_cp_type**을 AP_NETWORK_NODE, AP_END_NODE 또는 AP_APPN_NODE로 설정할 경우, 다음 필드를 설정해야 합니다.

  adj_cp_name
  tg_number
  solicit_sscp_sessions
  dspu_services
  hpr_supported
  hpr_link_lvl_error
  default_nn_server
  cp_cp_sess_support
  use_default_tg_chars
  tg_chars

- **adj_cp_type**을 AP_BACK_LEVEL_LEN_NODE로 설정할 경우, 다음 필드를 설정해야 합니다.

  adj_cp_name
  solicit_sscp_sessions
  dspu_services
  use_default_tg_chars
  tg_chars

- 국지 PU가 링크를 사용할 경우(**adj_cp_type**을 AP_HOST_XID3 또는 AP_HOST_XID0로 설정하거나 APPN 노드에 대한 링크에서 **solicit_sscp_sessions**을 AP_YES로 설정할 경우), 다음 필드를 설정해야 합니다.

  pu_name

- 다운스트림 PU가 링크를 사용하며 PU 집중으로부터 서비스를 받을 경우 (**dspu_services**를 AP_PU_CONCENTRATION으로 설정할 경우), 다음 필드를 설정해야 합니다.

  dspu_name

- 다운스트림 PU가 링크를 사용하며 DLUR로부터 서비스를 받을 경우 (**dspu_services**를 AP_DLUR로 설정할 경우), 다음 필드를 설정해야 합니다.

  dspu_name

  dlus_name

  bkup_dlus_name

VCB 구조

```
typedef struct define_ls
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* current format is zero  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   ls_name[8];     /* name of link station    */
        LS_DEF_DATA     def_data;       /* LS defined data         */
} DEFINE_LS;

typedef struct ls_def_data
{
        unsigned char   description[RD_LEN];
                                        /* resource description           */
        unsigned char   port_name[8];   /* name of associated port        */
        unsigned char   adj_cp_name[17];/* adjacent CP name               */
        unsigned char   adj_cp_type;    /* adjacent node type             */
        LINK_ADDRESS    dest_address;   /* destination address            */
        unsigned char   auto_act_supp;  /* auto-activate supported        */
        unsigned char   tg_number;      /* Pre-assigned TG number         */
        unsigned char   limited_resource;   /* limited resource           */
        unsigned char   solicit_sscp_sessions;
                                        /* solicit SSCP sessions          */
        unsigned char   pu_name[8];     /* Local PU name(reserved if      */
                                        /* solicit_sscp_sessions is set   */
                                        /* to AP_NO)                      */
        unsigned char   disable_remote_act; /* disable remote activation flag */
        unsigned char   dspu_services;  /* Services provided for          */
                                        /* downstream PU                  */
        unsigned char   dspu_name[8];   /* Downstream PU name(reserved    */
                                        /* if dspu_services is set to     */
                                        /* AP_NONE or AP_DLUR)            */
        unsigned char   dlus_name[17];  /* DLUS name if dspu_services     */
                                        /* set to AP_DLUR                 */
        unsigned char   bkup_dlus_name[17]; /* Backup DLUS name if        */
                                        /* dspu_services set to AP_DLUR   */
        unsigned char   hpr_supported;  /* does the link support HPR?     */
        unsigned char   hpr_link_lvl_error; /* does link use link-level   */
                                        /* error recovery for HPR frms?   */
        unsigned short  link_deact_timer;   /* HPR link deactivation timer */
        unsigned char   reserv1;        /* reserved                       */
        unsigned char   default_nn_server;  /* Use as deflt LS to NN server */
        unsigned char   ls_attributes[4];   /* LS attributes              */
        unsigned char   adj_node_id[4];     /* adjacent node ID           */
        unsigned char   local_node_id[4];   /* local node ID              */
        unsigned char   cp_cp_sess_support; /* CP-CP session support      */
        unsigned char   use_default_tg_chars;
                                        /* Use the default tg_chars       */
        TG_DEFINED_CHARS tg_chars;      /* TG characteristics             */
        unsigned short  target_pacing_count;/* target pacing count        */
```

```
        unsigned short  max_send_btu_size;  /* max send BTU size             */
        unsigned char   ls_role;            /* link station role to use      */
                                            /*  on this link                 */
        unsigned char   max_ifrm_rcvd;      /* max number of I-frames rcvd   */
        unsigned short  dlus_retry_timeout; /* DLUS retry timeout            */
        unsigned short  dlus_retry_limit;   /* DLUS retry limit              */
        unsigned char   conventional_lu_compression;
                                            /* Data compression requested for */
                                            /* conventional LU sessions      */
        unsigned char   conventional_lu_cryptography;
                                            /* Cryptography required for     */
                                            /* conventional LU sessions      */
        unsigned char   reserv3;            /* reserved                      */
        unsigned char   retry_flags;        /* conditions LU sessions        */
        unsigned short  max_activation_attempts;
                                            /* how many automatic retries:   */
        unsigned short  activation_delay_timer;
                                            /* delay between automatic retries*/
        unsigned char   branch_link_type;   /* branch link type              */
        unsigned char   adj_brn_cp_support; /* adjacent BrNN CP support      */
        unsigned char   reserv4[20];        /* reserved                      */
        unsigned short  link_spec_data_len; /* length of link specific data  */
} LS_DEF_DATA;

typedef struct tg_defined_chars
{
        unsigned char   effect_cap;         /* effective capacity            */
        unsigned char   reserve1[5];        /* reserved                      */
        unsigned char   connect_cost;       /* connection cost               */
        unsigned char   byte_cost;          /* byte cost                     */
        unsigned char   reserve2;           /* reserved                      */
        unsigned char   security;           /* security                      */
        unsigned char   prop_delay;         /* propagation delay             */
        unsigned char   modem_class;        /* modem class                   */
        unsigned char   user_def_parm_1;    /* user-defined parameter 1      */
        unsigned char   user_def_parm_2;    /* user-defined parameter 2      */
        unsigned char   user_def_parm_3;    /* user-defined parameter 3      */
} TG_DEFINED_CHARS;

typedef struct link_address
{
        unsigned short  length;              /* length                       */
        unsigned short  reserve1;            /* reserved                     */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                             /* address                      */
} LINK_ADDRESS;

typedef struct link_spec_data
{
       unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LS

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

ls_name

링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

ls_name 필드를 특수 값 “$ANYNET$”(ASCII 문자열)로 설정하면, 이것이 AnyNet DLC에 의해 경로지정되는 독립 LU 세션 트래픽이 전송되는 링크 스테이션임이 노드 연산자 기능에 알려집니다. AnyNet 경로지정이 요구될 경우, AnyNet DLC 상의 포트에 이 이름의 링크 스테이션을 정의해야 합니다.

def_data.description

자원 설명(QUERY_LS, QUERY_PU에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.port_name

이 링크 스테이션과 연관된 포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. 이 명명된 포트는 DEFINE_PORT 명령으로 이미 정의되어 있어야 합니다.

def_data.adj_cp_name

전체 정식 17바이트 인접 제어점(CP) 명으로, 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 APPN 노드에 대한 링크에만 관련되며, 그렇지 않은 경우에는 무시됩니다. APPN 노드에 대한 링크의 경우, **tg_number** 필드가 1-20 범위 내의 한 수로 설정되어 있지 않거나 **adj_cp_type** 필드가 AP_BACK_LEVEL_LEN_NODE로 설정되어 있지 않는 한, 모두 0으로 설정할 수 있습니다. 이름을 모두 0으로 설정하면, 이 이름이 XID 교환시 인접 노드에서 수신되는 이름과 비교되지 않습니다. 이름을 모두 0으로 설정하지 않으면, **adj_cp_type** 필드를 AP_BACK_LEVEL_LEN_NODE로 설정하지 않은 경우, 이 이름이 XID 교환시 인접 노드에서 수신되는 이름과 비교됩니다.(이 경우, 이 이름은 인접 노드를 식별하는 데 사용됩니다.)

def_data.adj_cp_type

인접 노드 유형.

AP_NETWORK_NODE

노드가 APPN 네트워크 노드임을 지정합니다.

AP_END_NODE

노드가 APPN 끝 노드 또는 업 레벨 노드임을 지정합니다.

AP_APPN_NODE
노드가 APPN 네트워크 노드, APPN 끝 노드 또는 업 레벨 노드임을 지정합니다. 노드 유형은 XID 교환시에 알 수 있습니다.

AP_BACK_LEVEL_LEN_NODE
노드가 back_level_len 노드임을 지정합니다. 즉, XID에서 제어점(CP) 명을 송신하지 않습니다. 독립 LU 세션을 지원하는 AnyNet DLC를 사용하는 링크의 경우, 반드시 AP_BACK_LEVEL_LEN_NODE를 지정해야 합니다.

AP_HOST_XID3
노드가 호스트이며 퍼스널 통신이나 통신 서버가 노드로부터의 polling XID에 대해 형식 3 XID로 응답하도록 지정합니다.

AP_HOST_XID0
노드가 호스트이며 퍼스널 통신이나 통신 서버가 노드로부터의 polling XID에 대해 형식 0 XID로 응답하도록 지정합니다. 독립 LU 세션을 지원하는 AnyNet DLC를 사용하는 링크의 경우, 반드시 AP_HOST_XID0를 지정해야 합니다.

AP_DSPU_XID
노드가 다운스트림 PU이며 퍼스널 통신이나 통신 서버가 링크 활성화에 XID 교환을 포함하도록 지정합니다.

AP_DSPU_NOXID
노드가 다운스트림 PU이며 퍼스널 통신이나 통신 서버가 링크 활성화에 XID 교환을 포함하지 않도록 지정합니다.

주: VRN에 대한 링크 스테이션은 항상 동적이므로 정의되지 않습니다.

def_data.dest_address.length
인접 노드 상의 목적지 링크 스테이션 주소 길이.

def_data.dest_address.length가 0으로 설정되어 있고 이 LS가 SATF 유형 포트와 연관되어 있으면, 프로그램은 이 링크 스테이션을 충칭 문자 링크 스테이션으로 간주합니다. 이 경우, 프로그램은 정의된 다른 링크 스테이션과는 부합하지 않는 입력 연결에 이 LS를 대응시킵니다.

def_data.dest_address.address
인접 노드 상의 목적지 링크 스테이션 주소. AnyNet DLC를 사용하는 링크의 경우, **dest_address**는 인접 노드 ID 또는 인접 제어점(CP) 명을 지정합니다. 인접 노드 ID를 지정할 경우, 길이가 4이어야 하며 주소는 4바이트 16진 노드 ID(1바이트 블럭 ID, 3바이트 PU ID)를 포함해야 합니다. 인접 제어점(CP) 명을 지정할 경우에는, 길이가 17이고 주소는 EBCDIC로 된 제어점(CP) 명을 포함하며 EBCDIC 공백으로 채워져야 합니다.

def_data.auto_act_supp

세션에서 링크를 요구할 때 링크를 자동으로 활성화할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO). APPN 노드에 대한 링크가 아닌 경우, 이 필드를 항상 AP_YES로 설정할 수 있으며 다른 매개변수에 대한 요구사항이 없습니다. APPN 노드에 대한 링크인 경우, 링크가 CP-CP 세션을 지원하면 이 필드를 AP_YES로 설정할 수 없습니다. 사전 지정된 TG 번호 역시 링크 **tg_number**에 대해 정의되어 있으며 이 번호가 1-20 사이의 한 값으로 설정되어 있는 경우에만 이 필드를 AP_YES로 설정할 수 있습니다. 이러한 요구사항은 **adj_cp_type**을 AP_BACK_LEVEL_LEN_NODE로 설정함으로써 해결할 수 있습니다. 이렇게 설정하면 **cp_cp_sess_support**와 **tg_number**가 무시되기 때문입니다.

def_data.tg_number

사전에 지정된 TG 번호. 이 필드는 인접 APPN 노드에 대한 링크일 경우에만 관련이 있으며, 그렇지 않은 경우에는 무시됩니다. **adj_cp_type**을 AP_BACK_LEVEL_LEN_NODE로 설정할 경우에도 무시되며 1로 설정된 것으로 간주됩니다. 인접 APPN 노드에 대한 링크인 경우, 이것은 1-20 범위 내에서 설정해야 합니다. 이 숫자는 링크를 활성화할 때 링크를 나타내는 데 사용됩니다. 퍼스널 통신이나 통신 서버는 이 링크를 활성화할 때 인접 노드에서 다른 번호를 받아들이지 않습니다. 사전에 지정된 TG 번호의 불일치로 인한 링크 활성화 장애를 피하기 위해서는, 인접 링크 스테이션 상의 인접 노드가 동일한 TG 번호를 정의해야 합니다(사전에 지정된 TG 번호를 사용할 경우). 사전에 지정된 TG 번호를 정의할 경우에는 **adj_cp_name** 또한 정의해야 하며(모두 0으로 설정할 수 없습니다) **adj_cp_type**을 AP_NETWORK_NODE 또는 AP_END_NODE로 설정해야 합니다. 0을 입력하면, TG 번호가 사전에 지정되지 않고 링크가 활성화될 때 조정됩니다.

def_data.limited_resource

링크를 사용하는 세션이 없을 때 이 링크 스테이션이 활성종료되는지의 여부를 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_NO

링크가 제한 자원이 아니며 자동으로 활성종료되지 않습니다.

AP_YES 또는 AP_NO_SESSIONS

링크가 제한 자원이며, 링크를 사용하는 활동중인 세션이 없을 때 자동으로 활성종료됩니다. 제한 자원 링크 스테이션은 CP-CP 세션을 지원하도록 구성할 수 있습니다.(이러한 구성은 이 필드를 AP_YES로 설정하고 **cp_cp_sess_support**를 AP_YES로 설정할 때 가능합니다.) 이 경우, CP-CP 세션이 이 링크에서 활성화되면, 퍼스널 통신이나 통신 서버는 이 링크를 제한 자원으로 취급하지 않습니다.(따라서 링크를 다운시키지 않습니다.)

AP_INACTIVITY
링크가 제한 자원이며, 이 링크를 사용하는 활동중인 세션이 없거나 **link_deact_timer** 필드에 지정된 기간 동안 링크에서 어떠한 데이터도 흐르지 않았을 때 자동으로 활성종료됩니다. 비교환식 포트 상의 링크 스테이션은 제한 자원으로 구성할 수 없습니다.

비교환식 포트 상의 링크 스테이션은 제한 자원으로 구성할 수 없습니다.

제한 자원 링크 스테이션은 CP-CP 세션을 지원하도록 구성할 수 있습니다.(이러한 구성은 이 필드를 AP_YES로 설정하고 **cp_cp_sess_support**를 AP_YES로 설정함으로써 가능합니다.) 이 경우, CP-CP 세션이 이 링크에서 활성화되면, 퍼스널 통신이나 통신 서버는 이 링크를 제한 자원으로 취급하지 않습니다.(따라서 링크를 다운시키지 않습니다.) 이는 이 필드를 AP_INACTIVITY로 설정할 경우에는 적용되지 않음에 주의하십시오.

def_data.solicit_sscp_sessions
AP_YES는 인접 노드에 SSCP와 국지 제어점(CP) 그리고 종속 LU간의 세션을 시작하도록 요청합니다.(이 경우, **pu_name**이 설정되어 있어야 합니다.) AP_NO는 이 링크에서 SSCP와의 세션을 요청하지 않습니다. 이 필드는 APPN 노드와의 링크에만 관련되며, 그렇지 않은 경우에는 무시됩니다. 인접 노드를 호스트로 정의한 경우(**adj_cp_type**을 AP_HOST_XID3 또는 AP_HOST_XID0로 설정한 경우), 퍼스널 통신이나 통신 서버는 항상 호스트에 SSCP와 국지 제어점(CP) 그리고 종속 LU 사이의 세션을 시작하도록 요청합니다.(이 경우에도 역시 **pu_name**이 설정되어 있어야 합니다.)

dspu_services를 AP_NONE으로 설정한 경우, 이 필드는 인접 APPN 노드에 대한 링크에서 AP_YES로만 설정할 수 있습니다. 이 필드를 AP_YES로 설정하고 이 LS가 사용하는 DCL을 hpr_only로 정의하면, DEFINE_LS가 거부되고 매개변수 점검과 2차 리턴 코드 AP_INVALID_SOLICIT_SSCP_SESS가 반환됩니다.

def_data.pu_name
인접 노드를 호스트로 정의했거나 APPN 노드에 대한 링크에서 **solicit_sscp_sessions**을 AP_YES로 설정한 경우에 이 링크를 사용하는 국지 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. 인접 노드가 호스트로 정의되어 있지 않으며 **solicit_sscp_sessions**이 AP_YES로 설정된 APPN 노드로 정의되어 있지 않으면, 이 필드는 무시됩니다.

def_data.disable_remote_act
이 링크의 원격 활성화가 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.dspu_services

국지 노드가 이 링크를 통해 다운스트림 PU에 제공하는 서비스를 지정합니다. 다음 중 하나로 설정됩니다.

AP_PU_CONCENTRATION

국지 노드가 다운스트림 PU에 PU 집중을 제공합니다.

AP_DLUR

국지 노드가 다운스트림 PU에 DLUR 서비스를 제공합니다. 이 설정은 국지 노드가 네트워크 노드일 경우에만 유효합니다.

AP_NONE

국지 노드가 이 다운스트림 PU에 서비스를 제공하지 않습니다.

이 필드를 AP_PU_CONCENTRATION 또는 AP_DLUR로 설정할 경우에는 **dspu_name** 또한 설정해야 합니다.

인접 노드를 다운스트림 PU로 정의한 경우(즉, **adj_cp_type**을 AP_DSPU_XID 또는 AP_DSPU_NOXID로 설정한 경우), 이 필드를 AP_PU_CONCENTRATION 또는 AP_DLUR로 설정해야 합니다. **solicit_sscp_sessions**을 AP_NO로 설정한 경우, APPN 노드에 대한 링크에서 이 필드를 AP_PU_CONCENTRATION 또는 AP_DLUR로 설정할 수 있습니다. 인접 노드를 호스트로 정의한 경우에는 이 필드가 무시됩니다.

이 필드를 AP_NONE으로 설정하고 이 LS가 사용하는 DLC를 hpr_only로 정의하면, DEFINE_LS가 거부되고 매개변수 점검과 2차 리턴 코드 SP_INVALID_DSPU_SERVICES가 반환됩니다.

def_data.dspu_name

다운스트림 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

dspu_services를 AP_PU_CONCENTRATION 또는 AP_DLUR로 설정할 경우에는 이 필드를 설정해야 하며, 그렇지 않을 경우에는 이 필드가 무시됩니다.

def_data.dlus_name

다운스트림 노드에 대한 링크가 활성화될 때부터 DLUR이 SSCP 서비스를 요청하는 DLUS 노드의 이름. 모두 0으로 설정되거나 EBCDIC 점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 문자열의 오른쪽은 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.) 이 필드를 모두 0으로 설정하면, 링크가 활성화될 때 글로벌 생략시 DLUS(DEFINE_DLUR_DEFAULTS 명령으로 정의한 경우)가 요청됩니다. **dlus_name**을 0으로 설정했을 때 글로벌 생략시 DLUS가 없을 경

우, DLUR은 링크가 활성화될 때 SSCP 접속을 시작하지 않습니다. **dspu_services**를 AP_DLUR로 설정하지 않은 경우에는 이 필드가 무시됩니다.

def_data.bkup_dlus_name
다운스트림 PU에 대한 백업으로 기능하는 DLUS 노드의 이름. 모두 0으로 설정하거나 EBCDIC 점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 문자열의 오른쪽은 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.) 이 필드를 모두 0으로 설정하면, 글로벌 백업 디폴트 DLUS(DEFINE_DLUR_DEFAULTS 명령으로 정의한 경우)가 이 PU에 대한 백업 DLUS로서 사용됩니다. **dspu_services**를 AP_DLUR로 설정하지 않은 경우에는 이 필드가 무시됩니다.

def_data.hpr_supported
이 링크에서 HPR이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 APPN 노드에 대한 링크일 경우에만 관련이 있으며, 그렇지 않은 경우에는 무시됩니다. APPN 노드에 대한 링크가 아닌 경우에 이 필드를 AP_YES로 설정하면, 명령이 거부되고 매개변수 점검과 2차 리턴 코드 INVALID_NODE_TYPE_FOR_HPR이 반환됩니다.

def_data.hpr_link_lvl_error
링크 레벨 오류 복구를 사용하는 이 링크에서 HPR 트래픽이 전송되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). **hpr_supported** 를 AP_NO로 설정한 경우에는 이 매개변수가 무시됩니다.

def_data.link_deact_timer
제한 자원 링크 활성종료 타이머(초 단위).

limited_resource를 AP_INACTIVITY로 설정한 경우, 이 타이머의 지속시간 동안 링크를 통해 아무런 데이터도 흐르지 않으면 링크가 자동으로 활성종료됩니다.

0을 지정하면, 생략시 값 30이 사용됩니다. 그러지 않을 경우, 최소값은 5입니다.(5보다 작은 값을 지정하면, 지정한 값이 무시되고 5가 사용됩니다.)**limited_resource**를 AP_NO로 설정한 경우에는 이 매개변수가 예약됩니다.

def_data.default_nn_server
네트워크 노드 서버에 대한 CP-CP 세션을 지원하기 위해 끝 노드가 자동으로 링크를 활성화할 수 있는지의 여부를 지정합니다. (AP_YES 또는 AP_NO). 이 필드가 효력을 가지려면, CP-CP 세션을 지원하도록 링크를 정의해야 합니다.

def_data.ls_attributes
인접 노드에 관한 추가 정보를 지정합니다.

def_data.ls_attributes[0]
호스트 유형.

AP_SNA
표준 SNA 호스트.

AP_FNA
FNA(VTAM-F) 호스트.

AP_HNA
HNA 호스트.

def_data.ls_attributes[1]
이 필드는 비트 필드입니다. AP_NO 또는 다음 값을 취할 수 있습니다(OR에 의해 비트와이즈로 연결).

AP_SUPPRESS_CP_NAME
백 레벨 LEN 노드와의 링크에 대한 네트워크명 CV suppression 옵션. 이 비트가 설정되어 있으면, 인접 노드와의 XID 교환에 네트워크명 CV가 포함되지 않습니다.(**adj_cp_type**을 AP_BACK_LEVEL_LEN_NODE 또는 AP_HOST_XID3으로 설정하지 않은 경우에는 이 비트가 무시됩니다.)

AP_REACTIVATE_ON_FAILURE
링크가 활동중에 있다가 실패하면 퍼스널 통신이나 통신 서버가 링크를 다시 활성화하려고 시도합니다. 재활성화 시도가 실패할 경우, 링크는 활동중이 아닌 상태로 유지됩니다.

AP_USE_PU_NAME_IN_XID_CVS
인접 노드를 호스트로 정의했거나 APPN 노드에 대한 링크에서 **solicit_sscp_sessions**을 tp AP_YES로 설정했으며 AP_SUPPRESS_CP_NAME 비트를 설정하지 않은 경우, 형식 3 XID에서 전송되는 네트워크명 CV 내의 전체 정식 CP 명이 **def_data.pu_name**에서 제공되고 해당 CP의 네트워크 ID로 전체 정식화된 이름으로 대체됩니다.

def_data.adj_node_id
인접 노드의 노드 ID. 이것은 4바이트 16진 문자열입니다. **adj_cp_type**이 인접 노드가 T2.1노드임을 지시하면, 이 필드가 nonzero가 아니며 **adj_cp_type**이 AP_BACK_LEVEL_LEN_NODE로 설정되어 있지 않거나 인접 노드가 XID3에서 네트워크명 CV를 전송할 경우, 이 필드는 무시됩니다. **adj_cp_type**을 AP_HOST_XID3 또는 AP_HOST_XID0로 설정하면, 이 필드는 항상 무시됩니다. **adj_cp_type**을 AP_DSPU_XID로 설정한 상태에서 이 필드가 nonzero이면, 이 필드는 다운스트림 PU의 ID를 점검하는 데 사용됩니다. **adj_cp_type**을 AP_DSPU_NOXID로 설정하면, 이 필드는 무시되거나(**dspu_services** 가 AP_PU_CONCENTRATION인 경우) DLUS에 다운스트림 PU를 식별하는 데 사용됩니다(**dspu_services**가 AP_DLUR인 경우).

def_data.local_node_id
이 링크 스테이션에서 XID로 전송되는 노드 ID. 4바이트 16진 문자

열입니다. 이 필드를 0으로 설정하면, **node_id**가 XID 교환에 사용됩니다. 이 필드가 nonzero이면, 이 LS에서의 XID 교환시 값을 대체합니다.

def_data.cp_cp_sess_support
CP-CP 세션이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 APPN 노드에 대한 링크일 경우에만 관련이 있으며, 그렇지 않은 경우에는 무시됩니다. **adj_cp_type**을 AP_BACK_LEVEL_LEN_NODE로 설정할 경우에도 무시되며 AP_NO로 설정된 것으로 간주됩니다.

def_data.use_default_tg_chars
DEFINE_PORT 명령에서 제공되는 생략시 TG 특성이 사용되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드를 AP_YES로 설정하면 **tg_chars** 필드가 무시됩니다. 이 필드는 APPN 노드에 대한 링크일 경우에만 관련이 있으며, 그렇지 않은 경우에는 무시됩니다.

def_data.tg_chars
TG 특성(35페이지의 『DEFINE_CN』을 참조하십시오). 이 필드는 APPN 노드에 대한 링크일 경우에만 관련이 있으며, 그렇지 않은 경우에는 무시됩니다.

def_data.target_pacing_count
이 TG에서의 BIND에 적절한 페이싱 창 크기를 지시하는 1 과 32 767 사이의 숫자값. 이 수는 고정 바인드 페이싱이 수행될 경우에만 유효합니다. 퍼스널 통신이나 통신 서버에서는 현재 이 값을 사용하지 않습니다.

def_data.max_send_btu_size
이 링크 스테이션에서 전송할 수 있는 최대 BTU 크기. 이 값은 링크 스테이션 쌍간에 전송 가능한 최대 BTU 크기를 조정하는 데 사용할 수 있습니다. 링크가 HPR이 가능한 링크가 아닐 경우, 이 필드는 99 이상의 한 값으로 설정해야 합니다. 링크가 HPR 가능 링크이면, 이 필드를 768 이상의 한 값으로 설정해야 합니다.

def_data.ls_role
이 링크 스테이션이 맡게 되는 링크 스테이션 역할. 조정가능, 1차 또는 2차 역할을 선택하는 AP_LS_NEG, AP_LS_PRI 또는 AP_LS_SEC가 될 수 있습니다. 이 필드는 또한 AP_USE_PORT_DEFAULTS로 설정하여 DEFINE_PORT 명령에 구성된 값을 선택하도록 할 수도 있습니다. **dlc_type**이 AP_TWINAX이면 AP_LS_SEC만 지원됩니다. **dlc_type**이 AP_ANYNET이면(그리고 **ls_name**이 “$ANYNET$”이면) AP_LS_PRI가 지원되지 않습니다.

def_data.max_ifrm_rcvd
확인 응답 전 XID 전송자가 수신할 수 있는 I 프레임 최대 수.

범위: 0 — 127

0을 지정하면, DEFINE_PORT에 있는 **max_ifrm_rcvd** 의 값이 생략시로 사용됩니다.

def_data.dlus_retry_timeout

def_data.dlus_name 및 **def_data.bkup_dlus_name** 필드에 지정된 DLUS에 접속하려는 두 번째 이후 시도들 사이의 간격(초 단위). 초기 시도와 첫번째 재시도 사이의 간격은 항상 1초입니다. 0을 지정하면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. **def_data.dspu_services**가 AP_DLUR로 설정되어 있지 않으면 이 필드가 무시됩니다.

def_data.dlus_retry_limit

def_data.dlus_name 및 **def_data.bkup_dlus_name** 필드에 지정한 DLUS에 접속하려는 초기 시도가 실패한 후의 재시도 최대 수. 0을 지정하면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. X'FFFF'를 지정하면, APPN이 무한정 재시도합니다. **def_data.dspu_services**가 AP_DLUR로 설정되어 있지 않으면 이 필드가 무시됩니다.

def_data.conventional_lu_compression

이 PU에 종속적인 상용 LU 세션에 대해 데이터 압축이 요청되는지의 여부를 지정합니다. 이 필드는 LU 0 - 3 트래픽을 전용하는 링크에만 유효합니다.

AP_NO

국지 노드가 이 PU를 사용하는 상용 LU 세션에서 흐르는 데이터를 압축하거나 압축해제하지 않습니다.

AP_YES

호스트가 압축을 요청할 경우, 이 PU에 종속적인 상용 LU 세션에 대해 데이터 압축이 사용가능하게 됩니다. 이 값을 설정해도 노드가 압축을 지원하지 않으면(START_NODE 명령에 정의), 링크 스테이션이 압축 지원없이 정의됩니다.

def_data.conventional_lu_cryptography

상용 LU 세션에 세션 레벨 암호화가 요구되는지의 여부를 지정합니다. 이 필드는 상용 LU 트래픽을 위한 링크에만 적용됩니다.

AP_NONE

프로그램이 세션 레벨 암호화를 수행하지 않습니다.

AP_MANDATORY

LU에서 반입(import) 키의 사용이 가능할 경우, 프로그램이 Mandatory 세션 레벨 암호화를 수행합니다. 그렇지 않을 경우에는, LU를 사용하는 응용프로그램이 의무 세션 레벨 암호화를 수행해야 합니다.(이것이 PU 집중일 경우에는 다운스트림 LU에 의해 수행됩니다.)

AP_OPTIONAL

이 값은 호스트 응용프로그램이 세션별로 암호화를 수행할 수 있도록 합니다. 호스트가 이 LS 상의 세션에 대해 암호화를 요청할 경우, 프로그램의 작동은 AP_MANDATORY일 때와 동

일합니다. 호스트가 암호화를 요청하지 않을 경우, 프로그램의 작동은 AP_NONE일 때와 동일합니다.

def_data.retry_flags

이 필드는 이 링크 스테이션의 활성화가 자동으로 재시도되는 조건을 지정합니다. 비트 필드이며, 다음 값을 취할 수 있습니다(OR에 의해 비트와이즈로 연결).

AP_RETRY_ON_START

활성화를 시도할 때 원격 노드에서 응답이 수신되지 않을 경우 링크 활성화를 재시도합니다. 활성화를 시도할 때 기본 포트가 활동중이 아닌 경우에는 프로그램이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_FAILURE

링크가 활동중에 또는 활성화 미정 상태에서 실패할 경우, 링크 활성화가 재시도됩니다. 활성화를 시도할 때 기본 포트가 실패한 경우, 프로그램이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_DISCONNECT

링크가 원격 노드에 의해 정상적으로 종료될 경우 링크 활성화를 재시도합니다.

AP_DELAY_APPLICATION_RETRIES

응용프로그램에 의해 시작되는(START_LS 또는 요구가 있을 때 링크 활성화에 의해) 링크 활성화 재시도는 **activation_delay_timer**로 페이싱됩니다.

AP_INHERIT_RETRY

이 필드에 플래그로 지정된 재시도 조건 이외에도, 기본 포트 정의의 **retry_flags** 필드에 지정된 재시도 조건 역시 사용됩니다.

def_data.max_activation_attempts

retry_flags 필드에 하나 이상의 플래그를 설정하지 않으면 이 필드는 아무런 효력이 없습니다.

이 필드는 원격 노드가 응답하지 않거나 기본 포트가 활동중이 아닐 경우 프로그램이 허용하는 재시도 수를 지정합니다. 여기에는 자동 재시도 수 및 응용프로그램에 의해 활성화 시도 수 둘다 포함됩니다.

이 한계값에 달하면, 자동으로 재시도하려는 시도가 더이상 행해지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는 OPEN_LU_SSCP_SEC_RQ는 활성화 시도가 한 번만 행해지도록 하며, 활성화에 실패해도 재시도가 행해지지 않습니다.

0은 '제한 없음'을 의미합니다. AP_USE_DEFAULTS 값을 지정하면, DEFINE_PORT에서 제공되는 **max_activiation_attempts**가 사용됩니다.

def_data.activation_delay_timer

retry_flags 필드에 하나 이상의 플래그를 설정하지 않으면 이 필드는 아무런 효력이 없습니다.

이 필드는 자동 재시도 사이, 또는 **def_data.retry_flags**에 AP_DELAY_APPLICATION_RETRIES 비트가 설정되어 있을 경우 응용프로그램에 의한 활성화 시도간에 프로그램이 대기하는 시간(초 수)을 지정합니다.

AP_USE_DEFAULTS 값을 지정하면, DEFINE_PORT에서 제공되는 **activiation_delay_timer**가 사용됩니다.

0을 지정하면, 프로그램이 생략시 타이머 지속시간 30초를 사용합니다.

def_data.branch_link_type

BrNN 전용. 링크가 업링크인지 다운링크인지를 지정합니다. 이 필드는 **def_data.adj_cp_type**을 AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE로 설정할 경우에만 적용됩니다.

AP_UPLINK

이 링크가 업링크입니다.

AP_DOWNLINK

이 링크가 다운링크입니다.

adj_cp_type 필드를 AP_NETWORK_NODE로 설정한 경우에는 이 필드를 AP_UPLINK로 설정해야 합니다.

다른 노드 유형: 이 필드가 무시됩니다.

def_data.adj_brnn_cp_support

BrNN 전용. 인접 CP가 NN(BrNN)(예를 들면, 외형상으로 NN과 유사한 BrNN)이 될 수 있는지, NN(BrNN)이 되어야 하는지, 또는 NN(BrNN)이 될 수 없는지를 지정합니다. 이 필드는 **adj_cp_type**을 AP_NETWORK_NODE 또는 AP_APPN_NODE로 설정할 경우에만 적용됩니다.(XID 교환시에 알려지는 노드 유형은 네트워크 노드입니다.)

AP_BRNN_ALLOWED

인접 CP가 NN(BrNN)이 될 수 있습니다(요구 사항이 아님).

AP_BRNN_REQUIRED

인접 CP가 NN(BrNN)이 되어야 합니다.

AP_BRNN_PROHIBITED

인접 CP가 NN(BrNN)이 될 수 없습니다.

adj_cp_type 필드를 AP_NETWORK_NODE로 설정하고 **auto_act_supp** 필드를 AP_YES로 설정한 경우, 이 필드를 AP_BRNN_REQUIRED 또는 AP_BRNN_PROHIBITED로 설정해야 합니다.

다른 노드 유형: 이 필드가 무시됩니다.

def_data.link_spec_data_len
이 필드는 항상 0으로 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_DEF_LINK_INVALID_SECURITY

AP_INVALID_CP_NAME
AP_INVALID_LIMITED_RESOURCE
AP_INVALID_LINK_NAME
AP_INVALID_LS_ROLE
AP_INVALID_NODE_TYPE
AP_INVALID_PORT_NAME
AP_INVALID_AUTO_ACT_SUPP
AP_INVALID_PU_NAME
AP_INVALID_SOLICIT_SSCP_SESS
AP_INVALID_DLUS_NAME
AP_INVALID_BKUP_DLUS_NAME
AP_INVALID_NODE_TYPE_FOR_HPR
AP_INVALID_TARGET_PACING_COUNT
AP_INVALID_BTU_SIZE
AP_HPR_NOT_SUPPORTED
AP_INVALID_TG_NUMBER
AP_MISSING_CP_NAME
AP_MISSING_CP_TYPE
AP_MISSING_TG_NUMBER
AP_PARALLEL_TGS_NOT_SUPPORTED
AP_INVALID_DLUS_RETRY_TIMEOUT
AP_INVALID_DLUS_RETRY_LIMIT
AP_INVALID_CLU_CRYPTOGRAPHY
AP_INVALID_RETRY_FLAGS
AP_BRNN_SUPPORT_MISSING
AP_INVALID_BRANCH_LINK_TYPE
AP_INVALID_BRNN_SUPPORT

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_LOCAL_CP_NAME

AP_DEPENDENT_LU_SUPPORTED
AP_DUPLICATE_DEST_ADDR
AP_INVALID_NUM_LS_SPECIFIED
AP_LS_ACTIVE
AP_PU_ALREADY_DEFINED
AP_DSPU_SERVICES_NOT_SUPPORTED
AP_DUPLICATE_TG_NUMBER
AP_TG_NUMBER_IN_USE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_UPLINK
AP_INVALID_DPWNLINK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LU_0_TO_3

이 명령은 0, 1, 2 또는 3 유형의 LU를 정의합니다. 이 명령은 LU 풀에 LU를 추가할 수 있도록 합니다. 아직 LU 풀이 없으면 LU 풀이 추가됩니다. 이 명령은 존재하는 정의의 **lu_model**, **모델명**, **우선순위**, **설명** 또는 **appc_spec_def_data**를 수정하는데 사용할 수 없으나 다른 필드에는 수정 가능합니다.

퍼스널 통신이나 통신 서버는 ACTLU에 의한 암시적 LU 유형 0, 1, 2 또는 3 정의를 지원합니다. 암시적 정의는 삭제할 수 없으나 LU가 비활성시에는 제거됩니다. 암시적 정의에 대한 정보를 얻으려면, QUERY_LU_0_TO_3을 사용하거나 LU_0_TO_3_INDICATION에 등록하십시오. **lu_name**, **pu_name** 또는 **nau_address**가 정확하고 **pool_name**이 모두 0일 경우(이 때 LU는 운영 요원이 우선적으로 구성한 것처럼 취급됩니다), DEFINE_LU_0_TO_3을 사용하여 암시적 LU 정의를 재정의할 수 있습니다.

VCB 구조

형 D 1

```
typedef struct define_lu_0_to_3
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* LU name                 */
        LU_0_TO_3_DEF_DATA
                        def_data;       /* defined data            */
} DEFINE_LU_0_TO_3;

typedef struct lu_0_to_3_def_data
{
       unsigned char   description           /* resource description       */
       unsigned char   nau_address;          /* LU NAU address             */
       unsigned char   pool_name[8];         /* LU pool name               */
       unsigned char   pu_name[8];           /* PU name                    */
       unsigned char   priority;             /* LU priority                */
       unsigned char   lu_model;             /* LU model                   */
       unsigned char   sscp_id[6]            /* SSCP ID                    */
       unsigned short  timeout;              /* Timeout                    */
       unsigned char   app_spec_def_data[16]; /* Application Specified Data */
       unsigned char   model_name[7];        /* LU model name for DDDLU    */
       unsigned char   reserv3[17];          /* reserved                   */
      } LU_0_TO_3_DEF_DATA;
```

VCB 구조

형 D 0

```
typedef struct define_lu_0_to_3
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* attributes              */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
```

```
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* LU name                 */
        LU_0_TO_3_DEF_DATA
                        def_data;       /* defined data            */
} DEFINE_LU_0_TO_3;

typedef struct lu_0_to_3_def_data
{
        unsigned char   description           /* resource description       */
        unsigned char   nau_address;          /* LU NAU address             */
        unsigned char   pool_name[8];         /* LU pool name               */
        unsigned char   pu_name[8];           /* PU name                    */
        unsigned char   priority;             /* LU priority                */
        unsigned char   lu_model;             /* LU model                   */
        unsigned char   sscp_id[6]            /* SSCP ID                    */
        unsigned short  timeout;              /* Timeout                    */
        unsigned char   app_spec_def_data[16]; /* Application Specified Data */
       } LU_0_TO_3_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LU_0_TO_3

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전 중 하나를 지정하려면, 이 필드를 0 또는 1로 설정하십시오.

lu_name

정의할 국지 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.description

자원 설명(QUERY_LU_0_TO_3에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.nau_address

LU의 네트워크 지정 장치(NAU) 주소로, 범위는 1—255이어야 합니다.

def_data.pool_name

이 LU가 속하는 LU 풀의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. LU가 풀에 속하지 않을 경우, 이 필드는 모두 2진 0으로 설정됩니다. 현재 풀이 없는 경우에는 풀이 작성됩니다.

def_data.pu_name

이 LU가 사용하는 PU의 이름(DEFINE_LS 명령에 지정된). 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

def_data.priority

호스트로 전송할 때의 LU 우선순위. 다음 값 중 하나로 설정됩니다.

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

def_data.lu_model

LU의 모델 유형 및 번호. 다음 값 중 하나로 설정됩니다.

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

형식 1의 경우에만, **model_name**을 모두 2진 0으로 설정하지 않으면 이 필드가 무시됩니다.

호스트 시스템이 DDDLU(종속 LU의 동적 정의)를 지원할 경우 AP_UNKNOWN 이외의 값을 지정하면, 노드가 호스트에서 국지 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. 형식 1의 경우에만, PSID 부속벡터가 이 필드의 값과 일치하는 시스템 유형이나 모델 번호를 포함합니다. 이 필드는 이 명령을 다시 발행함으로써 동적으로 변경할 수 있습니다. 변경사항은 LU를 닫고 활성종료할 때까지 효력을 가지지 않습니다.

def_data.sscp_id

이 필드는 이 LU를 활성화할 수 있는 SSCP의 ID를 지정합니다. 6바이트 2진 필드입니다. 이 필드를 2진 0으로 설정하면, 모든 SSCP가 LU를 활성화할 수 있습니다.

def_data.timeout

초 단위로 지정하는 LU에 대한 시간종료. 시간종료 값이 제공되고 LU 사용자가 OPEN_LU_SSCP_SEC_RQ에(또는 PU 집중의 경우, 다운스트림 LU 정의에) **allow_timeout**을 지정한 경우, 이 기간 동안 PLU-SLU 세션이 활동중이 아닌 상태로 있으며 다음 조건 중 하나에 해당될 때 LU가 활성종료됩니다.

- 세션이 제한 자원 링크를 통과합니다.

- 세션이 다시 사용되기 전에 다른 응용프로그램이 LU를 사용하려고 합니다.

시간종료를 0으로 설정하면 LU가 활성종료되지 않습니다.

def_data.app_spec_def_data
응용프로그램이 지정하는 정의된 데이터. 이 필드는 퍼스널 통신이나 통신 서버에 의해 해석되지는 않으나, 저장되었다가 QUERY_LU_0_TO_3 명령에서 반환됩니다.

def_data.model_name
퍼스널 통신이나 통신 서버는 이 필드가 EBCDIC 문자 A-Z, 0–9 및 @, # 및 $로 구성되는지 점검합니다. 호스트 시스템이 DDDLU(종속 LU의 동적 정의)를 지원할 때 이 필드를 모두 2진 0으로 설정하지 않으면, 노드가 호스트에서 로컬 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. PSID 부속벡터는 이 필드에 입력되는 이름을 포함합니다. 이 필드는 이 명령을 다시 발행함으로써 동적으로 변경할 수 있습니다. 변경사항은 LU를 닫고 활성종료될 때까지 효력을 가지지 않습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_CANT_MODIFY_VISIBILITY

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_PU_NOT_DEFINED
AP_LU_NAME_POOL_NAME_CLASH
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD

시스템에 종속 LU 지원이 내장되어 있지 않아 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LU_0_TO_3_RANGE

이 명령은 지정된 NAU 범위 내의 복수 LU 정의를 가능하게 합니다. 노드 운영요원이 기본명과 NAU 범위를 제공합니다. LU 명은 기본명과 NAU 주소의 조합으로 생성됩니다. 이 명령은 기존의 정의를 수정하는 데에는 사용할 수 없습니다.

예를 들어, 기본명 LUNME와 NAU 범위 1-4를 지정하면, LU가 LUNME001, LUNME002, LUNME003 또는 LUNME004로 정의됩니다. non-pad 문자가 5개 미만인 기본명은 non-pad 문자가 8개 미만인 LU 명을 생성합니다. 이 경우, 퍼스널 통신이나 통신 서버가 오른쪽을 채워 8자가 되도록 합니다.

VCB 구조

형 D 1

```
typedef struct define_lu_0_to_3_range
{
        unsigned short  opcode;                 /* verb operation code        */
        unsigned char   attributes;             /* verb attributes            */
        unsigned char   format;                 /* format                     */
        unsigned short  primary_rc;             /* primary return code        */
        unsigned long   secondary_rc;           /* secondary return code      */
        unsigned char   base_name[5];           /* base name                  */
        unsigned char   reserv3;                /* reserved                   */
        unsigned char   description;            /* resource description       */
        unsigned char   min_nau;                /* minimum NAU address        */
        unsigned char   max_nau;                /* maximum NAU address        */
        unsigned char   pool_name[8];           /* LU pool name               */
        unsigned char   pu_name[8];             /* PU name                    */
        unsigned char   priority;               /* LU priority                */
        unsigned char   lu_model;               /* LU model                   */
        unsigned char   sscp_id[6];             /* SSCP ID                    */
        unsigned short  timeout;                /* Timeout                    */
        unsigned char   app_spec_def_data[16]; /* application specified data */
        unsigned char   model_name[7];          /* LU model name for DDDLU    */
        unsigned char   name_attributes;        /* Attributes of base name    */
        unsigned char   base_number;            /* Base number for LU names   */
        unsigned char   reserv3[15];            /* reserved                   */
} DEFINE_LU_0_TO_3_RANGE;
```

VCB 구조

형 D 0

```
typedef struct define_lu_0_to_3_range
{
        unsigned short  opcode;                 /* verb operation code        */
        unsigned char   attributes;             /* verb attributes            */
        unsigned char   format;                 /* format                     */
        unsigned short  primary_rc;             /* primary return code        */
        unsigned long   secondary_rc;           /* secondary return code      */
        unsigned char   base_name[5];           /* base name                  */
        unsigned char   reserv3;                /* reserved                   */
        unsigned char   description;            /* resource description       */
        unsigned char   min_nau;                /* minimum NAU address        */
        unsigned char   max_nau;                /* maximum NAU address        */
        unsigned char   pool_name[8];           /* LU pool name               */
        unsigned char   pu_name[8];             /* PU name                    */
        unsigned char   priority;               /* LU priority                */
        unsigned char   lu_model;               /* LU model                   */
```

```
        unsigned char   sscp_id[6];           /* SSCP ID                     */
        unsigned short  timeout;              /* Timeout                     */
        unsigned char   app_spec_def_data;    /* application specified data */
} DEFINE_LU_0_TO_3_RANGE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LU_0_TO_3_RANGE

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전 중 하나를 지정하려면, 이 필드를 0이나 1로 설정하십시오.

base_name

기본 LU 명. 5바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. NAU 범위 내에 있는 각 LU의 경우, 이 기본명에 세 개의 A 유형 EBCDIC 숫자가 추가되어 NAU 주소의 십진값을 나타냅니다.

이 필드는 **name_attributes** 필드에 비트가 설정되지 않은 필드입니다. 비트를 설정하면 이 필드의 의미가 바뀝니다.

description

자원 설명(QUERY_LU_0_TO_3에서 반환). 이 필드의 길이는 4의 배수(바이트)이어야 하며 0이 될 수 없습니다.

min_nau

범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

max_nau

범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

pool_name

이 LU가 속하는 LU 풀의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. LU가 풀에 속하지 않을 경우, 이 필드는 모두 2진 0으로 설정됩니다.

pu_name

이 LU가 사용하는 PU의 이름(DEFINE_LS 명령에 지정된). 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

priority

호스트로 전송할 때의 LU 우선순위. 다음 값 중 하나로 설정됩니다.

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

lu_model

LU의 모델 유형 및 번호. 다음 값 중 하나로 설정됩니다.

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

형식 1의 경우에만, **model_name**을 모두 2진 0으로 설정하지 않으면 이 필드가 무시됩니다.

호스트 시스템이 DDDLU(종속 LU의 동적 정의)를 지원할 경우 AP_UNKNOWN 이외의 값을 지정하면, 노드가 호스트에서 국지 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. 형식 1의 경우에만, PSID 부속벡터가 이 필드의 값과 일치하는 시스템 유형이나 모델 번호를 포함합니다. 이 필드는 이 명령을 다시 발행함으로써 동적으로 변경할 수 있습니다. 변경사항은 LU를 닫고 활성종료할 때까지 효력을 가지지 않습니다.

lu_0_to_3_detail.def_data.sscp_id

이 필드는 이 LU를 활성화할 수 있는 SSCP의 ID를 지정합니다. 6바이트 2진 필드입니다. 이 필드를 2진 0으로 설정하면, 모든 SSCP가 LU를 활성화할 수 있습니다.

lu_0_to_3_detail.def_data.timeout

초 단위로 지정하는 LU에 대한 시간종료. 시간종료 값이 제공되고 LU 사용자가 OPEN_LU_SSCP_SEC_RQ에(또는 PU 집중의 경우, 다운스트림 LU 정의에) **allow_timeout**을 지정한 경우, 이 기간 동안 PLU-SLU 세션이 활동중이 아닌 상태로 있으며 다음 조건 중 하나에 해당될 때 LU가 활성종료됩니다.

- 세션이 제한 자원 링크를 통과합니다.
- 세션이 다시 사용되기 전에 다른 응용프로그램이 LU를 사용하려고 합니다.

시간종료를 0으로 설정하면 LU가 활성종료되지 않습니다.

model_name

퍼스널 통신이나 통신 서버는 이 필드가 EBCDIC 문자 A-Z, 0–9 및 @, # 및 $로 구성되는지 점검합니다. 호스트 시스템이 SDDLU(Self-Defining 종속 LU)를 지원할 때 이 필드를 모두 2진 0으로 설정하지

않으면, 노드가 호스트에서 로컬 LU를 동적으로 정의하기 위해 요청되지 않은 PSID NMVT 응답을 생성합니다. PSID 부속벡터는 이 필드에 입력되는 이름을 포함합니다.

name_attributes
이 비트 필드는 제공되는 **base_name**의 해석과 사용법을 수정합니다. 이 필드는 0 값을 취하거나 다음 값 중 하나 또는 모두를 취할 수 있습니다(OR에 의해 비트와이즈로 연결).

AP_USE_HEX_IN_NAME
이 비트를 설정하면, **base_name**의 해석이 다음과 같이 수정됩니다.

6바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. NAU 범위 내에 있는 각 LU의 경우, 이 기본명에 두 개의 EBCDIC 문자가 추가되어 NAU 주소의 16진 값을 나타냅니다.

AP_USE_BASE_NUMBER
이 비트를 설정하면, **base_name**의 해석이 다음과 같이 수정됩니다.

5바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. 이 기본명은 세 개의 EBCDIC 숫자가 추가되어 범위 내에 있는 LU의 십진 색인을 나타내며, **base_number**로 시작되고(**base_name + max_nau — min_nau**)로 끝납니다.

base_number
name_attributes에 AP_USE_BASE_NUMBER 비트를 설정하지 않은 경우, 이 필드가 무시됩니다. **name_attributes**에 AP_USE_BASE_NUMBER 비트를 설정한 경우에는, 이 필드가 앞에서 설명한 **base_name**의 해석을 수정합니다. 유효한 값은 0-(255 — **max_nau + min_nau**)입니다.

app_spec_def_data
응용프로그램이 지정하는 정의된 데이터. 이 필드는 퍼스널 통신이나 통신 서버에 의해 해석되지는 않으나, 저장되었다가 QUERY_LU_0_TO_3 명령에서 반환됩니다.(범위 내의 각 LU에 대해 동일한 데이터가 반환됩니다.)

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_BASE_NUMBER

AP_INVALID_LU_MODEL
AP_INVALID_LU_NAME
AP_INVALID_NAME_ATTRIBUTES
AP_INVALID_NAU_ADDRESS
AP_INVALID_PRIORITY

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PU_NOT_DEFINED

AP_INVALID_PU_NAME
AP_INVALID_PU_TYPE
AP_LU_NAME_POOL_NAME_CLASH
AP_LU_ALREADY_DEFINED
AP_LU_NAU_ADDR_ALREADY_DEFD
AP_IMPLICIT_LU_DEFINED
AP_CANT_MODIFY_VISIBILITY

시스템에 종속 LU 지원이 내장되어 있지 않아서 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LU_POOL

이 명령은 LU 풀을 정의하거나 기존 풀에 LU를 추가하는 데 사용됩니다. 추가할 LU는 DEFINE_LU_0_TO_3 명령이나 DEFINE_LU_0_TO_3_RANGE 명령으로 미리 정의되어 있어야 합니다. LU는 한 번에 한 개의 LU 풀에만 속할 수 있습니다. 지정한 LU가 이미 한 풀에 속해 있는 경우에는, LU가 기존 풀에서 새로이 정의되는 풀로 이동됩니다. 풀이 포함할 수 있는 LU의 총 수에는 제한이 없으나, 한 번에 최대 10개의 LU를 풀에 추가할 수 있습니다.

VCB 구조

```
typedef struct define_lu_pool
{
        unsigned short  opcode;                 /* verb operation code        */
        unsigned char   attributes;             /* verb attributes            */
        unsigned char   format;                 /* format                     */
        unsigned short  primary_rc;             /* primary return code        */
        unsigned long   secondary_rc;           /* secondary return code      */
        unsigned char   pool_name[8];           /* LU pool name               */
        unsigned char   description[RD_LEN];    /* resource description       */
        unsigned char   reserv3[4];             /* reserved                   */
        unsigned short  num_lus;                /* number of LUs to add       */
        unsigned char   lu_names[10][8]; /* LU names                          */
} DEFINE_LU_POOL;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LU_POOL

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pool_name

LU가 속하는 풀의 이름. 이 이름은 8바이트 문자열로, 오른쪽이 공백으로 채워집니다. EBCDIC 문자열이거나 국지로 표시가능한 문자 세트로 된 문자열일 수 있습니다.

description

자원 설명(QUERY_LU_POOL에서 반환). 이 필드의 길이는 4의 배수(바이트)이어야 하며 0이 될 수 없습니다.

num_lus

0-10 범위의 추가할 LU 수.

lu_names
풀에 추가할 LU의 이름. 각 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_NUM_LUS
AP_INVALID_POOL_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_LU_NAME_POOL_NAME_CLASH

AP_INVALID_POOL_NAME

시스템에 종속 LU 지원이 내장되어 있지 않아서 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_MODE

DEFINE_MODE 명령은 특정 모드(또는 세션 그룹)에 지정할 네트워킹 특성 집합을 정의합니다. 이 명령은 또한 이전에 정의된 모드 상의 필드를 수정하는 데에도 사용할 수 있습니다. SNASVCMG 모드를 재정의한 경우에는, **mode_name** 및 **cos_name**을 수정할 수 없습니다. CPSVCMG 모드는 재정의할 수 없습니다.

DEFINE_MODE 명령은 또한 알 수 없는 모드가 매핑되는 생략시 COS를 정의하는 데에도 사용할 수 있습니다. 이러한 작업은 **mode_name**을 모두 0으로 설정함으로써 수행됩니다. 생략시 COS의 초기값은 #CONNECT입니다.

주: 국지로 사용하려는 모드를 사용자의 네트워크 노드와 잠정적으로 상대방 노드에 정의해야 하나, 모든 모드를 다 정의할 필요는 없습니다. 정의되지 않은 모드를 지정하여 ALLOCATE를 발행할 경우, 노드는 DEFINE_DEFAULTS 명령에 지정된 모델 생략시 모드의 특성을 사용합니다. 그러한 모델이 지정되어 있지 않을 경우에는 공백 모드의 특성이 해당 모델에 사용됩니다.

VCB 구조

```
typedef struct define_mode
{
        unsigned short  opcode;                 /* verb operation code          */
        unsigned char   reserv2;                /* reserved                     */
        unsigned char   format;                 /* format                       */
        unsigned short  primary_rc;             /* primary return code          */
        unsigned long   secondary_rc;           /* secondary return code        */
        unsigned char   mode_name[8];           /* mode name                    */
        unsigned short  reserv3;                /* reserved                     */
        MODE_CHARS      mode_chars;             /* mode characteristics         */
} DEFINE_MODE;

typedef struct mode_chars
{
        unsigned char   description[RD_LEN]
                                                /* resource description         */
        unsigned short  max_ru_size_upp;        /* max RU size upper bound      */
        unsigned char   receive_pacing_win;     /* receive pacing window        */
        unsigned char   default_ru_size;        /* default RU size to maximize  */
                                                /* performance                  */
        unsigned short  max_neg_sess_lim;       /* max negotiable session limit */
        unsigned short  plu_mode_session_limit; /* LU-mode session limit        */
        unsigned short  min_conwin_src;         /* min source contention winner */
                                                /* sessions                     */
        unsigned char   cos_name[8];            /* class-of-service name        */
        unsigned char   cryptography;           /* cryptography                 */
        unsigned char   compression;            /* compression                  */
        unsigned short  auto_act;               /* initial auto-activation count*/
        unsigned short  min_conloser_src;       /* min source contention loser  */
        unsigned short  max_ru_size_low         /* maximum RU size lower bound  */
        unsigned short  max_receive_pacing_win;
                                                /* maximum receive pacing window*/
        unsigned char   max_compress_lvl;       /* maximum compression level    */
        unsigned char   max_decompression_lvl;  /* maximum decompression level  */
        unsigned char   comp_in_series;         /* support for LZ and RLE       */
        unsigned char   reserv4[24];            /* reserved                     */
} MODE_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DEFINE_MODE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0이나 1로 설정하십시오.

mode_name
모드 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. 이 필드를 모두 0으로 설정하면, 디폴트 COS가 **mode_chars.cos_name**으로 설정되고 다른 모든 **mode_chars** 필드가 무시됩니다.

mode_chars.compression
이 모드를 사용하여 활성화된 세션에 대한 압축 사용을 지정합니다.

AP_COMP_PROHIBITED
이 모드의 세션에서 RLE 압축이 지원되지 않습니다.

AP_COMP_REQUESTED
이 모드의 세션에서 RLE 압축이 지원되고 요청됩니다(요구 사항이 아님).

mode_chars.max_ru_size_upp
이 모드의 세션에서 송수신되는 RU의 최대 크기에 대한 상한선. 이 값은 세션 활성화시 최대 RU 크기를 조정하는 데 사용됩니다. 범위는 256—61440입니다. **default_ru_size**를 AP_YES로 설정한 경우에는 이 필드가 무시됩니다.

mode_chars.receive_pacing_win
이 모드의 세션에 대한 세션 페이싱 창. 고정 페이싱의 경우, 이 값은 수신 페이싱 창을 지정합니다. 적합 페이싱의 경우, 이 값은 선호 최소 창 크기로 사용됩니다. 인접 노드가 적합 페이싱을 지원하지 않는 것으로 지정하지 않는 한, 퍼스널 통신이나 통신 서버가 항상 적합 페이싱을 사용함에 주의하십시오. 범위는 1—63입니다. 0 값은 허용되지 않습니다.

mode_chars.default_ru_size
최대 RU 크기의 생략시 상한선이 사용되는지의 여부를 지정합니다. 이 매개변수가 AP_YES를 지정할 경우, **max_ru_size_upp**가 무시되며 최대 RU 크기의 상한선이 링크 BTU 크기에서 TH와 RH의 크기를 뺀 값으로 설정됩니다.

AP_YES
AP_NO

mode_chars.max_neg_sess_lim

이 모드에서 허용되는 국지 LU와 상대방 LU 사이의 최대 세션 수. 0 값을 지정하면, 암시적 CNOS 교환이 수행되지 않습니다. 범위는 0—32 767입니다.

mode_chars.plu_mode_session_limit

이 모드의 생략시 세션 한계. 이 모드에서 허용되는 로컬 LU와 상대방 LU 사이의 세션 수를 제한합니다. 이 값은 CNOS(세션 수 변경) 교환이 암시적으로 시작될 때 사용됩니다. 0 값을 지정하면, 암시적 CNOS 교환이 수행되지 않습니다. 범위는 0—32 767입니다.

mode_chars.min_conwin_src

이 모드를 사용하는 하나의 국지 LU가 활성화할 수 있는 회선경합 성공 세션 최대 수. 이 값은 CNOS(세션 수 변경) 교환이 암시적으로 시작될 때 사용됩니다. 0 값을 지정하면, 암시적 CNOS 교환이 수행되지 않습니다. 범위는 0—32 767입니다.

mode_chars.cos_name

이 모드에서 세션을 활성화할 때 요청할 서비스 클래스(COS)의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

mode_chars.cryptography

세션 레벨 암호화를 사용해야 하는지의 여부를 지정합니다(AP_NONE 또는 AP_MANDATORY).

mode_chars.compression

이 모드를 사용하여 활성화한 세션에 대한 압축 사용을 지정합니다.

AP_COMP_PROHIBITED

이 모드의 세션에서 압축이 지원되지 않습니다.

AP_COMP_REQUESTED

이 모드의 세션에서 압축이 지원되고 요청됩니다(요구 사항이 아님).

format 필드를 0으로 설정하면, 압축 및 압축해제 레벨이 노드가 지원하는 최대값으로 설정됩니다.

format 필드를 1로 설정하면, 압축이나 압축해제의 최대 레벨이 **max_compress_lvl** 및 **max_decompress_lvl** 필드에 의해 정의됩니다.

mode_chars.auto_act

이 모드에서 자동 활성화되는 세션 수를 지정합니다. 이 값은 세션 수 변경(CNOS) 교환이 암시적으로 시작될 때 사용됩니다.

범위는 0–32767입니다.

mode_chars.min_consloser_src

이 모드를 사용하는 하나의 국지 LU가 활성화할 수 있는 회선경합 실패 세션 최대 수를 지정합니다. 이 값은 CNOS(세션 수 변경) 교환이 암시적으로 개시될 때 사용됩니다. 범위는 0–32767입니다.

mode_chars.max_ru_size_low

이 모드의 세션에서 전송하거나 수신되는 RU의 최대 크기에 대한 하한선. 이 값은 세션 활성화시 최대 RU 크기를 조정하는 데 사용됩니다. 범위는 256–61140입니다.

0 값은 하한선이 없음을 의미합니다.

default_ru_size를 AP_YES로 설정한 경우에는 이 필드가 무시됩니다.

mode_chars.max_receive_pacing_win

이 모드의 세션에 대한 최대 페이싱 창을 지정합니다. 적합 페이싱의 경우, 이 값은 수신 페이싱 창을 제한하는 데 사용됩니다. 고정 페이싱의 경우에는 이 필드가 사용되지 않습니다. 인접 노드가 적합 페이싱을 지원하지 않는 것으로 지정하지 않는 한, 프로그램은 항상 적합 페이싱을 사용함에 주의하십시오. 범위는 0–32767입니다.

0 값은 상한선이 없음을 의미합니다.

mode_chars.max_compress_lvl

프로그램이 노드에서 지원하는 데이터 흐름을 위해 조정하려고 시도하는 최대 압축 레벨.

AP_NONE
AP_RLE_COMPRESSION
AP_LZ9_COMPRESSION
AP_LZ10_COMPRESSION
AP_LZ12_COMPRESSION

구성된 압축 레벨은 노드가 지원하는 압축 레벨(START_NODE의 **max_compress_lvl** 필드에 지정된) 이하이어야 합니다. 비확장 BIND로 압축을 조정할 경우, 압축 레벨은 RLE 압축으로 설정됩니다.

mode_chars.max_decompress_lvl

프로그램이 노드에서 지원하는 데이터 흐름을 위해 조정하려고 시도하는 최대 압축해제 레벨.

AP_NONE
AP_RLE_COMPRESSION
AP_LZ9_COMPRESSION
AP_LZ10_COMPRESSION
AP_LZ12_COMPRESSION

구성된 압축 레벨은 노드가 지원하는 압축 레벨(START_NODE의 **max_compress_lvl** 필드에 지정된) 이하이어야 합니다. 비확장 BIND로 압축을 조정할 경우, 압축해제 레벨은 LZ9 압축으로 설정됩니다.

mode_chars.comp_in_series

RLE 압축이 선행되는 LZ 압축을 사용할 수 있는지의 여부를 지정합니다. 이 필드를 AP_YES로 설정할 경우에는 **max_compress_lvl**을 AP_LZ9_COMPRESSION, AP_LZ10_COMPRESSION 또는 AP_LZ12_COMPRESSION으로 설정해야 합니다.

AP_YES

AP_NO

노드가 RLE 및 LZ 압축을 지원하지 않는 것으로 구성되어 있는 경우에는(START_NODE의 **comp_in_series** 필드에 지정), 이 필드를 AP_YES로 설정할 수 없습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_COS_NAME

AP_CPSVCMG_ALREADY_DEFD
AP_INVALID_CNOS_SLIM
AP_INVALID_COS_SNASVCMG_MODE
AP_INVALID_DEFAULT_RU_SIZE
AP_INVALID_MAX_NEGOT_SESS_LIM
AP_INVALID_MAX_RU_SIZE_UPPER
AP_INVALID_MAX_RU_SIZE_LOW
AP_RU_SIZE_LOW_UPPER_MISMATCH
AP_INVALID_COMPRESSION
AP_INVALID_MIN_CONWINNERS
AP_INVALID_MIN_CONLOSERS
AP_INVALID_MIN_CONTENTION_SUM
AP_INVALID_MODE_NAME
AP_INVALID_RECV_PACING_WINDOW
AP_INVALID_MAX_RECV_PACING_WIN
AP_INVALID_DEFAULT_RU_SIZES
AP_INVALID_SNASVCMG_MODE_LIMIT
AP_MODE_SESS_LIM_EXCEEDS_NEG
AP_INVALID_CRYPTOGRAPHY
AP_INVALID_MAX_COMPRESS_LVL
AP_INVALID_MAX_DECOMPRESS_LVL
AP_INVALID_COMP_IN_SERIES

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

재정의 결과

각 필드를 재정의한 경우의 결과는 다음과 같습니다.

description
갱신된 **description**은 차후의 QUERY_MODE 명령에서 반환됩니다.

compression

max_compress_lvl

max_decompress_lvl

comp_in_series

cryptography

max_ru_size_upp

receive_pacing_win

default_ru_size

max_ru_size_low

max_receive_pacing_win
갱신된 값은 이 모드에 대한 차후의 모든 세션 활성화 시도에 사용되며, 모든 후속 QUERY_MODE 명령에서 반환됩니다. 변경사항은 기존의 활동중인 세션에는 영향을 미치지 않습니다.

max_neg_sess_lim

plu_mode_session_limit

min_conwin_src

auto_act

min_conloser_src
갱신된 값은 그 다음 CNOS 명령이 시작될 때까지(국지 또는 원격으로 시작) 특정 국지 LU나 상대방 LU 쌍에는 사용되지 않습니다. 이전 값은 그 다음 CNOS 명령이 시작될 때까지 QUERY_MODE 명령에서 반환됩니다.

cos_name

갱신된 값은 이 모드에 대한 뒤에 오는 모든 세션 활성화 시도에 사용되며, 모든 후속 QUERY_MODE 명령에서 반환됩니다. 변경사항은 기존의 활동중인 세션에는 영향을 미치지 않습니다. 갱신된 값은 또한 뒤에 오는 모드-COS 매핑 작업에 사용되며(예를 들어, 이 노드가 네트워크 노드이고 모드-COS 매핑 서비스나 이 노드에서 서비스를 제공하는 끝 노드를 제공할 경우), 차후의 모든 QUERY_MODE_TO_COS_MAPPING 명령에서 반환됩니다.

주: 암시적 정의는 DEFINE_MODE에 의해 명시적으로 될 수 있습니다. 이는 **implicit set**가 AP_NO로 설정된 상태로 반환되는 차후의 QUERY_MODE 명령에서 반영됩니다.

DEFINE_PARTNER_LU

DEFINE_PARTNER_LU 명령은 국지 LU와 상대방 LU 사이의 LU-LU 세션을 위한 상대방 LU의 매개변수를 정의합니다. 선택적으로 DEFINE_PARTNER_LU를 사용하여 상대방 LU에 대해 이미 정의되어 있는 모든 매개변수(**fqplu_name**과 **plu_alias** 제외)를 수정할 수 있습니다.

VCB 구조

```
typedef struct define_partner_lu
{
        unsigned short  opcode;              /* verb operation code          */
        unsigned char   reserv2;             /* reserved                     */
        unsigned char   format;              /* format                       */
        unsigned short  primary_rc;          /* primary return code          */
        unsigned long   secondary_rc;        /* secondary return code        */
        PLU_CHARS       plu_chars;           /* partner LU characteristics   */
} DEFINE_PARTNER_LU;

typedef struct plu_chars
{
        unsigned char   fqplu_name[17];      /* fully qualified partner      */
                                             /* LU name                      */
        unsigned char   plu_alias[8];        /* partner LU alias             */
        unsigned char   description[RD_LEN];
                                             /* resource description         */
        unsigned char   plu_un_name[8];      /* partner LU uninterpreted name */
        unsigned char   preference           /* routing preference           */
        unsigned short  max_mc_ll_send_size; /* max MC send LL size          */
        unsigned char   conv_security_ver;   /* already_verified accepted?   */
        unsigned char   parallel_sess_supp;  /* parallel sessions supported? */
        unsigned char   reserv2[8];          /* reserved                     */
} PLU_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_PARTNER_LU

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

plu_chars.fqplu_name

상대방 LU의 전체 정식명. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

plu_chars.plu_alias

상대방 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8 바이트 문자열입니다. 연관된 별명이 없는 상대방 LU에 대해서는 이 필드를 모두 0으로 설정할 수 있습니다.

plu_chars.description

자원 설명(QUERY_PARTNER_LU 및

QUERY_PARTNER_LU_DEFINITION에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

plu_chars.plu_un_name
상대방 LU의 미해석 이름. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다.

plu_chars.max_mc_ll_send_size
상대방 LU에서 직접 대화 서비스에 의해 송수신되는 LL 레코드의 최대 크기. 범위: 1–32 767(32 767은 이 필드를 0으로 설정할 경우 지정됩니다).

plu_chars.preference
이 상대방 LU에 대한 세션 활성화에 사용되는 preferred 경로지정 프로토콜. 이 필드는 다음 값을 취할 수 있습니다.

AP_NATIVE
원시(APPN) 경로지정 프로토콜만을 사용하십시오.

AP_NONNATIVE
비원시(AnyNet) 프로토콜만을 사용하십시오.

AP_NATIVE_THEN_NONNATIVE
원시(APPN) 프로토콜을 시도하고, 상대방 LU를 찾을 수 없을 경우에는 비원시(AnyNet) 프로토콜을 사용하여 세션 활성화를 재시도하십시오.

AP_NONNATIVE_THEN_NATIVE
비원시(AnyNet) 프로토콜을 시도하고, 상대방 LU를 찾을 수 없을 경우에는 원시(APPN) 프로토콜을 사용하여 세션 활성화를 재시도하십시오.

AP_USE_DEFAULT_PREFERENCE
노드 시작시에 정의된 생략시 선택사항을 사용하십시오(QUERY_NODE로 다시 호출할 수 있습니다).

주: 비원시 경로지정은 노드 연산자 기능에서 AnyNet DLC를 사용할 수 있고 AnyNet 링크 스테이션이 정의되어 있는 경우에만 유효합니다.(Defined_LS를 참조하십시오.)

plu_chars.conv_security_ver
상대방 LU가 국지 LU를 대신하여 **user_ids**의 유효성을 검사할 권한이 있는지, 즉 상대방 LU가 접속 요청에 이미 검증된 표시기를 설정할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

plu_chars.parallel_sess_supp
상대방 LU가 병렬 세션을 지원하는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_ANYNET_NOT_SUPPORTED

AP_DEF_PLU_INVALID_FQ_NAME
AP_INVALID_UNINT_PLU_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PLU_ALIAS_CANT_BE_CHANGED

AP_PLU_ALIAS_ALREADY_USED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

재정의 결과

각 필드를 재정의한 경우의 결과는 다음과 같습니다.

fqplu_name
변경할 수 없습니다.

plu_alias

다른 **plu_alias**로 이전의 DEFINE_PARTNER_LU를 발행한 경우, DEFINE_PARTNER_LU가 실패합니다. **plu_alias**를 모두 0으로 설정하여 이전의 DEFINE_PARTNER_LU를 발행한 경우에는, 재정의가 받아들여지고 기존의 모든 PLU 레코드에 영향을 미칩니다. 이전의 DEFINE_PARTNER_LU를 발행하지 않은 경우, **plu_alias**를 모두 0으로 지정하지 않는 한 지정된 값이 암시적으로 정의된 해당되는 모든 상대방 LU 레코드로 복사됩니다. 이 필드를 모두 0으로 지정할 경우에는 암시적 **plu_alias**가 변경되지 않은 상태로 있습니다.

주: 실행중인 응용프로그램이 이전의 APPC 명령에서 이미 암시적 **plu_alias**를 얻어서 다음 ALLOCATE에서 이를 사용할 경우, non-zero **plu_alias**로 DEFINE_PARTNER_LU를 발행하면 응용프로그램이 실패할 수 있습니다.

description

갱신된 **description**이 뒤에 오는 QUERY_PARTNER_LU 명령에서 반환됩니다.

plu_un_name

갱신된 **plu_un_name**이 이 상대방 LU에 대한 뒤에 오는 모든 세션 활성화 요청에 사용되며, 모든 후속 QUERY_PARTNER_LU 명령에서 반환됩니다.

preference

갱신된 **preference**가 이 상대방 LU에 대한 뒤에 오는 모든 세션 활성화 요청에 사용되며, 뒤에 오는 모든 QUERY_PARTNER_LU 명령에서 반환됩니다.

max_mc_ll_send_size

갱신된 **선택사항**이 이 상대방 LU에 대한 뒤에 오는 모든 세션 활성화 요청에 사용됩니다. 변경사항은 기존의 대화에는 영향을 미치지 않습니다. 갱신된 값은 뒤에 오는 모든 QUERY_PARTNER_LU 명령에서 반환됩니다.

conv_security_ver

특정 국지 LU와 상대방 LU 사이의 세션 수가 0이 될 때까지 갱신된 값이 그 국지 LU에 사용되지 않습니다. BIND 및 RSP(BIND)는 이전의 설정을 사용하여 흐르게 되며, 이전 값은 세션 수가 0이 될 때까지 QUERY_PARTNER_LU 요청에서 반환됩니다. 이는 보안 지원이 활동중인 기존 세션의 보안 지원과 다를 경우 상대방 LU가 뒤에 오는 세션 활성화 시도를 거부할 수 있기 때문입니다.

parallel_sess_supp

conv_security_ver에서와 같이, 특정 로컬 LU와 지정된 상대방 LU 사이의 세션 수가 0이 될 때까지 갱신된 값이 그 국지 LU에 사용되지 않습니다. 이는 구조화된 LU6.2 세션 일관성 점검의 문제점을 피하기 위해서입니다.

주: 암시적 정의는 DEFINE_PARTNER_LU에 의해 명시적으로 될 수 있습니다. 이는 **implicit set**가 AP_NO로 설정된 상태로 반환되는 뒤에 오는 QUERY_PARTNER_LU 명령에서 반영됩니다.

DEFINE_PORT

DEFINE_PORT는 새로운 포트를 정의하거나 기존의 포트를 수정합니다. 이 포트는 지정된 DLC에 속하며, 이 DLC는 DEFINE_DLC 명령으로 이미 정의되어 있어야 합니다. DEFINE_PORT 명령은 노드 전반에 걸쳐 고유한 포트 명과 포트 고유 매개변수, 그리고 동적 링크 스테이션에서 사용할 디폴트 LS 특성을 제공합니다. 포트 고유 매개변수는 기본 구조에 연결됩니다. 생략 시 LS 특성은 포트 고유 매개변수 바로 다음에 연결됩니다.

포트가 재설정 상태이며(STOP_PORT를 발행한 후) 포트의 이전 정의 이후로 DEFINE_PORT에 지정한 **dlc_name**이 변경되지 않은 경우, DEFINE_PORT를 사용하여 기존 포트의 하나 이상의 필드를 수정할 수 있습니다.

포트가 활동중이면 다음 필드만을 수정할 수 있습니다.

description
implicit_dspu_services
implicit_deact_timer
implicit_cp_cp_sess_support
implicit_link_lvl_error
default_tg_chars
implicit_dspu_template
implicit_ls_limit
link_spec_data_len
link_spec_data

포트가 활동중일 때 포트 스펙 데이터가 변경되면, 이 명령이 거부되지는 않으나 수정이 무시됩니다.

DLC와 포트, 링크 스테이션 사이의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

VCB 구조

```
typedef struct define_port
{
        unsigned short  opcode;              /* verb operation code          */
        unsigned char   attributes;          /* verb attributes              */
        unsigned char   format;              /* format                       */
        unsigned short  primary_rc;          /* primary return code          */
        unsigned long   secondary_rc;        /* secondary return code        */
        unsigned char   port_name[8];        /* name of port                 */
        PORT_DEF_DATA   def_data;            /* port defined data            */
} DEFINE_PORT;

typedef struct port_def_data
{
        unsigned char   description;          /* resource description        */
        unsigned char   dlc_name[8];         /* DLC name associated with port */
        unsigned char   port_type;           /* port type                    */
        unsigned char   port_attributes[4];  /* port attributes              */
        unsigned char   implicit_uplink_to_en;/* Implicit links to EN are    */
                                             /* uplink                       */
        unsigned char   reserv3[2];          /* reserved                     */
        unsigned long   port_number;         /* port number                  */
```

```
        unsigned short  max_rcv_btu_size;     /* max receive BTU size          */
        unsigned short  tot_link_act_lim;     /* total link activation limit   */
        unsigned short  inb_link_act_lim;     /* inbound link activation limit */
        unsigned short  out_link_act_lim;     /* outbound link activation      */
                                              /* limit                         */
        unsigned char   ls_role;              /* initial link station role     */
        unsigned char   retry_flags;          /* conditions for automatic      */
                                              /* retries                       */
        usigned char    max_activation_attempts;
                                              /* how many automatic retries?   */
        unsigned char   adtivation_delay_timer;
                                              /* delay between automatic       */
                                              /* retries                       */
        unsigned char   reserv1[10];         /* reserved                       */
        unsigned char   implicit_dspu_template[8];
                                              /* reserved                      */
        unsigned char   implicit_ls_limit;    /* max number of implicit links  */
        unsigned char   reserv2;              /* reserved                      */
        unsigned char   implicit_dspu_services;
                                              /* implicit links support DSPUs  */
        unsigned char   implicit_deact_timer; /* Implicit link HPR link        */
                                              /* deactivation timer            */
        unsigned short  act_xid_exchange_limit;
                                              /* act.  XID exchange limit      */
        unsigned short  nonact_xid_exchange_limit;
                                              /* nonact.  XID exchange limit   */
        unsigned char   ls_xmit_rcv_cap;      /* LS transmit-receive           */
                                              /* capability                    */
        unsigned char   max_ifrm_rcvd;        /* max number of I-frames that   */
                                              /* can be received               */
        unsigned short  target_pacing_count;  /* Target pacing count           */
        unsigned short  max_send_btu_size;    /* Desired max send BTU size     */
        LINK_ADDRESS    dlc_data;             /* DLC data                      */
        LINK_ADDRESS    hpr_dlc_data;         /* HPR DLC data                  */
        unsigned char   implicit_cp_cp_sess_support;
                                              /* Implicit links allow CP-CP    */
                                              /* sessions                      */
        unsigned char   implicit_limited_resource;
                                              /* Implicit links are limited    */
                                              /* resource                      */
        unsigned char   implicit_hpr_support;
                                              /* Implicit links support HPR    */
        unsigned char   implicit_link_lvl_error;
                                              /* Implicit links support HPR    */
                                              /* link-level error recovery     */
        unsigned char   retired1;             /* reserved                      */
        TG_DEFINED_CHARS default_tg_chars;    /* Default TG chars              */
        unsigned char   discovery_support     /* Discovery function            */
                                              /* supported?                    */
        unsigned short  port_spec_data_len;   /* length of port spec data      */
        unsigned short  link_spec_data_len;   /* length of link spec data      */
} PORT_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;               /* length                        */
        unsigned short  reserve1;             /* reserved                      */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                              /* address                       */
} LINK_ADDRESS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_PORT

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

port_name
정의할 포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

def_data.description
자원 설명(QUERY_PORT에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

def_data.dlc_name
연관된 DLC의 이름으로, 국지로 표시가능한 문자 세트로 된 8 바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. 이 명명된 DLC는 DEFINE_DLC 명령으로 이미 정의되어 있어야 합니다.

def_data.port_type
포트에서 사용하는 회선의 유형을 지정합니다. 이 값은 다음 회선 유형 중 하나와 일치합니다.

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

이 필드를 AP_PORT_SATF로 설정할 경우에는 **ls_role**을 AP_LS_NEG로 설정해야 합니다.

def_data.port_attributes[0]
비트 필드입니다. AP_NO 값 또는 다음을 취할 수 있습니다.

AP_RESOLVE_BY_LINK_ADDRESS
수신된 XID3에서 전달되는 CP 명(또는 노드 ID)을 사용하여 입력 호출을 해결하기 전에 CONNECT_IN에 있는 링크 주소를 사용하여 입력 호출을 해결하려고 시도함을 지정합니다. **port_type** 필드를 AP_PORT_SWITCHED로 설정하지 않은 경우에는 이 비트가 무시됩니다.

def_data.implicit_uplink_to_en
BrNN 전용: 인접 노드가 끝 노드일 경우, 이 포트 상에 있지 않는 암

시적 링크 스테이션이 업링크인지 또는 다운링크인지를 지정합니다. 이 필드의 값은 링크 유형으로 사용되는 링크와 같은 동일 상대방이 존재하지않을 경우로 간주됩니다.

AP_NO
암시적 링크가 다운링크입니다.

AP_YES
암시적 링크가 업링크입니다.

다른 노드 유형: 이 필드가 무시됩니다.

def_data.port_number
포트 번호.

def_data.tot_link_act_lim
총 링크 활성화 한계. 동시에 활동할 수 있는 링크 스테이션의 최대 수를 지정합니다. 이 값은 **inb_link_act_lim** 필드와 **out_link_act_lim** 필드를 합한 값 이상이어야 합니다. **port_type**을 AP_PORT_NONSWITCHED로 설정하고 **ls_role**을 AP_LS_NEG 또는 AP_LS_SEC로 설정한 경우, 이 필드를 1로 설정해야 합니다. **ls_role**을 AP_LS_PRI로 설정한 경우에는, 이 필드의 값 범위가 1-256이어야 합니다. 이 포트가 AnyNet DLC용 포트이면 65535를 사용해야 합니다.

def_data.inb_link_act_lim
인바운드 링크 활성화 한계. 이 포트에서의 인바운드 활성화를 위해 예약된 링크 스테이션 수를 지정합니다. 따라서, 동시에 활동할 수 있는 아웃바운드 링크 스테이션의 최대 수는 **def_data.tot_link_act_lim - def_data.inb_link_act_lim**입니다. **port_type**을 AP_PORT_NONSWITCHED로 설정하고 **ls_role**을 AP_LS_NEG 또는 AP_LS_PRI로 설정한 경우, 이 필드를 0으로 설정해야 합니다. **port_type**을 AP_PORT_NONSWITCHED로 설정하고 **ls_role**을 AP_LS_SEC로 설정한 경우에는, 이 필드를 0이나 1로 설정해야 합니다. 이 포트가 AnyNet DLC용 포트이면 0을 사용해야 합니다.

def_data.out_link_act_lim
아웃바운드 링크 활성화 한계. 이 포트에서의 아웃바운드 활성화를 위해 예약된 링크 스테이션 수를 지정합니다. 따라서, 동시에 활동할 수 있는 인바운드 링크 스테이션의 최대 수는 **def_data.tot_link_act_lim - def_data.out_link_act_lim**입니다. **port_type**을 AP_PORT_NONSWITCHED로 설정하고 **ls_role**을 AP_LS_NEG로 설정한 경우, 이 필드를 0으로 설정해야 합니다. **ls_role**을 AP_LS_PRI로 설정한 경우에는, 이 필드가 **tot_link_act_lim**과 일치해야 합니다. **port_type**을 AP_PORT_NONSWITCHED로 설정하고 **ls_role**을 AP_LS_SEC로 설정한 경우에는, 이 필드를 0이나 1로 설정해야 합니다. 이 포트가 AnyNet DLC용 포트이면 0을 사용해야 합니다.

def_data.ls_role
링크 스테이션 역할. 조정가능(AP_LS_NEG), 1차(AP_LS_PRI) 또는 2

차(AP_LS_SEC)가 될 수 있습니다. 링크 스테이션 역할은 위에서 설명한 것과 같이 **tot_act_lim**, **inb_link_act_lim** 및 **out_link_act_lim** 필드로 지정된 값들 사이의 관계를 결정합니다. **port_type**을 AP_PORT_SATF로 설정한 경우에는 **ls_role**를 AP_LS_NEG로 설정해야 합니다.

def_data.retry_flags

이 필드는 DEFINE_LS의 **def_data.retry_flags**에 AP_INHERIT_RETRY 플래그가 설정되어 있을 경우 이 링크 스테이션의 활성화가 자동 재시도되는 조건을 지정합니다. 비트 필드이며, 다음 값을 취할 수 있습니다(OR에 의해 비트와이즈로 연결).

AP_RETRY_ON_START

활성화를 시도할 때 원격 노드에서 응답이 수신되지 않을 경우 링크 활성화를 재시도합니다. 활성화를 시도할 때 기본 포트가 활동중이 아닌 경우에는 APPN이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_FAILURE

링크가 활동중에 또는 활성화 미정 상태에서 실패할 경우, 링크 활성화가 재시도됩니다. 활성화를 시도할 때 기본 포트가 실패한 경우, APPN이 이를 활성화하려고 시도합니다.

AP_RETRY_ON_DISCONNECT

링크가 원격 노드에 의해 정상적으로 종료될 경우 링크 활성화를 재시도합니다.

AP_DELAY_APPLICATION_RETRIES

응용프로그램에 의해 시작되는(START_LS 또는 요구가 있을 때 링크 활성화에 의해) 링크 활성화 재시도는 **activation_delay_timer**로 페이싱됩니다.

AP_INHERIT_RETRY

이 필드에 플래그로 지정된 재시도 조건 이외에도, 기본 포트 정의의 **retry_flags** 필드에 지정된 재시도 조건 역시 사용됩니다.

def_data.max_activation_attempts

DEFINE_LS의 **def_data.retry_flags**에 하나 이상의 플래그를 설정하지 않고 DEFINE_LS의 **def_data.max_activation_attempts**를 AP_USE_DEFAULTS로 설정하지 않으면, 이 필드는 아무런 효력을 가지지 않습니다.

이 필드는 원격 노드가 응답하지 않거나 기본 포트가 활동중이 아닐 때 프로그램이 허용하는 재시도 수를 지정합니다. 여기에는 자동 재시도 수 및 응용프로그램에 의해 활성화 시도 수 둘다 포함됩니다.

이 한계값에 달하면, 자동으로 재시도하려는 시도가 더이상 행해지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는

OPEN_LU_SSCP_SEC_RQ는 활성화 시도가 한 번만 행해지도록 하며, 활성화에 실패해도 재시도가 행해지지 않습니다.

0은 '제한 없음'을 의미합니다. AP_USE_DEFAULTS 값을 지정하면, DEFINE_DLC에서 제공되는 **max_activiation_attempts**가 사용됩니다.

def_data.activation_delay_timer

DEFINE_LS의 **def_data.retry_flags**에 하나 이상의 플래그를 설정하지 않고 DEFINE_LS의 **activiation_delay_timer**를 AP_USE_DEFAULTS로 설정하지 않으면, 이 필드는 아무런 효력을 가지지 않습니다.

이 필드는 자동 재시도 사이, 또는 **def_data.retry_flags** 에 AP_DELAY_APPLICATION_RETRIES 비트가 설정되어 있을 경우 응용프로그램에 의한 활성화 시도간에 프로그램이 대기하는 시간(초 수)을 지정합니다.

AP_USE_DEFAULTS 값을 지정하면, DEFINE_DLC에서 제공되는 **activiation_delay_timer**가 사용됩니다.

0을 지정하면, 프로그램이 생략시 타이머 지속시간 30초를 사용합니다.

def_data.implicit_dspu_template

국지 노드가 이 포트에서 활성화되는 암시적 링크에 PU 집중을 제공할 경우의 정의에 사용되며 DEFINE_DSPU_TEMPLATE 명령으로 정의되는 DSPU 템플리트를 지정합니다. 링크가 활성화될 때 지정한 템플리트가 없을 경우(또는 이미 인스턴스 한계에 달한 경우)에는 활성화에 실패합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

def_data.implicit_dspu_services 필드를 AP_PU_CONCENTRATION으로 설정하지 않은 경우에는 이 필드가 예약됩니다.

def_data.implicit_ls_limit

동적 링크와 Discovery를 위해 활성화되는 링크를 포함하여, 이 포트에서 동시에 활동할 수 있는 암시적 링크 스테이션의 최대 수를 지정합니다. 0 값은 제한 없음을 의미하며, AP_NO_IMPLICIT_LINKS 값은 암시적 링크가 허용되지 않음을 의미합니다.

def_data.implicit.dspu_services

국지 노드가 이 포트에서 활성화된 암시적 링크를 통해 다운스트림 PU에 제공하는 서비스를 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_DLUR

국지 노드가 다운스트림 PU에 DLUR 서비스를 제공합니다(DEFINE_DLUR_DEFAULTS 명령을 통해 구성된 생략시 DLUS 사용). 이 설정은 국지 노드가 네트워크 노드일 경우에만 유효합니다.

AP_PU_CONCENTRATION
국지 노드가 다운스트림 PU에 PU 집중을 제공합니다.(**def_data.implicit_dspu_template** 필드에 지정된 DSPU 템플리트가 지정한 대로 정의를 제위치에 놓습니다.)

AP_NONE
국지 노드가 이 다운스트림 PU에 서비스를 제공하지 않습니다.

def_data.implicit_deact_timer
제한 자원 링크 활성종료 타이머(초 단위). **implicit_limited_resource**를 AP_YES 또는 AP_NO_SESSIONS으로 설정하면, 이 타이머의 지속시간 동안 HPR 가능 암시적 링크를 통해 아무런 데이타도 흐르지 않을 경우 이 링크가 자동으로 활성종료되고, 어떠한 세션도 이 링크를 사용하지 않습니다.

implicit_limited_resource를 AP_INACTIVITY로 설정하면, 이 타이머의 지속시간 동안 암시적 링크를 통해 아무런 데이터도 흐르지 않을 경우 이 링크가 자동으로 활성종료됩니다.

값은 0–1000초 범위 내의 한 정수값입니다. 생략시는 10초입니다.

0을 지정하면, 생략시 값 30이 사용됩니다. 그러지 않을 경우, 최소값은 5입니다.(5보다 작은 값을 지정하면, 지정한 값이 무시되고 5가 사용됩니다.)**implicit_limited_resource**를 AP_NO로 설정하지 않을 경우에는 이 매개변수가 예약됨에 주의하십시오.

def_data.act_xid_exchange_limit
활성화 XID 교환 한계.

def_data.nonact_xid_exchange_limit
비 활성화 XID 교환 한계.

def_data.ls_xmit_rcv_cap
링크 스테이션 송/수신 기능을 지정합니다. 2중 동시(AP_LS_TWS)(양방향(전송) 또는 동시 양방향(전송)이라고도 함) 또는 2중 교대(AP_LS_TWA)(비동시 양방향(전송)이라고도 함)입니다.

def_data.max_ifrm_rcvd
확인 응답이 전송되기 전에 국지 링크 스테이션이 수신할 수 있는 I 프레임 최대 수. 범위는 1—127입니다.

def_data.target_pacing_count
이 TG에서의 BIND에 적절한 페이싱 창 크기를 지시하는 1 과 32 767 사이의 숫자값. 이 수는 고정 바인드 페이싱이 수행될 때에만 유효합니다. 퍼스널 통신이나 통신 서버에서는 현재 이 값을 사용하지 않습니다.

def_data.max_send_btu_size
이 링크 스테이션에서 전송할 수 있는 최대 BTU 크기. 이 값은 링크 스테이션 쌍간에 전송 가능한 최대 BTU 크기를 조정하는 데 사용할 수 있습니다. 포트에서 암시적 HPR 가능 링크를 지원하지 않

을 경우에는 이 필드를 99 이상의 한 값으로 설정해야 합니다. 포트에서 암시적 HPR 가능 링크를 지원하면, 이 필드를 768 이상의 한 값으로 설정해야 합니다.

def_data.dlc_data.length
포트 주소 길이.

def_data.dlc_data.address
포트 주소.

def_data.hpr_dlc_data.length
HPR 포트 주소 길이.

def_data.hpr_dlc_data.address
HPR 포트 주소. 현재 HPR 링크를 지원할 때 사용됩니다. 이 필드는 퍼스널 통신이나 통신 서버가 이 포트를 사용하는 링크 스테이션에서 교환되는 XID3의 X'61' 제어 벡터 X'80' 부속필드에서 전송하는 정보를 지정합니다. 이 정보는 퍼스널 통신이나 통신 서버가 DLC로 발행하는 ACTIVATE_PORT에서 전달됩니다. 몇몇 DLC는 HPR 링크를 지원하는 포트에 대해 이 정보가 채워지도록 요구할 수 있습니다.

def_data.implicit_cp_cp_sess_support
이 포트 상에 있지 않는 암시적 링크 스테이션에서 CP-CP 세션이 허용되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.implicit_limited_resource
링크를 사용하는 세션이 없을 경우, 이 포트 상에 있지 않는 암시적 링크 스테이션이 활성종료되는지의 여부를 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_NO
암시적 링크가 제한 자원이 아니며 자동으로 활성종료되지 않습니다.

AP_YES 또는 AP_NO_SESSIONS
암시적 링크가 제한 자원이며 이들을 사용하는 활동중인 세션이 없을 때 자동으로 활성종료됩니다.

AP_INACTIVITY
암시적 링크가 제한 자원이며, 이들을 사용하는 활동중인 세션이 없거나 **implicit_deact_timer** 필드에 지정된 기간 동안 링크에 어떠한 데이터도 흐르지 않았을 때 자동으로 활성종료됩니다.

def_data.implicit_hpr_support
암시적 링크에서 HPR이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.implicit_link_lvl_error
링크 레벨 오류 복구를 사용하는 암시적 링크에서 HPR 트래픽이 전

송되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). **implicit_hpr_support**를 AP_NO로 설정할 경우에는 이 매개변수가 예약됩니다.

def_data.default_tg_chars
TG 특성(39페이지의 『DEFINE_COS』을 참조하십시오). 이 포트 상에 있지 않는 암시적 링크 스테이션과 **use_default_tg_chars**를 지정하는 정의된 링크 스테이션에 사용됩니다.

def_data.discovery_supported
이 포트에서 Discovery 함수가 수행되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

def_data.port_spec_data_len
ACTIVATE_PORT 신호에서 변경되지 않은 상태 포트에 전달되는 데이타 길이. 이 데이터는 기본 구조에 연결됩니다.

def_data.link_spec_data_len
이 필드는 항상 0으로 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PORT_NAME

AP_INVALID_DLC_NAME
AP_INVALID_PORT_TYPE
AP_INVALID_BTU_SIZE
AP_INVALID_LS_ROLE
AP_INVALID_LINK_ACTIVE_LIMIT
AP_INVALID_MAX_IFRM_RCVD
AP_INVALID_DSPU_SERVICES
AP_HPR_NOT_SUPPORTED
AP_DLUR_NOT_SUPPORTED
AP_PU_CONC_NOT_SUPPORTED
AP_INVALID_TEMPLATE_NAME
AP_INVALID_RETRY_FLAGS
AP_INVALID_IMPLICIT_UPLINK

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PORT_ACTIVE

AP_DUPLICATE_PORT_NUMBER
AP_CANT_MODIFY_WHEN_ACTIVE
AP_CANT_MODIFY_VISIBILITY
AP_INVALID_IMPLICIT_UPLINK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_TP

DEFINE_TP 명령은 노드 연산자 기능 TP 접속 관리자가 상대방 LU로부터의 입력 접속을 처리할 때 사용하는 트랜잭션 프로그램(TP)을 정의합니다. 이 명령은 또한 이전에 정의된 트랜잭션 프로그램(TP) 상의 하나 이상의 필드를 수정하는 데에도 사용할 수 있습니다.(단, 퍼스널 통신이나 통신 서버가 정의한 트랜잭션 프로그램(TP)을 수정하는 데에는 사용할 수 없습니다.)

VCB 구조

```
typedef struct define_tp
{
        unsigned short  opcode;            /* verb operation code           */
        unsigned char   attributes;        /* verb attributes               */
        unsigned char   reserv2;           /* reserved                      */
        unsigned char   format;            /* format                        */
        unsigned short  primary_rc;        /* primary return code           */
        unsigned long   secondary_rc;      /* secondary return code         */
        unsigned char   tp_name[64];       /* TP name                       */
        TP_CHARS        tp_chars;          /* TP characteristics            */
} DEFINE_TP;

typedef struct tp_chars
{
        unsigned char   description[RD_LEN]
                                           /* resource description          */
        unsigned char   conv_type;         /* conversation type             */
        unsigned char   security_rqd;      /* security support              */
        unsigned char   sync_level;        /* synchronization level support */
        unsigned char   dynamic_load;      /* dynamic load                  */
        unsigned char   enabled;           /* is the TP enabled?            */
        unsigned char   pip_allowed;       /* program initialization        */
                                           /* parameters supported          */
        unsigned char   duplex_support;    /* duplex supported              */
        unsigned char   reserv3[9];        /* reserved                      */
        unsigned short  tp_instance_limit; /* limit on currently active TP  */
                                           /* instances                     */
        unsigned short  incoming_alloc_timeout;
                                           /* incoming allocation timeout   */
        unsigned short  rcv_alloc_timeout; /* receive allocation timeout    */
        unsigned short  tp_data_len;       /* TP data length                */
        TP_SPEC_DATA    tp_data;           /* TP data                       */
} TP_CHARS;

typedef struct tp_spec_data
{
        unsigned char   pathname[256];     /* path and TP name              */
        unsigned char   parameters[64];    /* parameters for TP             */
        unsigned char   queued;            /* queued TP                     */
        unsigned char   load_type;         /* type of load-DETACHED/CONSOLE */
        unsigned char   dynamic_load       /* dynamic loading of TP enabled */
        unsigned char   reserved[5];       /* reserved                      */
} TP_SPEC_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_TP

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

tp_name
정의할 트랜잭션 프로그램(TP)의 이름. 64바이트 EBCDIC 문자열로서 오른쪽이 EBCDIC 공백으로 채워집니다. 퍼스널 통신이나 통신 서버는 이 필드의 문자 세트를 점검하지 않습니다.

tp_chars.description
자원 설명(QUERY_TP_DEFINITION 및 QUERY_TP에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효 바이트입니다.

tp_chars.conv_type
이 트랜잭션 프로그램(TP)이 지원하는 대화 유형을 지정합니다.

AP_BASIC
AP_MAPPED
AP_EITHER

tp_chars.security_rqd
트랜잭션 프로그램(TP)을 시작하는 데 대화 보안 정보가 필요한지의 여부를 지정합니다(AP_NO 또는 AP_YES).

tp_chars.sync_level
트랜잭션 프로그램(TP)이 지원하는 동기화 레벨을 지정합니다.

AP_NONE
트랜잭션 프로그램(TP)이 None 동기화 레벨을 지원합니다.

AP_CONFIRM_SYNC_LEVEL
트랜잭션 프로그램(TP)이 Confirm 동기화 레벨을 지원합니다.

AP_EITHER
트랜잭션 프로그램(TP)이 None 또는 Confirm 동기화 레벨을 지원합니다.

AP_SYNCPT_REQUIRED
트랜잭션 프로그램(TP)이 Sync-point 동기화 레벨을 지원합니다.

AP_SYNCPT_NEGOTIABLE
트랜잭션 프로그램(TP)이 None, Confirm 또는 Sync-point 동기화 레벨을 지원합니다.

tp_chars.dynamic_load
트랜잭션 프로그램(TP)을 동적으로 로드할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

tp_chars.enabled
트랜잭션 프로그램(TP)을 성공적으로 접속할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 생략시는 AP_NO입니다.

tp_chars.pip_allowed
트랜잭션 프로그램(TP)이 프로그램 초기화(PIP) 매개변수를 수신할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

tp_chars.duplex_support
트랜잭션 프로그램(TP)이 동시 양방향(전송)인지 또는 비동시 양방향(전송)인지를 지시합니다.

AP_FULL_DUPLEX
트랜잭션 프로그램(TP)이 동시 양방향(전송)임을 지정합니다.

AP_HALF_DUPLEX
트랜잭션 프로그램(TP) 비동시 양방향(전송)임을 지정합니다.

AP_EITHER_DUPLEX
트랜잭션 프로그램(TP)이 비동시 양방향(전송) 또는 동시 양방향(전송)이 될 수 있음을 지정합니다.

tp_chars.tp_instance_limit
동시에 활동중인 트랜잭션 프로그램(TP) 인스턴스 수 한계값. 0은 제한 없음을 의미합니다.

tp_chars.incoming_alloc_timeout
입력 접속이 대기행렬화되어 RECEIVE_ALLOCATE를 기다리는 시간(초 수)을 지정합니다. 0은 시간종료가 없음을 의미하며, 따라서 무한정 대기할 수 있습니다.

tp_chars.rcv_alloc_timeout
RECEIVE_ALLOCATE 명령이 대기행렬화하여 접속을 기다리는 시간(초 수)을 지정합니다. 0은 시간종료가 없음을 의미하며, 따라서 무한정 대기할 수 있습니다.

tp_chars.tp_data_len
구현 종속 트랜잭션 프로그램(TP) 데이터의 길이.

tp_spec_data
접속 관리자가 트랜잭션 프로그램(TP)을 시작할 때 사용하는 정보. 이 정보의 사용법에 대한 자세한 내용은 *퍼스널 통신 클라이언트/서버 통신 프로그래밍*을 참조하십시오.

tp_chars.tp_data.pathname
경로와 트랜잭션 프로그램명(TPN) 명을 지정합니다.

tp_chars.tp_data.parameters
트랜잭션 프로그램(TP)의 매개변수를 지정합니다.

tp_chars.tp_data.queued
트랜잭션 프로그램(TP)이 대기행렬화되는지의 여부를 지정합니다.

tp_chars.tp_data.load_type
트랜잭션 프로그램(TP)이 로드되는지의 여부를 지정합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_SYSTEM_TP_CANT_BE_CHANGED

AP_INVALID_CONV_TYPE
AP_INVALID_SYNC_LEVEL
AP_INVALID_DYNAMIC_LOAD
AP_INVALID_ENABLED
AP_INVALID_PIP_ALLOWED
AP_INVALID_DUPLEX_SUPPORT

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_CANT_MODIFY_VISIBILITY

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

재정의 결과: 각 필드의 재정의는 즉시 효력을 가집니다.(예를 들어, 트랜잭션 프로그램(TP)의 그 다음 인스턴스가 시작됩니다.) 단, **incoming_alloc_timeout** 및 **rcv_alloc_timeout**의 변경사항은 이미 대기행렬화되어 있는 접속이나 RECEIVE_ALLOCATES에는 영향을 미치지 않습니다.

DELETE_ADJACENT_NODE

DELETE_ADJACENT_NODE는 노드 디렉토리 데이터베이스에서 인접 노드에 있는 자원과 연관된 입력항목을 제거합니다.

디렉토리에서 노드 LU와 함께 노드의 제어점(CP)을 제거하려면, **num_of_lus**를 0으로 설정하십시오. **num_of_lus**가 nonzero일 경우, 이 명령은 제어점 정의는 그대로 두고 디렉토리에서 노드 LU를 제거하는 데 사용됩니다.

특정 이유로 인해 이 명령이 실패하면 어떠한 디렉토리 항목도 삭제되지 않습니다.

VCB 구조

DELETE_ADJACENT_NODE 명령에는 ADJACENT_NODE_LU 오버레이의 가변 수가 포함됩니다. ADJACENT_NODE_LU 구조는 DELETE_ADJACENT_NODE 구조의 끝에 연결됩니다.

```
typedef struct delete_adjacent_node
{
        unsigned short  opcode;          /* verb operation code        */
        unsigned char   reserv2;         /* reserved                   */
        unsigned char   format;          /* format                     */
        unsigned short  primary_rc;      /* primary return code        */
        unsigned long   secondary_rc;    /* secondary return code      */
        unsigned char   cp_name[17];     /* CP name                    */
        unsigned short  num_of_lus;      /* number of LUs              */
} DELETE_ADJACENT_NODE;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;     /* wildcard LU name indicator */
        unsigned char   fqlu_name[17];   /* fully qualified LU name    */
        unsigned char   reserv1[6];      /* reserved                   */
} ADJACENT_NODE_LU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_ADJACENT_NODE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

cp_name

인접 LEN 끝 노드에 있는 제어점(CP)의 전체 정식명. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

num_of_lus
삭제할 LU의 수. 전체 노드 정의를 삭제하려면 이 필드를 0으로 설정하십시오. 이 숫자는 DELETE_ADJACENT_NODE VCB 뒤에 오는 인접 LU 중복 수를 나타냅니다.

adjacent_node_lu.wildcard_lu
지정한 LU 명이 총칭 문자인지의 여부를 지시합니다(AP_YES 또는 AP_NO).

adjacent_node_lu.fqlu_name
삭제할 LU 명. 이 이름이 전체 정식명이 아닐 경우, CP 명의 네트워크 ID가 사용됩니다. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 하나의 A 유형 EBCDIC 문자열로 구성되거나 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_CP_NAME

AP_INVALID_LU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

DELETE_ADJACENT_NODE

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_CN

DELETE_CN은 연관된 모든 포트가 재설정될 경우, 연결 네트워크 제어 블럭용 메모리를 삭제하고 해제합니다. DELETE_CN은 또한 연결 네트워크에서 선택된 포트를 삭제하는 데에도 사용할 수 있습니다. 이를 위해서는 사용자가 **num_ports** 필드를 nonzero 값으로 설정하고 삭제할 포트의 포트명을 입력해야 합니다.

VCB 구조

```
typedef struct delete_cn
{
        unsigned short  opcode;         /* verb operation code        */
        unsigned char   attributes;     /* verb attributes            */
        unsigned char   format;         /* format                     */
        unsigned short  primary_rc;     /* primary return code        */
        unsigned long   secondary_rc;   /* secondary return code      */
        unsigned char   fqcn_name[17];  /* name of connection network */
        unsigned char   reserv1;        /* reserved                   */
        unsigned short  num_ports;      /* number of ports to delete  */
        unsigned char   port_name[8] [8];
                                        /* names of ports to delete   */
} DELETE_CN;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_CN

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

fqcn_name

삭제할 연결 네트워크의 이름(17바이트 길이). 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

num_ports

연결 네트워크 상의 삭제할 포트 수. 전체 노드 정의를 삭제하려면 이 필드를 0으로 설정해야 합니다.

port_name

num_ports가 nonzero일 경우에 삭제할 포트의 이름. 각 포트명은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. **num_ports** 필드가 0일 경우에는 이 필드가 예약됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_CN_NAME

AP_INVALID_NUM_PORTS_SPECIFIED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DELETE_COS

DELETE_COS는 SNA에 의해 정의된 생략시 서비스 클래스(COS)를 제외한 서비스 클래스(COS) 입력항목을 삭제합니다.

VCB 구조

```
typedef struct delete_cos
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   cos_name[8];    /* class-of-service name   */
} DELETE_COS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_COS

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

cos_name
서비스 클래스(COS) 명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_COS_NAME_NOT_DEFD

AP_SNA_DEFD_COS_CANT_BE_DELETE

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_DLC

DELETE_DLC는 DLC가 재설정될 경우 그와 연관된 모든 포트, 링크 스테이션이나 연결 네트워크 전송 그룹(TG)을 삭제합니다. 모든 DLC 제어 블럭이 삭제되고 메모리가 해제됩니다. 노드 연산자 기능은 DLC가 삭제되었는지의 여부를 지정하는 응답을 반환합니다.

PU와 연관된 링크 스테이션이 삭제되면(DLC와 연관되어 있으므로) PU에 정의된 모든 LU도 역시 삭제됨에 주의하십시오.

VCB 구조

```
typedef struct delete_dlc
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   dlc_name[8];    /* name of DLC             */

} DELETE_DLC;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_DLC

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dlc_name
삭제할 DLC의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLC_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_DLC_ACTIVE

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_DOWNSTREAM_LU

이 명령은 통신 서버에만 적용됩니다.

VCB 구조

```
typedef struct delete_downstream_lu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   dslu_name[8];   /* Downstream LU name      */
} DELETE_DOWNSTREAM_LU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_DOWNSTREAM_LU

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

OR을 사용하여 비트와이즈로 연결할 수 있는 값은 다음과 같습니다.

AP_DELAY_IF_REQUIRED

dslu_name으로 지정된 다운스트림 LU가 현재 활동중이며, 이 LU가 비활동 상태로 될 때까지 이 명령이 프로그램 내에서 대기행렬화되도록 지정합니다. 이 경우에는, LU가 비활동 상태로 될 때 이 명령이 처리됩니다.

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dslu_name

삭제할 다운스트림 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_DSLU_ACTIVE
AP_DELAYED_VERB_PENDING

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_LU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_DOWNSTREAM_LU_RANGE

이 명령은 통신 서버에만 적용됩니다.

예를 들어, 기본명 LUNME와 NAU 범위 1-4를 조합하면, LUNME001, LUNME002, LUNME003 및 LUNME004라는 이름을 가지는 LU가 삭제됩니다. non-pad 문자가 5개 미만인 기본명은 non-pad 문자가 8개 미만인 LU 명을 생성합니다.

이 명령은 범위 내의 모든 LU를 삭제합니다. 범위 내의 특정 LU가 없을 경우, 이 명령은 계속해서 존재하는 그 다음 LU를 삭제합니다. 이 명령은 지정된 범위 내의 어떠한 LU도 존재하지 않을 경우에만 실패합니다.

VCB 구조

```
typedef struct delete_downstream_lu_range
{
        unsigned short  opcode;            /* verb operation code      */
        unsigned char   attributes;        /* verb attributes          */
        unsigned char   format;            /* format                   */
        unsigned short  primary_rc;        /* primary return code      */
        unsigned long   secondary_rc;      /* secondary return code    */
        unsigned char   dslu_base_name[5]; /* Downstream LU base name  */
        unsigned char   min_nau;           /* min NAU address in range */
        unsigned char   max_nau;           /* max NAU address in range */
} DELETE_DOWNSTREAM_LU_RANGE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_DOWNSTREAM_LU_RANGE

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dslu_base_name

다운스트림 LU 명 범위의 기본명. 5바이트 영숫자 A 유형 EBCDIC 문자열 (문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. NAU 범위 내에 있는 각 LU의 경우, 이 기본명에 세 개의 A 유형 EBCDIC 숫자가 추가되어 NAU 주소의 십진값을 나타냅니다.

min_nau
범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

max_nau
범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_NAU_ADDRESS

AP_INVALID_LU_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

AP_INVALID_LU_NAME
AP_DSLU_ACTIVE
AP_DELAYED_VERB_PENDING

secondary_rc
AP_INVALID_LU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DELETE_DSPU_TEMPLATE

이 명령은 통신 서버에만 적용됩니다.

VCB 구조

형 D 1

```
typedef struct delete_dspu_template
{
        unsigned short  opcode;              /* verb operation code       */
        unsigned char   attributes;          /* verb attributes           */
        unsigned char   format;              /* format                    */
        unsigned short  primary_rc;          /* primary return code       */
        unsigned long   secondary_rc;        /* secondary return code     */
        unsigned char   template_name[8];    /* name of template          */
        unsigned short  num_of_dslu_templates;
                                             /* Number of DSLU templates  */
        unsigned char   reserv1[10];         /* reserved                  */
} DELETE_DSPU_TEMPLATE;

typedef struct dslu_template
{
        unsigned char   min_nau;             /* min NAU address in range  */
        unsigned char   max_nau;             /* max NAU address in range  */
        unsigned char   allow_timeout;       /* Allow timeout of host LU? */
        unsigned char   delayed_logon;       /* Allow delayed logon to    */
                                             /* host LU                   */
        unsigned char   reserv1[8];          /* reserved                  */
        unsigned char   host_lu[8];          /* host LU or pool name      */
} DSLU_TEMPLATE;
```

VCB 구조

형 D 0

```
typedef struct delete_dspu_template
{
        unsigned short  opcode;              /* verb operation code       */
        unsigned char   attributes;          /* verb attributes           */
        unsigned char   format;              /* format                    */
        unsigned short  primary_rc;          /* primary return code       */
        unsigned long   secondary_rc;        /* secondary return code     */
        unsigned char   template_name[8];    /* name of template          */
} DELETE_DSPU_TEMPLATE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_DSPU_TEMPLATE

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

template_name
DSPU 템플리트의 이름. (PORT_DEF_DATA의 **implicit_dspu_template** 필드에 지정하는 이름과 일치합니다). 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

num_of_dslu_templates
DEFINE_DSPU_TEMPLATE VCB 뒤에 오는 DSLU 템플리트 오버레이 수. 0-255 사이의 한 값이 될 수 있습니다. DSLU 템플리트는 중복으로 DELETE_DSPU_TEMPLATE VCB의 끝에 추가됩니다.

dslu_template.min_nau
범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

dslu_template.max_nau
범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

def_data.allow_timeout
이 필드는 예약됩니다.

def_data.delayed_logon
이 필드는 예약됩니다.

dslu_template.host_lu
이 필드는 예약됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TEMPLATE_NAME

AP_INVALID_NAU_RANGE

관련 START_NODE 매개변수가 설정되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_FOCAL_POINT

DELETE_FOCAL_POINT 명령은 지정한 유형 및 범주의 중재점을 삭제하는 데 사용할 수 있습니다. 중재점 유형에 관한 자세한 내용은 70페이지의 『DEFINE_FOCAL_POINT』를 참조하십시오. 활동중인 중재점을 삭제할 경우에는 중재점이 취소됩니다. 활동중인 중재점을 취소하려면 AP_ACTIVE 유형을 지정하십시오. 현재 활동중이 아닌 백업 또는 암시적 중재점을 삭제하면(AP_BACKUP 또는 AP_IMPLICIT를 지정하여), 그에 관해 저장된 정보가 제거됩니다.

DEFINE_FOCAL_POINT 명령을 사용하여 현재 활동중인 중재점을 취소할 수도 있음에 주의하십시오. 이러한 중복 기능은 하향 호환성을 위해 보유됩니다.

VCB 구조

```
typedef struct delete_focal_point
{
        unsigned short  opcode;         /* verb operation code          */
        unsigned char   reserv2;        /* reserved                     */
        unsigned char   format;         /* format                       */
        unsigned short  primary_rc;     /* primary return code          */
        unsigned long   secondary_rc;   /* secondary return code        */
        unsigned char   reserved;       /* reserved                     */
        unsigned char   ms_category[8]; /* management services category */
        unsigned char   type;           /* type of focal point          */
} DELETE_FOCAL_POINT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_FOCAL_POINT

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

ms_category

관리 서비스 범주. SNA 관리 서비스에 설명된 것과 같이 관리 서비스 범주에 대해 구조적으로 정의한 4바이트 값(오른쪽이 EBCDIC 공백으로 채워진) 중 하나이거나 8바이트 1134 유형 EBCDIC 설치 정의 이름일 수 있습니다.

type 삭제할 중재점의 유형을 지정합니다. 가능한 유형은 다음과 같습니다.

AP_ACTIVE

현재 활동중인 중재점(모든 유형이 가능함)이 취소됩니다.

AP_IMPLICIT
암시적 정의가 제거됩니다. 현재 활동중인 중재점이 암시적 중재점인 경우, 이 중재점이 취소됩니다.

AP_BACKUP
백업 정의가 제거됩니다. 현재 활동중인 중재점이 백업 중재점인 경우, 이 중재점이 취소됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TYPE

AP_INVALID_CATEGORY_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_INTERNAL_PU

DELETE_INTERNAL_PU 명령은 DLUR이 서비스를 제공하는 국지 PU의 삭제를 요청합니다. 이 명령은 PU에 활동중인 SSCP-PU 세션이 없는 경우에만 실행됩니다.

PU와 연관된 모든 LU가 삭제됩니다.

VCB 구조

```
typedef struct delete_internal_pu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   pu_name[8];     /* internal PU name        */
} DELETE_INTERNAL_PU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_INTERNAL_PU

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pu_name

삭제할 내부 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열 (문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PU_NOT_RESET

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LOCAL_LU

DELETE_LOCAL_LU 명령은 국지 LU 정의의 삭제를 요청합니다.

VCB 구조

```
typedef struct delete_local_lu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* local LU name           */
} DELETE_LOCAL_LU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_LOCAL_LU

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name
정의할 국지 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_CANT_DELETE_CP_LU

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LS

DELETE_LS는 링크 스테이션이 이미 정의되어 있고 재설정되었는지 점검합니다. 이 명령은 링크 스테이션 제어 블럭을 제거하며 링크 스테이션이 삭제되었는지의 여부를 지정하는 노드 연산자 기능으로부터의 응답을 반환합니다. 이 링크 스테이션을 사용하는 PU에 정의된 모든 LU도 역시 삭제됨에 주의하십시오.

VCB 구조

```
typedef struct delete_ls
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   ls_name[8];     /* name of link station    */
} DELETE_LS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_LS

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

ls_name
삭제할 링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LINK_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_LS_ACTIVE

AP_INVALID_LINK_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_0_TO_3

이 명령은 특정 LU를 삭제하는 데 사용됩니다.

VCB 구조

```
typedef struct delete_lu_0_to_3
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* LU name                 */
} DELETE_LU_0_TO_3;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_LU_0_TO_3

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

삭제할 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_LU_NAME

AP_CANT_DELETE_IMPLICIT_LU

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_LU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_0_TO_3_RANGE

이 명령은 특정 범위의 LU를 삭제하는 데 사용됩니다. 노드 운영요원이 기본명과 NAU 범위를 제공합니다. LU 명은 기본명과 NAU 주소의 조합으로 생성됩니다.

예를 들어, 기본명 LUNME와 NAU 범위 1-4를 조합하면, LUNME001, LUNME002, LUNME003 및 LUNME004라고 하는 LU가 삭제됩니다. non-pad 문자가 5개 미만인 기본명은 non-pad 문자가 8개 미만인 LU 명을 생성합니다.

범위 내의 모든 LU가 삭제됩니다. 범위 내의 특정 LU가 없을 경우, 이 명령은 계속해서 존재하는 그 다음 LU를 삭제합니다. 지정된 범위 내의 어떠한 LU도 존재하지 않으면 이 명령이 실패합니다.

VCB 구조

형 D 1

```
typedef struct delete_lu_0_to_3_range
{
        unsigned short  opcode;              /* verb operation code        */
        unsigned char   attributes;          /* verb attributes            */
        unsigned char   format;              /* format                     */
        unsigned short  primary_rc;          /* primary return code        */
        unsigned long   secondary_rc;        /* secondary return code      */
        unsigned char   base_name[6];        /* base name                  */
        unsigned char   min_nau;             /* minimum NAU address        */
        unsigned char   max_nau;             /* maximum NAU address        */
        unsigned char   name_attributes;     /* Attributes of base_name    */
        unsigned char   base_number;         /* Base number for LU names   */
        unsigned char   reserv5[16];         /* reserved                   */
} DELETE_LU_0_TO_3_RANGE;
```

VCB 구조

형 D 0

```
typedef struct delete_lu_0_to_3_range
{
        unsigned short  opcode;              /* verb operation code        */
        unsigned char   attributes;          /* verb attributes            */
        unsigned char   format;              /* format                     */
        unsigned short  primary_rc;          /* primary return code        */
        unsigned long   secondary_rc;        /* secondary return code      */
        unsigned char   base_name[5];        /* base name                  */
        unsigned char   min_nau;             /* minimum NAU address        */
        unsigned char   max_nau;             /* maximum NAU address        */
        unsigned char   reserv3;             /* reserved                   */
} DELETE_LU_0_TO_3_RANGE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_LU_0_TO_3_RANGE

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

base_name
기본 LU 명. 5바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다. NAU 범위 내에 있는 각 LU의 경우, 이 기본명에 세 개의 A 유형 EBCDIC 숫자가 추가되어 NAU 주소의 십진값을 나타냅니다.

min_nau
범위 내의 최소 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

max_nau
범위 내의 최대 NAU 주소. 1-255 사이의 한 값이 될 수 있습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_NAU_ADDRESS

AP_INVALID_LU_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_CANT_DELETE_IMPLICIT_LU

시스템에 종속 LU 지원이 내장되어 있지 않아서 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_POOL

이 명령은 LU 풀을 삭제하거나 풀에서 LU를 제거하는 데 사용됩니다. LU 명을 지정하지 않으면, 풀 전체가 제거됩니다. LU 풀에 지정한 LU가 더이상 존재하지 않거나 LU 풀 자체가 더이상 존재하지 않을 때 이 명령이 완료됩니다. 지정한 LU가 없거나 지정한 풀에 아무런 LU도 없을 경우에만 이 명령이 실패합니다.

VCB 구조

```
typedef struct delete_lu_pool
{
        unsigned short  opcode;                   /* verb operation code        */
        unsigned char   attributes;               /* verb attributes            */
        unsigned char   format;                   /* format                     */
        unsigned short  primary_rc;               /* primary return code        */
        unsigned long   secondary_rc;             /* secondary return code      */
        unsigned char   pool_name[8];             /* LU pool name               */
        unsigned short  num_lus;                  /* number of LUs to add       */
        unsigned char   lu_names[10][8]; /* LU names                        */
} DELETE_LU_POOL;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_LU_POOL

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pool_name

LU 풀의 이름. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

num_lus

lu_names 리스트에 지정된 LU의 수.

lu_names

제거할 LU의 이름. 각 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_POOL_NAME

AP_INVALID_LU_NAME
AP_INVALID_NUM_LUS

시스템에 종속 LU 지원이 내장되어 있지 않아서 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_MODE

DELETE_MODE 명령은 모드 정의 삭제를 요청합니다. CPSVCMG, SNASVCMG 및 기타 표준 SNA 모드에 대한 생략시 정의는 삭제되지 않습니다.

VCB 구조

```
typedef struct delete_mode
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   mode_name[8];   /* mode name               */
} DELETE_MODE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_MODE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

mode_name
모드 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작) 로, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_CP_OR_SNA_SVCMG_UNDELETABLE

AP_MODE_UNDELETABLE
AP_DEL_MODE_DEFAULT_SPCD
AP_MODE_NAME_NOT_DEFD

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc 갖
AP_UNEXPECTED_SYSTEM_ERRO6 TD 망먁맣매맸매먹매멕맹매멕머멸릭맥릴릴

DELETE_PARTNER_LU

DELETE_PARTNER_LU는 상대방 LU 정의의 삭제를 요청합니다.

VCB 구조

```
typedef struct delete_partner_lu
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   fqplu_name[17]; /* fully qualified partner */
                                        /* LU name                 */
} DELETE_PARTNER_LU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_PARTNER_LU

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

fqplu_name
상대방 LU의 전체 정식명. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백없이 8바이트입니다.)

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PLU_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_PORT

DELETE_PORT는 포트가 재설정될 경우, 포트와 연관된 모든 링크 스테이션이나 연결 네트워크 전송 그룹(TG)을 삭제합니다. 그런 후, 포트의 제어 블럭을 삭제하고 메모리를 해제하며, 포트가 삭제되었는지의 여부를 지시하는 노드 연산자 기능으로부터의 응답을 반환합니다.

PU와 연관된 링크 스테이션이 삭제되면(포트와 연관되어 있으므로) PU에 정의된 모든 LU도 역시 삭제됩니다.

VCB 구조

```
typedef struct delete_port
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   port_name[8];   /* name of port            */
} DELETE_PORT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_PORT

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

port_name
삭제할 포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PORT_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PORT_ACTIVE

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_TP

DELETE_TP는 트랜잭션 프로그램(TP) 정의의 삭제를 요청합니다.

VCB 구조

```
typedef struct delete_tp
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   attributes;     /* verb attributes         */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   tp_name[64];    /* TP name                 */
} DELETE_TP;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_TP format VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

tp_name
트랜잭션 프로그램(TP)의 이름. 프로그램은 이 필드의 문자 세트를 점검하지 않습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TP_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_TP

제5장 활성화 및 활성종료 명령

이 장에서는 활성화하고 활성종료하는 데 사용되는 명령에 대해 설명합니다.

- 데이터 링크 제어(DLC)
- 내부 PU
- 포트
- 링크 스테이션
- 세션
- 대화 그룹

이 장에서는 또한 고성능 경로지정(HPR)을 지원하는 연결에 대해 경로 전환을 요청하는 데 사용되는 명령에 대해서도 설명합니다.

START_DLC

START_DLC는 데이터 링크 제어(DLC)의 활성화를 요청합니다. 이 명령은 반환되어 DLC가 활성화되었는지의 여부를 지시합니다. DLC에 대해 포트가 정의되어 있지 않아도 DLC를 시작할 수 있습니다. DLC와 포트, 링크 스테이션 사이의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

VCB 구조

```
typedef struct start_dlc
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   dlc_name[8];    /* name of DLC           */
} START_DLC;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_START_DLC

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

dlc_name
시작할 데이터 링크 제어 인스턴스의 이름. 이 이름은 국지로 표시 가능한 문자 세트로 된 8바이트 문자열로서, DEFINE_DLC 명령으로 미리 정의되어 있어야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLC

DLC가 활성종료되므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_DLC_DEACTIVATING

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

START_INTERNAL_PU

START_INTERNAL_PU 명령은 종속 LU 리퀘스터(DLUR)로 하여금 DLUR이 서비스를 제공하는 사전 정의된 국지 PU에 대해 SSCP-PU 세션 활성화를 시작하도록 요청합니다.

VCB 구조

```
typedef struct start_internal_pu
{
        unsigned short  opcode;             /* verb operation code   */
        unsigned char   reserv2;            /* reserved              */
        unsigned char   format;             /* format                */
        unsigned short  primary_rc;         /* primary return code   */
        unsigned long   secondary_rc;       /* secondary return code */
        unsigned char   pu_name[8];         /* internal PU name      */
        unsigned char   dlus_name[17];      /* DLUS name             */
        unsigned char   bkup_dlus_name[17]; /* Backup DLUS name      */
} START_INTERNAL_PU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_START_INTERNAL_PU

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pu_name

SSCP-PU 세션 활성화 흐름을 요청할 내부 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

dlus_name

DLUR이 주어진 PU에 대한 SSCP-PU 세션 활성화를 요청하기 위해 접속하는 종속 LU 서버(DLUS) 노드의 이름. 모두 0으로 설정하거나 EBCDIC 점으로 연결되는 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로 설정해야 하며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 값은 DEFINE_INTERNAL_PU 명령에 지정한 값을 대체합니다. 이 필드를 모두 0으로 설정하면, DEFINE_INTERNAL_PU 명령에 지정한 DLUS가 사용됩니다. DEFINE_INTERNAL_PU 명령에 DLUS를 지정하지 않은 경우에는, 글로벌 생략시 값(DEFINE_DLUR_DEFAULTS 명령으로 지정한 경우)이 사용됩니다.

bkup_dlus_name

DLUR이 주어진 PU에 대한 백업 DLUS로 저장하는 DLUS 노드의 이름. 모두 0으로 설정하거나 EBCDIC 점으룖 니 개인 N

른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 값은 DEFINE_INTERNAL_PU 명령에 지정한 값을 대체합니다. 이 필드를 모두 0으로 설정하면, DEFINE_INTERNAL_PU 명령으로 지정한 백업 DLUS 명이 이 PU에 대한 백업 DLUS로 보유됩니다. DEFINE_INTERNAL_PU 명령으로 백업 DLUS를 지정하지 않으면, 글로벌 백업 생략시 DLUS (DEFINE_DLUR_DEFAULTS 명령으로 정의한 경우)가 이 PU에 대한 백업 생략시 값으로 보유됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLUS_NAME

AP_INVALID_BKUP_DLUS_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_NO_DEFAULT_DLUS_DEFINED

AP_PU_NOT_DEFINED
AP_PU_ALREADY_ACTIVATING
AP_PU_ALREADY_ACTIVE

명령이 정상적으로 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNSUCCESSFUL

secondary_rc
AP_DLUS_REJECTED

AP_DLUS_CAPS_MISMATCH
AP_PU_FAILED_ACTPU

START_INTERNAL_PU

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

START_LS

START_LS는 링크의 활성화를 요청합니다. 이 명령은 링크가 활성화되었는지의 여부를 지정하는 응답으로서 반환됩니다.

DLC와 포트, 링크 스테이션 사이의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

VCB 구조

```
typedef struct start_ls
{
        unsigned short  opcode;         /* verb operation code         */
        unsigned char   reserv2;        /* reserved                    */
        unsigned char   format;         /* format                      */
        unsigned short  primary_rc;     /* primary return code         */
        unsigned long   secondary_rc;   /* secondary return code       */
        unsigned char   ls_name[8];     /* name of link station        */
        unsigned char   enable;         /* whether the link is enabled */
        unsigned char   reserv3[3];     /* reserved                    */
} START_LS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_START_LS

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

ls_name

시작할 링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. **ls_name**의 값은 DEFINE_LS 명령에 있는 것과 일치해야 합니다.

enable

링크를 시작하려면 이 필드를 설정하십시오. 이 필드를 AP_ACTIVATE로 설정하면 링크가 시작됩니다. 그러지 않을 경우, 링크가 시작되지 않으며 다음 값들이 가능합니다. 이들 값은 OR로 연결할 수 있습니다.

AP_AUTO_ACT

요구가 있을 때 국지 노드에서 링크를 활성화할 수 있습니다. 이 값은 DEFINE_LS 명령에서 **auto_act_supp**를 AP_YES로 설정한 경우에만 유효합니다.

AP_REMOTE_ACT

원격 모드에서 링크를 활성화할 수 있습니다. **disable_remote_act**의 정의된 값을 변경하지 않습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LINK_NAME_SPECIFIED

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PORT_INACTIVE

AP_ACTIVATION_LIMITS_REACHED
AP_PARALLEL_TGS_NOT_SUPPORTED
AP_ALREADY_STARTING
AP_LINK_DEACT_IN_PROGRESS

링크가 활성화되기 전에 뒤에 오는 STOP_LS 또는 STOP_PORT가 명령을 취소하였으므로 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CANCELLED

secondary_rc
AP_LINK_DEACTIVATED

링크 소프트웨어가 상대방을 찾을 수 없기 때문에 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_LS_FAILURE

secondary_rc
AP_PARTNER_NOT_FOUND

링크를 설정하는 중에 링크 오류가 발생했으므로 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_LS_FAILURE

secondary_rc
AP_ERROR

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

START_PORT

START_PORT는 포트의 활성화를 요청합니다. 이 명령은 반환되어 포트가 활성화되었는지의 여부를 지시합니다. 포트에 대해 링크 스테이션을 정의하지 않은 경우에도 포트를 시작할 수 있으나, 부모(parent) DLC가 활동 중이 아닌 경우에는 포트가 시작되지 않습니다.

DLC와 포트, 링크 스테이션 사이의 관계에 대한 자세한 내용은 17페이지의 『DLC 프로세스, 포트 및 링크 스테이션』을 참조하십시오.

VCB 구조

```
typedef struct start_port
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   port_name[8];   /* name of port          */
} START_PORT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_START_PORT

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

port_name
시작할 포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열로, DEFINE_PORT 명령에 있는 것과 일치해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PORT_NAME

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_DLC_INACTIVE

AP_STOP_PORT_PENDING
AP_DUPLICATE_PORT

명령이 취소되었기 때문에 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CANCELLED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

STOP_DLC

STOP_DLC는 DLC를 종료할 것을 요청합니다. 이 명령은 반환되어 DLC가 종료되었는지의 여부를 지시합니다. STOP_DLC는 또한 프로그램에 이 DLC를 통한 포트 상의 링크 스테이션 활성화 재시도를 자동으로 중단하도록 지시하는 데에도 사용할 수 있습니다.

VCB 구조

```
typedef struct stop_dlc
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   stop_type;      /* stop type             */
        unsigned char   dlc_name[8];    /* name of DLC           */
} STOP_DLC;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_STOP_DLC

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

stop_type
DLC를 종료하는 방법.

AP_ORDERLY_STOP
노드가 DLC를 종료하기 전에 정리 작업을 수행합니다.

AP_IMMEDIATE_STOP
노드가 즉시 DLC를 종료합니다.

dlc_name
종료할 DLC의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열로, DEFINE_DLC 명령에 있는 것과 일치해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLC

AP_UNRECOGNIZED_DEACT_TYPE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_STOP_DLC_PENDING

명령이 취소되었기 때문에 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CANCELLED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

STOP_INTERNAL_PU

STOP_INTERNAL_PU 명령은 종속 LU 리퀘스터(DLUR)로 하여금 DLUR이 서비스를 제공하는 사전 정의된 국지 PU에 대해 SSCP-PU 세션 활성종료를 시작하도록 요청합니다.

VCB 구조

```
typedef struct stop_internal_pu
{
        unsigned short  opcode;         /* verb operation code    */
        unsigned char   reserv2;        /* reserved               */
        unsigned char   format;         /* format                 */
        unsigned short  primary_rc;     /* primary return code    */
        unsigned long   secondary_rc;   /* secondary return code  */
        unsigned char   pu_name[8];     /* internal PU name       */
        unsigned char   stop_type;      /* type of stop requested */
} STOP_INTERNAL_PU;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_STOP_INTERNAL_PU

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pu_name
SSCP-PU 세션을 활성종료할 내부 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

stop_type
PU에 대해 요청되는 종료 유형을 지정합니다. 순서화된 종료는 SSCP-PU 세션을 활성종료하기 전에 모든 기본 PLU-SLU이나 SSCP-LU 세션을 활성종료합니다.

AP_ORDERLY_STOP
AP_IMMEDIATE_STOP

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_STOP_TYPE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PU_NOT_DEFINED

AP_PU_ALREADY_DEACTIVATING
AP_PU_NOT_ACTIVE

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

STOP_LS

STOP_LS는 링크 스테이션의 활성종료를 요청합니다. 이 명령은 반환되어 링크가 종료되었는지의 여부를 지정합니다. STOP_LS는 또한 링크 스테이션의 원격 활성화를 사용불가능하게 하거나 요구가 있을 때의 링크 스테이션 활성화를 사용불가능하게 합니다. STOP_LS는 또한 프로그램에 링크 스테이션 활성화 재시도를 자동으로 중단하도록 지시하는 데에도 사용할 수 있습니다.

VCB 구조

```
typedef struct stop_ls
{
        unsigned short  opcode;         /* verb operation code          */
        unsigned char   reserv2;        /* reserved                     */
        unsigned char   format;         /* format                       */
        unsigned short  primary_rc;     /* primary return code          */
        unsigned long   secondary_rc;   /* secondary return code        */
        unsigned char   stop_type;      /* stop type                    */
        unsigned char   ls_name[8];     /* name of link station         */
        unsigned char   disable;        /* whether the link is disabled */
        unsigned char   reserved[3];    /* reserved                     */
} STOP_LS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_STOP_LS

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

stop_type

링크 스테이션을 종료하는 방법.

AP_ORDERLY_STOP

노드가 링크 스테이션을 종료하기 전에 정리 작업을 수행합니다.

AP_IMMEDIATE_STOP

노드가 즉시 링크 스테이션을 종료합니다.

ls_name

종료할 링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. **ls_name**의 값은 DEFINE_LS 명령에 있는 것과 일치해야 합니다.

disable

링크 스테이션의 원격 활성화나 요구가 있을 때의 활성화가 사용불

가능하게 되는지의 여부를 지시합니다. AP_NO로 설정하면, 링크 스테이션이 DEFINE_LS 명령에서 **auto_act_supp** 및 **disable_remote_act** 의 값으로 주어진 상태로 복귀합니다. AP_NO로 설정하지 않을 경우, 다음 값을 사용할 수 있습니다(OR로 연결할 수 있음).

AP_AUTO_ACT

요구가 있을 때 국지 노드에서 링크를 다시 활성화할 수 없습니다.

AP_REMOTE_ACT

원격 모드에서 링크를 활성화할 수 없습니다.

disable_remote_act를 AP_YES로 설정하여 구성한 링크는 이 비트가 무시됩니다.(원격 노드에 의한 활성화는 항상 STOP_LS에을 사

; 됩니다k) xA) k)

KO

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

STOP_PORT

STOP_PORT는 포트를 종료할 것을 요청합니다. 이 명령은 반환되어 포트가 종료되었는지의 여부를 지정합니다. STOP_PORT는 또한 프로그램에 포트 상의 링크 스테이션 활성화 재시도를 자동으로 중단하도록 지시하는 데에도 사용할 수 있습니다.

VCB 구조

```
typedef struct stop_port
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   stop_type;      /* Stop Type             */
        unsigned char   port_name[8];   /* name of port          */
} STOP_PORT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_STOP_PORT

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

stop_type
포트를 종료하는 방법.

AP_ORDERLY_STOP
노드가 포트를 종료하기 전에 정리 작업을 수행합니다.

AP_IMMEDIATE_STOP
노드가 즉시 포트를 종료합니다.

port_name
종료할 포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열로, DEFINE_PORT 명령에 있는 것과 일치해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PORT_NAME

AP_UNRECOGNIZED_DEACT_TYPE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_STOP_PORT_PENDING

명령이 취소되었기 때문에 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CANCELLED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

ACTIVATE_SESSION

ACTIVATE_SESSION 명령은 특정 모드의 특성을 사용하여 국지 LU와 지정한 상대방 LU간의 세션 활성화를 요청합니다.

VCB 구조

형 D 1

```
typedef struct activate_session
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* local LU name           */
        unsigned char   lu_alias[8];    /* local LU alias          */
        unsigned char   plu_alias[8];   /* partner LU alias        */
        unsigned char   mode_name[8];   /* mode name               */
        unsigned char   fqplu_name[17]; /* fully qualified partner */
                                        /* LU name                 */
        unsigned char   polarity;       /* requested session       */
                                        /* polarity                */
        unsigned char   session_id[8];  /* session identifier      */
        unsigned char   cnos_permitted; /* is implicit CNOS        */
                                        /* permitted?              */
        unsigned char   reserv4[15];    /* reserved                */
} ACTIVATE_SESSION;
```

형 D 0(이전 레벨)

```
typedef struct activate_session
{
        unsigned short  opcode;         /* verb operation code     */
        unsigned char   reserv2;        /* reserved                */
        unsigned char   format;         /* format                  */
        unsigned short  primary_rc;     /* primary return code     */
        unsigned long   secondary_rc;   /* secondary return code   */
        unsigned char   lu_name[8];     /* local LU name           */
        unsigned char   lu_alias[8];    /* local LU alias          */
        unsigned char   plu_alias[8];   /* partner LU alias        */
        unsigned char   mode_name[8];   /* mode name               */
        unsigned char   fqplu_name[17]; /* fully qualified partner */
                                        /* LU name                 */
        unsigned char   polarity;       /* requested session       */
                                        /* polarity                */
        unsigned char   session_id[8];  /* session identifier      */
} ACTIVATE_SESSION;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_ACTIVATE_SESSION

ACTIVATE_SESSION

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0이나 1로 설정하십시오.

lu_name

세션을 활성화하도록 요청되는 국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

세션을 활성화하도록 요청되는 국지 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정할 경우에만 유효하며, 이 경우 8바이트 모두 유효 바이트이고 반드시 설정해야 합니다. **lu_alias**와 **lu_name** 둘다를 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

plu_alias

국지 LU에서 사용되는 상대방 LU의 별명. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 필수 상대방 LU를 지정하는 데 사용됩니다.

mode_name

구성시에 정의한 네트워킹 특성 집합의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

fqplu_name

상대방 LU의 전체 정식 LU 명. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 **plu_alias** 필드를 모두 0으로 설정할 경우에만 유효합니다.

polarity

세션에 요청되는 극성. 가능한 값은 다음과 같습니다.

AP_POL_EITHER
AP_POL_FIRST_SPEAKER
AP_POL_BIDDER

AP_POL_EITHER를 선택할 경우 ACTIVATE_SESSION은 첫번째 스피커 세션(가능한 경우)을 활성화하며, 그렇지 않을 경우 송수신 요구자 세션이 활성화됩니다. AP_POL_FIRST_SPEAKER 또는 AP_POL_BIDDER의 경우, ACTIVATE_SESSION은 요청된 극성의 세션이 사용가능한 경우에만 실행됩니다.

cnos_permitted
이 필드는 AP_YES 또는 AP_NO로 설정할 수 있습니다. 지정한 모드의 세션 한계가 재설정되어 새로운 세션의 활성화가 불가능한 상태에서 이 필드를 AP_YES로 설정하면, 프로그램이 암시적 CNOS 처리를 시작하여 세션 한계를 초기화합니다. CNOS 처리가 행해지는 동안에는 이 명령이 일시중단됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

session_id
활성화된 세션의 8바이트 식별자(ID).

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_EXCEEDS_MAX_ALLOWED

AP_INVALID_CNOS_PERMITTED
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_MODE_NAME
AP_INVALID_PLU_NAME

명령이 모드의 세션 한계를 초과하면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

Secondary_rc
AP_EXCEEDS_MAX_ALLOWED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

ACTIVATE_SESSION

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

다른 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_ACTIVATION_FAIL_NO_RETRY

AP_ACTIVATION_FAIL_RETRY

DEACTIVATE_CONV_GROUP

DEACTIVATE_CONV_GROUP 명령은 지정한 대화 그룹에 해당하는 세션의 활성종료를 요청합니다. 이 명령은 노드 연산자 기능 API의 일부이지만, 주로 퍼스널 통신이나 통신 서버 APPC API를 사용하는 트랜잭션 프로그램(TP)을 작성하는 응용프로그램 프로그래머들을 위한 것입니다. 대화 그룹 식별자(CGID)는 *퍼스널 통신 클라이언트/서버 통신 프로그래밍*에 정의된 MC_ALLOCATE, ALLOCATE, MC_GET_ATTRIBUTES, GET_ATTRIBUTES 및 RECEIVE_ALLOCATE 명령에 의해 반환됩니다.

VCB 구조

```
typedef struct deactivate_conv_group
{
        unsigned short  opcode;         /* verb operation code           */
        unsigned char   reserv2;        /* reserved                      */
        unsigned char   format;         /* format                        */
        unsigned short  primary_rc;     /* primary return code           */
        unsigned long   secondary_rc;   /* secondary return code         */
        unsigned char   lu_name[8];     /* local LU name                 */
        unsigned char   lu_alias[8];    /* local LU alias                */
        unsigned long   conv_group_id;  /* conversation group identifier */
        unsigned char   type;           /* deactivation type             */
        unsigned char   reserv3[3];     /* reserved                      */
        unsigned long   sense_data;     /* deactivation sense data       */
} DEACTIVATE_CONV_GROUP;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEACTIVATE_CONV_GROUP

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

대화 그룹을 활성종료하도록 요청되는 국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

대화 그룹을 활성종료하도록 요청되는 국지 LU의 별명. 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정할 경우에만 유효하며, 이 경우 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. **lu_name**과 **lu_alias** 둘 다를 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

conv_group_id

활성종료할 세션의 대화 그룹 식별자(CGID).

type 활성종료 유형. 이 필드는 이 명령이 동기적으로 완료되는지 또는 비동기적으로 완료되는지를 지시하는 플래그와 활성종료 유형이 OR로 연결되는 비트 마스크입니다.

활성종료 유형은 다음과 같습니다.

AP_DEACT_CLEANUP
세션이 상대방 LU로부터의 응답을 기다리지 않고 즉시 종료됩니다.

AP_DEACT_NORMAL
세션을 사용하는 모든 대화가 종료된 후에 세션이 종료됩니다.

명령 작동은 다음과 같습니다.

AP_ASYNCHRONOUS_DEACTIVATION
명령이 즉시 반환됩니다.

AP_SYNCHRONOUS_DEACTIVATION
세션이 활성종료된 후에만 명령이 반환됩니다.

sense_data
CLEANUP 유형의 활성종료에서 사용할 감지 데이터를 지정합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CLEANUP_TYPE

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEACTIVATE_SESSION

DEACTIVATE_SESSION 명령은 특정 세션의 활성종료나 특정 모드 상의 모든 세션의 활성종료를 요청합니다.

VCB 구조

```
typedef struct deactivate_session
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   lu_name[8];     /* local LU name           */
        unsigned char   lu_alias[8];    /* local LU alias          */
        unsigned char   session_id[8];  /* session identifier      */
        unsigned char   plu_alias[8];   /* partner LU alias        */
        unsigned char   mode_name[8];   /* mode name               */
        unsigned char   type;           /* deactivation type       */
        unsigned char   reserv3[3];     /* reserved                */
        unsigned long   sense_data;     /* deactivation sense data */
        unsigned char   fqplu_name[17]; /* fully qualified partner */
                                        /* LU name                 */
        unsigned char   reserv4[20];    /* reserved                */
} DEACTIVATE_SESSION;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEACTIVATE_SESSION

format

VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

세션을 활성종료하도록 요청되는 국지 LU의 LU 명. 이 이름은 8 바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

세션을 활성종료하도록 요청되는 국지 LU의 별명. 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정할 경우에만 유효하며, 이 경우 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. **lu_name**과 **lu_alias** 필드 둘 다를 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

session_id
활성종료할 세션의 8바이트 식별자(ID). 이 필드를 모두 0으로 설정하면, 퍼스널 통신이나 통신 서버가 상대방 LU 및 모드의 모든 세션을 활성종료합니다.

plu_alias
국지 LU에서 사용되는 상대방 LU의 별명. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 필수 상대방 LU를 지정하는 데 사용됩니다.

mode_name
구성시에 정의한 네트워킹 특성 집합의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

type 활성종료 유형. 이 필드는 이 명령이 동기적으로 완료되는지 또는 비동기적으로 완료되는지를 지시하는 플래그와 활성종료 유형이 OR로 연결되는 비트 마스크입니다.

활성종료 유형은 다음과 같습니다.

AP_DEACT_CLEANUP
세션이 상대방 LU로부터의 응답을 기다리지 않고 즉시 종료됩니다.

AP_DEACT_NORMAL
세션을 사용하는 모든 대화가 종료된 후에 세션이 종료됩니다.

명령 작동은 다음과 같습니다.

AP_ASYNCHRONOUS_DEACTIVATION
명령이 즉시 반환됩니다.

AP_SYNCHRONOUS_DEACTIVATION
세션이 활성종료된 후에만 명령이 반환됩니다.

sense_data
CLEANUP 유형의 활성종료에 사용할 감지 데이터를 지정합니다.

fqplu_name
상대방 LU의 전체 정식 LU 명. 길이는 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 **plu_alias** 필드를 모두 0으로 설정할 경우에만 유효합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

session_id를 기존의 세션과 일치시킬 수 없을 경우, 이는 해당 세션이 이미 활성종료되었기 때문인 것으로 간주됩니다. 이러한 경우, 명령은 성공적으로 완료됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MODE_NAME

AP_INVALID_PLU_NAME
AP_INVALID_CLEANUP_TYPE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

PATH_SWITCH

PATH_SWITCH 명령은 퍼스널 통신이나 통신 서버에 고성능 경로지정(HPR)을 지원하는 연결에서 경로를 전환하도록 요청합니다. 보다 나은 경로가 없으면 연결이 변경되지 않은 상태로 있습니다.

VCB 구조

```
typedef struct path_switch
{
        unsigned short  opcode;         /* verb operation code   */
        unsigned char   reserv2;        /* reserved              */
        unsigned char   format;         /* format                */
        unsigned short  primary_rc;     /* primary return code   */
        unsigned long   secondary_rc;   /* secondary return code */
        unsigned char   rtp_connection_name[8];
                                        /* RTP connection name   */
} PATH_SWITCH;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_PATH_SWITCH

format
VCB 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

rtp_connection_name
경로 전환될 RTP 연결을 식별합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이며 반드시 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_RTP_CONNECTION

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

PATH_SWITCH

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PATH_SWITCH_IN_PROGRESS

경로 전환 시도의 실패로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNSUCCESSFUL

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

제6장 조회 명령

이 장에서는 노드 구성 및 상태에 관한 정보를 조회하는 데 사용되는 명령에 대해 기술합니다.

SNA API 클라이언트에서는 특정 매개변수들만이 지원됩니다.

상세한 정보는 이 장에 걸쳐 있는 노트 *패*드 아이콘을 참조하십시오.

QUERY_ADJACENT_NN

이 명령은 통신 서버에만 적용됩니다.

QUERY_ADJACENT_NN은 네트워크 노드에서만 사용되며 인접 네트워크 노드(즉, 세션 CP-CP 세션이 활동중이거나 언젠가 활동중이었던 그러한 네트워크 노드)에 관한 정보가 반환됩니다.

인접 노드 정보는 형식화된 리스트로 반환됩니다. 특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **adj_nncp_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **adj_nncp_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작됩니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_adjacent_nn
{
   unsigned short  opcode;                 /* verb operation code          */
   unsigned char   reserv2;                /* reserved                     */
   unsigned char   format;                 /* format                       */
   unsigned short  primary_rc;             /* primary return code          */
   unsigned long   secondary_rc;           /* secondary return code        */
   unsigned char   *buf_ptr;               /* pointer to buffer            */
   unsigned long   buf_size;               /* buffer size                  */
   unsigned long   total_buf_size;         /* total buffer size required   */
   unsigned short  num_entries;            /* number of entries            */
   unsigned short  total_num_entries;      /* total number of entries      */
   unsigned char   list_options;           /* listing options              */
   unsigned char   reserv3;                /* reserved                     */
        unsigned char   adj_nncp_name[17]; /* CP name of adj network node  */
} QUERY_ADJACENT_NN;

typedef struct adj_nncp_data
{
   unsigned short  overlay_size;           /* size of this entry           */
        unsigned char   adj_nncp_name[17]; /* CP name of adj. network node */
        unsigned char   cp_cp_sess_status; /* CP-CP session status         */
        unsigned long   out_of_seq_tdus;   /* out of sequence TDUs         */
        unsigned long   last_frsn_sent;    /* last FRSN sent               */
        unsigned long   last_frsn_rcvd;    /* last FRSN received           */
   unsigned char   reserva[20];            /* reserved                     */
} ADJ_NNCP_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_ADJACENT_NN

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **adj_nncp_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

adj_nncp_name

EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 전체 정식 17 바이트의 인접 네트워크 노드 이름(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다). **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 최대 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

adj_nncp_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

adj_nncp_data.adj_nncp_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 인접 네트워크 노드의 17 바이트 전체 정식 CP명(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다).

adj_nncp_data.cp_cp_sess_status
CP-CP 세션의 상태. 이는 다음 중 하나로 설정됩니다.

AP-ACTIVE
AP_CONWINNER_ACTIVE
AP_CONLOSER_ACTIVE
AP_INACTIVE

adj_nncp_data.out_of_seq_tdus
이 노드로부터 수신된 out_of_sequence TDU들의 갯수.

adj_nncp_data.last_frsn_sent
이 노드로 송신된 마지막 흐름 감소의 순서 번호.

adj_nncp_data.last_frsn_rcvd
이 노드로부터 수신된 마지막 흐름 감소의 순서 번호.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_ADJ_NNCP_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_ADJACENT_NODE

QUERY_ADJACENT_NODE는 DEFINE_ADJACENT_NODE에서 구성된 인접 노드에 관한 정보를 반환합니다.

정보는 정렬된 리스트로 반환됩니다. 리스트에 있는 각 정보는 인접 CP와 연관된 각 LU에 대한 ADJACENT_NODE_LU_DATA 중복이 뒤따라 오는 인접 CP에 관한 정보를 포함하고 있는 ADJACENT_NODE_DATA 중복으로 구성되어 있습니다.

입력항목은 **cp_name**으로 정렬된 후, **fqlu_name**으로 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_adjacent_node
{
   unsigned short  opcode;             /* Verb operation code           */
   unsigned char   reserv2;            /* reserved                      */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   cp_name[17];   /* CP name of adjacent node      */
} QUERY_ADJACENT_NODE;

typedef struct adjacent_node_data
{
   unsigned short  overlay_size;           /* size of this entry        */
        unsigned short  sub_overlay_size;  /* size of this stub entry   */
        unsigned char   cp_name[17];       /* CP name                   */
        DESCRIPTION     description;       /* resource description      */
        unsigned char   reserv3[19];       /* reserved                  */
        unsigned short  num_of_lus;        /* number of LUs             */
} ADJACENT_NODE_DATA;

typedef struct cn_det_data
{
        unsigned short  num_act_ports;     /* number of active ports    */
   unsigned char   reserva[20];            /* reserved                  */
} CN_DET_DATA;

typedef struct cn_def_data
{
   unsigned char   description[RD_LEN];
                                           /* resource description      */
```

```
        unsigned char   num_ports;         /* number of ports on CN     */
        unsigned char   reserv1[16];       /* reserved                  */
        TG_DEFINED_CHARS tg_chars;         /* TG characteristics        */
} CN_DEF_DATA;

typedef struct adjacent_node_lu_data
{
        unsigned short  overlay_size;      /* effective capacity        */
        unsigned char   reserve2[2];       /* reserved                  */
        ADJACENT_NODE_LU adj_lu_def_data;  /* Adjacent LU defined data  */
} ADJACENT_NODE_LU_DATA;

typedef struct adjacent_node_lu
{
        unsigned char   wildcard_lu;       /* Is this LU a wildcard?    */
        unsigned char   fqlu_name[17];     /* Fully-Qualified LU name   */
        unsigned char   reserve1[6];       /* reserved                  */
        ADJACENT_NODE_LU adj_lu_def_data;  /* Adjacent LU defined data  */
} ADJACENT_NODE_LU;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_ADJACENT_NODE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **cp_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 프로그램이 유지보수하는 디렉토리에 있는 첫번째 인접 노드로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

cp_name
인접 노드의 전체 정식 이름. 이 이름은 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

adjacent_node_data.overlay_size
ADJACENT_NODE_LU_DATA 구조체를 포함하여 이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

adjacent_node_data.sub_overlay_size
The ADJACENT_NODE_LU_DATA 구조체를 포함하지 않고, 입력항목의 노드 부분에 있는 바이트 수로 이는 입력항목에서 첫번째 ADJACENT_NODE_LU_DATA 필드에 대한 오프셋입니다.

adjacent_node_data.cp_name
인접 노드의 전체 정식 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

cn_data.det_data.num_act_ports
연결 네트워크상의 활동중 포트의 수를 제공하는 동적 값.

adjacent_node_data.description
자원 설명(DEFINE_ADJACENT_NODE에서 지정된 것과 같이). 이 필드의 길이는 4 바이트의 배수이어야 하며 0이 아니어야 합니다.

adjacent_node_data.num_of_lus
이 인접 노드에 대해 정의된 LU의 갯수. 각 LU에 대한 ADJACENT_NODE_LU_DATA 구조체가 이어집니다.

adjacent_node_lu_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

adjacent_node_lu_data.adj_lu_def_data.wildcard_lu
LU명이 총칭 문자로 정의되는지의 여부를 나타냅니다.

adjacent_node_lu_data.adj_lu_def_data.fqlu_name
인접 노드의 전체 정식 이름. 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CP_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_CN

QUERY_CN은 인접 연결 네트워크에 관한 정보를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_CN에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 형식화된 리스트로 반환됩니다. 특정 CN에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **fqcn_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **fqcn_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_cn
{
   unsigned short  opcode;             /* Verb operation code           */
   unsigned char   attributes;         /* verb attributes               */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   fqcn_name[17]; /* Name of connection network    */
} QUERY_CN;

typedef struct cn_data
{
   unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   fqcn_name[17]; /* Name of connection network    */
   unsigned char   reserv1;            /* reserved                      */
        CN_DET_DATA     det_data;      /* Determined data               */
        CN_DEF_DATA     def_data;      /* Defined data                  */
} CN_DATA;

typedef struct cn_det_data
{
        unsigned short  num_act_ports;     /* number of active ports    */
   unsigned char   reserva[20];            /* reserved                  */
} CN_DET_DATA;
```

```
typedef struct cn_def_data
{
   unsigned char   description[RD_LEN];
                                          /* resource description      */
        unsigned char   num_ports;        /* number of ports on CN     */
        unsigned char   reserv1[16];      /* reserved                  */
        TG_DEFINED_CHARS tg_chars;        /* TG characteristics        */
} CN_DEF_DATA;

typedef struct tg_defined_chars
{
        unsigned char   effect_cap;       /* effective capacity        */
        unsigned char   reserve1[5];      /* reserved                  */
        unsigned char   connect_cost;     /* connection cost           */
        unsigned char   byte_cost;        /* byte cost                 */
        unsigned char   reserve2;         /* reserved                  */
        unsigned char   security;         /* security                  */
        unsigned char   prop_delay;       /* propagation delay         */
        unsigned char   modem_class;      /* modem class               */
        unsigned char   user_def_parm_1;  /* user-defined parameter 1  */
        unsigned char   user_def_parm_2;  /* user-defined parameter 2  */
        unsigned char   user_def_parm_3;  /* user-defined parameter 3  */
} TG_DEFINED_CHARS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_CN

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **fqcn_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

fqcn_name

전체 정식의 17 바이트의 연결 네트워크 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

buf_size

버퍼에 반환되는 정보의 길이.

total_buf_size

요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries

실제로 반환된 입력항목의 갯수.

total_num_entries

반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

cn_data.overlay_size

이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

cn_data.fqcn_name

전체 정식의 17 바이트의 연결 네트워크 이름. 이 이름은 EBCDIC점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A

EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

cn_data.det_data.num_act_ports
연결 네트워크상의 활동중인 포트의 수를 제공하는 동적 값.

cn_data.def_data.description
자원 설명(DEFINE_CN에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

cn_data.def_data.num_ports
연결 네트워크상의 포트 수.

cn_data.def_data.tg_chars.effect_cap
유효 용량의 실제 단위. 이 값은 0.1mmm * 2 eeeee 공식으로 표현되는 1 바이트의 부동 소수로 인코딩됩니다. 여기서 바이트의 비트 표현은 eeeeemmm입니다. 유효 용량의 각 단위는 초당 300 비트와 일치합니다.

cn_data.def_data.tg_chars.connect_cost Cost per connect time.
유효 값은 0—255 범주의 정수 값으로, 여기서 0은 연결 시간당 최저 비용이며 255는 최고 비용입니다.

cn_data.def_data.tg_chars.byte_cost
바이트당 비용. 유효 값은 0—255 범주의 정수 값으로, 여기서 0은 바이트당 최저 비용이며 255는 최고 비용입니다.

cn_data.def_data.tg_chars.security
아래의 리스트에서 기술된 대로의 보안 값.

AP_SEC_NONSECURE
보안 없음.

AP_SEC_PUBLIC_SWITCHED_NETWORK
이 연결 네트워크를 통해 전송되는 데이터는 공중 교환 통신망(PSN)을 통해 흐를 것입니다.

AP_SEC_UNDERGROUND_CABLE
안전한 지하 케이블을 통해 데이터가 전송됩니다.

AP_SEC_SECURE_CONDUIT
회선이 보호되어 있지 않은 안전 배관(conduit)입니다.

AP_SEC_GUARDED_CONDUIT
배관이 물리적 탭핑에 대비하여 보호됩니다.

AP_SEC_ENCRYPTED
회선을 통한 암호화.

AP_SEC_GUARDED_RADIATION
회선이 물리적 또는 방사 탭핑에 대비하여 보호됩니다.

cn_data.def_data.tg_chars.prop_delay
0신호가 링크를 다 통과하는 데 걸리는 시간을 백만분의 1초 단위로

나타내는 전달 지연. 이 값은 0.1mmm * 2 eeeee 공식으로 표현되는 1 바이트의 부동 소수로 인코딩됩니다. 여기서 바이트의 비트 표현은 eeeeemmm입니다. 생략시 값이 아래에 나열됩니다.

AP_PROP_DELAY_MINIMUM
전달 지연 없음.

AP_PROP_DELAY_LAN
480 백만분의 1초 미만의 지연.

AP_PROP_DELAY_TELEPHONE
480과 49 512 백만분의 1초간의 지연.

AP_PROP_DELAY_PKT_SWITCHED_NET
49 512와 245 760 백만분의 1초간의 지연.

AP_PROP_DELAY_SATELLITE
245 760 백만분의 1초보다 더 긴 지연.

AP_PROP_DELAY_MAXIMUM
최대 전달 지연.

cn_data.def_data.tg_chars.modem_class
예약됨. 이 필드는 항상 0으로 설정되어야 합니다.

cn_data.def_data.tg_chars.user_def_parm_1
0—255 범주의 사용자 정의 매개변수.

cn_data.def_data.tg_chars.user_def_parm_2
0—255 범주의 사용자 정의 매개변수.

cn_data.def_data.tg_chars.user_def_parm_3
0—255 범주의 사용자 정의 매개변수.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CN_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_CN_PORT

QUERY_CN_PORT는 인접 연결 네트워크에서 정의된 포트에 관한 정보를 반환합니다. 정보는 형식화된 리스트로 반환됩니다. 특정 port에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **port_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. **fqcn_name** 필드가 항상 유효한 연결 네트워크의 이름으로 설정되어야 함에 주의하십시오.

리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

VCB 구조

```
typedef struct query_cn_port
{
   unsigned short  opcode;             /* Verb operation code           */
   unsigned char   reserv2;            /* reserved                      */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   fqcn_name[17]; /* Name of connection network    */
   unsigned char   port_name[8];       /* port name                     */
} QUERY_CN_PORT;

typedef struct cn_port_data
{
   unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   fqcn_name[17]; /* Name of connection network    */
        unsigned char   port_name[8];  /* name of port                  */
        unsigned char   tg_num;        /* transmission group number     */
   unsigned char   reserva[20];        /* reserved                      */
} CN_PORT_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_CN_PORT

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램

은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **fqcn_name**과 **port_name**의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

fqcn_name
전체 정식의 17 바이트의 연결 네트워크 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 항상 설정되어야 합니다.

port_name
국지로 표시가능한 문자 세트로 된 8 바이트의 문자열. 8 바이트 모두 유효하며 설정되어야 합니다. **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

cn_port_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

cn_port_data.fqcn_name
전체 정식의 17 바이트의 연결 네트워크 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

cn_port_data.port_name
8 바이트의 국지로 표시가능한 문자 세트로 된 포트명. 8 바이트 모두 유효합니다.

cn_port_data.tg_num
지정된 포트에 대한 전송 그룹(TG) 번호.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_CN_NAME

AP_INVALID_PORT_NAME
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_CONVERSATION

QUERY_CN_PORT는 지정된 LU를 통해 실행중인 대화에 관한 리스트 정보를 반환합니다. 특정 대화에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **conv_id** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. **lu_alias** 필드가 항상 설정되어야 함에 주의하십시오. lu_name이 0이 아니면 lu_alias에 우선하여 사용될 것입니다.

리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **conv_id**에 의해 정렬됩니다. AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 색인에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_conversation
{
   unsigned short  opcode;             /* Verb operation code          */
   unsigned char   reserv2;            /* reserved                     */
   unsigned char   format;             /* format                       */
   unsigned short  primary_rc;         /* Primary return code          */
   unsigned long   secondary_rc;       /* Secondary return code        */
   unsigned char   *buf_ptr;           /* pointer to buffer            */
   unsigned long   buf_size;           /* buffer size                  */
   unsigned long   total_buf_size;     /* total buffer size required   */
   unsigned short  num_entries;        /* number of entries            */
   unsigned short  total_num_entries;  /* total number of entries      */
   unsigned char   list_options;       /* listing options              */
   unsigned char   reserv3;            /* reserved                     */
        unsigned char   lu_name[8];        /* local LU name             */
        unsigned char   lu_alias[8];       /* local LU alias            */
        unsigned long   conv_id;       /* conversation identifier      */
        unsigned char   session_id[8];     /* session identifier        */
        unsigned char   reserv4[12];       /* reserved                  */
} QUERY_CONVERSATION;

typedef struct conv_summary
{
   unsigned short  overlay_size;       /* size of this entry           */
        unsigned long   conv_id;       /* conversation identifier      */
        unsigned char   local_tp_name[64]; /* Name of local TP         */
        unsigned char   partner_tp_name[64];
                                       /* Name of partner TP           */
        unsigned char   tp_id[8];          /* TP identifier            */
        unsigned char   sess_id[8];        /* session identifier       */
        unsigned long   conv_start_time;/* time conversation was       */
                                        /* started                     */
        unsigned long   bytes_sent;     /* bytes sent so far           */
        unsigned long   bytes_received; /* bytes received so far       */
        unsigned char   conv_state;     /* conversation state          */
        unsigned char   duplex_type;    /* conversation duplex type    */
} CONV_SUMMARY;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_CONVERSATION

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **index**(다음을 참조)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
국지 LU의 이름. 이는 8–바이트 영숫자 유형 A EBCDIC 문자열(숫자로 시작하지 않음)이며, 오른쪽이 EBCDIC 공백으로 채워집니다.

lu_alias
국지 TP에게 국지 LU가 알려지는 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다.

conv_id
대화 ID.

session_id
모두 2진 0이면, 반환된 대화를 필터하는 데 이 필드가 사용되지 않습니다. 0이 아니면, 세션 ID가 제공된 값과 일치하는 대화들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

conv_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

conv_summary.conv_id
대화 ID.

이 매개변수의 값은 트랜잭션 조치 시동시에는 ALLOCATE 명령에 의해 반환되거나 트랜잭션 프로그램 시동시에는 RECEIVE_ALLOCATE에 의해 반환됩니다.

conv_summary.local_tp_name
국지 트랜잭션 프로그램(TP)의 이름.

conv_summary.partner_tp_name
상대방 트랜잭션 프로그램(TP)의 이름. 이는 국지로 초기화된 대화에 대해서만 유효합니다. 원격으로 초기화된 대화의 경우에는 공백입니다.

conv_summary.tp_id
트랜잭션 프로그램(TP)에 지정된 트랜잭션 프로그램(TP) 식별자(ID). 이 식별자는 API stub나 NOF 트랜잭션 프로그램 관리자에 의해 지정됩니다.

conv_summary.sess_id
이 대화에 할당된 세션의 식별자(ID).

QUERY_CONVERSATION

conv_summary.conv_start_time
노드가 시작된 시간부터 대화가 시작된 시간까지의 경과 시간으로 100분의 1초 단위.

conv_summary.bytes_sent
이 대화상에서 지금까지 송신된 바이트 수.

conv_summary.bytes_received
이 대화상에서 지금까지 수신된 바이트 수.

conv_summary_conv_state
conv_id로 식별되는 대화의 현재 상태. 비동시 양방향(전송) 대화의 경우, 다음 중 하나입니다.

AP_RESET_STATE

AP_SEND_STATE
AP_RECEIVE_STATE
AP_CONFIRM_STATE
AP_CONFIRM_SEND_STATE
AP_CONFIRM_DEALL_STATE
AP_PEND_POST_STAT
AP_PEND_DEALL_STATE
AP_END_CONV_STATE
AP_SEND_PENDING_STATE
AP_POST_ON_RECEIPT_STATE

동시 양방향(전송) 대화의 경우, 다음 중 하나입니다.

AP_RESET_STATE
AP_SEND_RECEIVE_STATE
AP_SEND_ONLY_STATE
AP_RECEIVE_ONLY_STATE

conv_summary.duplex_type
이 대화가 비동시 양방향(전송)인지 동시 양방향(전송)인지를 지정합니다.

AP_HALF_DUPLEX
AP_FULL_DUPLEX

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_INVALID_LU_ALIAS
AP_INVALID_LU_NAME

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_COS

QUERY_COS는 특정 서비스 클래스(COS)에 대한 전송경로 계산 정보를 반환합니다. 정보는 형식된 리스트로 반환됩니다. 특정 COS에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **cos_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오. 이 리스트는 **cos_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 정의된 정렬하기에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_cos
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char   reserv2;             /* reserved                     */
   unsigned char   format;              /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   *buf_ptr;            /* pointer to buffer            */
   unsigned long   buf_size;            /* buffer size                  */
   unsigned long   total_buf_size;      /* total buffer size required   */
   unsigned short  num_entries;         /* number of entries            */
   unsigned short  total_num_entries;   /* total number of entries      */
   unsigned char   list_options;        /* listing options              */
   unsigned char   reserv3;             /* reserved                     */
        unsigned char   cos_name[8];    /* COS name                   */
} QUERY_COS;

typedef struct cos_data
{
   unsigned short  overlay_size;        /* size of this entry           */
        unsigned char   cos_name[8];    /* COS name                   */
   unsigned char   description[RD_LEN];
                                        /* resource description         */
        unsigned char   transmission_priority;
                                        /* transmission priority        */
   unsigned char   reserv1;             /* reserved                     */
        unsigned short  num_of_node_rows;/* number of node rows         */
        unsigned short  num_of_tg_rows; /* number of TG rows            */
        unsigned long   trees;          /* number of tree caches for COS*/
        unsigned long   calcs;          /* number of route calculation  */
                                        /* for this COS                 */
        unsigned long   rejs;           /* number of route rejects      */
                                        /* for COS                      */
   unsigned char   reserva[20];         /* reserved                   */
} COS_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_COS

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **cos_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

cos_name
서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

cos_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

cos_data.cos_name
서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 오른쪽은 EBCDIC 공백으로 채워집니다.

cos_data.description
자원 설명(DEFINE_COS에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

cos_data.transmission_priority
전송 우선순위. 다음 값 중 하나로 설정됩니다.

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

cos_data.num_of_node_rows
이 COS에 대한 노드 행의 수.

cos_data.num_of_tg_rows
이 COS에 대한 TG 행의 수.

cos_data.trees
최종 초기화 이래 COS에 대해 구축된 전송경로 트리 캐쉬(cache)의 수.

cos_data.calcs
이 서비스 클래스(COS)를 지정하는 세션 활성화 요청의 수(그 결과 전송경로 계산).

cos_data.rejs
네트워크를 통해 이 노드에서 명명된 목적지로 받아들일 수 있는(지정된 서비스 클래스를 사용하여) 전송경로가 없으므로 실패한 세션

활성화 요청의 수. 전송경로는 전체가 지정된 서비스 클래스(COS)를 제공할 수 있는 활동중 TG 및 노드들로 이루어진 경우에만 받아들일 수 있습니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_COS_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DEFAULT_PU

QUERY_DEFAULT_PU는 사용자로 하여금 DEFINE_DEFAULT_PU 명령을 사용하여 정의된 생략시 PU를 조회할 수 있게 합니다.

VCB 구조

```
typedef struct query_default_pu
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char   reserv2;             /* reserved                   */
   unsigned char   format;              /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
        unsigned char   def_pu_name[8]; /* default PU name            */
   unsigned char   description[RD_LEN];
                                        /* resource description       */
        unsigned char   def_pu_sess[8]; /* PU name of active          */
                                        /* default session            */
        unsigned char   reserv3[16];    /* reserved                   */
} QUERY_DEFAULT_PU;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DEFAULT_PU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

def_pu_name

가장 최근의 DEFINE_DEFAULT_PU 명령에서 지정된 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 발행된 DEFINE_DEFAULT_PU 명령이 없으면, 이 필드가 모두 0으로 설정됩니다.

description

자원 설명(DEFINE_DEFAULT_PU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

def_pu_sess

현재 활동중인 생략시 PU 세션과 연관된 PU의 이름. 생략시 PU가 정의된 경우에는 **def_pu_name** 필드와 다를 것이지만, 그것과 연관된

세션은 활동중이지 않습니다. 이 경우, 퍼스널 통신이나 통신 서버는 정의된 생략시 PU가 활동중으로 될 때까지 계속해서 이전 생략시 PU와 연관된 세션을 사용합니다. 활동중 PU 세션이 없으면, 이 필드가 모두 0으로 설정됩니다.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DEFAULTS

QUERY_DEFAULTS는 사용자로 하여금 DEFINE_DEFAULTS 명령을 사용하여 정의된 기본설정 값을 조회할 수 있게 합니다.

VCB 구조

```
typedef struct query_defaults
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char   reserv2;             /* reserved                   */
   unsigned char   format;              /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
        DEFAULT_CHARS   default_chars;  /* default information        */
} QUERY_DEFAULTS;

typedef struct default_chars
{
   unsigned char   description[RD_LEN];
                                        /* resource description       */
        unsigned char   mode_name[8];   /* default mode name          */
        unsigned char   implicit_plu_forbidden;
                                        /* disallow implicit          */
                                        /* PLUs ?                     */
        unsigned char   specific_security_codes;
                                        /* generic security           */
                                        /* sense codes                */
        unsigned char   limited_timeout;/* timeout for limited        */
                                        /* sessions                   */
        unsigned char   reserv[244];    /* reserved                   */
} DEFAULT_CHARS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DEFAULTS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

default_chars.description

자원 설명(DEFINE_DEFAULTS에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

default_chars.mode_name
가장 최근의 DEFINE_DEFAULTS 명령에서 지정된 모드의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 발행된 DEFINE_DEFAULTS 명령이 없으면, 이 필드가 모두 0으로 설정됩니다.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DIRECTORY_ENTRY

QUERY_DIRECTORY_LU는 디렉토리 데이터베이스로부터 LU의 리스트를 반환합니다. 정보는 요약이나 상세 정보, 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LU에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **resource_name** 및 **resource_type** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

국지 노드가 네트워크 노드이면, 다음과 같이 정보가 반환됩니다.

첫번째 네트워크 노드
- 네트워크 노드에 위치한 첫번째 LU
- 네트워크 노드에 위치한 두 번째 LU
- ...
- 네트워크 노드에 위치한 n번째 LU

이 네트워크 노드가 서비스하는 첫번째 끝 노드
- 끝 노드(1)에 위치한 첫번째 LU
- 끝 노드(1)에 위치한 두 번째 LU
- ...
- 끝 노드(1)에 위치한 n번째 LU

...

이 네트워크 노드가 서비스하는 n번째 끝 노드
- 끝 노드(n)에 위치한 첫번째 LU
- 끝 노드(n)에 위치한 두 번째 LU
- ...

두 번째 네트워크 노드

...등등..

프로그램이 끝 노드로 작동하고 있을 때에는 자원 리스트에서 반환된 첫번째 입력항목이 EN CP입니다.(끝 노드의 네트워크 노드 서버에 대해서는 어떤 입력항목도 반환되지 않습니다.)

반환된 이 디렉토리 입력항목 리스트는 부모(parent) 이름(및 유형)에 의해 필터링됩니다. 이 경우, **parent_name**과 **parent_type** 필드가 모두 설정되어야 합니다.(그렇지 않으면, 이 필드들이 모두 0으로 설정됩니다.) 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

형D 1

```
typedef struct query_directory_entry{
   unsigned short  opcode;                  /* verb operation code         */
   unsigned char   reserv2;                 /* reserved                    */
   unsigned char   format;                  /* format                      */
   unsigned short  primary_rc;              /* primary return code         */
   unsigned long   secondary_rc;            /* secondary return code       */
   unsigned char   *buf_ptr;                /* pointer to buffer           */
   unsigned long   buf_size;                /* buffer size                 */
   unsigned long   total_buf_size;          /* total buffer size required  */
   unsigned short  num_entries;             /* number of entries           */
   unsigned short  total_num_entries;       /* total number of entries     */
   unsigned char   list_options;            /* listing options             */
   unsigned char   reserv3;                 /* reserved                    */
        unsigned char   resource_name[17];  /* network qualified res name  */
        unsigned char   reserv4;            /* reserved                    */
        unsigned short  resource_type;      /* Resource type               */
        unsigned char   parent_name[17];    /* parent name filter          */
        unsigned char   reserv5;            /* reserved                    */
        unsigned short  parent_type;        /* parent type                 */
        unsigned char   reserv6[24];        /* reserved                    */
} QUERY_DIRECTORY_ENTRY;

typedef struct directory_entry_summary
{
   unsigned short  overlay_size;            /* size of this entry          */
        unsigned char   resource_name[17];  /* network qualified res name  */
        unsigned char   reserve1;           /* reserved                    */
        unsigned short  resource_type;      /* Resource type               */
   unsigned char   description[RD_LEN];     /* resource description        */
        unsigned char   real_owning_cp_type; /* real owning CP type        */
        unsigned char   real_owning_cp_name[17];
                                            /* real owning CP name         */
} DIRECTORY_ENTRY_SUMMARY;

typedef struct directory_entry_detail
{
   unsigned short  overlay_size;            /* size of this entry          */
        unsigned char   resource_name[17];  /* network qualified res name  */
        unsigned char   reserv1a;           /* reserved                    */
        unsigned short  resource type;      /* Resource type               */
   unsigned char   description[RD_LEN];     /* resource description        */
        unsigned char   parent_name[17];    /* network qualified           */
                                            /* parent name                 */
        unsigned char   reserv1b;           /* reserved                    */
        unsigned short  parent_type;        /* parent resource type        */
        unsigned char   entry_type;         /* Type of the directory entry */
        unsigned char   location;           /* Resource location           */
        unsigned char   real_owning_cp_type; /* real owning CP type        */
        unsigned char   real_owning_cp_name[17];
                                            /* real owning CP name         */
        unsigned char   reserva;            /* reserved                    */
} DIRECTORY_LU_DETAIL;
```

VCB 구조

형 D 0(이전 레벨)

```
typedef struct query_directory_entry{
   unsigned short  opcode;                  /* verb operation code         */
   unsigned char   reserv2;                 /* reserved                    */
   unsigned char   format;                  /* format                      */
   unsigned short  primary_rc;              /* primary return code         */
   unsigned long   secondary_rc;            /* secondary return code       */
   unsigned char   *buf_ptr;                /* pointer to buffer           */
   unsigned long   buf_size;                /* buffer size                 */
   unsigned long   total_buf_size;          /* total buffer size required  */
   unsigned short  num_entries;             /* number of entries           */
   unsigned short  total_num_entries;       /* total number of entries     */
   unsigned char   list_options;            /* listing options             */
   unsigned char   reserv3;                 /* reserved                    */
```

```
        unsigned char   resource_name[17];   /* network qualified res name  */
        unsigned char   reserv4;             /* reserved                    */
        unsigned short  resource_type;       /* Resource type               */
        unsigned char   parent_name[17];     /* parent name filter          */
        unsigned char   reserv5;             /* reserved                    */
        unsigned short  parent_type;         /* parent type                 */
} QUERY_DIRECTORY_ENTRY;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DIRECTORY_ENTRY

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오. VCB의 형식에 영향을 미치는 것 외에, 형식 1만이 AP_DLUR_LU_RESOURCE의 자원을 반환합니다.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **resource_name**과 **resource_type**의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며, 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

resource_type
자원 유형. 다음 중 하나를 참조하십시오.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE
AP_DLUR_LU_RESOURCE

list_options가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

parent_name
부모(parent) 이름 필터. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드가 설정되면, 지정된 부모(parent)에 속하는 디렉토리 입력항목만이 반환됩니다.(그리고 이 경우에는 **parent_name** 필드 역시 설정되어야 합니다.) 이 필드는 마치 모두 0으로 설정된 것 같습니다.

parent_type
parent_name 필드에 지정된 부모(parent)의 유형. **parent_name**이 0이 아니면 유형이 지정되어야 하며, 그렇지 않은 경우에는 이 필드가 0으로 설정되어야 합니다. 다음 중 하나로 설정될 수 있습니다.

AP_ENCP_RESOURCE
AP_NNCP_RESOURCE

list_options가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
반환된 디렉토리 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

directory_entry_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

directory_entry_summary.resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_entry_summary.resource_type
자원 유형. 이는 다음 중 하나일 수 있습니다.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE
AP_DLUR_LU_RESOURCE

(형 D 이 0으로 설정되면 반환되지 않습니다.)

directory_entry_summary.description
다음에서 지정된 것과 같이의 자원 설명.

DEFINE_LOCAL_LU
DEFINE_DIRECTORY_ENTRY
DEFINE_ADJACENT_LEN_NODE 또는
DEFINE_ADJACENT_NODE

directory_entry_summary.real_owning_cp_type
NN 및 BrNN 만: 실제 소유 CP 유형. 이는 다음 중 하나일 수 있습니다.

AP_NONE
실제 소유 CP는 부모(parent) 자원입니다.

AP_ENCP_RESOURCE
실제 소유 CP는 부모(parent) 자원이 아니고 EN입니다.

기타 노드 유형: 이 필드가 AP_NONE으로 설정됩니다.

directory_entry_summary.real_owning_cp_name
NN 및 BrNN 만: 전체 정식 소유 CP명. 이 이름의 길이는 17 바이

트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

실제 소유 CP가 부모(parent)이면, 이 필드가 2진 0으로 설정됩니다.

실제 소유 CP가 부모(parent)가 아닌 경우에는, 이 필드가 실제 소유 CP의 이름으로 설정됩니다.

자원을 BrNN의 도메인에 있는 EN이 소유하는 경우에는, 실제 소유 CP가 BrNN의 NNS의 디렉토리에서 부모(parent)가 아닙니다. 이 경우 실제 소유 CP는 EN이지만, 부모(parent)는 BrNN입니다.

기타 노드 유형: 이 필드가 2진 0으로 설정됩니다.

directory_entry_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

directory_entry_detail.resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_entry_detail.resource_type
자원 유형. 이는 다음 중 하나일 수 있습니다.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

directory_entry_detail.description
다음에서 지정된 것과 같이의 자원 설명.

DEFINE_LOCAL_LU
DEFINE_DIRECTORY_ENTRY
DEFINE_ADJACENT_LEN_NODE 또는
DEFINE_ADJACENT_NODE

directory_entry_detail.parent_name
LU를 서비스하는 노드의 전체 정식 부모(parent) 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_entry_detail.parent_type
부모(parent) 자원 유형. 이는 다음 중 하나일 수 있습니다.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE

directory_lu_detail.entry_type
디렉토리 입력항목의 유형을 지정합니다. 이는 다음 값 중 하나일 수 있습니다.

AP_HOME
국지 자원.

AP_CACHE
캐쉬(cache) 입력항목.

AP_REGISTER
등록된 자원(NN 만).

directory_entry_detail.location
다음 값 중 하나일 수 있는 자원의 위치를 지정합니다.

AP_LOCAL
자원이 국지 노드에 있습니다.

AP_DOMAIN
자원이 접속된 끝 노드에 속합니다.

AP_CROSS_DOMAIN
자원이 국지 노드의 도메인 내에 없습니다.

directory_entry_detail.real_owning_cp_type
NN 및 BrNN 만: 실제 소유 CP 유형. 이는 다음 중 하나일 수 있습니다.

AP_NONE
실제 소유 CP는 부모(parent) 자원입니다.

AP_ENCP_RESOURCE
실제 소유 CP는 부모(parent) 자원이 아니고 EN입니다.

기타 노드 유형: 이 필드가 AP_NONE으로 설정됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_RES_NAME

AP_INVALID_RES_TYPE
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DIRECTORY_LU

QUERY_DIRECTORY_LU는 디렉토리 데이터베이스로부터 LU의 리스트를 반환합니다. 정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LU에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **lu_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **lu_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

디렉토리에 있는 DLUS의 서비스를 받는 LU 역시 이 조회로 반환됨에 주의하십시오.

VCB 구조

```
typedef struct query_directory_lu
{
   unsigned short  opcode;                  /* verb operation code         */
   unsigned char   reserv2;                 /* reserved                    */
   unsigned char   format;                  /* format                      */
   unsigned short  primary_rc;              /* primary return code         */
   unsigned long   secondary_rc;            /* secondary return code       */
   unsigned char   *buf_ptr;                /* pointer to buffer           */
   unsigned long   buf_size;                /* buffer size                 */
   unsigned long   total_buf_size;          /* total buffer size required  */
   unsigned short  num_entries;             /* number of entries           */
   unsigned short  total_num_entries;       /* total number of entries     */
   unsigned char   list_options;            /* listing options             */
   unsigned char   reserv3;                 /* reserved                    */
        unsigned char   lu_name[17];        /* network qualified LU name   */
} QUERY_DIRECTORY_LU;

typedef struct directory_lu_summary
{
   unsigned short  overlay_size;            /* size of this entry          */
        unsigned char   lu_name[17];        /* network qualified LU name   */
   unsigned char   description[RD_LEN];     /* resource description        */
} DIRECTORY_LU_SUMMARY;

typedef struct directory_lu_detail
{
   unsigned short  overlay_size;            /* size of this entry          */
        unsigned char   lu_name[17];        /* network qualified LU name   */
   unsigned char   description[RD_LEN];     /* resource description        */
        unsigned char   server_name[17];    /* network qualified           */
                                            /* server name                 */
        unsigned char   lu_owner_name[17];  /* network qualified           */
                                            /* LU owner name               */
        unsigned char   location;           /* Resource location           */
        unsigned char   entry_type;         /* Type of the directory entry */
        unsigned char   wild_card;          /* type of wildcard entry      */
        unsigned char   apparent_lu_owner_name[17];
                                            /* apparent LU owner name      */
        unsigned char   reserva[3];         /* reserved                    */
} DIRECTORY_LU_DETAIL;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DIRECTORY_LU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **lu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name

네트워크 정식 LU명. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
반환된 디렉토리 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

directory_lu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

directory_lu_summary.lu_name
네트워크 정식 LU명. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_lu_summary.description
자원 설명(DEFINE_LOCAL_LU 또는 DEFINE_ADJACENT_NODEDEFINE_DEFAULT_PU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

directory_lu_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

directory_lu_detail.lu_name
네트워크 정식 LU명. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_lu_detail.description
자원 설명(DEFINE_LOCAL_LU 또는 DEFINE_ADJACENT_NODEDEFINE_DEFAULT_PU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

directory_lu_detail.server_name

LU를 서비스하는 노드의 네트워크 정식 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_lu_detail.lu_owner_name

LU를 소유하는 노드의 네트워크 정식 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_lu_detail.location

다음 값 중 하나일 수 있는 자원의 위치를 지정합니다.

AP_LOCAL
자원이 국지 노드에 있습니다.

AP_DOMAIN
자원이 접속된 끝 노드에 속합니다.

AP_CROSS_DOMAIN
자원이 국지 노드의 도메인 내에 없습니다.

directory_lu_detail.entry_type

디렉토리 입력항목의 유형을 지정합니다. 이는 다음 값 중 하나일 수 있습니다.

AP_HOME
국지 자원.

AP_CACHE
캐쉬(cache)된 입력항목.

AP_REGISTER
등록된 자원(NN 만).

directory_lu_detail.wild_card

LU가 대응할 총칭 문자의 유형을 지정합니다.

AP_OTHER
LU 입력항목의 알 수 없는 유형.

AP_EXPLICIT
전체 **lu_name**이 이 LU의 위치를 찾는 데 사용될 것입니다.

AP_PARTIAL_WILDCARD
lu_name의 공백 없는 부분만이 LU의 위치를 찾는 데 사용될 것입니다.

AP_FULL_WILDCARD
모든 **lu_names**이 이 LU로 향하게 될 것입니다.

directory_lu_detail.apprent_lu_owner_name

NN 및 BrNN 만: 전체 정식 외견상의 LU 소유자 CP명. 이 이름의

길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

외견상의 LU 소유자가 실제 LU 소유자이면, 이 필드가 2진 0으로 설정됩니다.

외견상의 LU 소유자가 실제 소유자가 아닌 경우에는, 이 필드가 외견상의 LU 소유자의 이름으로 설정됩니다.

자원을 BrNN의 도메인에 있는 EN이 소유하는 경우에는, 실제 LU 소유자가 BrNN의 NNS의 디렉토리에서 외견상의 LU 소유자가 아닙니다. 이 경우 실제 LU 소유자는 EN이지만, 외견상의 소유자는 BrNN입니다.

기타 노드 유형: 이 필드가 2진 0으로 설정됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DIRECTORY_STATS

이 명령은 통신 서버에만 적용됩니다.

QUERY_DIRECTORY_STATS는 디렉토리 데이터베이스 통계를 반환합니다 (캐쉬 정보를 참조하는 통계는 끝 노드의 경우 예약됩니다.) 네트워크 위치 트래픽의 레벨을 측정하는 데 이 명령을 사용할 수 있습니다. 네트워크 노드의 경우에는 이 정보를 노드 초기화시 구성할 수 있는 디렉토리 캐쉬 (cache)의 크기를 조정하는 데 사용할 수 있습니다.

VCB 구조

```
typedef struct query_directory_stats
{
   unsigned short  opcode;                 /* verb operation code           */
   unsigned char   reserv2;                /* reserved                      */
   unsigned char   format;                 /* format                        */
   unsigned short  primary_rc;             /* primary return code           */
   unsigned long   secondary_rc;           /* secondary return code         */
        unsigned long   max_caches;        /* max number of cache entries   */
        unsigned long   cur_caches;        /* cache entry count             */
        unsigned long   cur_home_entries;  /* home entry count              */
        unsigned long   cur_reg_entries;   /* registered entry count        */
        unsigned long   cur_directory_entries;
                                           /* current number of dir entries */
        unsigned long   cache_hits;        /* count of cache finds          */
        unsigned long   cache_misses;      /* count of resources found by   */
                                           /* broadcast search (not cache)  */
        unsigned long   in_locates;        /* locates in                    */
        unsigned long   in_bcast_locates;  /* broadcast locates in          */
        unsigned long   out_locates;       /* locates out                   */
        unsigned long   out_bcast_locates; /* broadcast locates out         */
        unsigned long   not_found_locates; /* unsuccessful locates          */
        unsigned long   not_found_bcast_locates;
                                           /* unsuccessful broadcast        */
                                           /* locates                       */
        unsigned long   locates_outstanding;
                                           /* total outstanding locates     */
   unsigned char   reserva[20];            /* reserved                      */
} QUERY_DIRECTORY_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DIRECTORY_STATS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

QUERY_DIRECTORY_STATS

primary_rc
AP_OK

max_caches
예약됨.

cur_caches
예약됨.

cur_home_entries
홈 입력항목의 현재 갯수.

cur_reg_entries
등록된 입력항목의 현재 갯수.

cur_directory_entries
현재 디렉토리에 있는 총 입력항목 갯수.

cache_hits
예약됨.

cache_misses
예약됨.

in_locates
수신된 지정 위치의 갯수.

in_bcast_locates
수신된 브로드캐스트 위치의 갯수.

out_locates
송신된 지정 위치의 갯수.

out_bcast_locates
송신된 브로드캐스트 위치의 갯수.

not_found_locates
“not found.”로 반환된 지정 위치의 갯수.

not_found_bcast_locates
“not found.”로 반환된 브로드캐스트 위치의 갯수.

locates_outstanding
지정된 것과 브로드캐스트 모두 미결 위치의 현재 갯수.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLC

QUERY_DLC는 노드에서 정의된 DLC들에 대한 정보의 리스트를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_DLC에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 DLC에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **dlc_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **dlc_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 정의된 정렬하기에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_dlc
{
   unsigned short  opcode;                 /* verb operation code         */
        unsigned char   attributes;        /* ver attributes              */
   unsigned char   format;                 /* format                      */
   unsigned short  primary_rc;             /* primary return code         */
   unsigned long   secondary_rc;           /* secondary return code       */
   unsigned char   *buf_ptr;               /* pointer to buffer           */
   unsigned long   buf_size;               /* buffer size                 */
   unsigned long   total_buf_size;         /* total buffer size required  */
   unsigned short  num_entries;            /* number of entries           */
   unsigned short  total_num_entries;      /* total number of entries     */
   unsigned char   list_options;           /* listing options             */
   unsigned char   reserv3;                /* reserved                    */
   unsigned char   dlc_name[8];            /* name of DLC                 */
} QUERY_DLC;

typedef struct dlc_summary
{
   unsigned short  overlay_size;           /* size of this entry          */
   unsigned char   dlc_name[8];            /* name of DLC                 */
   unsigned char   description[RD_LEN];
                                           /* resource description        */
        unsigned char   state;             /* State of the DLC            */
   unsigned char   dlc_type;               /* DLC type                    */
} DLC_SUMMARY;

typedef struct dlc_detail
{
   unsigned short  overlay_size;           /* size of this entry          */
   unsigned char   dlc_name[8];            /* name of DLC                 */
        unsigned char   reserv2[2];        /* reserved                    */
        DLC_DET_DATA     det_data;         /* Determined data             */
        DLC_DEF_DATA     def_data;         /* Defined data                */
} DLC_DETAIL;
```

```
typedef struct dlc_det_data
{
        unsigned char   state;             /* State of the DLC           */
   unsigned char   reserv3[3];             /* reserved                   */
   unsigned char   reserva[20];            /* reserved                   */
} DLC_DET_DATA;

typedef struct dlc_def_data
{
        DESCRIPTION     description;       /* resource description       */
   unsigned char   dlc_type;               /* DLC type                   */
        unsigned char   neg_ls_supp;       /* negotiable LS support      */
        unsigned char   port_types;        /* allowable port types       */
        unsigned char   retry_flags;       /* conditions for automatic   */
                                           /* retries                    */
        unsigned short  max_activaion_attempts;
                                           /* how many automatic retries? */
   unsigned short  activation_delay_timer;
                                           /* delay between automatic    */
                                           /* retries                    */
        unsigned char   reserv3[6];        /* reserved                   */
        unsigned short  dlc_spec_data_len; /* Length of DLC specific data */
} DLC_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DLC

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **dlc_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환 & 된 색인 값에 의해v$ 돌려줄 다음에.
스트에- 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 v 인 값에 의해

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를] 환합니다.

primary_rc
AP_OK

buf_size
v 퍼에] 환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를] 환하는 데 필요한 v 퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**8 다 높을 수도 있습니다.

num_entries
실제N 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

dlc_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlc_summary.dlc_name
DLC 이름. 이는 국지로 표시가능한 . 자 세트로 된 16 Y 이트의 문자열입니다. 8 바이트 모두 유효합니다.

dlc_summary.description
자원 설명(DEFINE_DLC에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

dlc_summary.state
DLC의 상태. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

dlc_summary.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

dlc_detail.overlay_size
이 입력항목에 있는 바이트의 수(dlc_spec_data를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlc_detail.dlc_name
DLC 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

dlc_detail.det_data.state
DLC의 상태. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_ACTIVE
AP_NOT_ACTIVE
AP_PENDING_INACTIVE

dlc_detail.def_data.description
자원 설명(DEFINE_DLC에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

dlc_detail.def_data.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC

AP_SDLC
AP_TWINAX
AP_X25

dlc_detail.def_data.neg_ls_supp
DLC가 조정가능 링크 스테이션을 지원하는지의 여부를 지정합니다 (AP_YES 또는 AP_NO).

dlc_detail.def_data.port_types
제공된 **dlc_type**에 대해 허용가능한 포트 유형을 지정합니다. 값은 다음 값 중 하나 또는 그 이상에 해당합니다.

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

dlc__detail.def_data.retry_flags
이 필드는 플래그 AP_INHERIT_RETRY가 **def_data.retry_flags**의 DEFINE_LS와 DEFINE_PORT 둘다에서 설정되는 경우 이 DLC에서 정의된 링크 스테이션이 어떤 조건하에서 자동 재시도를 하게 되는지 지정합니다. 이는 비트 필드로 다음 값 중 어느 것이든 취할 수 있습니다.

AP_RETRY_ON_START
활성화 시도시 원격 노드로부터 응답이 수신되지 않으면 링크 활성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 활동중이 아니면, 프로그램이 이를 활성화시키려 할 것입니다.

AP_RETRY_ON_FAILURE
활동중 또는 대기 활동중인 동안 링크가 실패하면 링크 활성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 실패했으면, 프로그램이 이를 활성화시키려 할 것입니다.

AP_RETRY_ON_DISCONNECT
링크가 원격 노드에 의해 정상적으로 중단되면 링크 활성화가 재시도될 것입니다.

AP_DELAY_APPLICATION_RETRIES
응용프로그램에 의해 초기화되는(START_LS 또는 요구가 있을때 링크 활성화를 사용하여) 링크 활성화 재시도는 **activation_delay_timer**를 사용하여 속도 조정될 것입니다.

AP_INHERIT_RETRY
이 플래그는 아무런 영향도 미치지 않습니다.

dlc_detail.def_data.max_activation_attempts
최소한 하나의 플래그가 **def_data.retry_flags**에 있는 DEFINE_LS에서 설정되고, DEFINE_LS상의 **def_data.max_activation_attempts**가 AP_USE_DEFAULTS로 설정되며 그리고 DEFINE_PORT상의

def_data.max_activation_attempts가 AP_USE_DEFAULTS로 설정되지 않는 한, 이 필드는 아무런 영향도 미치지 않습니다.

이 필드는 원격 노드가 응답하고 있지 않거나 하부 포트가 활동중이 아닐 때, 프로그램이 허용하는 재시도 회수를 지정합니다. 여기에는 자동 재시도와 응용프로그램 지향 활성화 시도가 모두 포함됩니다.

이 한도에 도달하게 되면, 더 이상 자동 재시도가 이루어지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는 OPEN_LU_SSCP_SEC_RQ는 활성화가 실패하면 더 이상 재시도하지 않는 단일 활성화 시도의 결과를 낳습니다.

0은 '제한 없음'을 의미합니다. 값 AP_USE_DEFAULTS는 '제한 없음'을 의미합니다.

dlc_detail.def_data.activation_delay_timer

최소한 하나의 플래그가 **def_data.retry_flags**에 있는 DEFINE_LS에서 설정되고, DEFINE_LS상의 **def_data.max_activation_attempts**가 AP_USE_DEFAULTS로 설정되며 그리고 DEFINE_PORT상의 **def_data.max_activation_attempts**가 AP_USE_DEFAULTS로 설정되지 않는 한, 이 필드는 아무런 영향도 미치지 않습니다.

이 필드는 **def_data.retry_flags**에서 AP_DELAY_APPLICATION_RETRIES 비트가 설정되는 경우 프로그램가 자동 재시도 사이, 그리고 응용프로그램 지향 활성화 시도 사이에서 대기하는 초 수를 지정합니다.

0 또는 AP_USE_DEFAULTS의 값은 30초의 생략시 타이머 지속시간을 사용하게 합니다.

dlc_detail.def_data.dlc_spec_data_len

DLC의 유형에 특유한 데이터의 채워지지 않은 바이트 단위의 길이. 데이터는 DLC_DETAIL 구조체에 연결될 것입니다. 이 데이터는 4 바이트의 경계에서 끝으로 채워집니다. 이 필드는 항상 0으로 설정되어야 합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_DLC_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUR_DEFAULTS

QUERY_DLUR_DEFAULTS는 사용자로 하여금 DEFINE_DLUR_DEFAULTS 명령을 사용하여 정의된 기본설정 값을 조회할 수 있게 합니다.

VCB 구조

```
typedef struct query_dlur_defaults
{
   unsigned short  opcode;                 /* verb operation code       */
   unsigned char   reserv2;                /* reserved                  */
   unsigned char   format;                 /* format                    */
   unsigned short  primary_rc;             /* primary return code       */
   unsigned long   secondary_rc;           /* secondary return code     */
        DESCRIPTION     description;       /* resource description      */
        unsigned char   dlus_name[17];     /* DLUS name                 */
        unsigned char   bkup_dlus_name[17];/* Backup DLUS name          */
   unsigned char   reserv3;                /* reserved                  */
        unsigned short  dlus_retry timeout;/* DLUS Retry Timeout        */
        unsigned short  dlus_retry_limit;  /* DLUS Retry Limit          */
        unsigned char   reserv4[16];       /* reserved                  */
} QUERY_DLUR_LU;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DLUR_LU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

description

자원 설명. 이 필드의 길이는 4 바이트의 배수이어야 하며 0이 아니어야 합니다.

dlus_name

생략시로 서비스할 DLUS 노드의 이름. 이는 모두 0으로 설정되거나 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

bkup_dlus_name

백업 생략시로 서비스할 DLUS 노드의 이름. 이는 모두 0으로 설정

되거나 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) LU의 위치. 반환되는 유일한 값은 다음과 같습니다.

dlus_retry_timeout
DLUS에 접속하기 위한 두 번째와 차후의 시도간의 간격으로 초 단위. 초기 시도와 첫번째 재시도간의 간격은 항상 1초입니다.

dlus_retry_limit
최초 DLUS 접속 실패 후 최대 재시도 횟수. X'FFFF'가 지정되면, 프로그램이 무기한 재시도합니다.

관련 START_NODE 매개변수가 설정되지 않았으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

시스템에 DLUR 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

STOP_NODE 명령이 발행되었으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUR_LU

QUERY_DLUR_LU는 DLUR에 의해 지원되는 LU에 대한 정보의 리스트를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LU에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **lu_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **lu_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

반환되는 LU 리스트는 **pu_name**에 의해 필터링되거나 LU가 국지인지 다운스트림인지에 따라 필터링되거나, 둘다에 의해 필터링될 수 있습니다. PU에 의한 필터링하려면, **pu_name** 필드를 설정해야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.) 위치에 의한 필터링하려면, **filter** 필드를 AP_INTERNAL이나 AP_DOWNSTREAM으로 설정해야 합니다.(그렇지 않고, 필터링이 필요하지 않으면, 이 필드를 AP_NONE으로 설정해야 합니다.)

VCB 구조

```
typedef struct query_dlur_lu
{
   unsigned short  opcode;             /* verb operation code        */
   unsigned char   reserv2;            /* reserved                   */
   unsigned char   format;             /* format                     */
   unsigned short  primary_rc;         /* primary return code        */
   unsigned long   secondary_rc;       /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer          */
   unsigned long   buf_size;           /* buffer size                */
   unsigned long   total_buf_size;     /* total buffer size required */
   unsigned short  num_entries;        /* number of entries          */
   unsigned short  total_num_entries;  /* total number of entries    */
   unsigned char   list_options;       /* listing options            */
   unsigned char   reserv3;            /* reserved                   */
        unsigned char   lu_name[8];    /* name of LU                 */
        unsigned char   pu_name[8];    /* name of PU to filter on    */
        unsigned char   filter;        /* reserved                   */
} QUERY_DLUR_LU;

typedef struct dlur_lu_summary
{
   unsigned short  overlay_size;       /* size of this entry         */
        unsigned char   lu_name[8];    /* name of LU                 */
} DLUR_LU_SUMMARY;
```

```
typedef struct dlur_lu_detail
{
   unsigned short  overlay_size;       /* size of this entry          */
        unsigned char   lu_name[8];    /* name of LU                  */
        unsigned char   pu_name[8];    /* name of owning PU           */
        unsigned char   dlus_name[17]; /* DLUS name if SSCP-LU        */
                                       /* session active               */
        unsigned char   lu_location;   /* downstream or local LU      */
        unsigned char   nau_address;   /* NAU address of LU           */
        unsigned char   plu_name[17];  /* PLU name if PLU-SLU session*/
                                       /* active                       */
        unsigned char   reserv1[27];   /* reserved                    */
        unsigned char   rscv_len;      /* length of appended RSCV     */
} DLUR_LU_DETAIL;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DLUR_LU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **lu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
조회되고 있는 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

pu_name
PU명 필터. 이는 모두 0으로 설정되거나 오른쪽은 EBCDIC 공백으로 채워지면서 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 설정되어야 합니다. 이 필드가 설정되면 지정된 PU와 연관된 LU만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

filter 위치 필터. 반환되는 LU가 위치(AP_INTERNAL 또는 AP_DOWNSTREAM)에 의해 필터링되는지 여부를 지정합니다. 필터가 필요없으면, 이 필드를 AP_NONE으로 설정해야 합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

dlur_lu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlur_lu_summary.lu_name
LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

dlur_lu_detail.overlay_size
이 입력항목에 있는 바이트의 수(RSCV를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlur_lu_detail.lu_name
LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

dlur_lu_detail.pu_name
LU와 연관된 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

dlur_lu_detail.dlus_name
SSCP-LU 세션이 활동중인 경우 DLUS 노드의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) SSCP-LU 세션이 활동중이지 않으면, 이 필드가 모두 0으로 설정될 것입니다.

dlur_lu_detail.lu_location
LU의 위치. 반환되는 유일한 값은 다음과 같습니다.

AP_INTERNAL
AP_DOWNSTREAM

dlur_lu_detail.nau_address
LU의 네트워크 지정 장치(NAU) 주소. 1—255 범주에 있습니다.

dlur_lu_detail.plu_name
LU에 활동중 PLU-SLU 세션이 있는 경우 PLU의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) PLU-SLU 세션이 활동중이지 않으면, 이 필드가 모두 0으로 설정될 것입니다.

dlur_lu_detail.rscv_len
이 값은 항상 0일 것입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_FILTER_OPTION
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUR_PU

QUERY_DLUR_PU는 DLUR에 의해 지원되는 PU들에 대한 정보의 리스트를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 PU에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **pu_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **pu_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

반환되는 PU들의 리스트는 **dlus_name**에 의해 필터링되거나 PU가 국지인지 다운스트림인지에 따라 필터링되거나, 둘다에 의해 필터링될 수 있습니다. DLUS에 의한 필터링하려면, **dlus_name** 필드를 설정해야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.) PU 위치에 의한 필터링하려면, **filter** 필드를 AP_INTERNAL이나 AP_DOWNSTREAM으로 설정해야 합니다.(그렇지 않고, 필터링이 필요하지 않으면, 이 필드를 AP_NONE으로 설정해야 합니다.)

VCB 구조

```
typedef struct query_dlur_pu
{
   unsigned short  opcode;                  /* verb operation code           */
   unsigned char   reserv2;                 /* reserved                      */
   unsigned char   format;                  /* format                        */
   unsigned short  primary_rc;              /* primary return code           */
   unsigned long   secondary_rc;            /* secondary return code         */
   unsigned char   *buf_ptr;                /* pointer to buffer             */
   unsigned long   buf_size;                /* buffer size                   */
   unsigned long   total_buf_size;          /* total buffer size required    */
   unsigned short  num_entries;             /* number of entries             */
   unsigned short  total_num_entries;       /* total number of entries       */
   unsigned char   list_options;            /* listing options               */
   unsigned char   reserv3;                 /* reserved                      */
        unsigned char   pu_name[8];         /* name of PU                    */
        unsigned char   dlus_name[17];      /* fully qualified DLUS name     */
        unsigned char   filter;             /* local/downstream filter       */
} QUERY_DLUR_PU;

typedef struct dlur_pu_summary
{
   unsigned short  overlay_size;            /* size of this entry            */
        unsigned char   pu_name[8];         /* name of PU                    */
   unsigned char   description[RD_LEN];
                                            /* resource description          */
} DLUR_PU_SUMMARY;
```

```
typedef struct dlur_pu_detail
{
   unsigned short  overlay_size;             /* size of this entry            */
        unsigned char   pu_name[8];          /* name of PU                    */
   unsigned char   description[RD_LEN];
                                             /* resource description          */
        unsigned char   defined_dlus_name[17];
                                             /* defined DLUS name             */
        unsigned char   bkup_dlus_name[17];  /* backup DLUS name              */
        unsigned char   pu_id[4];            /* PU identifier                 */
        unsigned char   pu_location;         /* downstream or local PU        */
        unsigned char   active_dlus_name[17];
                                             /* active DLUS name              */
        unsigned char   ans_support;         /* Auto-Network shutdown support */
        unsigned char   pu_status;           /* status of the PU              */
        unsigned char   dlus_session_status; /* status of the DLUS pipe       */
   unsigned char   reserv3;                  /* reserved                      */
        FQPCID  fqpcid;                      /* FQPCID used on pipe           */
        unsigned short  dlus_retry_timeout;  /* DLUS retry timeout            */
        unsigned short  dlus_retry_limit;    /* DLUS retry limit              */
} DLUR_PU_DETAIL;

typedef struct fqpcid
{
        unsigned char   pcid[8];             /* proc correlator identifier    */
        unsigned char   fqcp_name[17];       /* originator's network          */
                                             /* qualified CP name             */
   unsigned char   reserve3[3];              /* reserved                      */
} FQPCID;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DLUR_PU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **pu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

pu_name
조회되고 있는 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dlus_name
DLUS 필터. 이는 모두 0으로 설정되거나 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정되어야 합니다. 이 필드가 설정되면 지정된 DLUS 노드로의 SSCP-PU 세션과 연관된 PU들만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

filter 이 필드는 AP_NONE으로 설정되어야 합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

dlur_pu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlur_pu_summary.pu_name
PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

dlur_pu_summary.description
자원 설명(DEFINE_INTERNAL_PU에서 지정된 것과 같이). 이는 국로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

dlur_pu_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlur_pu_detail.pu_name
PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

dlur_pu_detail.description
자원 설명(DEFINE_INTERNAL_PU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

dlur_pu_detail.defined_dlus_name
DEFINE_INTERNAL_PU 명령 또는 DEFINE_LS 명령(**dspu_services**가 AP_DLUR로 설정된 상태로)에 의해 정의된 DLUS 노드의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

dlur_pu_detail.bkup_dlus_name
DEFINE_INTERNAL_PU 명령 또는 DEFINE_LS 명령(**dspu_services**가 AP_DLUR로 설정된 상태로)에 의해 정의된 백업 DLUS 노드의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

dlur_pu_detail.pu_id
DEFINE_INTERNAL_PU 명령에서 정의되었거나 다운스트림 PU로부터 XID에서 확보한 PU 식별자(ID). 이는 4 바이트의 16진 문자열입니다. 0—11의 비트들은 블럭 번호로 설정되고 12—31의 비트들은 PU를 고유하게 식별하는 ID 번호로 설정됩니다.

dlur_pu_detail.pu_location
PU의 위치. 반환되는 유일한 값은 다음과 같습니다.

AP_INTERNAL
AP_DOWNSTREAM

dlur_pu_detail.active_dlus_name

PU가 현재 사용하고 있는 DLUS 노드의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열입니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) SSCP-PU 세션이 활동중이지 않으면, 이 필드가 모두 0으로 설정될 것입니다.

dlur_pu_detail.ans_support

자동 네트워크 종료 지원. SSCP-LU 세션이 활동중이 아닌 경우에는 이 필드가 예약됩니다. SSCP-PU 활성화시 지원 설정이 DLUS로부터 DLUR로 보내집니다. 부속영역 노드가 PU를 제어하고 있는 SSCP에 대한 자동 네트워크 종료 절차를 초기화하는 경우 링크 레벨 접속이 계속되어야 하는지의 여부를 지정합니다. 이는 다음 값 중 하나일 수 있습니다.

AP_CONT
AP_STOP

dlur_pu_detail.pu_status

PU의 상태(DLUR에 의해 보여지는 대로). 이는 다음 값 중 하나로 설정될 수 있습니다.

AP_RESET
PU가 재설정 상태에 있습니다.

AP_PEND_ACTPU
PU가 호스트로부터 ACTPU를 기다리고 있습니다.

AP_PEND_ACTPU_RSP
PU로 ACTPU를 이송했으므로, DLUR은 이제 PU가 응답하기를 기다리고 있습니다.

AP_ACTIVE
PU가 활동중입니다.

AP_PEND_DACTPU_RSP
PU로 DACTPU를 이송했으므로, DLUR이 PU가 응답하기를 기다리고 있습니다.

AP_PEND_INOP
DLUR이 PU를 비활성화시키기 전에 모든 필수 이벤트들이 완료되기를 기다리고 있습니다.

dlur_pu_detail.dlus_session_status

현재 PU가 사용하고 있는 DLUS 파이프의 상태. 이는 다음 값 중 하나일 수 있습니다.

AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE
AP_INACTIVE

dlur_pu_detail.fqpcid.pcid
파이프에서 사용된 절차 상관자 ID. 이는 8 바이트의 16진 문자열입니다. SSCP-PU 세션이 활동중이지 않으면, 이 필드가 0으로 설정될 것입니다.

dlur_pu_detail.fqpcid.fqcp_name
파이프에서 사용된 전체 정식 제어점 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) SSCP-PU 세션이 활동중이지 않으면, 이 필드가 0으로 설정될 것입니다.

dlur_pu_detail.dlus_retry_timeout
dlus_name과 **bkup_dlus_name** 필드에서 지정된 DLUS에 접속하기 위한 두 번째와 차후의 시도간의 간격으로 초 단위. 초기 시도와 첫번째 재시도간의 간격은 항상 1초입니다. 0이 지정되면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다.

def_data.dlus_retry_limit
dlus_name과 **bkup_dlus_name** 필드에서 지정된 최초 DLUS 접속 실패 후 최대 재시도 횟수. 0이 지정되면 DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. X'FFFF'가 지정되면 프로그램이 무기한 재시도합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_FILTER_OPTION
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DLUS

QUERY_DLUS는 DLUR로 알려지는 DLUS들에 대한 정보의 리스트를 반환합니다.

정보는 리스트로 반환됩니다. 특정 DLUS 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **dlus_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **dlus_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

이 명령이 파이프 통계를 반환함에 주의하십시오.

VCB 구조

```
typedef struct query_dlus
{
   unsigned short  opcode;             /* verb operation code           */
   unsigned char   reserv2;            /* reserved                      */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* primary return code           */
   unsigned long   secondary_rc;       /* secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   dlus_name[17];      /* fully qualified DLUS name     */
} QUERY_DLUS;

typedef struct dlus_data
{
   unsigned short  overlay_size;           /* size of this entry          */
        unsigned char   dlus_name[17];     /* fully qualified DLUS name   */
        unsigned char   is_default;        /* is the DLUS the default     */
        unsigned char   is_backup_default; /* is DLUS the backup default  */
        unsigned char   pipe_state;        /* state of CPSVRMGR pipe      */
        unsigned short  num_active_pus;    /* num of active PUs using pipe */
        PIPE_STATS      pipe_stats;        /* pipe statistics             */
} DLUS_DATA;

typedef struct pipe_stats
{
        unsigned long   reqactpu_sent;     /* REQACTPUs sent to DLUS       */
        unsigned long   reqactpu_rsp_received;
                                           /* RSP(REQACTPU)s received      */
                                           /* from DLUS                    */
        unsigned long   actpu_received;    /* ACTPUs received from DLUS    */
        unsigned long   actpu_rsp_sent;    /* RSP(ACTPU)s sent to DLUS     */
        unsigned long   reqdactpu_sent;    /* REQDACTPUs sent to DLUS      */
```

```
        unsigned long   reqdactpu_rsp_received;
                                        /* RSP(REQDACTPU)s received      */
                                        /* from DLUS                     */
        unsigned long   dactpu_received;   /* DACTPUs received from DLUS    */
        unsigned long   dactpu_rsp_sent;   /* RSP(DACTPU)s sent to DLUS     */
        unsigned long   actlu_received;    /* ACTLUs received from DLUS     */
        unsigned long   actlu_rsp_sent;    /* RSP(ACTLU)s sent to DLUS      */
        unsigned long   dactlu_received;   /* DACTLUs received from DLUS    */
        unsigned long   dactlu_rsp_sent;   /* RSP(DACTLU)s sent to DLUS     */
        unsigned long   sscp_pu_mus_rcvd;  /* MUs for SSCP-PU               */
                                        /* sessions received             */
        unsigned long   sscp_pu_mus_sent;  /* MUs for SSCP-PU sessions sent */
        unsigned long   sscp_lu_mus_rcvd;  /* MUs for SSCP-LU sessions      */
                                        /* received                      */
        unsigned long   sscp_lu_mus_sent;  /* MUs for SSCP-LU sessions sent */
} PIPE_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DLUS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **dlus_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

dlus_name
조회되고 있는 DLUS의 이름. 이는 모두 0으로 설정되거나 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정되어야 합니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

dlus_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dlus_data.dlus_name
DLUS의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

dlus_data.is_default
DLUS 노드가 DEFINE_DLUR_DEFAULTS 명령에 의해 생략시로 지명되었는지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

dlus_data.is_backup_default
DLUS 노드가 DEFINE_DLUR_DEFAULTS 명령에 의해 백업 생략시로 지명되었는지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

dlus_data.pipe_state
DLUS에 대한 파이프의 상태. 다음 값 중 하나를 가질 수 있습니다.

AP_ACTIVE
AP_PENDING_ACTIVE
AP_INACTIVE
AP_PENDING_INACTIVE

dlus_data.num_active_pus
현재 DLUS에 대한 파이프를 사용하고 있는 PU들의 수.

dlus_data.pipe_stats.reqactpu_sent
파이프를 통해 DLUS로 보내진 REQACTPU들의 수.

dlus_data.pipe_stats.reqactpu_rsp_received
파이프를 통해 DLUS로부터 수신된 RSP(REQACTPU)들의 수.

dlus_data.pipe_stats.actpu_received
파이프를 통해 DLUS로부터 수신된 ACTPU들의 수.

dlus_data.pipe_stats.actpu_rsp_sent
파이프를 통해 DLUS로 보내진 RSP(ACTPU)들의 수.

dlus_data.pipe_stats.reqdactpu_sent
파이프를 통해 DLUS로 보내진 REQDACTPU들의 수.

dlus_data.pipe_stats.reqdactpu_rsp_received
파이프를 통해 DLUS로부터 수신된 RSP(REQDACTPU)들의 수.

dlus_data.pipe_stats.dactpu_received
파이프를 통해 DLUS로부터 수신된 DACTPU들의 수.

dlus_data.pipe_stats.dactpu_rsp_sent
파이프를 통해 DLUS로 보내진 RSP(DACTPU)들의 수.

dlus_data.pipe_stats.actlu_received
파이프를 통해 DLUS로부터 수신된 ACTLU의 수.

dlus_data.pipe_stats.actlu_rsp_sent
파이프를 통해 DLUS로 보내진 RSP(ACTLU)들의 수.

dlus_data.pipe_stats.dactlu_received
파이프를 통해 DLUS로부터 수신된 DACTLU의 수.

dlus_data.pipe_stats.dactlu_rsp_sent
파이프를 통해 DLUS로 보내진 RSP(DACTLU)들의 수.

dlus_data.pipe_stats.sscp_pu_mus_rcvd
파이프를 통해 DLUS로부터 수신된 SSCP-PU MU들의 수.

dlus_data.pipe_stats.sscp_pu_mus_sent
파이프를 통해 DLUS로 보내진 SSCP-PU MU들의 수.

dlus_data.pipe_stats.sscp_lu_mus_rcvd
파이프를 통해 DLUS로부터 수신된 SSCP-LU MU들의 수.

dlus_data.pipe_stats.sscp_lu_mus_sent
파이프를 통해 DLUS로 보내진 SSCP-LU MU들의 수.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DLUS_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DOWNSTREAM_LU

이 명령은 통신 서버에만 적용됩니다.

QUERY_DOWNSTREAM_LU는 DLUR이나 PU 또는 이들 둘다의 서비스를 받는 다운스트림 LU에 대한 정보를 반환합니다. 이 정보는 결정된 데이터(실행동안 동적으로 모아진 데이터)와 정의된 데이터로 구성됩니다.(정의된 데이터는 DEFINE_DOWNSTREAM_LU 명령에서 응용프로그램에 의해 제공된 데이터입니다. DLUR의 지원을 받는 LU의 경우에는, 다운스트림 LU가 활성화될 때 암시적으로 정의된 데이터가 대신하게 됩니다.)

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 local에 관한 정보를 확보하거나 여러 개의 청크(chunk)로 리스트 정보를 확보하려면, **dslu_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다.

반환된 LU는 국지 노드가 제공하는 서비스 유형이나 LU의 연관 다운스트림 PU에 의해, 또는 둘다에 의해 필터링됩니다. 서비스 유형에 의한 필터링을 원하면, **dspu_services** 필드를 AP_PU_CONCENTRATION이나 AP_DLUR로 설정해야 합니다.(그렇지 않고, 필터링이 필요하지 않으면, 이 필드를 AP_NONE으로 설정해야 합니다.) PU에 의한 필터링을 원하면, **dspu_name** 필드를 설정해야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.)

VCB 구조

```
typedef struct query_downstream_lu
{
   unsigned short  opcode;             /* verb operation code           */
   unsigned char   attributes;         /* Verb attributes               */
   unsigned char   reserv2;            /* reserved                      */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* primary return code           */
   unsigned long   secondary_rc;       /* secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   dslu_name[8];  /* Downstream LU name            */
        unsigned char   dspu_name[8];  /* Downstream PU name filter     */
        unsigned char   dspu_services; /* filter on DSPU services type  */
} QUERY_DOWNSTREAM_LU;

typedef struct downstream_lu_summary
{
   unsigned short  overlay_size;          /* size of this entry          */
        unsigned char   dslu_name[8];     /* LU name                     */
        unsigned char   dspu_name[8];     /* PU name                     */
   unsigned char   description[RD_LEN];
                                          /* resource description        */
        unsigned char   dspu_services;    /* type of service provided to */
                                          /* downstream node             */
```

```
        unsigned char   nau_address;      /* NAU address                   */
        unsigned char   lu_sscp_sess_active;
                                          /* Is LU-SSCP session active     */
        unsigned char   plu_sess_active;  /* Is PLU-SLU session active     */
} DOWNSTREAM_LU_SUMMARY

typedef struct downstream_lu_detail
{
   unsigned short  overlay_size;           /* size of this entry           */
        unsigned char   dslu_name[8];      /* LU name                      */
   unsigned char   reserv1[2];             /* reserved                     */
        DOWNSTREAM_LU_DET_DATA det_data;   /* Determined data              */
        DOWNSTREAM_LU_DEF_DATA def_data;   /* Defined data                 */
} DOWNSTREAM_LU_DETAIL;

typedef struct downstream_lu_det_data
{
        unsigned char   lu_sscp_sess_active;
                                          /* Is LU-SSCP session active     */
        unsigned char   plu_sess_active;  /* Is PLU-SLU session active     */
        unsigned char   dspu_services;    /* type of services provided to  */
                                          /* downstream node               */
   unsigned char   reserv1;               /* reserved                      */
        SESSION_STATS   lu_sscp_stats;    /* LU-SSCP session statistics    */
        SESSION_STATS   ds_plu_stats;     /* downstream PLU-SLU session    */
                                          /* statistics                    */
        SESSION_STATS   us_plu_stats;     /* upstream PLU_SLU sess stats   */
        unsigned char   host_lu_name[8];  /* Determined host LU name       */
        unsigned char   host_pu_name[8];  /* Determined host PU name       */
        unsigned char   reserva[4];       /* reserved                      */
} DOWNSTREAM_LU_DET_DATA;

typedef struct downstream_lu_def_data
{
   unsigned char   description[RD_LEN];
                                          /* resource description          */
        unsigned char   nau_address;      /* NAU address                   */
        unsigned char   dspu_name[8];     /* Downstream PU name            */
        unsigned char   host_pu_name;     /* host LU or pool name          */
        unsigned char   allow_timeout;    /* Allow timeout of host LU?     */
        unsigned char   delayed_logon;    /* Allow delayed logo to host LU */
        unsigned char   reserv2[6];       /* reserved                      */
} DOWNSTREAM_LU_DEF_DATA

typedef struct session_stats
{
   unsigned short  rcv_ru_size;           /* session receive RU size       */
   unsigned short  send_ru_size;          /* session send RU size          */
   unsigned short  max_send_btu_size;     /* max send BTU size             */
   unsigned short  max_rcv_btu_size;      /* max rcv BTU size              */
        unsigned short  max_send_pac_win; /* max send pacing win size      */
        unsigned short  cur_send_pac_win; /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;  /* max receive pacing win size   */
   unsigned short  cur_rcv_pac_win;       /* current receive pacing        */
                                          /* window size                   */
   unsigned long   send_data_frames;      /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                          /* num of FMD data frames sent   */
   unsigned long   send_data_bytes;       /* number of data bytes sent     */
   unsigned long   rcv_data_frames;       /* num data frames received      */
   unsigned long   rcv_fmd_data_frames;
                                          /* num of FMD data frames recvd  */
   unsigned long   rcv_data_bytes;        /* number of data bytes received */
   unsigned char   sidh;                  /* session ID high byte          */
   unsigned char   sidl;                  /* session ID low byte           */
   unsigned char   odai;                  /* ODAI bit set                  */
   unsigned char   ls_name[8];            /* Link station name             */
        unsigned char   pacing type;      /* type of pacing in use         */
} SESSION_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_DOWNSTREAM_LU

attributes
명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **dslu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

dslu_name
조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dspu_name
PU명 필터. 이는 모두 0으로 설정되거나 오른쪽은 EBCDIC 공백으로 채워지면서 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 설정되어야 합니다. 이 필드가 설정되면 지정된 PU와 연관된 LU만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

dspu_services
DSPU 서비스 필터. AP_PU_CONCENTRATION으로 설정되면, PU 집중(concentration)에 의해 서비스되는 다운스트림 LU만이 반환됩니다. AP_DLUR로 설정되면, DLUR의 지원을 받는 LU만 반환됩니다. 그렇지 않고 AP_NONE으로 설정되면 모든 다운스트림 LU에 대한 정보가 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

downstream_lu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

downstream_lu_summary.dslu_name
조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

downstream_lu_summary.dspu_name
이 LU가 사용하고 있는 국지 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

downstream_lu_summary.description
자원 설명(DEFINE_DOWNSTREAM_LU나 DEFINE_DOWNSTREAM_LU_RANGE에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

downstream_lu_summary.dspu_services
국지 노드가 링크를 거쳐 다운스트림 LU에 제공하는 서비스를 지정합니다. 이는 다음 중 하나로 설정됩니다.

AP_PU_CONCENTRATION
다운스트림 LU에 대한 PU 집중을 제공하는 국지 노드.

AP_DLUR
다운스트림 LU에 대한 DLUR 지원을 제공하는 국지 노드.

downstream_lu_summary.nau_address
LU의 네트워크 지정 장치(NAU) 주소로 그 범주는 1—255.

downstream_lu_summary.lu_sscp_sess_active
LU-SSCP 세션이 활동중인지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

downstream_lu_summary.plu_sess_active
PLU-SLU 세션이 활동중인지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

downstream_lu_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

downstream_lu_detail.dslu_name
조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

downstream_lu_detail.det_data.lu_sscp_sess_active
다운스트림 LU로의 LU-SSCP 세션이 활동중인지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

downstream_lu_detail.det_data.plu_sess_active
다운스트림 LU로의 PLU-SLU 세션이 활동중인지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

downstream_lu_detail.det_data.dspu_services
국지 노드가 링크를 거쳐 다운스트림 LU에 제공하는 서비스를 지정합니다. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_PU_CONCENTRATION
다운스트림 LU에 대한 PU 집중을 제공하는 국지 노드.

AP_DLUR
다운스트림 LU에 대한 DLUR 지원을 제공하는 국지 노드.

downstream_lu_detail.det_data.lu_sscp_stats.rcv_ru_size
최대 수신 RU 크기. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.lu_sscp_stats.send_ru_size
최대 송신 RU 크기. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.lu_sscp_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.lu_sscp_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

downstream_lu_detail.det_data.lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

downstream_lu_detail.det_data.lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

downstream_lu_detail.det_data.lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

downstream_lu_detail.det_data.lu_sscp_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

downstream_lu_detail.det_data.lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

downstream_lu_detail.det_data.lu_sscp_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정하며, BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

downstream_lu_detail.det_data.lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

downstream_lu_detail.det_data.ds_plu_stats.rcv_ru_size
최대 수신 RU 크기.

downstream_lu_detail.det_data.ds_plu_stats.send_ru_size
최대 송신 RU 크기.

downstream_lu_detail.det_data.ds_plu_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.ds_plu_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.ds_plu_stats.max_send_pac_win
이 세션상의 최대 송신 페이싱 창 크기.

downstream_lu_detail.det_data.ds_plu_stats.cur_send_pac_win
이 세션상의 현재 송신 페이싱 창 크기.

downstream_lu_detail.det_data.ds_plu_stats.max_rcv_pac_win
이 세션상의 최대 수신 페이싱 창 크기.

downstream_lu_detail.det_data.ds_plu_stats.cur_rcv_pac_win
이 세션상의 현재 수신 페이싱 창 크기.

downstream_lu_detail.det_data.ds_plu_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.ds_plu_stats.sidh
세션 ID 상위(high) 바이트.

downstream_lu_detail.det_data.ds_plu_stats.sidl
세션 ID 하위(low) 바이트.

downstream_lu_detail.det_data.ds_plu_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정하며, BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

downstream_lu_detail.det_data.ds_plu_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

downstream_lu_detail.det_data.plu_stats.pacing_type
다운스트림 PLU-SLU 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE 또는 AP_PACING_FIXED를 취할 수 있습니다.

downstream_lu_detail.det_data.lu_sscp_pacing_type
LU-SSCP 세션상에서 사용중인 수신 페이싱. 이는 값 AP_NONE을 취합니다.

downstream_lu_detail.det_data.us_plu_stats.send_ru_size
최대 송신 RU 크기.

downstream_lu_detail.det_data.us_plu_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.us_plu_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

downstream_lu_detail.det_data.us_plu_stats.max_send_pac_win
이 세션상의 최대 송신 페이싱 창 크기.

downstream_lu_detail.det_data.us_plu_stats.cur_send_pac_win
이 세션상의 현재 송신 페이싱 창 크기.

downstream_lu_detail.det_data.us_plu_stats.max_rcv_pac_win
이 세션상의 최대 수신 페이싱 창 크기.

downstream_lu_detail.det_data.us_plu_stats.cur_rcv_pac_win
이 세션상의 현재 수신 페이싱 창 크기.

downstream_lu_detail.det_data.us_plu_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.us_plu_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.us_plu_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.us_plu_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

downstream_lu_detail.det_data.us_plu_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_lu_detail.det_data.us_plu_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

downstream_lu_detail.det_data.us_plu_stats.sidh
세션 ID 상위(high) 바이트. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.us_plu_stats.sidl
세션 ID 하위(low) 바이트. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.us_plu_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정하며, BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.us_plu_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_lu_detail.det_data.us_plu_stats.pacing_type
업스트림 PLU-SLU 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE 또는 AP_PACING_FIXED를 취할 수 있습니다.

downstream_lu_detail.det_data.host_lu_name
다운스트림 LU가 매핑되거나 PLU-SLU 세션이 마지막으로 활동중일 때 매핑되었던 호스트 LU의 이름. 이는 호스트 LU 풀의 이름일 수도 있기 때문에 **def_data.host_lu_name**과 다를 지도 모릅니다.

downstream_lu_detail.det_data.host_pu_name
다운스트림 PU가 매핑되거나 PLU-SLU 세션이 마지막으로 활동중일 때 매핑되었던 호스트 PU의 이름.

downstream_lu_detail.def_data.description
자원 설명(DEFINE_DOWNSTREAM_LU나 DEFINE_DOWNSTREAM_LU_RANGE에서 지정된 대로).

downstream_lu_detail.def_data.nau_address
LU의 네트워크 지정 장치(NAU) 주소로 그 범주는 1—255.

downstream_lu_detail.def_data.dspu_name
LU와 연관된 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

downstream_lu_detail.def_data.host_lu_name
다운스트림 LU가 매핑되는 호스트 LU나 호스트 LU 풀의 이름. LU

의 경우 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. LU 풀의 경우에는 퍼스널 통신이나 통신 서버가 이 필드에 대한 문자 세트를 지정하지 않습니다. 이 필드는 DLUR의 서비스를 받는 다운스트림 LU를 위해 예약됩니다.

downstream_lu_detail.def_data.allow_timeout
호스트 LU 정의에서 지정된 **timeout** 시간 동안 세션이 활동중이지 않은 상태로 남아 있는 경우, 이 다운스트림 LU에 의해 사용된 호스트 LU를 퍼스널 통신이나 통신 서버가 시간종료시킬 수 있도록 허용하는지를 지정합니다(AP_YES 또는 AP_NO).

downstream_lu_detail.def_data.delayed_logon
다운스트림 LU로부터 첫번째 데이터를 수신할 때까지 퍼스널 통신이나 통신 서버가 호스트 LU로의 다운스트림 LU 연결을 지연해야 하는지의 여부를 지정합니다. 대신, 시뮬레이트된 로그온 화면이 다운스트림 LU로 보내질 것입니다(AP_YES 또는 AP_NO).

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DOWNSTREAM_PU

```
   unsigned short  cur_rcv_pac_win;        /* current receive pacing        */
                                           /* window size                   */
   unsigned long   send_data_frames;       /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                           /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;        /* number of data bytes sent    */
   unsigned long   rcv_data_frames;        /* num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                           /* num of FMD data frames recvd  */
   unsigned long   rcv_data_bytes;         /* number of data bytes received */
   unsigned char   sidh;                   /* session ID high byte          */
   unsigned char   sidl;                   /* session ID low byte           */
   unsigned char   odai;                   /* ODAI bit set                  */
   unsigned char   ls_name[8];             /* Link station name             */
   unsigned char   pacing_type;            /* type of pacing in use         */
} SESSION_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DOWNSTREAM_PU

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **dslu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

dspu_name
조회되고 있는 다운스트림 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dspu_services
DSPU 서비스 필터. AP_PU_CONCENTRATION으로 설정되면, PU 집중에 의해 서비스되는 다운스트림 LU만이 반환됩니다. AP_DLUR로 설정되면, DLUR의 지원을 받는 LU만이 반환됩니다. 그렇지 않고, AP_NONE으로 설정되면 모든 다운스트림 LU에 대한 정보가 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

downstream_pu_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

downstream_pu_data.dspu_name
다운스트림 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

downstream_pu_data.description
자원 설명(DEFINE_LS에서 정의된 대로).

downstream_pu_data.ls_name
링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

downstream_pu_data.pu_sscp_sess_active
다운스트림 PU로의 PU_SSCP 세션이 활동중인지의 여부를 지정합니다. AP_YES나 AP_NO로 설정하십시오.

downstream_pu_data.dspu_services
국지 노드가 링크를 거쳐 다운스트림 PU에 제공하는 서비스를 지정합니다. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_PU_CONCENTRATION
다운스트림 LU에 대한 PU 집중을 제공하는 국지 노드.

AP_DLUR
다운스트림 LU에 대한 DLUR 지원을 제공하는 국지 노드.

downstream_pu_data.pu_sscp_stats.rcv_ru_size
최대 수신 RU 크기. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_pu_data.pu_sscp_stats.send_ru_size
최대 송신 RU 크기. **downstream_lu_detail.det_data.dspu_services**가 AP_PU_CONCENTRATION으로 설정되면, 이 필드가 예약됩니다.

downstream_pu_data.pu_sscp_stats.max_send_btu_size

downstream_pu_data.pu_sscp_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_pu_data.pu_sscp_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

downstream_pu_data.pu_sscp_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

downstream_pu_data.pu_sscp_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

downstream_pu_data.pu_sscp_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

downstream_pu_data.pu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

downstream_pu_data.pu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

downstream_pu_data.pu_sscp_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정하며, BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

downstream_pu_data.pu_sscp_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

downstream_pu_data.pu_sscp_stats.pacing_type
업스트림 PU-SSCP 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE을 취할 것입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_DSPU_TEMPLATE

이 명령은 통신 서버에만 적용됩니다.

QUERY_DSPU_TEMPLATE는 암시적 링크를 통한 PU 집중에 사용된 정의된 다운스트림 PU에 관한 정보를 반환합니다. 이 정보는 리스트로 반환됩니다. 특정 다운스트림 PU 템플리트에 관한 정보를 확보하거나 여러 개의 청크(chunk)로 리스트 정보를 확보하려면, **template_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

VCB 구조

```
typedef struct query_dspu_template
{
   unsigned short  opcode;                /* verb operation code           */
   unsigned char   attributes;            /* Verb attributes               */
   unsigned char   reserv2;               /* reserved                      */
   unsigned char   format;                /* format                        */
   unsigned short  primary_rc;            /* primary return code           */
   unsigned long   secondary_rc;          /* secondary return code         */
   unsigned char   *buf_ptr;              /* pointer to buffer             */
   unsigned long   buf_size;              /* buffer size                   */
   unsigned long   total_buf_size;        /* total buffer size required    */
   unsigned short  num_entries;           /* number of entries             */
   unsigned short  total_num_entries;     /* total number of entries       */
   unsigned char   list_options;          /* listing options               */
   unsigned char   reserv3;               /* reserved                      */
        unsigned char   template_name[8]; /* name of DSPU template          */
} QUERY_DSPU_TEMPLATE;

typedef struct dspu_template_data
{
   unsigned short  overlay_size;          /* size of this entry            */
        unsigned char   template_name[8]; /* name of DSPU template          */
   unsigned char   description;           /* resource description          */
        unsigned char   reserv1[12];      /* reserved                      */
        unsigned short  max_instance;     /* max active template instances */
        unsigned short  active instance;  /* current active instances      */
        unsigned short  num_of_dslu_templates;
                                          /* number of DSLU templates      */
} DSPU_TEMPLATE_DATA;
```

각 **dspu_template_data** 뒤에는 **num_of_dslu_templates** 다운스트림 LU 템플리트가 따라옵니다. 각 다운스트림 LU 템플리트는 다음과 같은 형식을 가집니다.

```
typedef struct dslu_template_data
{
   unsigned short  overlay_size;       /* size of this entry         */
   unsigned char   reserv1[2];         /* reserved                   */
        DSLU_TEMPLATE   dslu_template; /* downstream LU template     */
} DSLU_TEMPLATE_DATA;

typedef struct dslu_template
{
        unsigned char   min_nau;       /* min NAU address in range   */
```

```
        unsigned char   max_nau;        /* max NAU address in range   */
    unsigned char   reserv1[10];        /* reserved                   */
        unsigned char   host_lu[8];     /* host LU or pool name       */
} DSLU_TEMPLATE;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_DSPU_TEMPLATE

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 첨부할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

지정된 **template_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

template_name
DSPU 템플리트의 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. **list_options**이 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

dspu_template_data.overlay_size
이 입력항목에 있는 바이트의 수(LU 템플리트를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dspu_template_data.template_name
DSPU 템플리트의 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다.

dspu_template_data.description
자원 설명(QUERY_DSPU_TEMPLATE에서 정의된 대로).

dspu_template_data.max_instance
이는 동시에 활동중일 수 있는 템플리트의 최대 인스턴스 갯수입니다.

dspu_template_data.active_instance
이는 현재 활동중인 템플리트의 인스턴스 갯수입니다.

dspu_template_data.num_of_dslu_templates
이 다운스트림 PU 템플리트에 대한 다운스트림 LU 템플리트의 갯수. 이 필드 뒤에는 중재점 범주에 등록된 각 응용프로그램에 대해 하나씩 **num_of_dslu_templates_application_id** 입력항목이 있습니다.

dslu_template_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

dslu_template_data.dslu_template.min_nau
범위내 최소 NAU 주소.

dslu_template_data.dslu_template.max_nau
범위내 최대 NAU 주소.

def_data.allow_timeout
호스트 LU 정의에서 지정된 **timeout** 시간 동안 세션이 활동중이지 않은 상태로 남아 있는 경우, 이 다운스트림 LU에 의해 사용된 호스트 LU를 프로그램이 시간종료시킬 수 있도록 허용하는지를 지정합니다 (AP_YES 또는 AP_NO).

def_data.delayed_logon
다운스트림 LU로부터 첫번째 데이터를 수신할 때까지 프로그램이 호스트 LU로의 다운스트림 LU 연결을 지연해야 하는지의 여부를 지정합니다. 대신, 시뮬레이트된 로그온 화면이 다운스트림 LU로 보내집니다(AP_YES 또는 AP_NO).

dslu_template_data.dslu_template.host_lu_name
범위내에 있는 모든 다운스트림 LU가 매핑될 호스트 LU나 호스트 LU 풀의 이름. 이는 8 바이트 영숫자 유형 A A-EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TEMPLATE_NAME

AP_INVALID_LIST_OPTION

관련 START_NODE 매개변수가 설정되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_FOCAL_POINT

QUERY_FOCAL_POINT는 퍼스널 통신이나 통신 서버가 알고 있는 중재점에 관한 정보를 반환합니다.

이 정보는 리스트로 반환됩니다. 특정 중재점에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **ms_category** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

주: 어떤 중재점도 찾을 수 없으면, **fp_data.fp_type**이 AP_NO_FP로 설정된 상태로 FP_DATA 구조체가 반환될 것입니다. 다음 구조체를 참조하십시오.

VCB 구조

```
typedef struct query_focal_point
{
   unsigned short  opcode;               /* verb operation code           */
   unsigned char   reserv2;              /* reserved                      */
   unsigned char   format;               /* format                        */
   unsigned short  primary_rc;           /* primary return code           */
   unsigned long   secondary_rc;         /* secondary return code         */
   unsigned char   *buf_ptr;             /* pointer to buffer             */
   unsigned long   buf_size;             /* buffer size                   */
   unsigned long   total_buf_size;       /* total buffer size required    */
   unsigned short  num_entries;          /* number of entries             */
   unsigned short  total_num_entries;    /* total number of entries       */
   unsigned char   list_options;         /* listing options               */
   unsigned char   reserv3;              /* reserved                      */
        unsigned char   ms_category[8];  /* name of MS category           */
} QUERY_FOCAL_POINT;

typedef struct fp_data
{
   unsigned short  overlay_size;         /* size of this entry            */
        unsigned char   ms_appl_name[8];  /* focal point application name */
        unsigned char   ms_category[8];   /* focal point category         */
   unsigned char   description[RD_LEN];
                                         /* resource description          */
        unsigned char   fp_fqcp_name[17];  /* focal pt fully qual CP name */
        unsigned char   bkup_appl_name[8]; /* backup focal pt appl name   */
        unsigned char   bkup_fp_fqcp_name[17];
                                         /* backup FP fully qualified     */
                                         /* CP name                       */
        unsigned char   implicit_appl_name[8];
                                         /* implicit FP appl name         */
        unsigned char   implicit_fp_fqcp_name[17];
                                         /* implicit FP fully             */
                                         /* qualified CP name             */
        unsigned char   fp_type;         /* focal point type              */
        unsigned char   fp_status;       /* focal point status            */
        unsigned char   fp_routing;      /* type of MDS routing to use    */
   unsigned char   reserva[20];          /* reserved                      */
        unsigned short  number_of_appls; /* number of applications       */
} FP_DATA;
```

각 **fp_data** 다음에는 **number_of_appls** 응용프로그램 이름이 있습니다. 각 응용프로그램 이름은 다음의 형식을 가집니다.

```
typedef struct application_id
{
       unsigned char    appl_name[8];   /* application name             */
} APPLICATION_ID;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_FOCAL_POINT

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **ms_category**(다음 매개변수 참조)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

ms_category

관리 서비스 범주. 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 범주를 위한 4 바이트 구조로 정의된 값(EBCDIC 공백으로 오른쪽 채워넣기됨) 중 하나이거나 8 바이트의 유형 1134 EBCDIC

설치 정의 이름일 수 있습니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

fp_data.overlay_size
이 입력항목에 있는 바이트의 수(응용프로그램 이름을 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

fp_data.ms_appl_name
현재 활동중인 중재점 응용프로그램의 이름. 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 응용프로그램을 위한 4 바이트 구조로 정의된 값(오른쪽은 EBCDIC 공백으로 채워짐) 중 하나이거나 8 바이트의 유형 1134 EBCDIC 설치 정의 이름일 수 있습니다.

fp_data.ms_category
관리 서비스 범주. 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 범주를 위한 4 바이트 구조로 정의된 값(오른쪽은 EBCDIC 공백으로 채워짐) 중 하나이거나 8 바이트의 유형 1134 EBCDIC 설치 정의 이름일 수 있습니다.

fp_data.description
자원 설명(DEFINE_FOCAL_POINT에서 정의된 대로). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

fp_data.fp_fqcp_name
현재 활동중인 중재점의 전체 정식 제어점(CP) 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

fp_data.bkup_appl_name

백업 중재점 응용프로그램의 이름. 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 응용프로그램을 위한 4 바이트 구조로 정의된 값(오른쪽은 EBCDIC 공백으로 채워짐) 중 하나이거나 8 바이트의 유형 1134 EBCDIC 설치 정의 이름일 수 있습니다.

fp_data.bkup_fp_fqcp_name

현재 활동중인 백업 중재점의 전체 정식 제어점(CP) 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

fp_data.implicit_appl_name

암시적 중재점 응용프로그램의 이름(DEFINE_FOCAL_POINT 명령을 사용하여 지정된). 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 응용프로그램을 위한 4 바이트 구조로 정의된 값(오른쪽은 EBCDIC 공백으로 채워짐) 중 하나이거나 8 바이트의 유형 1134 EBCDIC 설치 정의 이름일 수 있습니다. 암시적 중재점이 현재 활동중인 중재점인 경우에는 이 필드가 **ms_appl_name**과 동일할 것입니다.

fp_data.bkup_fp_fqcp_name

암시적 중재점의 전체 정식 제어점(CP) 이름(DEFINE_FOCAL_POINT 명령을 사용하여 지정된 것과 같이). 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 암시적 중재점이 현재 활동중인 중재점인 경우에는 이 필드가 **fp_fqcp_name**과 동일할 것입니다.

fp_data.fp_type

중재점의 유형. 더 상세한 것은 *SNA 관리 서비스*를 참조하십시오. 이는 다음 값 중 하나일 것입니다.

AP_EXPLICIT_PRIMARY_FP
AP_BACKUP_FP
AP_DEFAULT_PRIMARY_FP
AP_IMPLICIT_PRIMARY_FP
AP_DOMAIN_FP
AP_HOST_FP
AP_NO_FP

fp_data.fp_status

중재점의 상태. 이는 다음 값 중 하나일 수 있습니다.

AP_NOT_ACTIVE
중재점이 현재 활동중이지 않습니다.

AP_ACTIVE
중재점이 현재 활동중입니다.

AP_PENDING
중재점이 대기 활동중입니다. 이는 암시적 요청이 중재점으로 송신된 후와 응답이 수신되기 전에 발생합니다.

AP_NEVER_ACTIVE
범주에 대한 응용프로그램 등록이 승인되었을 지라도 지정된 범주에 사용가능한 중재점 정보가 없습니다.

fp_data.fp_routing
MDS 트랜스포트를 사용하여 중재점에 데이터를 보낼 때 응용프로그램이 지정해야 하는 경로지정의 유형.

AP_DEFAULT
MDS_MU를 중재점으로 전달하는 데 생략시 경로지정이 사용됩니다.

AP_DIRECT
MDS_MU가 세션상에서 직접 중재점으로 전송경로될 것입니다.

fp_data.number_of_appls
이 중재점 범주에 등록된 응용프로그램의 수. 이 필드 뒤에는 중재점 범주에 등록된 각 응용프로그램에 대해 하나씩 **number_of_appls application_id** 입력항목이 있을 것입니다.

appl_name
중재점 범주에 등록된 응용프로그램의 이름. 이는 SNA 관리 서비스에서 기술된 것과 같이 관리 서비스 응용프로그램을 위한 4 바이트 구조로 정의된 값(오른쪽은 EBCDIC 공백으로 채워짐) 중 하나이거나 8 바이트의 유형 1134 EBCDIC 설치 정의 이름일 수 있습니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MS_CATEGORY

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_HPR_STATS

이 명령은 통신 서버에만 적용됩니다.

QUERY_HPR_STATS는 노드의 HPR 성능을 설명하는 통계를 반환합니다. QUERY_HPT_STATS는 RTP Tower를 지원하는 노드에 의해서만 지원됩니다.

VCB 구조

```
typedef struct query_hpr_stats
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        unsigned COUNTER
                        num-orig_rs_sent;   /* RS requests sent as origin  */
        unsigned COUNTER
                        num_orig_rs_rej;    /* RS rejections at orign      */
        unsigned COUNTER
                        num_inter_rs_rcvd;  /* Intermediate RS requests    */
        unsigned COUNTER
                        num_inter_rs_rej;   /* Intermediate RS rejections  */
        unsigned COUNTER
                        num_dest_rs_rcvd;   /* RS reqs as destination      */
        unsigned COUNTER
                        num_dest_rs_rej;    /* RS rej sent as destination  */
        unsigned char   reserv[28];         /* reserved                    */
} QUERY_HPR_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_HPR_STATS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

num_orig_rs_sent

노드가 시작된 이래 이 노드에서 생겨 송신된 총 HPR 전송경로 설정 요청 수.

num_orig_rs_rej

노드가 시작된 이래 이 노드에서 생기고 다른 노드에 의해 거부된 총 HPR 전송경로 설정 요청 수.

num_inter_rs_rcvd
노드가 시작된 이래 중계 노드로 활동하고 있는 이 노드에 의해 처리된 총 HPR 전송경로 설정 요청 수.

num_inter_rs_rej
노드가 시작된 이래 중계 노드로 활동하고 있는 이 노드에 의해 처리되고 노드에 의해 거부된 총 HPR 전송경로 설정 요청 수.

num_dest_rs_rcvd
노드가 시작된 이래 이 노드에 의해 수신되고, 목적지로 이 노드를 가지는 총 HPR 전송경로 설정 요청 수.

num_dest_rs_rej
노드가 시작된 이래 목적지로 이 노드를 가지고 노드에 의해 거부된 이 노드에 수신된 총 HPR 전송경로 설정 요청 수.

active_isr_hpr_sessions
노드에서 현재 활동중인 HPR-APPN 경계 기능을 사용하는 ISR 세션의 수.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 HPR RTP Tower 기능을 지원하지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

QUERY_ISR_SESSION

이 명령은 통신 서버에만 적용됩니다.

QUERY_ISR_SESSION은 네트워크 노드에서만 사용되며 네트워크 노드가 중계 세션 경로지정(ISR)을 제공하고 있는 세션에 대한 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 세션에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **fqpcid** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 구조체에 있는 필드들이 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 먼저 **fqpcid.pcid**에 의해 정렬된 후 **fqpcid.fqcp_name**에서 EBCDIC 사전식 정렬에 의해 정렬됩니다. **fqpcid.pcid_name**에 의한 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

fqpcid 구조체의 형식은 8 바이트의 절차 상관자 식별자(PCID)이며 세션 창설자의 네트워크 정식 CP명입니다.

각 세션에 대한 상세 정보 외에, 이것이 START_NODE 매개변수에 지정되는 경우에는 전송경로 선택 제어 벡터(RSVC)가 반환됩니다. 이 RSVC는 세션이 hop-by-hop 형태에서 취하는 네트워크를 통한 전송경로를 정의합니다.

VCB 구조

형 D 2

```
typedef struct query_isr_session
{
   unsigned short  opcode;            /* verb operation code          */
   unsigned char   reserv2;           /* reserved                     */
   unsigned char   format;            /* format                       */
   unsigned short  primary_rc;        /* primary return code          */
   unsigned long   secondary_rc;      /* secondary return code        */
   unsigned char   *buf_ptr;          /* pointer to buffer            */
   unsigned long   buf_size;          /* buffer size                  */
   unsigned long   total_buf_size;    /* total buffer size required   */
   unsigned short  num_entries;       /* number of entries            */
   unsigned short  total_num_entries; /* total number of entries      */
   unsigned char   list_options;      /* listing options              */
   unsigned char   session_type;      /* is this query for DLUR or    */
                                      /* regular ISR sessions?        */
   FQPCID          fqpcid;            /* fully qualified procedure    */
                                      /* correlator ID                */
} QUERY_ISR_SESSION;
```

```
typedef struct isr_session_summary
{
   unsigned short  overlay_size;      /* size of this entry           */
   FQPCID          fqpcid;            /* fully qualified procedure    */
                                      /* correlator ID                */
} ISR_SESSION_SUMMARY;

typedef struct isr_session_detail
{
   unsigned short  overlay_size;      /* size of this entry           */
   FQPCID          fqpcid;            /* fully qualified procedure    */
   unsigned short  sub_overlay_size;  /* offset to appended RSCV      */
                                      /* correlator ID                */
   unsigned char   trans_pri;         /* Transmission priority:       */
   unsigned char   cos_name[8];       /* Class-of-service name        */
   unsigned char   ltd_res;           /* Session spans a limited      */
   unsigned char   reserv1[8];        /* reserved                     */
                                      /* resource                     */
   SESSION_STATS   pri_sess_stats;    /* primary hop session stats    */
   SESSION_STATS   sec_sess_stats;    /* secondary hop session        */
                                      /* statistics                   */
   unsigned char   sess_lu_type;      /* session LU type              */
   unsigned char   sess_lu_level;     /* session LU level             */
   unsigned char   pri_tg_number;     /* Primary session TG number    */
   unsigned char   sec_tg_number;     /* Secondary session TG number  */
   unsigned long   rtp_tcid;          /* RTP TC identifier            */
   unsigned long   time_active;       /* time elapsed since           */
                                      /* activation                   */
   unsigned char   isr_state;         /* current state of ISR session */
   unsigned char   reserv2[11];       /* reserved                     */
   unsigned char   mode_name[8];      /* mode name                    */
   unsigned char   pri_lu_name[17];   /* primary LU name              */
   unsigned char   sec_lu_name[17];   /* secondary LU name            */
   unsigned char   pri_adj_cp_name[17];
                                      /* primary stage adj CP name    */
   unsigned char   sec_adj_cp_name[17];
                                      /* secondary stage adj CP name  */
   unsigned char   reserv3[3];        /* reserved                     */
   unsigned char   rscv_len;          /* Length of following RSCV     */
} ISR_SESSION_DETAIL;

typedef struct fqpcid
{
   unsigned char   pcid[8];           /* pro correlator identifier    */
   unsigned char   fqcp_name[17];     /* orig's network qualified     */
                                      /* CP name                      */
   unsigned char   reserve3[3];       /* reserved                     */
} FQPCID;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;       /* session receive RU size      */
   unsigned short  send_ru_size;      /* session send RU size         */
   unsigned short  max_send_btu_size; /* Maximum send BTU size        */
   unsigned short  max_rcv_btu_size;  /* Maximum rcv BTU size         */
   unsigned short  max_send_pac_win;  /* Max send pacing window size  */
   unsigned short  cur_send_pac_win;  /* Curr send pacing window size */
   unsigned short  max_rcv_pac_win;   /* Max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;   /* Curr rec pacing window size  */
   unsigned long   send_data_frames;  /* Number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                      /* num of FMD data frames sent  */
```

```
   unsigned long   send_data_bytes;   /* Number of data bytes sent    */
   unsigned long   rcv_data_frames;   /* Num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                      /* num of FMD data frames recvd */
   unsigned long   rcv_data_bytes;    /* Num data bytes received      */
   unsigned char   sidh;              /* Session ID high byte         */
   unsigned char   sidl;              /* Session ID low byte          */
   unsigned char   odai;              /* ODAI bit set                 */
   unsigned char   ls_name[8];        /* Link station name            */
   unsigned char   pacing_type;       /* type of pacing in use        */
} SESSION_STATS;
```

VCB 구조

형 D 1(이전 레벨)

```
typedef struct isr_session_detail
{
   unsigned short  overlay_size;      /* size of this entry           */
   FQPCID          fqpcid;            /* fully qualified procedure    */
   unsigned short  sub_overlay_size;  /* offset to appended RSCV      */
                                      /* correlator ID                */
   unsigned char   trans_pri;         /* Transmission priority:       */
   unsigned char   cos_name[8];       /* Class-of-service name        */
   unsigned char   ltd_res;           /* Session spans a limited      */
   unsigned char   reserv1[2];        /* reserved                     */
                                      /* resource                     */
   SESSION_STATS   pri_sess_stats;    /* primary hop session stats    */
   SESSION_STATS   sec_sess_stats;    /* secondary hop session        */
                                      /* statistics                   */
   unsigned char   sess_lu_type;      /* session LU type              */
   unsigned char   sess_lu_level;     /* session LU level             */
   unsigned char   pri_tg_number;     /* Primary session TG number    */
   unsigned char   sec_tg_number;     /* Secondary session TG number  */
   unsigned long   rtp_tcid;          /* RTP TC identifier            */
   unsigned long   time_active;       /* time elapsed since           */
                                      /* activation                   */
   unsigned char   isr_state;         /* current state of ISR session */
   unsigned char   reserv2[11];       /* reserved                     */
   unsigned char   mode_name[8];      /* mode name                    */
   unsigned char   pri_lu_name[17];   /* primary LU name              */
   unsigned char   sec_lu_name[17];   /* secondary LU name            */
   unsigned char   pri_adj_cp_name[17];
                                      /* primary stage adj CP name    */
   unsigned char   sec_adj_cp_name[17];
                                      /* secondary stage adj CP name  */
   unsigned char   reserv3[3];        /* reserved                     */
   unsigned char   rscv_len;          /* Length of following RSCV     */
} ISR_SESSION_DETAIL;

typedef struct fqpcid
{
   unsigned char   pcid[8];           /* pro correlator identifier    */
   unsigned char   fqcp_name[17];     /* orig's network qualified     */
                                      /* CP name                      */
   unsigned char   reserve3[3];       /* reserved                     */
} FQPCID;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;       /* session receive RU size      */
```

```
   unsigned short  send_ru_size;      /* session send RU size         */
   unsigned short  max_send_btu_size; /* Maximum send BTU size        */
   unsigned short  max_rcv_btu_size;  /* Maximum rcv BTU size         */
   unsigned short  max_send_pac_win;  /* Max send pacing window size  */
   unsigned short  cur_send_pac_win;  /* Curr send pacing window size */
   unsigned short  max_rcv_pac_win;   /* Max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;   /* Curr rec pacing window size  */
   unsigned long   send_data_frames;  /* Number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                      /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;   /* Number of data bytes sent    */
   unsigned long   rcv_data_frames;   /* Num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                      /* num of FMD data frames recvd */
   unsigned long   rcv_data_bytes;    /* Num data bytes received      */
   unsigned char   sidh;              /* Session ID high byte         */
   unsigned char   sidl;              /* Session ID low byte          */
   unsigned char   odai;              /* ODAI bit set                 */
   unsigned char   ls_name[8];        /* Link station name            */
   unsigned char   pacing_type;       /* type of pacing in use        */
} SESSION_STATS;
```

VCB 구조

형 D 0(이전 레벨)

```
typedef struct isr_session_detail
{
   unsigned short  overlay_size;      /* size of this entry           */
   FQPCID          fqpcid;            /* fully qualified procedure    */
   unsigned char   trans_pri;         /* Transmission priority:       */
   unsigned char   cos_name[8];       /* Class-of-service name        */
   unsigned char   ltd_res;           /* Session spans a limited      */
   unsigned char   reserv1[8];        /* reserved                     */
                                      /* resource                     */
   SESSION_STATS   pri_sess_stats;    /* primary hop session stats    */
   SESSION_STATS   sec_sess_stats;    /* secondary hop session        */
                                      /* statistics                   */
   unsigned char   reserv3[3];        /* reserved                      */
   unsigned char   reserva[20];       /* reserved                      */
   unsigned char   rscv_len;          /* Length of following RSCV     */
} ISR_SESSION_DETAIL;
```

주: ISR 세션 상세 중복 뒤에는 *SNA formats*에 의해 정의된 것과 같이 전송경로 선택 제어 벡터(RSCV)가 따라올 것입니다. 이 제어 벡터는 네트워크를 통한 세션 전송경로를 정의하며 BIND 상에서 운반됩니다. 이 RSCV를 포함할지는 노드가 시작될 때 결정되며(START_NODE의 한 옵션으로), DEFINE_ISR_STATS를 사용하여 나중에 변경될 수 있습니다. RSCV들이 보관되지 않도록 지정하는 데 이 명령들이 사용되었으면, **rscv_len**이 0으로 설정됩니다.

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_ISR_SESSION

format
VCB의 형식과 또는 반환된 중복 형식을 식별합니다. 위에서 나열된 VCB의 버전이나 중복을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **fqpcid**(다음 매개변수를 보십시오)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

session_type
이 명령이 DLUR에 의해 유지보수되는 세션을 조회합니까? 또는 정규 ISR 세션을 조회합니까?

AP_ISR_SESSION	ISR 세션
AP_DLUR_SESSIONS	DLUR 세션

fqpcid.pcid
절차 상관자 ID. 이는 8 바이트의 16진 문자열입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

fqpcid.pcid_name
전체 정식 제어점 이름. 이 이름의 길이는 17-bytes 바이트이며 오른

쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

isr_session_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

isr_session_summary.fqpcid.pcid
절차 상관자 ID.

isr_session_summary.fqpcid.fqcp_name
전체 정식 제어점 이름. 이 이름의 길이는 17-bytes 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

isr_session_detail.overlay_size
이 입력항목에 있는 바이트의 수(RSCV를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

isr_session_detail.sub_overlay_size
이 필드는 이 상세 중복의 크기를 부여합니다. RSCV가 첨부되면, 이것이 RSCV의 시작에 대한 오프셋입니다. 이 필드는 하나의 상세 구조체의 형식 크기와 같거나 클 수 있습니다(향후 확장 허용).

isr_session_detail.fqpcid.pcid
절차 상관자 ID.

isr_session_detail.fqpcid.fqcp_name
전체 정식 제어점 이름. 이 이름의 길이는 17-bytes 바이트이며 오른

쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

session_detail.trans_pri
전송 우선순위. 다음 값 중 하나로 설정됩니다.

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

session_detail.cos_name
서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

session_detail.ltd_res
세션이 자원 링크를 사용할지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

isr_session_detail.pri_sess_stats.rcv_ru_size
최대 수신 RU 크기.

isr_session_detail.pri_sess_stats.send_ru_size
최대 송신 RU 크기.

isr_session_detail.pri_sess_stats.max_send_btu_size
1차 세션 홉(hop)상에서 송신될 수 있는 최대 BTU 크기.

isr_session_detail.pri_sess_stats.max_rcv_btu_size
1차 세션 홉상에서 수신될 수 있는 최대 BTU 크기.

isr_session_detail.pri_sess_stats.max_send_pac_win
1차 세션 홉상의 최대 송신 페이싱 창 크기.

isr_session_detail.pri_sess_stats.cur_send_pac_win
1차 세션 홉상의 현재 송신 페이싱 창 크기.

isr_session_detail.pri_sess_stats.max_rcv_pac_win
1차 세션 홉상의 최대 수신 페이싱 창 크기.

isr_session_detail.pri_sess_stats.cur_rcv_pac_win
1차 세션 홉상의 현재 수신 페이싱 창 크기.

isr_session_detail.pri_sess_stats.send_data_frames
1차 세션 홉상에서 송신된 정상 흐름 데이터 프레임의 갯수.

isr_session_detail.pri_sess_stats.send_data_frames
1차 세션 홉상에서 송신된 정상 흐름 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.send_fmd_data_frames
1차 세션 홉상에서 송신된 정상 흐름 FMD 데이터 프레임의 갯수.

DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.send_data_bytes
1차 세션 홉상에서 송신된 정상 흐름 데이터 바이트의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.rcv_data_frames
1차 세션 홉상에서 수신된 정상 흐름 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.rcv_fmd_data_frames
1차 세션 홉상에서 수신된 정상 흐름 FMD 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.rcv_data_bytes
1차 세션 홉상에서 수신된 정상 흐름 데이터 바이트의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.pri_sess_stats.sidh
세션 ID 상위(high) 바이트.

isr_session_detail.pri_sess_stats.sidl
세션 ID 하위(low) 바이트.

isr_session_detail.pri_sess_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정합니다. BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

isr_session_detail.pri_sess_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 세션 통계를 세션 데이터가 흐르는 링크와 상관시키는 데 이 필드가 사용될 수 있습니다.

isr_session_detail.sec_sess_stats.rcv_ru_size
최대 수신 RU 크기.

isr_session_detail.pri_sess_stats.pacing_type
1차 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE를 취할 수도 있습니다.

isr_session_detail.sec_sess_stats.send_ru_size
최대 송신 RU 크기.

isr_session_detail.sec_sess_stats.max_send_btu_size
2차 세션 홉상에서 송신될 수 있는 최대 BTU 크기.

isr_session_detail.sec_sess_stats.max_rcv_btu_size
2차 세션 홉상에서 수신될 수 있는 최대 BTU 크기.

isr_session_detail.sec_sess_stats.max_send_pac_win
2차 세션 홉상의 최대 송신 페이싱 창 크기.

isr_session_detail.sec_sess_stats.cur_send_pac_win
2차 세션 홉상의 현재 송신 페이싱 창 크기.

isr_session_detail.sec_sess_stats.max_rcv_pac_win
2차 세션 홉상의 최대 수신 페이싱 창 크기.

isr_session_detail.sec_sess_stats.cur_rcv_pac_win
2차 세션 홉상의 현재 수신 페이싱 창 크기.

isr_session_detail.sec_sess_stats.send_data_frames
2차 세션 홉상에서 송신된 정상 흐름 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.send_fmd_data_frames
2차 세션 홉상에서 송신된 정상 흐름 FMD 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.send_data_bytes
2차 세션 홉상에서 송신된 정상 흐름 데이터 바이트의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.rcv_data_frames
2차 세션 홉상에서 수신된 정상 흐름 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.rcv_fmd_data_frames
2차 세션 홉상에서 수신된 정상 흐름 FMD 데이터 프레임의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.rcv_data_bytes
2차 세션 홉상에서 수신된 정상 흐름 데이터 바이트의 갯수. DEFINE_ISR_STATS를 사용하여 통계의 수집이 사용불가능하게 되었으면, 이 필드에 0이 반환될 것입니다.

isr_session_detail.sec_sess_stats.sidh
세션 ID 상위(high) 바이트.

isr_session_detail.sec_sess_stats.sidl
세션 ID 하위 바이트(LFSID로부터).

isr_session_detail.sec_sess_stats.odai

원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정합니다. BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

isr_session_detail.sec_sess_stats.ls_name

통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 중간 세션 통계를 특정 링크 스테이션과 상관시키는 데 이 필드가 사용될 수 있습니다.

isr_session_detail.sec_sess_stats.pacing_type

1차 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE를 취할 수 있습니다.

isr_session.detail.sess_lu_type

BIND상에서 지정된 세션의 LU 유형. 이 필드는 다음 값 중 하나를 취할 수 있습니다.

AP_LU_TYPE_0
AP_LU_TYPE_1
AP_LU_TYPE_2
AP_LU_TYPE_3
AP_LU_TYPE_4
AP_LU_TYPE_6
AP_LU_TYPE_7
AP_LU_TYPE_UNKNOWN
(LU 유형 5는 의도적으로 생략되었습니다.)

DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한 AP_LU_TYPE_UNKNOWN이 항상 반환될 것입니다.

isr_session.detail.sess_lu_level

세션의 LU 레벨. 이 필드는 다음 값 중 하나를 취할 수 있습니다.

AP_LU_LEVEL_0
AP_LU_LEVEL_1
AP_LU_LEVEL_2
AP_LU_LEVEL_UNKNOWN

LU 유형 6 외의 다른 LU 유형의 경우,이 필드가 AP_LU_LEVEL_0으로 설정됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한 AP_LU_LEVEL_UNKNOWN이 항상 반환될 것입니다.

isr_session.detail.pri_tg_number

1차 세션 홉에 의해 통과되는 링크와 연관된 TG 번호. 1차 세션 스

테이지가 RTP 연결을 통과하면 0이 반환됩니다.
DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한, 0이 항상 반환될 것입니다.

isr_session.detail.sec_tg_number
1차 세션 홉에 의해 통과되는 링크와 연관된 TG 번호. 1차 세션 스테이지가 RTP 연결을 통과하면, 0이 반환됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한, 0이 항상 반환될 것입니다.

isr_session.detail.rtp_tcid
이 ISR 세션이 ANR/ISR 경계의 일부를 형성하는 경우에 반환되는 RTP 연결에 대한 국지 TC ID. 다른 경우에는 이 필드가 0으로 설정됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한 0이 항상 반환될 것입니다.

isr_session.detail.time_active
세션 활성화이래 경과된 시간으로 100분의 1초 단위로 측정됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한 0이 항상 반환될 것입니다.

isr_session.detail.isr_state
세션의 현재 상태. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_ISR_INACTIVE
AP_ISR_PENDING_ACTIVE
AP_ISR_ACTIVE
AP_ISR_PENDING_INACTIVE

isr_session.detail.mode_name
세션의 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.
DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않는 한, 모두 2진 0이 항상 반환될 것입니다.

isr_session.detail.pri_lu_name
세션의 1차 LU 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다. 각 이름은 내포된 공백 없이 최대 8 바이트를 가질 수 있습니다. 이 이름이 사용가능하지 않으면, 이 필드에 모두 2진 0이 반환됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않은 한 모두 2진 0이 항상 반환될 것입니다.

isr_session.detail.sec_lu_name
세션의 2차 LU 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다. 각 이름은 내포된 공백 없이 최대 8 바이트를 가질 수 있습니다. 이 이름이 사용가능하지

않으면, 이 필드에 모두 2진 0이 반환됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않은 한 모두 2진 0이 항상 반환될 것입니다.

isr_session.detail.pri_adj_cp_name
이 세션의 1차 스테이지 인접 CP명. 1차 세션 스테이지가 RTP 연결을 통과하면, 원격 RTP 끝점의 CP명이 반환됩니다. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다. 각 이름은 내포된 공백없이 최대 8 바이트를 가질 수 있습니다. 이 이름이 사용가능하지 않으면, 이 필드에 모두 2진 0이 반환됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않은 한 모두 2진 0이 항상 반환될 것입니다.

isr_session.detail.sec_adj_cp_name
이 세션의 2차 스테이지 인접 CP명. 2차 세션 스테이지가 RTP 연결을 통과하면, 원격 RTP 끝점의 CP명이 반환됩니다. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다. 각 이름은 내포된 공백없이 최대 8 바이트를 가질 수 있습니다. 이 이름이 사용가능하지 않으면, 이 필드에 모두 2진 0이 반환됩니다. DEFINE_ISR_STATS를 사용하여 이름의 수집이 사용가능해지지 않은 한 모두 2진 0이 항상 반환될 것입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_FQPCID

AP_INVALID_LIST_OPTION
AP_INVALID_SESSION_TYPE

관련 START_NODE 매개변수가 설정되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

노드에 네트워크 노드 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LOCAL_LU

QUERY_LOCAL_LU는 국지 LU에 대한 정보를 반환합니다. 퍼스널 통신이나 통신 서버 제어점 LU에 관한 정보를 검색하기 위해 QUERY_LOCAL_LU를 발행할 수 있습니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 국지 LU에 관한 정보를 확보하거나 여러 개의 청크(chunk)로 리스트 정보를 확보하려면, **lu_name**이나 **lu_alias** 필드가 설정되어야 합니다. **lu_name** 필드가 0이 아니면 색인을 결정하는 데 사용될 것입니다. **lu_name** 필드가 모두 0으로 설정되면, **lu_alias**가 색인을 결정하는데 사용될 것입니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU가 사용될 것입니다. **list_options** 필드가 AP_FIRST_IN_LIST로 설정되면 두 필드 모두 무시될 것입니다.(이 경우, AP_LIST_BY_ALIAS **list_options**가 설정되면 반환되는 리스트가 LU alias에 의해 정렬될 것이고, 그렇지 않으면 LU명에 의해 정렬될 것입니다.) 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 지정된 옵션에 따라 **lu_alias**나 **lu_name**상에서 정렬됩니다. EBCDIC 사전식 정렬하기에 의해 필드가 정렬됩니다.

반환되는 국지 LU의 리스트는 그들이 연관되어 있는 PU의 이름에 의해 필터링될 수 있습니다. 이 경우, **pu_name** 필드가 설정되어야 합니다.(그렇지 않은 경우에는 이 필드가 모두 0으로 설정되어야 합니다.)

VCB 구조

형 D 1

```
typedef struct query_local_lu
{
   unsigned short  opcode;             /* verb operation code         */
   unsigned char   reserv2;            /* reserved                    */
   unsigned char   format;             /* format                      */
   unsigned short  primary_rc;         /* primary return code         */
   unsigned long   secondary_rc;       /* secondary return code       */
   unsigned char   *buf_ptr;           /* pointer to buffer           */
   unsigned long   buf_size;           /* buffer size                 */
   unsigned long   total_buf_size;     /* total buffer size required  */
   unsigned short  num_entries;        /* number of entries           */
   unsigned short  total_num_entries;  /* total number of entries     */
   unsigned char   list_options;       /* listing options             */
   unsigned char   reserv3;            /* reserved                    */
   unsigned char   lu_name[8];         /* LU name                     */
   unsigned char   lu_alias[8];        /* LU alias                    */
        unsigned char   pu_name[8];    /* PU name filter              */
} QUERY_LOCAL_LU;

typedef struct local_lu_summary
{
   unsigned short  overlay_size;       /* size of this entry          */
```

```
   unsigned char   lu_name[8];         /* LU name                    */
   unsigned char   lu_alias[8];        /* LU alias                   */
   unsigned char   description;        /* resource description       */
} LOCAL_LU_SUMMARY;

typedef struct local_lu_detail
{
   unsigned short  overlay_size;       /* size of this entry         */
   unsigned char   lu_name[8];         /* LU name                    */
        LOCAL_LU_DEF_DATA def_data;    /* defined data               */
        LOCAL_LU_DEF_DATA det_data;    /* determined data            */
} LOCAL_LU_DETAIL;

typedef struct local_lu_def_data
{
   unsigned char   description[RD_LEN];
                                       /* resource description     */
        unsigned char   lu_alias[8];       /* local LU alias           */
        unsigned char   nau_address;       /* NAU address              */
        unsigned char   syncpt_support;    /* Reserved                 */
        unsigned short  lu_session_limit;  /* LU session limit         */
        unsigned char   default_pool;      /* member of default_lu_pool */
   unsigned char   reserv2;                /* reserved                 */
   unsigned char   pu_name[8];             /* PU name                  */
        unsigned char   lu_attributes;     /* LU attributes            */
   unsigned char   sscp_id[6];             /* SSCP ID                  */
        unsigned char   disable;           /* disable or enable Local LU*/
        unsigned char   attach_routing_data[128];
                                       /* routing data for         */
                                       /* incoming attaches        */
        unsigned char   lu_model;          /* LU model name for SDDLU  */
        unsigned char   model_name[8];     /* LU model name for SDDLU  */
        unsigned char   reserv4[16];       /* reserved                 */
 } LOCAL_LU_DEF_DATA;

typedef struct local_lu_det_data
{
        unsigned char   lu_sscp_sess_active;
                                       /* Is LU-SSCP session active */
        unsigned char   appl_conn_active;  /* Is LU-SSCP session active */
   unsigned char   reserv1[2];             /* reserved                 */
        SESSION_STATS   lu_sscp_stats;     /* LU-SSCP session statistics*/
   unsigned char   sscp_id[6];             /* SSCP ID                  */
} LOCAL_LU_DET_DATA;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;            /* session receive RU size      */
   unsigned short  send_ru_size;           /* session send RU size         */
   unsigned short  max_send_btu_size;      /* max send BTU size            */
   unsigned short  max_rcv_btu_size;       /* max rcv BTU size             */
        unsigned short  max_send_pac_win;  /* max send pacing win size     */
        unsigned short  cur_send_pac_win;  /* current send pacing win size */
        unsigned short  max_rcv_pac_win;   /* max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;        /* current receive pacing       */
                                       /* window size                  */
   unsigned long   send_data_frames;       /* number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                       /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;        /* number of data bytes sent    */
   unsigned long   rcv_data_frames;        /* num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                       /* num of FMD data frames recvd */
```

```
   unsigned long   rcv_data_bytes;         /* number of data bytes received*/
   unsigned char   sidh;                   /* session ID high byte         */
   unsigned char   sidl;                   /* session ID low byte          */
   unsigned char   odai;                   /* ODAI bit set                 */
   unsigned char   ls_name[8];             /* Link station name            */
        unsigned char   pacing _type;      /* Type of pacing in use        */
} SESSION_STATS;
```

VCB 구조

형 D 0

```
typedef struct local_lu_def_data
{
   unsigned char   description[RD_LEN];
                                           /* resource description      */
        unsigned char   lu_alias[8];       /* local LU alias            */
        unsigned char   nau_address;       /* NAU address               */
        unsigned char   syncpt_support;    /* Reserved                  */
        unsigned short  lu_session_limit;  /* LU session limit          */
        unsigned char   default_pool;      /* member of default_lu_pool */
   unsigned char   reserv2;                /* reserved                  */
   unsigned char   pu_name[8];             /* PU name                   */
        unsigned char   lu_attributes;     /* LU attributes             */
   unsigned char   sscp_id[6];             /* SSCP ID                   */
        unsigned char   disable;           /* disable or enable Local LU*/
        unsigned char   attach_routing_data[128];
                                           /* routing data for          */
                                           /* incoming attaches         */
 } LOCAL_LU_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LOCAL_LU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **lu_name**(또는 **lu_name**이 모두 0으로 설정된 경우, **lu_alias**)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

AP_LIST_BY_ALIAS
반환되는 리스트는 **lu_alias**에 의해 정렬됩니다. 이 옵션은 AP_FIRST_IN_LIST가 지정될 때에만 유효합니다. AP_LIST_FROM_NEXT 또는 AP_LIST_INCLUSIVE가 지정되는 경우, 리스트 정렬하기는 **lu_name**이나 **lu_alias**가 시작점으로 제공되었는지의 여부에 따라 다를 것입니다.

lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용될 것입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

pu_name
PU 명 필터. 이는 모두 0으로 설정되거나 오른쪽은 EBCDIC 공백으로 채워지면서 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 설정되어야 합니다. 이 필드가 설정되면 이 PU와 연관된 국지 LU만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

local_lu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

local_lu_summary.lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다.

local_lu_summary.lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

local_lu_summary.description
자원 설명(DEFINE_LOCAL_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

local_lu_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

local_lu_detail.lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다.

local_lu_detail.def_data.description
자원 설명(DEFINE_LOCAL_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

local_lu_detail.def_data.lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

local_lu_detail.def_data.nau_address
LU의 네트워크 지정 장치(NAU) 주소로 그 범주는 0—255. 0이 아닌 값은 LU가 종속 LU임을 암시합니다. 0은 LU가 독립 LU임을 암시합니다.

local_lu_detail.def_data.syncpt_support
예약됨.

local_lu_detail.def_data.lu_session_limit
국지 LU에 대한 최대 세션 수. 0의 값은 제한이 없음을 나타냅니다.

local_lu_detail.def_data.default_pool
LU가 종속 LU 6.2 생략시 풀의 구성원인 경우에는 AP_YES. 독립 LU의 경우에는 항상 AP_NO입니다.

local_lu_detail.def_data.pu_name
이 LU가 사용할 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 이 필드는 종속 LU에 의해서만 사용되며, 독립 LU에 대해서는 모두 2진 0으로 설정될 것입니다.

local_lu_detail.def_data.lu_attributes
구성된 LU 속성. 이 필드는 값 AP_NONE을 취하거나 다음 옵션들을 함께 취합니다.

AP_DISABLE_PWSUB
국지 LU에 대해 사용불가능하게 된 암호 대체 지원.

local_lu_detail.def_data.sscp_id
이 필드는 이 LU를 활성화시키도록 허용된 SSCP의 ID를 지정합니다. 이는 6 바이트의 2진 필드입니다. 이 필드는 종속 LU에 의해서만 사용되며, 독립 LU에 대해서는 모두 2진 0으로 설정되어야 합니다. 그렇지 않으면 모든 SSCP가 LU를 활성화할 수 있습니다.

local_lu_detail.def_data.disable
이 필드는 LOCAL LU가 사용불가능해야 하는지 또는 사용가능해야 하는지의 여부를 나타냅니다. 이 매개변수를 적당하게 설정한 상태로(AP_YES 또는 AP_NO) DEFINE_LOCAL_LU를 재발행함으로써, LU를 동적으로 사용가능 또는 사용불가능하게 할 수 있습니다. 사용불가능하게 된 LU가 사용가능해질 때, 프로그램은 NOTIFY(online)를 발행합니다. 사용가능한 LU가 사용불가능하게 될 때에는, 프로그램이 NOTIFY(off-line)를 발행합니다. LU가 사용불가능하게 될 때 바인드되면, 프로그램이 NOTIFY(off-line)가 뒤따르는 UNBIND를 발행합니다.

local_lu_detail.def_data.attach_routing_data
이 필드는 트랜잭션 프로그램(TP)을 위해 이 국지 LU에 도달하는 접속으로 생기는 DYNAMIC_LOAD_INDICATION상에서 변경되지 않고 건네지는 데이터를 나타냅니다. 예를 들어, 트랜잭션 프로그램(TP) 작업 디렉토리에 대한 경로를 설정하는 이 필드가 사용될 수도 있습니다.

def_data.lu_model
모델 유형 및 LU의 갯수. 이 필드는 종속 LU에 의해서만 사용되며 독립 LU에 대해서는 AP_UNKNOWN으로 설정되어야 합니다. 종속 LU의 경우, 이는 다음 값 중 하나로 설정됩니다.

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

종속 LU의 경우에는, **model_name**이 모두 2진 0으로 설정되지 않으면, 이 필드가 무시됩니다. AP_UNKNOWN외의 다른 값이 지정되고 호스트 시스템이 SDDLU(자체 정의 종속 LU)를 지원하면, 노드는 호스트에서 국지 LU를 동적으로 정의하기 위해서 요청되지 않은 PSID NMVT 응답을 생성할 것입니다. PSID 부속벡터에는 이 필드의 값에 해당하는 기계 유형이나 모델 번호가 들어 있을 것입니다. 이 필드는 명령을 재발행함으로써 동적으로 변경될 수도 있습니다. LU가 닫히고 활성종료된 후에야 변경사항이 영향을 미치게 됩니다.

def_data.model_name

LU의 모델명. 이 필드는 종속 LU에 의해서만 사용되며 독립 LU에 대해서는 2진 0으로 설정되어야 합니다.

이 필드가 2진 0으로 설정되지 않고 호스트 시스템이 SDDLU(자체 정의 종속 LU)를 지원하면, 노드는 호스트에서 국지 LU를 동적으로 정의하기 위해서 요청되지 않은 PSID NMVT 응답을 생성합니다. PSID 부속벡터에는 이 필드에서 제공된 이름이 들어 있습니다. 명령을 재발행함으로써 필드가 동적으로 변경될 수도 있습니다. LU가 닫히고 활성종료된 후에야 변경사항이 영향을 미치게 됩니다.

local_lu_detail.det_data.lu_sscp_session_active

LU-SSCP 세션이 활동중인지의 여부(AP_YES 또는 AP_NO)를 지정합니다. **def_data.nau_address**가 0이면, 이 필드가 예약됩니다.

local_lu_detail.det_data.appl_conn_active

응용프로그램이 LU를 사용하고 있는지의 여부(AP_YES 또는 AP_NO)를 지정합니다. **def_data.nau_address**가 0이면, 이 필드가 예약됩니다.

local_lu_detail.det_data.lu_sscp_stats.rcv_ru_size

이 필드는 항상 예약됩니다.

local_lu_detail.det_data.lu_sscp_stats.send_ru_size

이 필드는 항상 예약됩니다.

local_lu_detail.det_data.lu_sscp_stats.max_send_btu_size

송신될 수 있는 최대 BTU 크기.

local_lu_detail.det_data.lu_sscp_stats.max_rcv_btu_size

수신될 수 있는 최대 BTU 크기.

local_lu_detail.det_data.lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

local_lu_detail.det_data.lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

local_lu_detail.det_data.lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

local_lu_detail.det_data.lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

local_lu_detail.det_data.lu_sscp_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

local_lu_detail.det_data.lu_sscp_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

local_lu_detail.det_data.lu_sscp_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

local_lu_detail.det_data.lu_sscp_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

local_lu_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

local_lu_detail.det_data.lu_sscp_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

local_lu_detail.det_data.lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

local_lu_detail.det_data.lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

local_lu_detail.det_data.lu_sscp_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 ACTLU의 송신자가 이 필드를 0으로 설정하며, ACTLU 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

local_lu_detail.det_data.lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 이 세션을 세션이 흐르는 링크와 상관시키는 데 이 필드가 사용될 수 있습니다.

주: LU-SSCP 통계(**local_lu_detail.det_data.lu_sscp_stats**) 는 **nau_address**가 0이 아닐 때에만 유효합니다. 그렇지 않으면 필드가 예약됩니다.

local_lu_detail.det_data.lu_sscp_stats.pacing_type
LU-SSCP 세션상에서 사용중인 수신 페이싱 유형. AP_NONE으로 설정될 것입니다.

local_lu_detail.det_data.sscp_id
이 LU가 사용하는 PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하고 있는 6 바이트의 필드.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LOCAL_TOPOLOGY

모든 APPN 노드는 모든 인접 노드에 대한 전송 그룹(TG)에 관한 정보를 보유하고 있는 국지 위상 데이터베이스를 유지보수합니다.

QUERY_LOCAL_TOPOLOGY가 이러한 TG에 관한 정보를 반환될 수 있게 합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 국지 TG에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **dest**, **dest_type**, 그리고 **tg_num** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드들이 무시될 것입니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오. 이 리스트는 먼저 **dest**에서 정렬되고, 그 다음에는 **dest_type**에서 정렬되며, 마지막으로 **tg_num**에서 정렬됩니다. **dest** 이름은 먼저 이름 길이에 의해 정렬되고 난 후, 동일한 길이를 가진 이름에 대해서는 사전식 순서로 정렬됩니다. **dest_type** 필드는 다음의 순서를 따릅니다. AP_LEN_NODE, AP_NETWORK_NODE, AP_END_NODE, AP_VRN. **tg_num**은 번호로 정렬됩니다.

AP_LIST_INCLUSIVE가 선택되면, 반환되는 리스트가 그 이름의 첫번째 유효 레코드부터 시작합니다.

AP_LIST_FROM_NEXT가 선택되면, 지정된 것 다음의 이름으로 첫번째 유효 레코드에서 리스트가 시작할 것입니다.

VCB 구조

```
typedef struct query_local_topology
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   dest[17];      /* TG destination node           */
        unsigned char   dest_type;     /* TG destination node type      */
        unsigned char   tg_num;        /* TG number                     */
} QUERY_LOCAL_TOPOLOGY;

typedef struct local_topology_summary
{
   unsigned short  overlay_size;       /* size of this entry           */
```

```
        unsigned char   dest[17];       /* TG destination node          */
        unsigned char   dest_type;      /* TG destination node type     */
        unsigned char   tg_num;         /* TG number                    */
} LOCAL_TOPOLOGY_SUMMARY;

typedef struct local_topology_detail
{
   unsigned short  overlay_size;        /* size of this entry           */
        unsigned char   dest[17];       /* TG destination node          */
        unsigned char   dest_type;      /* TG destination node type     */
        unsigned char   tg_num;         /* TG number                    */
   unsigned char   reserv1;             /* reserved                     */
        LINK_ADDRESS    dlc_data;       /* DLC signalling data          */
        unsigned long   rsn;            /* resource sequence number     */


        unsigned char   status;         /* TG status                    */
        TG_DEFINED_CHARS tg_chars;      /* TG characteristics           */
        unsigned char   cp_cp_session_active;
                                        /* CP-CP session is active      */
        unsigned char   branch_tg;      /* branch link type             */
        unsigned char   reserva[13];    /* reserved                     */
} LOCAL_TOPOLOGY_DETAIL;

typedef struct link_address
{
        unsigned short  length;         /* length                       */
        unsigned short  reserve1;       /* reserved                     */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                        /* address                      */
} LINK_ADDRESS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LOCAL_TOPOLOGY

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **dest**, **dest_type** 그리고 **tg_num**의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

dest TG에 대한 전체 정식 목적지 노드 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dest_type
이 TG에 대한 목적지 노드의 노드 유형. 이는 다음 값 중 하나일 수 있습니다.

AP_NETWORK_NODE
AP_VRN
AP_END_NODE
AP_LEARN_NODE

dest_type을 알 수 없는 경우에는, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

tg_num
TG와 연관된 번호. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

local_topology_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

local_topology_summary.dest
TG에 대한 전체 정식 목적지 노드 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

local_topology_summary.dest_type
이 TG에 대한 목적지 노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

dest_type이 AP_END_NODE로 설정되는 경우, TG 목적지가 LEN 노드에 대한 것인지 또는 끝 노드에 대한 것인지를 지정함에 주의하십시오.

local_topology_summary.tg_num
TG와 연관된 번호.

local_topology_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

local_topology_detail.dest
TG에 대한 전체 정식 목적지 노드 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

local_topology_detail.dest_type
이 TG에 대한 목적지 노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN
AP_END_NODE

dest_type이 AP_END_NODE로 설정되는 경우에는, TG 목적지가 LEN 노드에 대한 것인지 또는 끝 노드에 대한 것인지를 지정함에 주의하십시오.

local_topology_detail.tg_num
TG와 연관된 번호.

local_topology_detail.dlc_data.length
VRN에 대한 연결의 DLC 주소 길이(**dest_type**이 AP_VRN이 아니면 0으로 설정됨).

local_topology_detail.dlc_data.address
VRN에 대한 연결의 DLC 주소.

local_topology_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

local_topology_detail.status
TG의 상태를 지정합니다. 이는 다음 값 중 하나 또는 그 이상일 수 있습니다.

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

local_topology_detail.tg_chars
TG 특성(35페이지의 『DEFINE_CN』 참조).

local_topology_detail.cp_cp_session_active
국지 노드의 회선경합 성공 CP-CP 세션이 활동중인지의 여부(AP_NO 또는 AP_YES)를 지정합니다.

local_topology_detail.branch_link_type
BrNN 만. 이 TG의 분기 링크 유형. 이는 다음 중 하나로 설정됩니다.

AP_UPLINK
이 링크가 업링크입니다.

AP_DOWNLINK
링크가 EN에 대한 다운링크입니다.

AP_DOWNLINK_TO_BRNN
TG가 EN face를 보여주고 있는 BrNN에 대한 다운링크입니다.

AP_OTHERLINK
이 링크는 기타 링크입니다.

기타 노드 유형: 이 필드는 의미가 없으며 항상 AP_BRNN_NOT_SUPPORTED로 설정됩니다.

local_topology_detail.branch_tg
NN 만. TG가 분기 TG인지의 여부를 지정합니다.

AP_NO
TG가 분기 TG가 아닙니다.

AP_YES
TG가 분기 TG입니다.

기타 노드 유형: 이 필드는 의미가 없으며 항상 AP_NO로 설정됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TG

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LS

QUERY_LS는 노드에서 정의된 링크 스테이션에 관한 정보의 리스트를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_LS에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LS에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **ls_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **ls_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이에 따라 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라).
AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

반환되는 링크 스테이션들의 리스트는 그들이 속해 있는 포트의 이름에 의해 필터링될 수 있습니다. 이 경우, **port_name** 필드가 설정되어야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.)

VCB 구조

형 D 1

```
typedef struct query_ls
{
   unsigned short  opcode;             /* verb operation code           */
   unsigned char   attributes;         /* Verb attributes               */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   ls_name[8];    /* name of link station          */
        unsigned char   port_name[8];  /* name of link station          */
} QUERY_LS;

typedef struct ls_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   ls_name[8];    /* link station name             */
   unsigned char   description[RD_LEN];
                                       /* resource description          */
   unsigned char   dlc_type;           /* DLC type                      */
        unsigned char   state;         /* link station state            */
        unsigned short  act_sess_count;     /* currently active sess count    */
        unsigned char   det_adj_cp_name[17];
                                            /* determined adj CP name        */
```

```
        unsigned char   det_adj_cp_type;    /* determined adj node type       */
   unsigned char   port_name[8];            /* port name                      */
        unsigned char   adj_cp_name[17];    /* adjacent CP name               */
        unsigned char   adj_cp_type;        /* adjacent node type             */
} LS_SUMMARY;

typedef struct ls_detail
{
   unsigned short  overlay_size;            /* size of this entry             */
        unsigned char   ls_name[8];         /* link stations name             */
        LS_DET_DATA     det_data;           /* determined data                */
        LS_DEF_DATA     def_data;           /* defined data                   */
} LS_DETAIL;

typedef struct ls_det_data
{
   unsigned short  act_sess_count;          /* curr active sessions count     */
   unsigned char   dlc_type;                /* DLC type                       */
        unsigned char   state;              /* link station state             */
        unsigned char   sub_state;          /* link station sub state         */
        unsigned char   det_adj_cp_name[17];
                                            /* adjacent CP name               */
        unsigned char   det_adj_cp_type;    /* adjacent node type             */
   unsigned char   dlc_name[8];             /* name of DLC                    */
        unsigned char   dynamic;            /* is LS is dynamic ?             */
        unsigned char   migration;          /* supports migration partners    */
        unsigned char   tg_num;             /* TG number                      */
        LS_STATS        ls_stats;           /* link station statistics        */
        unsigned long   start_time;         /* time LS started                */
        unsigned long   stop_time;          /* time LS stopped                */
        unsigned long   up_time;            /* total time LS active           */
        unsigned long   current_state_time; /* time in current state          */
        unsigned char   deact_cause;        /* deactivation cause             */
        unsigned char   hpr_support;        /* TG HPR support                 */
        unsigned char   anr_label[2];       /* local ANR label                */
        unsigned char   hpr_link_lvl_error; /* HPR link-level error           */
        unsigned char   auto_act;           /* auto activate                  */
        unsigned char   ls_role;            /* link station role              */
        unsigned char   reserva;            /* reserved                       */
        unsigned char   node_id[4];         /* determined node id             */
        unsigned short  active_isr_count;   /* currently active ISR sessions  */
        unsigned short  active_lu_sess_count;
                                            /* active LU-LU session count     */
        unsigned short  active_sscp_sess_count;
                                            /* active SSCP session count      */
        ANR_LABEL       reverse_anr_labe;   /* reverse ANR label              */
        unsigned short  max_send_btu_size;  /* negotiated max BTU length      */
        unsigned char   brnn_link_type;     /* branch link type               */
        unsigned char   adj_cp_is_brnn;     /* adjacent CP is a BrNN          */
        unsigned char   reservb[6];         /* reserved                       */
} LS_DET_DATA;

typedef struct anr_label
{
        unsigned short  length;             /* ANR label length               */
        unsigned short  reserv;             /* reserved                       */
        unsigned char   label[MAX_ANR_LABEL_SIZE];
                                            /* ANR label                      */
} ANR_LABEL;

typedef struct ls_def_data
{
   unsigned char   description[RD_LEN];
                                            /* resource description           */
        unsigned char   port_name[8];       /* name of associated port        */
        unsigned char   adj_cp_name[17];    /* adjacent CP name               */
        unsigned char   adj_cp_type;        /* adjacent node type             */
        LINK_ADDRESS    dest_address;       /* destination address            */
        unsigned char   auto_act_supp;      /* auto-activate supported        */
        unsigned char   tg_number;          /* Pre-assigned TG number         */
        unsigned char   limited_resource;   /* limited resource               */
        unsigned char   solicit_sscp_sessions;
```

```
                                        /* solicit SSCP sessions          */
        unsigned char   pu_name[8];     /* Local PU name (reserved if     */
                                        /* solicit_sscp_sessions is set   */
                                        /* to AP_NO)                      */
        unsigned char   disable_remote_act; /* disable remote activation flag */
        unsigned char   dspu_services;  /* Services provided for          */
                                        /* downstream PU                  */
        unsigned char   dspu_name[8];   /* Downstream PU name (reserved   */
                                        /* if dspu_services is set to     */
                                        /* AP_NONE or AP_DLUR)            */
        unsigned char   dlus_name[17];  /* DLUS name if dspu_services     */
                                        /* is set to AP_DLUR              */
        unsigned char   bkup_dlus_name[17]; /* Backup DLUS name if        */
                                        /* dspu_services is set           */
                                        /* to AP_DLUR                     */
        unsigned char   hpr_supported;  /* does the link support HPR?     */
        unsigned char   hpr_link_lvl_error; /* does the link support HPR  */
                                        /* link-level error recovery?     */
        unsigned short  link_deact_timer;   /* HPR link deactivation timer    */
    unsigned char   reserv1;            /* reserved                       */
        unsigned char   default_nn_server;  /* Use as default LS to NN server */
        unsigned char   ls_attributes[4];   /* LS attributes                  */
        unsigned char   adj_node_id[4];     /* adjacent node ID               */
        unsigned char   local_node_id[4];   /* local node ID                  */
        unsigned char   cp_cp_sess_support; /* CP-CP session support          */
        unsigned char   use_default_tg_chars;
                                        /* Use default tg_chars           */
        TG_DEFINED_CHARS tg_chars;      /* TG characteristics             */
    unsigned short  target_pacing_count;
                                        /* target pacing count            */
    unsigned short  max_send_btu_size;  /* max send BTU size              */
        unsigned char   ls_role;        /* link station role to use       */
                                        /* on this link                   */
        unsigned char   max_ifrm_rcvd;  /* max number of I-frames rcvd    */
        unsigned short  dlus_retry_timeout; /* DLUS retry timeout         */
        unsigned short  dlus_retry_limit;   /* DLUS retry limit           */
    unsigned char   conventional_lu_compression;
                                        /* Data compression requested for */
                                        /* conventional LU sessions       */
    unsigned char   conventional_lu_cryptography;
                                        /* Cryptography required for      */
                                        /* conventional LU sessions       */
    unsigned char   reserv3;            /* reserved                       */
        unsigned char   retry_flags;    /* conditions for automatic       */
                                        /* retries                        */
        unsigned short  max_activation_attempts;
                                        /* how many automatic retries:    */
    unsigned short  activation_delay_timer;
                                        /* delay between automatic        */
                                        /* retries                        */
        unsigned char   branch_link_type;   /* branch link type           */
        unsigned char   adj_brnn_cp_support;/* adjacent BrNN CP support   */
        unsigned char   reserv4[20];    /* reserved                       */
        unsigned short  link_spec_data_len; /* length of link specific data   */
} LS_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;         /* length                         */
        unsigned short  reserve1;       /* reserved                       */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                        /* address                        */
} LINK_ADDRESS;

typedef struct link_spec_data
{
       unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_DATA;
```

```
typedef struct tg_defined_chars
{
        unsigned char   effect_cap;         /* Effective capacity            */
        unsigned char   reserve1[5];        /* Reserved                      */
        unsigned char   connect_cost;       /* Connection Cost               */
        unsigned char   byte_cost;          /* Byte cost                     */
        unsigned char   reserve2;           /* Reserved                      */
        unsigned char   security;           /* Security                      */
        unsigned char   prop_delay;         /* Propagation delay             */
        unsigned char   modem_class;        /* Modem class                   */
        unsigned char   user_def_parm_1;    /* User-defined parameter 1      */
        unsigned char   user_def_parm_2;    /* User-defined parameter 2      */
        unsigned char   user_def_parm_3;    /* User-defined parameter 3      */
} TG_DEFINED_CHARS;

typedef struct ls_stats
{
        unsigned long   in_xid_bytes;       /* number of XID bytes received  */
        unsigned long   in_msg_bytes;       /* num message bytes received    */
        unsigned long   in_xid_frames;      /* num XID frames received       */
        unsigned long   in_msg_frames;      /* num message frames received   */
        unsigned long   out_xid_bytes;      /* num XID bytes sent            */
        unsigned long   out_msg_bytes;      /* num message bytes sent        */
        unsigned long   out_xid_frames;     /* num XID frames sent           */
        unsigned long   out_msg_frames;     /* num message frames sent       */
   unsigned long   in_invalid_sna_frames;
                                            /* num invalid frames received   */
        unsigned long   in_session_control_frames;
                                            /* num control frames received   */
        unsigned long   out_session_control_frames;
                                            /* num control frames sent       */
        unsigned long   echo_rsps;          /* response from adj LS count    */
        unsigned long   current_delay;      /* time taken for last test sig  */
        unsigned long   max_delay;          /* max delay by test signal      */
        unsigned long   min_delay;          /* min delay by test signal      */
        unsigned long   max_delay_time;     /* time since longest delay      */
        unsigned long   good_xids;          /* successful XID on LS count    */
        unsigned long   bad_xids;           /* unsuccessful XID on LS count  */
} LS_STATS;
```

VCB 구조

형 D 0(이전 레벨)

```
typedef struct ls_det_data
{
   unsigned short  act_sess_count;          /* curr active sessions count    */
   unsigned char   dlc_type;                /* DLC type                      */
        unsigned char   state;              /* link station state            */
        unsigned char   sub_state;          /* link station sub state        */
        unsigned char   det_adj_cp_name[17];
                                            /* adjacent CP name              */
        unsigned char   det_adj_cp_type;    /* adjacent node type            */
   unsigned char   dlc_name[8];             /* name of DLC                   */
        unsigned char   dynamic;            /* is LS is dynamic ?            */
        unsigned char   migration;          /* supports migration partners   */
        unsigned char   tg_num;             /* TG number                     */
        LS_STATS        ls_stats;           /* link station statistics       */
        unsigned long   start_time;         /* time LS started               */
        unsigned long   stop_time;          /* time LS stopped               */
        unsigned long   up_time;            /* total time LS active          */
        unsigned long   current_state_time; /* time in current state         */
        unsigned char   deact_cause;        /* deactivation cause            */
        unsigned char   hpr_support;        /* TG HPR support                */
        unsigned char   anr_label[2];       /* local ANR label               */
        unsigned char   hpr_link_lvl_error; /* HPR link-level error          */
        unsigned char   auto_act;           /* auto activate                 */
        unsigned char   ls_role;            /* link station role             */
        unsigned char   reserva;            /* reserved                      */
```

```
        unsigned char   node_id[4];         /* determined node id            */
        unsigned short  active_isr_count;   /* currently active ISR sessions */
        unsigned char   reservb[30];        /* reserved                      */
} LS_DET_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LS

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 형식 1 버전을 지정하려면, 이 필드를 1로 설정하십시오. 0으로 설정되면, 프로그램이 형식 0 LS_DET_DATA 구조체를 반환합니다.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **ls_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

ls_name
링크 스테이션명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

port_name
포트명 필터. 이는 모두 0으로 설정되거나 오른쪽은 EBCDIC 공백으로 채워지면서 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 설정되어야 합니다. 이 필드가 설정되면 이 포트에 속하는 링크 스테이션들만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

ls_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

ls_summary.ls_name
링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

ls_summary.description
자원 설명(DEFINE_LS에서 정의된 대로). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

ls_summary.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

DEFINE_DLC 명령에서 새로운 유형을 지정함으로써 추가 DLC 유형이 정의될 수 있습니다. 더 자세한 정보는 51페이지의 『DEFINE_DLC』를 참조하십시오.

ls_summary.state
링크 스테이션의 상태. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

ls_summary.act_sess_count
링크를 사용하는 총 활동중인 세션 수(끝점과 중계 모두).

ls_summary.det_adj_cp_name
링크 활성화 동안 결정되는 전체 정식 17 바이트의 인접 CP명. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) LS가 활동중이 아니면 널(null)일 것입니다.

ls_summary.adj_cp_type이 AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE 중 하나가 아니면, 이 필드가 예약됩니다.

ls_summary.det_adj_cp_type
링크 활성화 동안 결정되는 인접 노드의 유형. 이는 다음 값 중 하나입니다.

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

LS가 활동중이 아니면 AP_LEARN_NODE일 것입니다.

ls_summary.adj_cp_type이 AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE 중 하나가 아니면, 이 필드가 예약됩니다.

ls_summary.port_name
이 링크 스테이션과 연관된 포트의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

ls_summary.adj_cp_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 전체 정식 17 바이트의 인접 제어점(CP) 이름(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다). 암시적 링크의 경우에는 이것이 널(null)일 것입니다.

ls_summary.adj_cp_type
인접 노드의 유형. 이는 다음 값 중 하나입니다.

AP_END_NODE
AP_NETWORK_NODE
AP_APPN_NODE
AP_BACK_LEVEL_LEN__NODE
AP_HOST_XID3
AP_HOST_XID0
AP_DSPU_XID
AP_DSPU_NOXID

ls_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

ls_detail.ls_name
링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

ls_detail.det_data.act_sess_count
링크를 사용하는 총 활동중인 세션 수(끝점과 중계 모두).

ls_detail.det_data.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC

AP_SDLC
AP_TWINAX
AP_X25

DEFINE_DLC verb에서 새로운 유형을 지정함으로써 추가 DLC 유형이 정의될 수 있습니다. 더 자세한 정보는 51페이지의 『DEFINE_DLC』를 참조하십시오.

ls_detail.det_data.state
링크 스테이션의 상태. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

ls_detail.det_data.sub_state
이 필드는 링크 스테이션의 상태에 관한 보다 상세한 정보를 제공합니다. 이 필드는 다음 값 중 하나로 설정됩니다.

AP_SENT_CONNECT_OUT
AP_PENDING_XID_EXCHANGE
AP_SENT_ACTIVATE_AS
AP_SENT_SET_MODE
AP_ACTIVE
AP_SENT_DEACTIVATE_AS_ORDERLY
AP_SENT_DISCONNECT
AP_WAITING_STATS
AP_RESET

ls_detail.det_data.det_adj_cp_name
링크 활성화 동안 결정되는 전체 정식 17 바이트의 인접 제어점(CP) 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

ls_summary.adj_cp_type이 AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE 중 하나가 아니면, 이 필드가 예약됩니다.

ls_detail.det_data.det_adj_cp_type
링크 활성화 동안 결정되는 인접 노드의 유형. 이는 다음 값 중 하나입니다.

AP_END_NODE
AP_NETWORK_NODE
AP_LEARN_NODE
AP_VRN

ls_summary.adj_cp_type이 AP_NETWORK_NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE 중 하나가 아니면, 이 필드가 예약됩니다.

ls_detail.det_data.dlc_name
DLC의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

ls_detail.det_data.dynamic
링크가 명시적으로 (DEFINE_LS 명령에 의해) 정의되었는지, 암시적으로 또는 동적으로(인접 노드로부터의 연결 요청에 대한 응답으로, 또는 연결 네트워크를 통해 다른 노드에 동적으로 연결하기 위해) 정의되었는지를 지정합니다. 이는 AP_YES 또는 AP_NO일 수 있습니다.

ls_detail.det_data.migration
인접 노드가 이주 레벨 노드(하위 네트워크(LEN) 노드와 같은)인지, 또는 전체 APPN 네트워크 노드나 끝 노드인지를 지정합니다(AP_YES, AP_NO 또는 AP_UNKNOWN).

ls_detail.det_data.tg_num
TG와 연관된 번호.

ls_detail.det_data.ls_stats.in_xid_bytes
이 링크 스테이션에서 수신된 총 XID(교환 식별자) 바이트 수.

ls_detail.det_data.ls_stats.in_msg_bytes
이 링크 스테이션에서 수신된 총 데이터 바이트 수.

ls_detail.det_data.ls_stats.in_xid_frames
이 링크 스테이션에서 수신된 총 XID(교환 식별자) 프레임 수.

ls_detail.det_data.ls_stats.in_msg_frames
이 링크 스테이션에서 수신된 총 데이터 프레임 수.

ls_detail.det_data.ls_stats.out_xid_bytes
이 링크 스테이션에서 송신된 총 XID(교환 식별자) 바이트 수.

ls_detail.det_data.ls_stats.out_msg_bytes
이 링크 스테이션에서 송신된 총 데이터 바이트 수.

ls_detail.det_data.ls_stats.out_xid_frames
이 링크 스테이션에서 송신된 총 XID(교환 식별자) 프레임 수.

ls_detail.det_data.ls_stats.out_msg_frames
이 링크 스테이션에서 송신된 총 데이터 프레임 수.

ls_detail.det_data.ls_stats.in_invalid_sna_frames
이 링크 스테이션에서 수신된 총 SNA 부정확 프레임 수.

ls_detail.det_data.ls_stats.in_session_control_frames
이 링크 스테이션에서 수신된 총 세션 제어 프레임 수.

ls_detail.det_data.ls_stats.out_session_control_frames
이 링크 스테이션에서 송신된 총 세션 제어 프레임 수.

ls_detail.det_data.ls_stats.echo_rsps
인접 노드로부터 수신된 반향 응답의 수. 인접 노드로의 전달 지연을 측정하기 위해 주기적으로 반향 요청이 보내집니다.

ls_detail.det_data.ls_stats.current_delay
이 링크 스테이션으로부터 인접 링크 스테이션으로 최종 검사 신호가 송신되고 반환되는 데 걸린 시간(밀리세컨드(1000분의 1초) 단위).

ls_detail.det_data.ls_stats.max_delay
이 링크 스테이션으로부터 인접 링크 스테이션으로 최종 검사 신호가 송신되고 반환되는 데 걸린 최장 시간(밀리세컨드(1000분의 1초) 단위).

ls_detail.det_data.ls_stats.min_delay
이 링크 스테이션으로부터 인접 링크 스테이션으로 최종 검사 신호가 송신되고 반환되는 데 걸린 최단 시간(밀리세컨드(1000분의 1초) 단위).

ls_detail.det_data.ls_stats.max_delay_time
최장 지연이 발생했을 때의 시스템 시작 후 시간(100분의 1초 단위).

ls_detail.det_data.ls_stats.good_xids
이 링크 스테이션이 시작된 이래 여기서 발생한 총 성공 XID 교환 수.

ls_detail.det_data.ls_stats.bad_xids
이 링크 스테이션이 시작된 이래 여기서 발생한 총 비성공 XID 교환 수.

ls_detail.det_data.start_time
링크 스테이션이 마지막으로 활성화되었을 때(즉, 모드 설정 명령이 완료했을 때)의 시스템 시작 이래 시간(100분의 1초 단위).

ls_detail.det_data.stop_time
링크 스테이션이 마지막으로 활성종료되었을 때의 시스템 시작 이래 시간(100분의 1초 단위).

ls_detail.det_data.up_time
시스템 시작 이래 이 링크 스테이션이 활동중이었던 총 시간(100분의 1초 단위).

ls_detail.det_data.current_state_time
이 링크 스테이션이 현재 상태에 있었던 총 시간(100분의 1초 단위).

ls_detail.det_data.deact_cause
링크 스테이션의 마지막으로 활성종료 원인. 다음 값 중 하나로 설정됩니다.

AP_NONE
링크 스테이션이 활성종료된 적이 없습니다.

AP_DEACT_OPER_ORDERLY
운영요원으로부터의 오래된 STOP 명령의 결과로 링크 스테이션이 활성종료되었습니다.

AP_DEACT_OPER_IMMEDIATE
운영요원으로부터의 즉시 STOP 명령의 결과로 링크 스테이션이 활성종료되었습니다.

AP_DEACT_AUTOMATIC
가령, 링크 스테이션을 사용하는 세션이 더 이상 없는 이유로 링크 스테이션이 자동으로 활성종료되었습니다.

AP_DEACT_FAILURE
장애로 인해 링크 스테이션이 활성종료되었습니다.

ls_detail.det_data.hpr_support
이 TG상에서 지원되는 고성능 경로지정(HPR)의 레벨(AP_NONE, AP_BASE 또는 AP_RTP)로 국지 및 인접 노드의 능력을 고려에 넣음.

ls_detail.det_data.anr_label
국지 링크에 할당된 HPR 자동 네트워크 경로지정(ANR) 레이블.

ls_detail.det_data.hpr_link_lvl_error
링크상의 HPR 트래픽에 링크 레벨 오류 복구가 사용되고 있는지의 여부를 지정합니다.

ls_detail.def_data.auto_act
링크가 현재 원격 활성화나 요구가 있을때의 활성화를 허용하는지의 여부를 지정합니다. 다음 값이 반환될 수 있습니다.

AP_AUTO_ACT
링크가 국지 노드에 의해 요구가 있을 때 활성화될 수 있습니다.

AP_REMOTE_ACT
링크가 원격 노드에 의해 활성화될 수 있습니다.

ls_detail.det_data.ls_role
이 링크 스테이션에 대한 링크 스테이션 역할. 초기에는 링크 스테이션을 위해 정의된 링크 스테이션 역할로 설정됩니다. 정의된 역할이 조정가능하면, XID 교환 동안 이 값이 조정된 역할(1차나 2차)로 바뀌고, 링크가 활성종료될 때 다시 조정가능으로 되돌아갑니다.

AP_LS_NEG
링크 스테이션 역할이 조정가능합니다.

AP_LS_PRI
링크 스테이션 역할이 1차입니다.

AP_LS_SEC
링크 스테이션 역할이 2차입니다.

ls_detail.det_data.node_id
XID 교환 동안 인접 노드로부터 수신된 노드 ID. 이는 4 바이트의 16진 문자열입니다.

ls_detail.det_data.active_isr_count
링크를 사용하는 활동중 중개 세션의 수.

ls_detail.det_data.active_lu_sess_count
링크를 사용하는 활동중 LU-LU 세션의 수.

ls_detail.det_data.active_sscp_sess_count
링크를 사용하는 활동중 LU-SSCP 및 PU-SSCP 세션의 수.

ls_detail.det_data.reverse_anr_label.length
링크 스테이션을 위한 역 자동 네트워크 경로지정(anr) 레이블의 길이. 링크가 HPR을 지원하지 않거나 레이블을 알 수 없으면, 이 필드가 0으로 됩니다.

ls_detail.det_data.local_address
이 링크 스테이션의 국지 주소.

ls_detail.det_data.max_send_btu_size
인접 노드와의 조정에 의해 결정된 대로, 이 링크상에서 송신될 수 있는 최대 BTU 크기. 링크 활성화가 아직 시도되지 않았으면, 0이 반환됩니다.

ls_detail.det_data.brnn_link_type
BrNN 만. 분기 링크 유형. 이는 다음 중 하나입니다.

AP_UPLINK
이 링크가 업링크입니다.

AP_DOWNLINK
링크가 다운링크입니다.

AP_OTHERLINK
이 링크는 기타 링크입니다.

AP_UNKNOWN_LINK_TYPE
이 링크는 기타 링크입니다.

AP_BRNN_NOT_SUPPORTED
이 링크는 PU 2.0 트래픽만 지원합니다.

기타 노드 유형: 이 필드는 의미가 없으며 항상 AP_BRNN_NOT_SUPPORTED로 설정됩니다.

ls_detail.det_data.adj_cp_is_brnn
모든 노드 유형: 인접 노드가 BrNN인지의 여부를 지정합니다.

AP_UNKNOWN
인접 노드가 BrNN인지의 여부를 알 수 없습니다.

AP_NO
인접 노드가 BrNN이 아닙니다.

AP_YES
인접 노드가 BrNN입니다.

ls_detail.def_data.description
자원 설명(DEFINE_LS에서 정의된 대로). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

ls_detail.def_data.port_name
이 링크 스테이션과 연관된 포트의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 링크가 VRN인 경우, 이 필드는 VRN에 연결하기 위해 사용된 실제 포트의 이름을 지정합니다(DEFINE_CN 명령에서 지정된 것과 같이).

ls_detail.def_data.adj_cp_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성되는 전체 정식 17 바이트의 인접 제어점(CP) 이름(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다). 이는 **back_lvl_len_end_node**가 AP_NO로 설정되지 않거나, 링크 스테이션과 연관된 포트가 교환식으로 정의되는 경우에 정의합니다.

ls_detail.def_data.adj_cp_type
인접 노드 유형.

AP_NETWORK_NODE
노드가 APPN 네트워크 노드임을 지정합니다.

AP_END_NODE
노드가 APPN 끝 노드이거나 위 레벨 LEN 노드임을 지정합니다.

AP_APPN_NODE
노드가 APPN 네트워크 노드, APPN 끝 노드 또는 위 레벨 LEN 노드임을 지정합니다. 노드 유형은 XID 교환 동안 알게 될 것입니다.

AP_BACK_LEVEL_LEN_NODE
노드가 이전 레벨 LEN 노드임을 지정합니다.

AP_HOST_XID3
노드가 호스트이며 노드 연산자 기능이 형식 3 XID를 가진 노드로부터의 폴링 XID에 응답한다고 지정합니다.

AP_HOST_XID0
노드가 호스트이며 노드 연산자 기능이 형식 0 XID를 가진 노드로부터의 폴링 XID에 응답한다고 지정합니다.

AP_DSPU_XID
노드가 다운스트림 PU이며 노드 연산자 기능이 링크 활성화에서 XID 교환을 포함한다고 지정합니다.

AP_DSPU_NOXID
노드가 다운스트림 PU이며 노드 연산자 기능이 링크 활성화에서 XID 교환을 포함하지 않는다고 지정합니다.

주: VRN에 대한 링크 스테이션이 항상 동적이며 따라서 정의되지 않습니다.

ls_detail.def_data.dest_address.length
인접 노드상의 목적지 링크 스테이션 주소 길이.

ls_detail.def_data.dest_address.address
인접 노드상의 링크 스테이션의 목적지 주소.

ls_detail.def_data.auto_act_supp
START_LS 명령에 의해 시작되고 STOP_LS에 의해 종료된 후 링크가 자동으로 활성화될 것인지의 여부를 지정합니다(AP_YES 또는 AP_NO).

ls_detail.def_data.tg_number
사전 지정된 TG 번호(1-20의 범위). 링크가 활성화될 때 링크를 표현하는 데 이 번호가 사용됩니다. 0은 TG 번호가 사전지정되지 않고 링크가 활성화될 때 조정됨을 나타냅니다.

ls_detail.def_data.limited_resource
링크를 사용하고 있는 세션이 없을 때 이 링크 스테이션이 활성종료될 것인지의 여부를 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_NO
링크가 제한된 자원이 아니며 자동으로 활성종료되지 않을 것입니다.

AP_YES 또는 AP_NO_SESSIONS
링크가 제한된 자원이며 그것을 사용하는 활동중인 세션이 없을 때 자동으로 활성종료될 것입니다.

AP_INACTIVITY
링크가 제한된 자원이며 그것을 사용하는 활동중인 세션이 없거나 **link_deact_timer** 필드에 의해 지정된 시간 동안 링크상에서 흐른 데이터가 없는 경우 자동으로 활성종료될 것입니다.

ls_detail.def_data.solicit_sscp_sessions
AP_YES는 호스트에게 SSCP와 국지 제어점(CP) 및 종속 LU간의 세션을 초기화하도록 요청합니다. AP_NO는 이 링크에서 SSCP와의 어떤 세션도 요청하지 않습니다.

ls_detail.def_data.pu_name
solicit_sscp_sessions이 AP_YES로 설정된 경우 이 링크를 사용할 국지 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.
solicit_sscp_sessions가 AP_NO로 설정되면, 이 필드가 예약됩니다.

ls_detail.def_data.disable_remote.act

이 링크의 원격 활성화가 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

ls_detail.def_data.dspu_services

solicit_sscp_sessions가 AP_NO로 설정되면 국지 노드가 이 링크를 통해 다운스트림 PU에 제공하는 서비스들을 지정합니다. 이는 다음 중 하나로 설정됩니다.

AP_PU_CONCENTRATION

국지 노드가 다운스트림 PU를 위한 PU 집중(concentration)을 제공할 것입니다.

AP_DLUR

국지 노드가 다운스트림 PU를 위한 DLUR 서비스를 제공할 것입니다.

AP_NONE

국지 노드가 다운스트림 PU를 위한 어떤 서비스도 제공하지 않을 것입니다.

solicit_sscp_sessions가 AP_YES로 설정되면, 이 필드가 예약됩니다.

ls_detail.def_data.dspu_name

다운스트림 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 **solicit_sscp_sessions**가 AP_NO로 설정되는 경우에만 유효합니다.

ls_detail.def_data.dlus_name

다운스트림 노드로의 링크가 활성화될 때 DLUR이 SSCP 서비스를 청하는 DLUS의 이름. 이는 모두 0으로 설정되거나 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정되어야 합니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 필드가 모두 0으로 설정되면, 글로벌 생략시 DLUS(DEFINE_DLUR_DEFAULTS 명령에 의해 정의되었으면)가 청해집니다. **dlus_name**이 0으로 설정되고 글로벌 생략시 DLUS가 없으면, 링크가 활성화될 때 DLUR이 SSCP 접속을 초기화하지 않을 것입니다. **dspu_services**가 AP_DLUR로 설정되면, 이 필드가 예약됩니다.

ls_detail.def_data.bkup_dlus_name

다운스트림 PU를 위한 백업으로 서비스하는 DLUS 노드의 이름. 이는 모두 0으로 설정되거나 EBCDIC 점으로 연결되고, 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 17 바이트의 문자열로 설정되어야 합니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 필드가 모두 0으로 설정되면, 글로벌 백업 생략시 DLUS(DEFINE_DLUR_DEFAULTS 명령에 의해 정의되었으면)가 이 PU를 위한 백업으로 사용됩니다. **dspu_services**가 AP_DLUR로 설정되면, 이 필드가 예약됩니다.

ls_detail.def_data.hpr_supported
이 링크에서 HPR이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

ls_detail.def_data.hpr_link_lvl_error
이 링크에서 HPR 링크 레벨 오류 복구 타워가 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). **hpr_supported**가 AP_NO로 설정되면 매개변수가 예약됨에 주의하십시오.

ls_detail.def_data.link_deact_timer
제한된 자원 링크 활성종료 타이머(초 단위).

limited_resource가 AP_YES나 AP_NO_SESSIONS로 설정되면, 이 타이머의 주기 동안 링크를 통과하는 데이터가 없고 링크를 사용하는 세션이 없는 경우 링크가 자동으로 활성종료됩니다.

limited_resource가 AP_INACTIVITY로 설정되면 이 타이머 주기 동안 링크를 통과하는 데이터가 없는 경우 링크가 자동으로 활성종료됩니다.

ls_detail.def_data.default_nn_server
네트워크 노드 서버로의 CP-CP 세션을 지원하기 위해 링크가 끝 노드에 의해 자동으로 활성화될 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드가 영향을 미치려면 링크가 CP-CP 세션을 지원하도록 정의되어야 합니다.

ls_detail.def_data.ls_attributes
인접 노드에 관한 정보를 더 지정합니다.

ls_detail.def_data.ls_attributes[0]
호스트 유형.

AP_SNA
표준 SNA 호스트.

AP_FNA
FNA(VTAM-F) 호스트.

AP_HNA
HNA 호스트.

def_data.ls_attributes[1]
이는 한 비트 필드입니다. 값 AP_NO를 취하거나 다음 값 중 어느 것이든 취할 수 있습니다.

AP_SUPPRESS_CP_NAME
이전 레벨 LEN 노드에 대한 링크를 위한 네트워크명 CV 억제 옵션. 이 비트가 설정되면, 인접 노드와의 XID 교환에 네트워크명 CV가 포함되지 않습니다.(**adj_cp_type**이 AP_BACK_LEVEL_LEN_NODE나 AP_HOST_XID3으로 설정되지 않는 한, 이 비트는 무시됩니다.)

AP_REACTIVATE_ON_FAILURE
링크가 활성화된 후 실패하면, 퍼스널 통신이나 통신 서버가 그 링크를 재활성화하려고 시도할 것입니다. 재활성화 시도가 실패하면, 링크가 활동중이 아닌 상태로 남아 있을 것입니다.

AP_USE_PU_NAME_IN_XID_CVS
인접 노드가 호스트로 정의되거나 APPN 노드에 대한 링크에서 **solicit_sscp_sessions**가 AP_YES로 설정되고, AP_SUPPRESS_CP_NAME 비트가 설정되지 않으면, 형식 3 XID들에서 송신된 네트워크명 CV들에 있는 전체 정식 CP명이 CP의 네트워크 ID로 전체 정식이 된 **def_data.pu_name**에서 제공된 이름으로 대체됩니다.

ls_detail.def_data.adj_node_id
인접 노드의 정의된 노드 ID.

ls_detail.def_data.local_node_id
이 링크 스테이션상의 XID들에서 송신된 노드 ID. 이는 4 바이트의 16진 문자열입니다. 이 필드가 0으로 설정되면, XID 교환에서 **node_id**가 사용됩니다. 이 필드가 0이 아니면, 이 LS상의 XID 교환을 위한 값을 대체합니다.

ls_detail.def_data.cp_cp_sess_support
CP-CP 세션이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

ls_detail_def_data.use_default_tg_chars
DEFINE_LS상에서 제공된 TG 특성이 DEFINE_PORT 상에서 제공된 생략시 특성을 위해 버려졌는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 암시적 링크에는 이 필드가 적용되지 않습니다.

ls_detail.def_data.tg_chars
TG 특성(35페이지의 『DEFINE_CN』 참조).

ls_detail.def_data.target_pacing_count
이 TG상의 BIND들을 위한 적절한 페이싱 창 크기를 나타내는 1과 32 767을 포함하여 이들 사이에 있는 숫자 값. 수는 고정 바인드 페이싱이 수행되고 있는 경우에만 유효합니다. 퍼스널 통신이나 통신 서버는 현재 이 값을 사용하고 있지 않음에 주의하십시오.

ls_detail.def_data.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

ls_detail.def_data.ls_role
이 링크 스테이션이 취해야 하는 링크 스테이션 역할. 조정가능 1차 또는 2차의 역할을 선택하기 위해 AP_LS_NEG, AP_LS_PRI 또는 AP_LS_SEC 중 어느 것이든 가능합니다. DEFINE_PORT 명령에서 구성된 값을 선택하기 위해 이 필드가 AP_USE_PORT_DEFAULTS로 설정될 수도 있습니다.

ls_detail.def_data.max_ifrm_rcvd

승인전에 XID 송신자에 의해 수신될 수 있는 최대 I-Frames 수. DEFINE_PORT로부터의 생략시 값이 사용되어야 한다면 0으로 설정하십시오.

ls_detail.def_data.dlus_retry_timeout

Is_detail.def_data.dlus_name과 **Is_detail.def_data.bkup_dlus_name** 필드에서 지정된 DLUS에 접속하기 위한 두 번째와 차후의 시도간의 간격으로 초 단위. 초기 시도와 첫번째 재시도간의 간격은 항상 1초입니다. 0이 지정되면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. **def_data.dspu_services**가 AP_DLUR로 설정되지 않으면, 이 필드가 무시됩니다.

ls_detail.def_data.dlus_retry_limit

Is_detail.def_data.dlus_name과 **Is_detail.def_data.bkup_dlus_name** 필드에서 지정된 최초 DLUS에 접속 실패후 최대 재시도 회수. 0이 지정되면, DEFINE_DLUR_DEFAULTS를 통해 구성된 생략시 값이 사용됩니다. X'FFFF'가 지정되면, 프로그램이 무기한 재시도합니다. **def_data.dspu_services**가 AP_DLUR로 설정되지 않으면, 이 필드가 무시됩니다.

ls_detail.def_data.link_spec_data_len

초기화 동안 변경되지 않은 상태로 링크 스테이션에 건네진 데이터의 채워지지 않은 바이트 단위의 길이. 데이터는 LS_DETAIL 구조체로 연결됩니다. 이 데이터는 4 바이트의 경계에서 끝으로 채워질 것입니다.

ls_detail.def_data.convention_lu_compression

이 링크상의 세션을 위해 데이터 압축이 요청되는지의 여부를 지정합니다. 이 필드는 LU 0에서 3으로의 트래픽을 운반하는 링크에 대해서만 유효함에 주의하십시오.

AP_NO

국지 노드는 이 링크를 통한 전통적 LU 데이터 흐름을 압축하거나 압축해제해서는 안됩니다.

AP_YES

호스트가 압축을 요청하면 이 링크상의 전통적 LU 세션에 대해 데이터 압축이 사용가능해야 합니다.

ls_detail.def_data.convention_lu_cryptography

전통적 LU 세션을 위해 세션 레벨 암호화가 요청되는지의 여부를 지정합니다. 이 필드는 전통적 LU 트래픽을 운반하는 링크에 대해서만 적용됩니다.

AP_NONE

프로그램에 의해 세션 레벨 암호화가 수행되지 않습니다.

AP_MANDATORY

LU에 반입 키가 사용가능하면 프로그램에 의해 필수 세션 레벨 암호화가 수행됩니다. 그렇지 않으면, LU를 사용하는 응

용프로그램에 의해 수행되어야 합니다.(PU 집중(concentration) 이면, 다운스트림 LU에 의해 수행됩니다.)

AP_OPTIONAL
이 값은 사용된 암호화가 세션당 기준으로 호스트 응용프로그램에 의해 가동되게 합니다. 호스트가 이 PU에 의존적인 세션을 위한 암호화를 요청하는 경우, 프로그램의 행동은 AP_MANDATORY에 관한 것입니다. 호스트가 암호화를 요청하지 않으면, 행동이 AP_NONE과 동일합니다.

ls_detail.def_data.retry_flags
이 필드는 이 링크 스테이션의 활성화가 자동 재시도를 하게 되는 조건을 지정합니다. 이는 비트 필드로 다음 값 중 어느 것이든 취할 수 있습니다.

AP_RETRY_ON_START
활성화 시도시 원격 노드로부터 응답이 수신되지 않으면 링크 활성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 활동중이 아니면, 프로그램이 이를 활성화시키려고 할 것입니다.

AP_RETRY_ON_FAILURE
활동중이나 대기 활동중인 동안 링크가 실패하면 링크 활성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 실패했으면, 프로그램이 이를 활성화시키려고 할 것입니다.

AP_RETRY_ON_DISCONNECT
링크가 원격 노드에 의해 정상적으로 중단되면 링크 활성화가 재시도될 것입니다.

AP_DELAY_APPLICATION_RETRIES
응용프로그램에 의해 초기화되는(START_LS 또는 요구가 있을 때 링크 활성화를 사용하여) 링크 활성화 재시도는 **activation_delay_timer**를 사용하여 속도 조정될 것입니다.

AP_DELAY_INHERIT_RETRY
이 필드에서 플래그에 의해 지정된 재시도 조건 외에, 하부 포트 정의의 **retry_flags** 필드에서 정의된 조건들 역시 사용될 것입니다.

ls_detail.def_data.max_activation_attempts
retry_flags에서 최소한 하나의 플래그가 설정되지 않는 한, 이 필드가 아무런 영향도 미치지 않습니다.

이 필드는 원격 노드가 응답하고 있지 않거나 하부 포트가 활동중이 아닐 때 프로그램이 허용하는 재시도 회수를 지정합니다. 여기에는 자동 재시도와 응용프로그램 지향 활성화 시도가 모두 포함됩니다.

이 한도에 도달하게 되면, 더 이상 자동 재시도가 이루어지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는

OPEN_LU_SSCP_SEC_RQ는 활성화가 실패하면 더 이상 재시도하지 않는 단일 활성화 시도의 결과를 낳습니다.

0은 '제한 없음'을 의미합니다. 값 AP_USE_DEFAULTS는 DEFINE_PORT에서 제공된 **max_activiation_attempts**가 사용되게 합니다.

ls_detail.def_data.activation_delay_timer

retry_flags에서 최소한 하나의 플래그가 설정되지 않는 한, 이 필드가 아무런 영향도 미치지 않습니다.

이 필드는 **def_data.retry_flags**에서 AP_DELAY_APPLICATION_RETRIES 비트가 설정되는 경우 프로그램이 자동 재시도 사이, 그리고 응용프로그램 지향 활성화 시도 사이에서 대기하는 초 수를 지정합니다.

값 AP_USE_DEFAULTS는 DEFINE_PORT에서 제공된 **activiation_delay_timer**가 사용되게 합니다.

0이 지정되면, 프로그램이 30초의 생략시 타이머 기간을 사용합니다.

def_data.branch_link_type

BrNN 만. 링크가 업링크인지 다운링크인지를 지정합니다. 이 필드는 **def_data.adj_cp_type** 필드가 AP_NETWORK, NODE, AP_END_NODE, AP_APPN_NODE 또는 AP_BACK_LEVEL_LEN_NODE로 설정되는 경우에만 적용됩니다.

AP_UPLINK
이 링크가 업링크입니다.

AP_DOWNLINK
링크가 다운링크입니다.

필드 **adj_cp_type**이 AP_NETWORK_NODE로 설정되면, 이 필드가 AP_UPLINK로 설정되어야 합니다.

기타 노드 유형: 이 필드가 무시됩니다.

ls_detail.det_data.adj_cp_is_brnn

BrNN만. 인접 CP가 NN(BrNN), 가령 NN 페이스를 보여주는 BrNN으로 허용되는지, 요구되는지 또는 금지되는지의 여부를 지정합니다. 이 필드는 **adj_cp_type** 필드가 AP_NETWORK_ NODE나 AP_APPN_NODE로 설정되는(그리고 XID 교환 동안 알게 된 노드 유형이 네트워크 노드) 경우에만 적용됩니다.

AP_BRNN_ALLOWED
인접 CP가 NN(BrNN)이 되게 허용됩니다.(그러나 꼭 필요한 것은 아닙니다.)

AP_BRNN_REQUIRED
인접 CP가 NN(BrNN)이 되게 허용되지 않습니다.

AP_BRNN_PROHIBITED
인접 CP가 NN(BrNN)이 되게 허용되지 않습니다.

필드 **adj_cp_type**이 AP_NETWORK_NODE로 설정되고 필드 **auto_act_supp**가 AP_YES로 설정되면, 이 필드가 AP_BRNN_REQUIRED 또는 AP_BRNN_PROHIBITED로 설정되어야 합니다.

기타 노드 유형: 이 필드가 무시됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LINK_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LS_EXCEPTION

QUERY_LS는 노드에서 정의된 링크 스테이션에 관한 정보의 리스트를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_LS에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LS에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **ls_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **ls_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이에 따라 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

반환되는 링크 스테이션들의 리스트는 그들이 속해 있는 포트의 이름에 의해 필터링될 수 있습니다. 이 경우, **port_name** 필드가 설정되어야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.)

VCB 구조

```
typedef struct query_ls_exception
{
   unsigned short  opcode;               /* verb operation code         */
   unsigned char   reserv2;              /* reserved                    */
   unsigned char   format;               /* format                      */
   unsigned short  primary_rc;           /* Primary return code         */
   unsigned long   secondary_rc;         /* Secondary return code       */
   unsigned char   *buf_ptr;             /* pointer to buffer           */
   unsigned long   buf_size;             /* buffer size                 */
   unsigned long   total_buf_size;       /* total buffer size required  */
   unsigned short  num_entries;          /* number of entries           */
   unsigned short  total_num_entries;    /* total number of entries     */
   unsigned char   list_options;         /* listing options             */
   unsigned char   reserv3;              /* reserved                    */
        unsigned long   exception_index; /* index of LS exception entry */
        unsigned char   ls_name;         /* name of link station        */
} QUERY_LS_EXCEPTION;

typedef struct LS_EXCEPTION
{
   unsigned short  overlay_size;         /* size of this entry          */
        unsigned long   exception_indes; /* index of this entry         */
        unsigned_DATE_TIME
                        time;            /* date and time               */
        unsigned char   ls_name[8];      /* link station name           */
        unsigned char   adj_cp_name[17];
                                         /* adjacent CP name            */
```

```
        unsigned char   adj_node_id[4];
                                        /* adjacent node id           */
        unsigned short  tg_number;      /* TG number                  */
        unsigned long   general_sense;  /* general sense data         */
        unsigned char   retry;          /* wil retry request          */
        unsigned long   end_sense;      /* termination sense data     */
        unsigned long   xid_local_sense; /* XID local sense data      */
        unsigned long   xid_remote_sense;/* XID remote sense data     */
        unsigned short  xid_error_byte;  /* offset of byte in error   */
        unsigned short  xid_error_bit;   /* offset of bit in error    */
   unsigned char   dlc_type;            /* DLC type                  */
        LINK_ADDRESS    local_addr;     /* local address             */
        LINK_ADDRESS    destination_addr;/* destination address      */
        unsigned char   reserved[20];   /* reserved                  */
} LS_EXCEPTION;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LS_EXCEPTION

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 형식 1 버전을 지정하려면, 이 필드를 1로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

다음 매개변수에서 지정된 **index**는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

exception_index
LS 예외 입력항목의 색인. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

ls_name
반환된 입력항목이 관련된 링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 이 필드가 널(null)로 설정되면, 임의의 링크 스테이션 또는 모든 링크 스테이션에 연관된 입력항목들이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

ls_exception.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

ls_exception.exception_index
이 LS 예외 입력항목에 지정된 색인. 색인의 값은 0에서 시작하며 2**31–1(2,147,483,647)의 최대 값까지 증분되고 난후 행바꾸기합니다.

ls_exception.time
LS 예외 입력항목이 생성된 날짜 및 시간.

ls_exception.ls_name
링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

ls_exception.adj_cp_name
전체 정식의 17 바이트의 인접 CP명. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문

자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드의 값은 다음과 같이 결정됩니다.

XID에서 인접 CP명이 수신되었으면, 그것이 반환됩니다.

XID에서 인접 CP명이 수신되었으나 국지로 정의된 값이 사용가능하면, 그것이 반환됩니다.

그렇지 않으면, 널(null)이 반환됩니다.

ls_exception.node_id
XID 교환 동안 인접 노드로부터 수신된 노드 ID(또는 아무것도 수신되지 않은 경우 널(null)). 이는 4 바이트의 16진 문자열입니다.

ls_exception.tg_number
이 링크 스테이션에 대한 TG와 연관된 번호(0-256의 범위). 256의 값은 장애시 TG 번호를 알 수 없었음을 나타냅니다.

ls_exception.general_sense
XID 순서의 시작까지 링크의 활성화 시작 순서와 연관된 오류 감지 데이터. 이는 노드에 의해 생성됩니다.

ls_exception.retry
노드가 링크를 활성화 시키기 위해 시작 요청을 재시도할 것인지의 여부를 나타냅니다.

AP_NO
노드가 시작 요청을 재시도하지 않을 것입니다.

AP_YES
노드가 시작 요청을 재시도할 것입니다.

ls_exception.end_sense
활성화 시도의 종료와 연관된 감지 데이터. 이는 DLC 계층에 의해 생성됩니다.

ls_exception.xid_local_sense
XID에서 송신된 국지 생성 감지 데이터.

ls_exception.xid.remote_sense
XID에서 수신된 원격 생성 감지 데이터.

ls_exception.xid_error_byte
XID에서 오류 바이트에 있는 오류 비트의 오프셋(0-65535의 범위). 값 65535는 이 필드가 아무 의미도 없음을 나타냅니다.

ls_exception.xid_error_bit
XID에서 오류 바이트에 있는 오류 비트의 오프셋(0-7의 범위). 값 8은 이 필드가 아무 의미도 없음을 나타냅니다.

ls_exception.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_SDLC
AP_X25
AP_TR

DEFINE_DLC 명령에서 새로운 유형을 지정함으로써 추가 DLC 유형이 정의될 수 있습니다. 더 자세한 정보는 51페이지의 『DEFINE_DLC』를 참조하십시오.

ls_exception.local_addr.length
국지 링크 스테이션의 주소 길이.

ls_exception.local_address.address
국지 링크 스테이션의 주소.

ls_exception.destination_addr.length
인접 노드상의 목적지 링크 스테이션 주소 길이.

ls_exception.destination_addr.address
인접 노드상의 목적지 링크 스테이션의 주소.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_EXCEPTION_INDEX

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LU_0_TO_3

QUERY_LU_0_TO_3은 유형 0, 1, 2이나 3의 국지 LU에 관한 정보를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_LU_0_TO_3에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 국지 LU에 관한 정보를 확보하거나 여러 개의 청크(chunk)로 리스트 정보를 확보하려면, **lu_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다.

SNA API 클라이언트에서는 특정 매개변수들만이 지원됩니다. 특정 세부사항은 노트 *패드*를 보십시오.

이 아이콘은 통신 서버와 퍼스널 통신에 영향을 미칠 수 있는 중요한 정보를 나타냅니다.

VCB 구조

```
typedef struct query_lu_0_to_3
{
   unsigned short  opcode;                /* verb operation code        */
   unsigned char   attributes;            /* Verb attributes            */
   unsigned char   reserv2;               /* reserved                   */
   unsigned char   format;                /* format                     */
   unsigned short  primary_rc;            /* primary return code        */
   unsigned long   secondary_rc;          /* secondary return code      */
   unsigned char   *buf_ptr;              /* pointer to buffer          */
   unsigned long   buf_size;              /* buffer size                */
   unsigned long   total_buf_size;        /* total buffer size required */
   unsigned short  num_entries;           /* number of entries          */
   unsigned short  total_num_entries;     /* total number of entries    */
   unsigned char   list_options;          /* listing options            */
   unsigned char   reserv3;               /* reserved                   */
        unsigned char   pu_name[8];       /* PU name filter             */
   unsigned char   lu_name[8];            /* LU name                    */
        unsigned char   host_attachment;  /* Host attachment filter     */
} QUERY_LU_0_TO_3;

typedef struct lu_0_to_3_summary
{
   unsigned short  overlay_size;          /* size of this entry            */
   unsigned char   pu_name[8];            /* PU name                       */
   unsigned char   lu_name[8];            /* LU name                       */
   unsigned char   description[RD_LEN];
                                          /* resource description          */
        unsigned char   nau_address;      /* NAU address                   */
        unsigned char   lu_sscp_sess_active;
                                          /* Is LU-SSCP session active     */
        unsigned char   appl_conn_active; /* Is connection to appl active? */
        unsigned char   plu_sess_active;  /* Is PLU-SLU session active     */
        unsigned char   host_attachment;  /* LU's host attachment          */
} LU_0_TO_3_SUMMARY;

typedef struct lu_0_to_3_detail
{
   unsigned short  overlay_size;          /* size of this entry            */
   unsigned char   lu_name[8];            /* LU name                       */
```

```
   unsigned char   reserv1[2];             /* reserved                     */
        LU_0_TO_3_DET_DATA det_data;       /* Determined data              */
        LU_0_TO_3_DEF_DATA def_data;       /* Defined data                 */
} LU_0_TO_3_DETAIL;

typedef struct lu_0_to_3_det_data
{
        unsigned char   lu_sscp_sess_active;
                                           /* Is LU-SSCP session active    */
        unsigned char   appl_conn_active;  /* Application is using LU      */
        unsigned char   plu_sess_active;   /* Is PLU-SLU session active    */
   unsigned char   host_attachment;        /* Host attachment              */
        SESSION_STATS   lu_sscp_stats;     /* LU-SSCP session statistics   */
        SESSION_STATS   plu_stats;         /* PLU-SLU session statistics   */
        unsigned char   plu_name[8];       /* PLU name                     */
        unsigned char   session_id[8];     /* Internal ID of PLU-SLU sess  */
        unsigned char   app_spec_det_data[256];
                                           /* Application Specified Data   */
        unsigned char   app_type;          /* Application type             */
   unsigned char   sscp_id[6];             /* SSCP ID                      */
        unsigned char   bind_lu_type;      /* LU type issuing BIND         */
   unsigned char   reserva[12];            /* reserved                     */
} LU_0_TO_3_DET_DATA;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;       /* session receive RU size       */
   unsigned short  send_ru_size;      /* session send RU size          */
   unsigned short  max_send_btu_size; /* max send BTU size             */
   unsigned short  max_rcv_btu_size;  /* max rcv BTU size              */
        unsigned short  max_send_pac_win;  /* max send pacing win size      */
        unsigned short  cur_send_pac_win;  /* current send pacing win size  */
        unsigned short  max_rcv_pac_win;   /* max receive pacing win size   */
   unsigned short  cur_rcv_pac_win;   /* current receive pacing        */
                                      /* window size                   */
   unsigned long   send_data_frames;  /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                      /* num of FMD data frames sent   */
   unsigned long   send_data_bytes;   /* number of data bytes sent     */
   unsigned long   rcv_data_frames;   /* num data frames received      */
   unsigned long   rcv_fmd_data_frames;
                                      /* num of FMD data frames recvd  */
   unsigned long   rcv_data_bytes;    /* number of data bytes received */
   unsigned char   sidh;              /* session ID high byte          */
   unsigned char   sidl;              /* session ID low byte           */
   unsigned char   odai;              /* ODAI bit set                  */
   unsigned char   ls_name[8];        /* Link station name             */
   unsigned char   pacing_type;       /* type of pacing in use         */
} SESSION_STATS;

typedef struct lu_0_to_3_def_data
{
   unsigned char   description[RD_LEN];
                                        /* resource description         */
        unsigned char   nau_address;    /* LU NAU address               */
        unsigned char   pool_name[8];   /* LU Pool name                 */
   unsigned char   pu_name[8];          /* PU name                      */
        unsigned char   priority;       /* LU priority                  */
        unsigned char   lu_model;       /* LU model                     */
        unsigned char   sscp_id[6];
        unsigned char   timeout;        /* Timeout                      */
        unsigned char   app_spec_def_data[16];
                                        /* Application Specified Data   */
        unsigned char   model_name[7]; /* LU model                     */
        unsigned char   reserv3[17];    /* reserved                     */
} LU_0_TO_3_DEF_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LU_0_TO_3

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_SUMMARY 값 역시 SNA API 클라이언트를 위해 지원됩니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **lu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_FIRST_IN_LIST 값 역시 SNA API 클라이언트를 위해 지원됩니다.

.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

SNA API 클라이언트에 대해 list_options 값이 무시됩니다.

pu_name
PU 명. 이 PU를 사용하는 LU만이 반환될 것입니다. 모든 LU의 리스트가 요구되면, 이 필드를 모두 2진 0으로 설정해야 합니다. SNA API 클라이언트에 대해 list_options 값이 무시됩니다.

SNA API 클라이언트에 대해 pu_name 값이 무시됩니다.

host_attachment
호스트 연결을 위한 필터.

AP_NONE
모든 LU에 대한 정보를 반환합니다.

AP_NONE이 SNA API 클라이언트를 위해 지원되는 유일한 값입니다.

AP_DLUR_ATTACHED
DLUR에 의해 지원되는 모든 LU에 대한 정보를 반환합니다.

AP_DIRECT_ATTACHED
호스트 시스템에 직접 연결되는 LU에 대한 정보만을 반환합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries

실제로 반환된 입력항목의 갯수.

total_num_entries

반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

lu_0_to_3_summary.overlay_size

이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

lu_0_to_3_summary.pu_name

이 LU가 사용하고 있는 국지 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

lu_0_to_3_summary.pu_name 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.lu_name

조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

lu_0_to_3_summary.description

자원 설명(DEFINE_LU_0_TO_3에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

lu_0_to_3_summary.description 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.nau_address

LU의 네트워크 지정 장치(NAU) 주소로 그 범주는 1—255.

lu_0_to_3_summary.nau_address 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.lu_sscp_sess_active

LU-SSCP 세션이 활동중인지의 여부를 지정합니다(AP_YES or AP_NO).

lu_0_to_3_summary.lu_sscp_sess_active 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.appl_conn_active

응용프로그램이 LU를 사용하고 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

lu_0_to_3_summary.aapl_conn_active 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.plu_sess_active
PLU-SLU 세션이 활동중인지의 여부를 지정합니다(AP_YES 또는 AP_NO).

lu_0_to_3_summary.plu_sess_active 값은 SNA API 클라이언트에서 반환되지 않습니다.

lu_0_to_3_summary.host_attachment
LU 호스트 연결 유형.

AP_DLUR_ATTACHED
LU가 DLUR을 사용하여 호스트 시스템에 연결됩니다.

AP_DIRECT_ATTACHED
LU가 직접 호스트 시스템에 연결됩니다.

lu_0_to_3_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

lu_0_to_3_detail.lu_name
조회되고 있는 국지 LU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

lu_0_to_3_detail.det_data.lu_sscp_sess_active
LU-SSCP 세션이 활동중인지의 여부를 지정합니다(AP_YES or AP_NO).

lu_0_to_3_detail.det_data.appl_conn_active
이 LU가 현재 응용프로그램에 의해 사용되고 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

lu_0_to_3_detail.det_data.plu_sess_active
PLU-SLU 세션이 활동중인지의 여부를 지정합니다(AP_YES 또는 AP_NO).

lu_0_to_3_detail.det_data.host_attachment
LU 호스트 연결 유형.

AP_DLUR_ATTACHED
LU가 DLUR을 사용하여 호스트 시스템에 연결됩니다.

AP_DIRECT_ATTACHED
LU가 직접 호스트 시스템에 연결됩니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.lu_sscp_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

lu _0_to_3_detail.det_data.lu_sscp_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.lu_sscp_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

lu_0_to_3_detail.det_data.lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

lu_0_to_3_detail.det_data.lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

lu_0_to_3_detail.det_data.lu_sscp_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 ACTLU의 송신자가 이 필드를 0으로 설정하며, ACTLU 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세

트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 이 세션을 세션이 흐르는 링크와 상관시키는 데 이 필드가 사용될 수 있습니다.

lu_0_to_3_detail.det_data.lu_sscp_stats.pacing_type
LU-SSCP 세션상에서 사용중인 수신 페이싱 유형. AP_NONE으로 설정될 것입니다.

lu_0_to_3_detail.det_data.plu_stats.rcv_ru_size
최대 수신 RU 크기.

lu_0_to_3_detail.det_data.plu_stats.send_ru_size
최대 송신 RU 크기.

lu_0_to_3_detail.det_data.plu_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

lu_0_to_3_detail.det_data.plu_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

lu_0_to_3_detail.det_data.plu_stats.max_send_pac_win
이 세션상의 최대 송신 페이싱 창 크기.

lu_0_to_3_detail.det_data.plu_stats.cur_send_pac_win
이 세션상의 현재 송신 페이싱 창 크기.

lu_0_to_3_detail.det_data.plu_stats.max_rcv_pac_win
이 세션상의 최대 수신 페이싱 창 크기.

lu_0_to_3_detail.det_data.plu_stats.cur_rcv_pac_win
이 세션상의 현재 수신 페이싱 창 크기.

lu_0_to_3_detail.det_data.plu_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.plu_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.plu_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

lu_0_to_3_detail.det_data.plu_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.plu_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

lu_0_to_3_detail.det_data.plu_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

lu_0_to_3_detail.det_data.plu_stats.sidh
세션 ID 상위(high) 바이트.

lu_0_to_3_detail.det_data.plu_stats.sidl
세션 ID 하위(low) 바이트.

lu_0_to_3_detail.det_data.plu_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정하며, BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

lu_0_to_3_detail.det_data.plu_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

lu_0_to_3_detail.det_data.plu_stats.pacing_type
PLU-SSCP 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE 또는 AP_PACING_FIXED를 취할 수 있습니다.

lu_0_to_3_detail.det_data.plu_name
1차 LU명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.(PLU-SLU 세션이 활동중이 아니면, 이 필드가 예약됩니다.)

lu_0_to_3_detail.det_data.session_id
PLU-SLU 세션의 8 바이트 내부 식별자(ID).

lu_0_to_3_detail.det_data.app_spec_det_data
예약됨.

lu_0_to_3_detail.det_data.sscp_id
이 LU가 사용하는 PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하고 있는 6 바이트의 필드.

lu_sscp_sess_active가 AP_YES가 아니면, 이 필드가 0으로 될 것입니다.

lu_0_to_3_detail.det_data.app_type
예약됨.

lu_0_to_3_detail.lu_0_to_3_det_data.bind_lu_type
원래의 BIND를 발행한 LU의 LU 유형. 활동중 LU-LU 세션이 없으면, 다음 중 하나일 수 있습니다.

AP_LU_TYPE_0
AP_LU_TYPE_1
AP_LU_TYPE_2
AP_LU_TYPE_3
AP_LU_TYPE_6(다운스트림 종속 LU 6.2의 경우).

활동중 LU—LU 세션이 없으면, 이 필드는 다음 값을 취합니다.

AP_LU_TYPE_UNKNOWN

lu_0_to_3_detail.def_data.description
자원 설명(DEFINE_LU_0_TO_3에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

lu_0_to_3_detail.def_data.nau_address
LU의 네트워크 지정 장치(NAU) 주소로 그 범주는 1—255.

lu_0_to_3_detail.def_data.pool_name
이 LU가 속하는 풀의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. LU가 풀에 속하지 않으면, 이 필드가 모두 2진 0으로 설정됩니다.

lu_0_to_3_detail.def_data.pu_name
이 LU가 사용할 PU의 이름(DEFINE_LS 명령에서 지정된 것과 같이). 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

lu_0_to_3_detail.def_data.priority
호스트에 송신할 때의 LU 우선순위. 다음 값 중 하나로 설정됩니다.

AP_NETWORK
AP_HIGH
AP_MEDIUM
AP_LOW

lu_0_to_3_detail.def_data.lu_model
모델 유형 및 LU의 갯수. 다음 값 중 하나로 설정됩니다.

AP_3270_DISPLAY_MODEL_2
AP_3270_DISPLAY_MODEL_3
AP_3270_DISPLAY_MODEL_4
AP_3270_DISPLAY_MODEL_5
AP_RJE_WKSTN
AP_PRINTER
AP_SCS_PRINTER
AP_UNKNOWN

lu_0_to_3_detail.def_data.sscp_id
이 필드는 이 LU를 활성화시키도록 허용된 SSCP의 ID를 지정합니다. 이는 6 바이트의 2진 필드입니다. 필드가 2진 0으로 설정되면, 어떤 SSCP에 의해서든 LU가 활성화될 수 있습니다.

lu_0_to_3_detail.def_data.timeout
초 단위로 지정된 LU에 대한 시간종료. 시간종료가 제공되고 LU의 사용자가 OPEN_LU_SSCP_SEC_RQ에서(PU 집중의 경우에는, 다운스트림 LU 정의에서) **allow_timeout**을 정의했으면, PLU-SLU 세션이 이 기간동안 활동중이 아니며 다음 조건들 중 하나에 해당하면 LU가 활성종료될 것입니다.

- 세션이 제한된 자원 링크 통과합니다.

- 세션이 다시 사용되기 전에 또 다른 응용프로그램이 LU를 사용하려고 합니다.

시간종료가 0으로 설정되면, LU가 활성종료되지 않을 것입니다.

lu_0_to_3_detail.def_data.app_spec_def_data
DEFINE_LU_0_TO_3으로부터의 응용프로그램 특정 데이터; 프로그램이 이 필드를 해석하지 않고 단지 보관만 하여 QUERY_LU_0_TO_3 명령에서 반환합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_LU_POOL

QUERY_LU_POOL은 풀의 리스트와 이들에 속하는 LU를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 LU 풀에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **pool_name** 및 **lu_name** 필드가 설정되어야 합니다. **lu_name** 필드가 모두 0으로 설정되면, 반환되는 정보가 지정된 풀의 첫번째 LU로부터 시작합니다. **list_options** 필드가 AP_FIRST_IN_LIST로 설정되면 두 필드 모두 무시됩니다.

VCB 구조

```
typedef struct query_lu_pool
{
   unsigned short  opcode;             /* verb operation code           */
   unsigned char   attributes;         /* verb attributes               */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* primary return code           */
   unsigned long   secondary_rc;       /* secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   pool_name[8];  /* pool name                     */
   unsigned char   lu_name[8];         /* LU name                       */
} QUERY_LU_POOL;

typedef struct lu_pool_summary
{
   unsigned short  overlay_size;        /* size of this entry          */
        unsigned char   pool_name[8];   /* pool name                   */
   unsigned char   description[RD_LEN]; /* resource description        */
        unsigned short  num_active_lus; /* num of currently active LUs */
        unsigned char   num_avail_lus;  /* num of currently available  */
                                        /* LUs                         */
} LU_POOL_SUMMARY;

typedef struct lu_pool_detail
{
   unsigned short  overlay_size;        /* size of this entry          */
        unsigned char   pool_name[8];   /* pool name                   */
   unsigned char   description[RD_LEN]; /* resource description        */
   unsigned char   lu_name[8];          /* LU name                     */
        unsigned char   lu_sscp_sess_active; /* Is LU-SSCP session active */
        unsigned char   appl_conn_active;    /* Is SSCP connection open   */
        unsigned char   plu_sess_active;     /* Is PLU-SLU session active */
} LU_POOL_DETAIL;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_LU_POOL

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **pool_name**과 **lu_name**의 조합(다음 매개변수 참조)은 반환될I

lu_name
LU명. 이 이름은 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 모두 0으로 설정되면, 지정된 풀에 속하는 LU가 풀의 시작으로부터 나열됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
반환된 디렉토리 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

lu_pool_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

lu_pool_summary.pool_name
지정된 LU가 속하는 LU 풀의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.(요청에서 이 필드가 지정되고 **lu_name** 필드가 모두 2진 0으로 설정되면, 풀에 있는 LU만이 반환됨에 주의하십시오.)

lu_pool_summary.description
LU 풀 설명(DEFINE_LU_POOL에서 지정된 것과 같이).

lu_pool_summary.num_active_lus
활동중 LU-SSCP 세션을 가지는 지정된 풀 내의 LU 갯수.

lu_pool_summary.num_avail_lus
open_force가 AP_YES로 설정된 OPEN_LU_SSCP_SEC_REQ를 만족시킬 수 있게 사용가능한 지정된 풀 내의 LU 갯수. 여기에는 PU가 활동중이거나 호스트 링크가 자동 활동가능한 모든 LU와 그 연결이 자유로운 모든 LU가 포함됩니다. 이 집계는 LU **model_type**, **model_name** 및 PU의 DDDLU가 지원에 관계없습니다. **model_type**에 대한 특정 값을 지정하는 OPEN_LU_SSCP_SEC_REQ를 충족시키기 위해 사용가능한 LU는 더 조금 있을 것입니다.

lu_pool_detail.num_active_lus
활동중 LU-SSCP 세션을 가지는 지정된 풀 내의 LU 갯수.

lu_pool_detail.num_avail_lus
사용가능한 LU-SSCP 세션을 가지는 지정된 풀 내의 LU 갯수.

lu_pool_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

lu_pool_detail.pool_name
지정된 LU가 속하는 LU 풀의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.(요청에서 이 필드가 지정되고 **lu_name** 필드가 모두 2진 0으로 설정되면, 풀에 있는 LU만이 반환됨에 주의하십시오.)

lu_pool_detail.description
LU 설명(DEFINE_LU_0_TO_3에서 지정된 것과 같이).

lu_pool_detail.lu_name
풀에 속하는 LU의 LU명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 이 이름이 모두 0으로 설정되면, 지정된 풀이 비어있음을 의미합니다.

lu_pool_detail.lu_sscp_sess_active
LU-SSCP 세션이 활동중인지의 여부를 지정합니다(AP_YES or AP_NO).

lu_pool_detail.appl_conn_active
LU 세션이 현재 응용프로그램에 의해 사용되고 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

lu_pool_detail.plu_sess_active
PLU-SLU 세션이 활동중인지의 여부를 지정합니다(AP_YES 또는 AP_NO).

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LIST_OPTION

AP_INVALID_POOL_NAME
AP_INVALID_LU_NAME

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

QUERY_LU_POOL

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MDS_APPLICATION

QUERY_MDS_APPLICATION는 MDS 레벨 메세지를 위해 등록한 응용프로그램의 리스트를 반환합니다.

응용프로그램은 697페이지의 『제15장 관리 서비스 명령』에서 기술된 REGISTER_MS_APPLICATION 명령을 발행함으로써 등록합니다.

특정 응용프로그램에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **application** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

VCB 구조

```
typedef struct query_mds_application
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char   reserv2;             /* reserved                   */
   unsigned char   format;              /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   *buf_ptr;            /* pointer to buffer          */
   unsigned long   buf_size;            /* buffer size                */
   unsigned long   total_buf_size;      /* total buffer size required */
   unsigned short  num_entries;         /* number of entries          */
   unsigned short  total_num_entries;   /* total number of entries    */
   unsigned char   list_options;        /* listing options            */
   unsigned char   reserv3;             /* reserved                   */
        unsigned char   application[8]; /* application                */
} QUERY_MDS_APPLICATION;

typedef struct mds_application_data
{
   unsigned short  overlay_size;        /* size of this entry         */
        unsigned char   application[8]; /* application name           */
        unsigned short  max_rcv_size;   /* max data size application  */
                                        /* can receive                */
   unsigned char   reserva[20];         /* reserved                   */
} MDS_APPLICATION_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_MDS_APPLICATION

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램

은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
지정된 **application**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

application
응용프로그램 명. 이 이름은 8 바이트의 영숫자 유형 A EBCDIC 문자열입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

mds_application_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mds_application_data.application
등록된 응용프로그램의 이름. 이 이름은 8 바이트의 영숫자 유형 A EBCDIC 문자열입니다.

mds_application_data.max_rcv_size
하나의 청크에서 응용프로그램이 수신할 수 있는 최대 바이트 수(이는 응용프로그램이 MDS에 등록할 때 지정됩니다). MDS 레벨 응용프로그램 등록에 관한 더 자세한 정보는 제15장 관리 서비스 명령를 참조하십시오.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_APPLICATION_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MDS_STATISTICS

QUERY_MDS_STATISTICS는 관리 서비스 통계를 반환합니다. MDS 경로지정 트래픽의 레벨을 측정하는 데 이 명령을 사용할 수 있습니다.

VCB 구조

```
typedef struct query_mds_statistics
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        unsigned long   alerts_sent;       /* number of alert sends       */
        unsigned long   alert_errors_rcvd; /* error messages received     */
                                           /* for alert sends             */
        unsigned long   uncorrelated_alert_errors;
                                           /* uncorrelated alert          */
                                           /* errors received             */
        unsigned long   mds_mus_rcvd_local;/* number of MDS_MUs received  */
                                           /* from local applications     */
        unsigned long   mds_mus_rcvd_remote;
                                           /* number of MDS_MUs received  */
                                           /* from remote applications    */
        unsigned long   mds_mus_delivered_local;
                                           /* num of MDS_MUs delivered    */
                                           /* to local applications       */
        unsigned long   mds_mus_delivered_remote;
                                           /* num of MDS_MUs              */
                                           /* delivered to remote appls   */
        unsigned long   parse_errors;      /* number of MDS_MUs received  */
                                           /* with parse errors           */
        unsigned long   failed_deliveries; /* number of MDS_MUs where     */
                                           /* delivery failed             */
        unsigned long   ds_searches_performed;
                                           /* number of DS searches done  */
        unsigned long   unverified_errors; /* number of unverified errors */
   unsigned char   reserva[20];            /* reserved                    */
} QUERY_MDS_STATISTICS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_MDS_STATISTICS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

alerts_sent
MDS 트랜스포트 시스템을 사용하여 송신된 국지에서 생긴 경고 수.

alert_errors_rcvd
경고를 포함하고 있는 메세지 전달 실패를 나타내는 MDS에 의해 수신된 오류 메세지 수.

uncorrelated_errors_rcvd
경고를 포함하고 있는 메세지 전달 실패를 나타내는 MDS에 의해 수신된 오류 메세지 수. 오류 메세지가 MDS 송신 경고 대기행렬 상의 경고와 상관될 수 없을 때에 전달 실패가 발생합니다. MDS는 문제점 진단 중재점에 보내진 경고를 캐쉬(cache)하는 고정 크기의 대기행렬을 유지보수합니다. 일단 대기행렬이 최대 크기에 도달하면, 가장 오래된 경고가 버려지고 새로운 경고로 대체됩니다. 전달 오류 메세지가 수신되면, MDS는 문제점 진단 중재점이 복원될 때까지 경고가 보존될 수 있도록 오류 메세지를 캐쉬(cache)된 경고에 상관시키려 시도합니다.

주: 두 개의 계수, **alert_errors_rcvd**와 **uncorrelated_errors_rcvd**가 경고 송신 대기행렬의 크기를 조정할 수 있도록 유지보수됩니다. 시간이 흐르면서 **uncorrelated_errors_rcvd**가 늘어나면, 이는 경고 송신 대기행렬 크기가 너무 작음을 의미합니다.

mds_mus_rcvd_local
국지 응용프로그램으로부터 수신된 MDS_MU들의 갯수.

mds_mus_rcvd_remote
MDS_RECEIVE 및 MSU_HANDLER 트랜잭션 프로그램(TP)을 사용하여 원격 노드로부터 수신된 MDS_MU들의 갯수.

mds_mus_delivered_local
성공적으로 국지 응용프로그램에 전달된 MDS_MU들의 갯수.

mds_mus_delivered_remote
MDS_SEND를 사용하여 원격 노드에 성공적으로 전달된 MDS_MU들의 갯수.

parse_errors
헤더 형식 오류를 포함하고 있는 수신된 MDS_MU들의 갯수.

failed_deliveries
이 노드가 전달하지 못한 MDS_MU들의 갯수.

ds_searches_performed
예약됨.

unverified_errors
예약됨.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE

QUERY_MODE는 특정 상대방 LU와의 국지 LU에 의해 사용된 모드에 관한 정보를 반환합니다. 정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 모드에 관한 정보를 확보하거나 여

```
                                        /* resource description          */
        unsigned short  sess_limit;     /* current session limit         */
   unsigned short  act_sess_count;      /* curr active sessions count    */
        unsigned char   fqplu_name[17];/* partner LU name               */
   unsigned char   reserv1[3];          /* reserved                      */
} MODE_SUMMARY;

typedef struct mode_detail
{
   unsigned short  overlay_size;        /* size of this entry            */
   unsigned char   mode_name[8];        /* mode name                     */
   unsigned char   description[RD_LEN];
                                        /* resource description          */
        unsigned short  sess_limit;          /* session limit                */
        unsigned short  act_sess_count;      /* currently active sess count  */
        unsigned char   fqplu_name[17];      /* partner LU name              */
   unsigned char   reserv1[3];               /* reserved                     */
        unsigned short  min_conwinners_source;
                                             /* min conwinner sess limit     */
        unsigned short  min_conwinners_target;
                                             /* min conloser limit           */
        unsigned char   drain_source;        /* drain source?                */
        unsigned char   drain_partner;       /* drain partner?               */
        unsigned short  auto_act;            /* auto activated conwinner     */
                                             /* session limit                */
        unsigned short  act_cw_count;        /* active conwinner sess count  */
        unsigned short  act_cl_count;        /* active conloser sess count   */
        unsigned char   sync_level;          /* synchronization level        */
        unsigned char   default_ru_size;     /* default RU size to maximize  */
                                             /* performance                  */
        unsigned short  max_neg_sess_limit; /* max negotiated session limit */
        unsigned short  max_rcv_ru_size;     /* max receive RU size          */
        unsigned short  pending_session_count;
                                             /* pending sess count for mode  */
        unsigned short  termination_count;  /* termination count for mode   */
   unsigned char   implicit;                 /* implicit or explicit entry   */
        unsigned char   reserva[15];         /* reserved                     */
} MODE_DETAIL;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_MODE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **lu_name**(또는 **lu_name**이 모두 0으로 설정된 경우 **lu_alias**), **plu_alias**(또는 **plu_alias**가 모두 0으로 설정된 경우 **fqplu_name**) 그리고 **mode_name**(다음 매개변수 참조)의 조합은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다. 상대방 LU 색인이 지정될 때, 가능하면 다른 상대방 LU에 대한 정보가 포함됩니다.

AP_FIRST_IN_LIST

plu_alias 및 **fqplu_name**이 모두 0으로 설정되면, 반환되는 리스트가 리스트의 첫번째 상대방 LU로부터 시작하고 **mode_name** 색인은 무시됩니다. **plu_alias**나 **fqplu_name** 어느것이든 하나가 지정되면, 리스트가 이 색인에서 시작하지만, **mode_name** 색인 값은 무시되고 반환된 리스트는 리스트의 첫번째 모드 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name

LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다.

lu_alias

국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 이 필드는 **lu_name** 필드가 모두 0으로 설정된 경우에만 유효하지만, 이 경우 8 바이트 모두가 유효하며 설정되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용됩니다.

plu_alias

상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 이 필드가 모두 0으로 설정되면, **fqplu_alias** 필드가 색인 결정에 사용될 것입니다.

fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

active_sessions
활동중인 세션 필터. 반환된 모드가 모드에 현재 활동중인 세션이 있는지에 따라 필터링되어야 하는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

mode_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mode_summary.mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

mode_summary.description
자원 설명(DEFINE_MODE에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

mode_summary.sess_limit
현재 세션 한계.

mode_summary.act_sess_count
모드를 사용하고 있는 총 활동중인 세션 수. **active_sessions** 필터가 AP_YES로 설정되었으면, 이 필드가 항상 0보다 클 것입니다.

mode_summary.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

mode_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mode_detail.mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

mode_detail.description
자원 설명(DEFINE_MODE에서 지정된 것과 같이).

mode_detail.sess_limit
현재 세션 한계.

mode_detail.act_sess_count
모드를 사용하고 있는 총 활동중인 세션 수. **active_sessions** 필터가 AP_YES로 설정되었으면, 이 필드가 항상 0보다 클 것입니다.

mode_detail.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

mode_detail.min_conwinners_source
국지 LU가 그곳에서 회선경합 성공 세션(또는 “첫번째 스피커”)인 최소 세션 수를 지정합니다.

mode_detail.min_conwinners_target
국지 LU가 그곳에서 회선경합 실패 세션(또는 “명령자(bidder)”)인 최소 세션 수를 지정합니다.

mode_detail.drain_source
세션 한계가 변경되거나 재설정될 때 세션을 활성종료하기 전에 국지 LU가 대기 세션 요청을 만족시키는지의 여부를 지정합니다(AP_NO 또는 AP_YES).

mode_detail.drain_partner
세션 한계가 변경되거나 재설정될 때 세션을 활성종료하기 전에 상대방 LU가 대기 세션 요청을 만족시키는지의 여부를 지정합니다 (AP_NO 또는 AP_YES).

mode_detail.auto_act
상대방 LU와의 세션 수 변경 교환에 뒤따라 자동으로 활성화되는 회선경합 성공 세션의 수.

mode_detail.act_cw_count
이 모드를 사용하는 활동중인 회선경합 성공("첫번째 스피커") 세션 수(국지 LU는 이러한 세션들 중 하나를 사용하기 전에 송수신을 요구할 필요가 없습니다).

mode_detail.act_cl_count
이 모드를 사용하는 활동중인 회선경합 실패("명령자(bidder)") 세션 수 (국지 LU는 이러한 세션들 중 하나를 사용하기 전에 송수신을 요구해야 합니다).

mode_detail.sync_level
모드에 의해 지원되는 동기화 레벨을 지정합니다(AP_NONE, AP_CONFIRM 또는 AP_SYNCPT).

mode_detail.default_ru_size
규격 RU 크기에 대한 생략시 상위 바운드가 사용될지의 여부를 지정합니다. 이 매개변수가 AP_YES의 값을 가지면, **define_mode**에서 지정된 **mode_chars.max_ru_size_upp** 필드가 무시되며, 최대 RU 크기에 대한 상위 바운드가 BTU 크기에서 TH 및 RH의 크기를 뺀 것으로 설정됩니다.

AP_YES
AP_NO

mode_detail.max_neg_sess_limit
최대 조정가능 세션 한계. 목표 LU로 CNOS 처리 동안 국지 LU가 사용할 수 있는 모드명에 대한 최대 세션 한계를 지정합니다.

mode_detail.max_rcv_ru_size
최대 수신 RU 크기.

mode_detail.pending_session_count
세션 대기(세션 활성화가 완료되기를 기다리는) 수를 지정합니다.

mode_detail.termination_count
이전의 CNOS 명령이 모드 세션 한계를 0으로 재설정했으면, 이러한 세션들을 사용중이거나 사용하려고 기다리고 있는 대화가 있을 것입니다. 이 필드는 이러한 세션 중 얼마나 많은 세션들이 아직 활성종료되지 않았는가에 대한 계수입니다.

mode_detail.implicit
암시적(AP_YES) 또는 명시적(AP_NO) 정의로 인해 입력항목이 놓여졌는지의 여부를 지정합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MODE_NAME

AP_INVALID_PLU_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE_DEFINITION

QUERY_MODE_DEFINITION은 DEFINE_MODE 명령상에서 이전에 건네진 정보와 SNA 정의 생략시 모드에 관한 정보를 모두 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 모드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **mode_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **mode_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라).

AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

이 명령은 정의 정보만을 반환합니다. QUERY_MODE 명령은 일단 상대방 LU와의 국지 LU에 의해 사용되기 시작된 후 결정되는 정보를 반환합니다.

VCB 구조

```
typedef struct query_mode_definition
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   mode_name[8];       /* mode name                     */
} QUERY_MODE_DEFINITION;

typedef struct mode_def_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   mode_name[8];       /* mode name                     */
   unsigned char   description[RD_LEN];
                                       /* resource description          */
} MODE_DEF_SUMMARY;

typedef struct mode_def_detail
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   mode_name[8];       /* mode name                     */
        MODE_CHARS      mode_chars;    /* mode characteristics          */
} MODE_DEF_DETAIL;

typedef struct mode_chars
{
   unsigned char   description[RD_LEN];
                                          /* resource description        */
        unsigned short  max_ru_size_upp;  /* max RU size upper bound     */
```

```
        unsigned char   receive_pacing_win; /* receive pacing window        */
        unsigned char   default_ru_size;    /* default RU size to maximize  */
                                            /* performance                  */
        unsigned short  max_neg_sess_lim;   /* max negotiable session limit */
        unsigned short  plu_mode_session_limit;
                                            /* LU-mode session limit        */
        unsigned short  min_conwin_src;     /* min source contention winner */
                                            /* sessions                     */
   unsigned char   cos_name[8];             /* class-of-service name        */
        unsigned char   cryptography;       /* cryptography                 */
        unsigned char   compression;        /* compression                  */
        unsigned short  auto_act;           /* initial auto-activation count*/
        unsigned short  min_conloser_src;   /* min source contention loser  */
        unsigned short  max_ru_size_low     /* maximum RU size lower bound  */
        unsigned short  max_receive_pacing_win;
                                            /* maximum receive pacing window*/
} MODE_CHARS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_MODE_DEFINITION

format

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

mode_def_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mode_def_summary.mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 8 바이트의 모드명.

mode_def_summary.description
자원 설명(DEFINE_MODE에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

mode_def_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mode_def_detail.mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

mode_def_detail.mode_chars.description
자원 설명(DEFINE_MODE에서 지정된 것과 같이). 이는 국지로 표시 가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

mode_def_detail.mode_chars.max_ru_size_upp
이 모드명으로 세션상에서 사용될 최대 RU 크기에 대한 상위 경계.

mode_def_detail.mode_chars.receive_pacing_win
고정 페이싱이 사용될 때의 세션을 위한 세션 페이싱 창을 지정합니다. 적합 페이싱이 사용될 때 선호 최소 창 크기를 지정합니다.

mode_def_detail.mode_chars.default_ru_size
규격 RU 크기에 대한 생략시 상위 바운드가 사용될지의 여부를 지정합니다. 이 매개변수가 AP_YES를 지정하면, **max_ru_size_upp**가 무시됩니다.

AP_YES
AP_NO

mode_def_detail.mode_chars.max_neg_sess_lim
최대 조정가능 세션 한계. 지정된 모드명에 대한 국지 LU와 상대방 사이의 허용가능 최대 세션 수를 조정하는 데 사용된 값.

mode_def_detail.mode_chars.plu_mode_session_limit
이 모드상에서 초기에 조정하기 위한 세션 한계. 이 값은 선호 세션 한계를 나타내며 암시적 CNOS에 사용됩니다.

범위: 0—32 767

mode_def_detail.mode_chars.min_conwin_src
이 모드를 사용하는 국지 LU에 의해 활성화될 수 있는 최대 회선경합 성공 세션 수.

범위: 0—32767

mode_def_detail.mode_chars.cos_name
서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

mode_def_detail.mode_chars.cryptography
이 모드를 사용하는 세션에서 암호화가 사용되는지의 여부를 지정합니다(AP_NONE 또는 AP_MANDATORY).

mode_def_detail.mode_chars.compression
이 모드를 사용하요 활성화된 세션에 대한 압축 사용 여부를 지정합니다.

AP_COMP_PROHIBITED
RLE 압축이 이 모드에 대한 세션상에서 지원되지 않습니다.

AP_COMP_REQUESTED
이 모드에 대한 세션상에서 RLE 압축이 지원되며 요청(필수는 아님)됩니다.

mode_def_detail.mode_chars.auto_act
이 모드에 대해 자동 활성화될 세션 수를 지정합니다. 이 값은 암시적 CNOS에 사용됩니다.

범위: 0—32767

mode_def_detail.mode_chars.min_consloser_src
이 모드에 대한 임의의 한 국지 LU에 의해 활성화될 최소 회선경합 실패 세션 수를 지정합니다. CNOS(세션 수 변경) 교환이 암시적으로 초기화될 때 이 값이 사용됩니다.

범위: 0—32767

mode_def_detail.mode_chars.max_ru_size_low
이 모드에서 세션상에 송신되거나 수신된 RU들의 최대 크기에 대한 하위 바운드를 지정합니다. 세션 활성화 동안 최대 RU 크기가 조정될 때 이 값이 사용됩니다.

범위: 0—61140

default_ru_size가 AP_YES로 설정되면 필드가 무시됩니다.

mode_def_detail.mode_chars.max_receive_pacing_win
이 모드에서의 세션을 위한 최대 페이싱 창을 지정합니다. 적합 페이싱의 경우, 그것이 부여하는 수신 페이싱 창을 제한하는 데 이 값이 사용됩니다. 고정 페이싱의 경우에는, 이 필드가 사용되지 않습니다. 인접 노드가 그것을 지원하지 않는다고 지정하지 않는 한, 프로그램은 항상 적합 페이싱을 사용함에 주의하십시오.

범위: 0—32767

0의 값은 상한 바운드가 없음을 의미합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MODE_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_MODE_TO_COS_MAPPING

QUERY_MODE_TO_COS_MAPPING은 COS 매핑에 관한 정보를 반환합니다.

정보는 형식된 리스트로 반환됩니다. 특정 모드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **mode_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **mode_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

생략시 COS(알 수 없는 모드가 매핑되는)가 DEFINE_MODE를 사용하여 대체되었으면, QUERY_MODE_TO_COS_MAPPING 역시 널(null) **mode_name**(모두 0)과 생략시 COS를 가진 입력항목을 반환합니다. 정렬하기에서 이 입력항목이 처음입니다.

VCB 구조

```
typedef struct query_mode_to_cos_mapping
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   mode_name[8];       /* mode name                     */
} QUERY_MODE_TO_COS_MAPPING;

typedef struct mode_to_cos_mapping_data
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   mode_name[8];       /* mode name                     */
        unsigned char   cos_name[8];   /* COS name                      */
   unsigned char   reserva[20];        /* reserved                      */
} MODE_TO_COS_MAPPING_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_MODE_TO_COS_MAPPING

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **mode_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

mode_name

세션 그룹에 대한 네트워크 특성을 지정하는 모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다. 생략시 COS에 대한 입력항목을 나타내기 위해 모두 0으로 설정될 수 있습니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

mode_to_cos_mapping_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

mode_to_cos_mapping_data.mode_name
세션 그룹에 대한 네트워크 특성을 지정하는 8 바이트의 모드명. 모두 0으로 설정되면, 생략시 COS에 대한 입력항목을 나타냅니다.

mode_to_cos_mapping_data.cos_name
모드명과 연관된 서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_MODE_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NMVT_APPLICATION

QUERY_NMVT_APPLICATION은 이전에 REGISTER_NMVT_APPLICATION 명령을 발행함으로써 네트워크 관리 벡터 전송(NMVT) 레벨 메세지를 위해 등록한 응용프로그램의 리스트를 반환합니다.(더 자세한 것은 697페이지의 『제 15장 관리 서비스 명령』를 보십시오.)

정보는 리스트로 반환됩니다. 특정 응용프로그램에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **application** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

VCB 구조

```
typedef struct query_nmvt_application
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;            /* pointer to buffer            */
   unsigned long   buf_size;            /* buffer size                  */
   unsigned long   total_buf_size;      /* total buffer size required   */
   unsigned short  num_entries;         /* number of entries            */
   unsigned short  total_num_entries;   /* total number of entries      */
   unsigned char   list_options;        /* listing options              */
   unsigned char   reserv3;             /* reserved                     */
        unsigned char   application[8]; /* application                  */
} QUERY_NMVT_APPLICATION;

typedef struct nmvt_application_data
{
   unsigned short  overlay_size;        /* size of this entry           */
        unsigned char   application[8]; /* application name             */
        unsigned short  ms_vector_key_type; /* MS vector key accepted       */
                                            /* by appl                      */
        unsigned char   conversion_required;
                                            /* conversion to MDS_MU required */
        unsigned char   reserv[5];          /* reserved                     */
   unsigned char   reserva[20];             /* reserved                     */
} NMVT_APPLICATION_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_NMVT_APPLICATION

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램

은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
지정된 **application**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

application
응용프로그램 명. 8 바이트의 유형 A EBCDIC 문자열이거나 모드 EBCDIC 0입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

nmvt_application_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

nmvt_application_data.application
등록된 응용프로그램의 이름. 이 이름은 8 바이트의 영숫자 유형 A EBCDIC 문자열입니다.

nmvt_application_data.ms_vector_key_type
응용프로그램에 의해 승인된 관리 서비스 벡터 키. 응용프로그램이 NMVT 메세지에 대해 등록될 때, 어느 관리 서비스 벡터 키를 승인할지 지정합니다. NMVT 응용프로그램 등록에 관한 더 자세한 정보는 697페이지의 『제15장 관리 서비스 명령』을 보십시오.

nmvt_application_data.conversion_required
등록된 응용프로그램이 메세지를 NMVT에서 MDS_MU 형식으로 변환시켜야 하는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 응용프로그램이 NMVT 메세지에 대해 등록할 때, 이 변환이 필요한지의 여부를 지정할 것입니다. NMVT 응용프로그램 등록에 관한 더 자세한 정보는 697페이지의 『제15장 관리 서비스 명령』을 보십시오.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_APPLICATION_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NN_TOPOLOGY_NODE

이 명령은 통신 서버에만 적용됩니다.

각 네트워크 노드는 네트워크에 있는 네트워크 노드, VRN 및 네트워크 노드 대 네트워크 노드 TG들에 관한 정보를 갖고 있는 네트워크 위상 데이터베이스를 유지보수합니다.

QUERY_NN_TOPOLOGY_NODE는 이 데이터베이스에 있는 네트워크 노드 및 VRN 입력항목에 관한 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **node_type** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드들이 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **node_name**, **node_type** 그리고 **frsn**에 의한 것입니다. **node_name**은 먼저 이름 길이로 정렬되고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). **node_type** 필드는 다음의 순서를 따릅니다. AP_NETWORK_NODE, AP_VRN. **frsn**은 번호로 정렬됩니다.

AP_LIST_INCLUSIVE가 선택되면, 반환되는 리스트가 그 이름의 첫번째 유효 레코드부터 시작합니다.

AP_LIST_FROM_NEXT가 선택되면, 지정된 것 다음의 이름으로 첫번째 유효 레코드에서 리스트가 시작할 것입니다.

frsn 필드(흐름 순서 번호)가 0이 아닌 값으로 설정되면, 이것보다 높은 FRSN을 가진 데이터베이스 입력항목들만이 반환됩니다. 이렇게 하면 노드의 현재 FRSN을 먼저 확보함으로써 많은 “chunks”에서 일관된 위상 데이터베이스가 반환됩니다. 이는 다음과 같이 작동할 것입니다.

1. 노드의 현재 FRSN을 반환하는 QUERY_NODE를 발행합니다.
2. “chunks.”에 있는 모든 데이터베이스 입력항목을 확보하는 데 필요한 만큼의 QUERY_NN_TOPOLOGY_NODE(FRSN이 0으로 설정된 상태로)를 발행합니다.
3. QUERY_NODE를 다시 발행하여 새로운 FRSN을 단계 1에서 반환된 것과 비교합니다.
4. 두 개의 FRSN이 서로 다르면 데이터베이스가 변경된 것이며, 그래서 FRSN을 단계 1에서 제공된 FRSN 보다 1만큼 크게 설정한 상태로 QUERY_NN_TOPOLOGY_NODE를 발행합니다.

VCB 구조

```
typedef struct query_nn_topology_node
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
        unsigned char   node_name[17]; /* network qualified node name   */
        unsigned char   node_type;     /* node type                     */
        unsigned long   frsn;          /* flow reduction sequence num   */
} QUERY_NN_TOPOLOGY_NODE;
```

주: **frsn** 필드가 0이 아닌 값으로 설정되면, 지정된 것보다 큰 FRSN을 가진 노드 입력항목들만이 반환됩니다. 0으로 설정되면 모든 노드 입력항목이 반환됩니다.

```
typedef struct nn_topology_node_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   node_name[17]; /* network qualified node name   */
        unsigned char   node_type;     /* node type                     */
} NN_TOPOLOGY_NODE_SUMMARY;
```

```
typedef struct nn_topology_node_detail
{
   unsigned short  overlay_size;           /* size of this entry           */
        unsigned char   node_name[17];     /* network qualified node name  */
        unsigned char   node_type;         /* node type                    */
        unsigned short  days_left;         /* days left until entry purged */
   unsigned char   reserv1[2];             /* reserved                     */
        unsigned long   frsn;              /* flow reduction sequence num */
        unsigned long   rsn;               /* resource sequence number     */
        unsigned char   rar;               /* route additional resistance */
        unsigned char   status;            /* node status                  */
        unsigned char   function_support;  /* function support             */
   unsigned char   reserv2;                /* reserved                     */
        unsigned char   branch_aware;      /* node is branch aware         */
   unsigned char   reserva[20];            /* reserved                     */
} NN_TOPOLOGY_NODE_DETAIL;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_NN_TOPOLOGY_NODE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

QUERY_NN_TOPOLOGY_NODE

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **node_name**, **node_type** 그리고 **frsn**의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

node_name

네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

node_type

노드의 유형. 이는 다음 값 중 하나일 수 있습니다.

AP_NETWORK_NODE
AP_VRN

node_type을 알 수 없으면, AP_LEARN_NODE가 지정되어야 합니다.

frsn 흐름 감소 순서 번호. 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 노드들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

nn_topology_node_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

nn_topology_node_summary.node_name
네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

nn_topology_node_summary.node_type
노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

nn_topology_node_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

nn_topology_node_detail.node_name
네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

nn_topology_node_detail.node_type
노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

nn_topology_node_detail.days_left
위상 데이터베이스에서 이 노드 입력항목을 삭제하기 전의 일 수. 국지 노드 입력항목에 대해서는 0으로 설정될 것입니다.(이 입력항목은 절대 삭제되지 않습니다.)

nn_topology_node_detail.frsn
흐름 감소 순서 번호. 국지 노드에서 이 자원이 최종 갱신된 시간을 나타냅니다.

nn_topology_node_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

nn_topology_node_detail.rar
노드의 전송경로 추가 저항.

nn_topology_node_detail.status
노드의 상태를 지정합니다. 이는 AP_UNCONGESTED이거나 다음 값 중 하나 또는 그 이상일 수 있습니다.

AP_CONGESTED
ISR 세션의 수가 **isr_sessions_upper_threshold** 보다 큽니다.

AP_ERR_DEPLETED
끝점 세션의 수가 지정된 최대치에 도달했습니다.

AP_IRR_DEPLETED
ISR 세션의 수가 최대치에 도달했습니다.

AP_QUIESCING
STOP_NODE나 유형 AP_QUIESCE 또는 AP_QUIESCE_ISR이 발행되었습니다.

nn_topology_node_detail.function_support
어느 기능이 지원되는지를 지정합니다. 이는 다음 값 중 하나 또는 그 이상일 수 있습니다.

AP_PERIPHERAL BORDER_NODE
주변 경계 노드 기능이 지원됩니다.

AP_EXTENDED BORDER_NODE
확장 경계 노드 기능이 지원됩니다.

AP_CDS
노드가 중앙 디렉토리 서버 기능을 지원합니다.

AP_GATEWAY
노드가 게이트웨이 노드입니다.(이 기능은 아직 구조적으로 정의되지 않습니다.)

AP_INTERCHANGE_NODE
이 노드가 게이트웨이 노드입니다.(이 기능은 아직 구조적으로 정의되지 않습니다.)

AP_ISR
노드가 중계 세션 경로지정(ISR)을 지원합니다.

AP_HPR
노드가 고성능 경로지정(HPR)의 기본 기능을 지원합니다.

AP_RTP_TOWER
노드가 HPR의 RTP 타워를 지원합니다.

AP_CONTROL_OVER_RTP_TOWER
노드가 RTP 타워를 통한 제어 흐름을 지원합니다.

주: AP_CONTROL_OVER_RTP_TOWER는 AP_HPR과 AP_RTP_TOWER 둘다의 설정값에 해당합니다.

nn_topology_node_detail.branch_aware
노드가 분기를 인식하는지의 여부를 지정합니다.

AP_NO
노드가 분기를 인식하지 못합니다.

AP_YES
노드가 분기를 인식합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_NODE

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NN_TOPOLOGY_STATS

이 명령은 통신 서버에만 적용됩니다.

QUERY_NN_TOPOLOGY_STATS는 위상 데이터베이스에 관한 통계 정보를 반환하며 네트워크 노드에서만 발행됩니다.

VCB 구조

```
typedef struct query_nn_topology_stats
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        unsigned long   max_nodes;          /* max num of nodes in database  */
        unsigned long   cur_num_nodes;      /* current number of nodes in    */
                                            /* database                      */
        unsigned long   node_in_tdus;       /* number of TDUs received       */
        unsigned long   node_out_tdus;      /* number of TDUs sent           */
        unsigned long   node_low_rsns;      /* node updates received with    */
                                            /* low RSNs                      */
        unsigned long   node_equal_rsns;    /* node updates in with          */
                                            /* equal RSNs                    */
        unsigned long   node_good_high_rsns;
                                            /* node updates in with          */
                                            /* high RSNs                     */
        unsigned long   node_bad_high_rsns;
                                            /* node updates in with          */
                                            /* high and odd RSNs             */
        unsigned long   node_state_updates; /* number of node updates sent   */
        unsigned long   node_errors;        /* number of node entry          */
                                            /* errors found                  */
        unsigned long   node_timer_updates; /* number of node records built  */
                                            /* due to timer updates          */
        unsigned long   node_purges;        /* num node records purged       */
        unsigned long   tg_low_rsns;        /* TG updates received with      */
                                            /* low RSNs                      */
        unsigned long   tg_equal_rsns;      /* TG updates in with equal RSNs */
        unsigned long   tg_good_high_rsns;  /* TG updates in with high RSNs  */
        unsigned long   tg_bad_high_rsns;   /* TG updates in with high       */
                                            /* and odd RSNs                  */
        unsigned long   tg_state_updates;   /* number of TG updates sent     */
        unsigned long   tg_errors;          /* number of TG entry errors     */
                                            /* found                         */
        unsigned long   tg_timer_updates;   /* number of node records        */
                                            /* built due to timer updates    */
        unsigned long   tg_purges;          /* num node records purged       */
        unsigned long   total_route_calcs;  /* num routes calculated for COS */
        unsigned long   total_route_rejs;   /* num failed route calculations */
        unsigned long   total_tree_cache_hits;
                                            /* total num of tree cache hits  */
        unsigned long   total_tree_cache_misses;
                                            /* total num of tree cache       */
                                            /* misses                        */
        unsigned counter
                        total_tdu_wars;    /* total number TDU war           */
   unsigned char   reserva[16];            /* reserved                       */
} QUERY_NN_TOPOLOGY_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_NN_TOPOLOGY_STATS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_OK

max_nodes

위상 데이터베이스에서의 최대 노드 레코드 수(0은 무제한을 의미합니다).

cur_num_nodes

이 노드의 위상 데이터베이스에 있는 현재 노드 수. 이 값이 허용 최대 노드 수를 초과하면, 경고가 발행됩니다.

node_in_tdus

이 노드에 의해 수신된 총 위상 데이터베이스 갱신(TDU) 수.

node_out_tdus

최종 초기화 이래 모든 인접 네트워크 노드로 송신되도록 이 노드에 의해 구축된 총 위상 데이터베이스 갱신(TDU) 수.

node_low_rsns

현재 RSN 보다 작은 RSN으로 이 노드에 의해 수신된 총 위상 노드 갱신 수. 짝수 RSN과 홀수 RSN 모두 이 집계에 포함됩니다.(이 TDU들은 오류가 아니라, TDU들이 모든 인접 네트워크 노드에 대한 브로드캐스트일 때의 결과입니다. 이 노드 위상 데이터베이스에 어떠한 갱신도 발생하지 않으면, 이 노드는 더 높은 RSN을 가진 TDU를 이 낮은 RSN을 송신하는 인접 노드에 송신합니다.)

node_equal_rsns

현재 RSN과 동등한 RSN으로 이 노드에 의해 수신된 총 위상 노드 갱신 수. 짝수 RSN과 홀수 RSN 모두 이 집계에 포함됩니다.(이 TDU들은 오류가 아니라, TDU들이 모든 인접 네트워크 노드에 대한 브로드캐스트일 때의 결과입니다. 이 노드 위상 데이터베이스에 어떠한 갱신도 발생하지 않습니다.)

node_good_high_rsns

현재 RSN 보다 큰 RSN으로 이 노드에 의해 수신된 총 위상 노드 갱신 수. 노드는 자신의 위상을 갱신하고 모든 인접 네트워크 노드에 TDU를 브로드캐스트합니다. 이 갱신의 송신자에게 TDU를 보낼 필요는 없는데, 그 이유는 그 노드가 이미 갱신을 갖고 있기 때문입니다.

QUERY_NN_TOPOLOGY_STATS

node_bad_high_rsns

현재 RSN 보다 큰 홀수 RSN으로 이 노드에 의해 수신된 총 위상 노드 갱신 수. 이러한 갱신들은 APPN 네트워크 노드들 중 하나에 의해 감지된 위상 불일치를 나타냅니다. 노드는 자신의 위상을 갱신하고 모든 인접 네트워크 노드에 TDU를 브로드캐스트합니다.

node_state_updates

APPN 위상 및 경로지정에 영향을 미치는 내부적으로 감지된 노드 상태 변경의 결과로 구축된 총 위상 노드 갱신 수. 갱신들이 TDU들에 의해 모든 인접 네트워크 노드로 송신됩니다.

node_errors

이 노드에 의해 감지된 총 위상 노드 갱신 불일치 수. 이는 이 노드가 위상 데이터베이스를 갱신하려 하고 데이터 불일치를 감지할 때 발생합니다. 이 노드는 다음 홀수 번호로 점증된 현재 RSN으로 TDU를 작성하여 이를 모든 인접 네트워크 노드에 브로드캐스트합니다.

node_timer_updates

타이머 갱신으로 인해 이 노드의 자원에 대해 구축된 총 위상 노드 갱신 수. 갱신들이 TDU들에 의해 모든 인접 네트워크 노드로 송신됩니다. 이러한 갱신은 다른 네트워크 노드들이 그들의 위상 데이터베이스에서 이 노드를 삭제하지 않게 합니다.

node_purges

이 노드의 위상 데이터베이스로부터 일소된 총 위상 노드 레코드 수. 이는 지정된 시간 내에 노드 레코드가 갱신되지 않았을 경우에 발생됩니다. 소유 노드는 네트워크 위상에 보존하려는 자원에 대한 갱신의 브로드캐스트에 책임이 있습니다.

tg_low_rsns

현재 RSN 보다 큰 RSN으로 이 노드에 의해 수신된 총 위상 TG 갱신 수. 짝수 RSN과 홀수 RSN 모두 이 집계에 포함됩니다.(이 TDU들은 오류가 아니라, TDU들이 모든 인접 네트워크 노드에 대한 브로드캐스트일 때의 결과입니다. 이 노드 위상 데이터베이스에 어떠한 갱신도 발생하지 않으면, 이 노드는 더 높은 RSN을 가진 TDU를 이 낮은 RSN을 송신하는 인접 노드에 송신합니다.)

tg_equal_rsns

현재 RSN과 동등한 RSN으로 이 노드에 의해 수신된 총 위상 TG 갱신 수. 짝수 RSN과 홀수 RSN 모두 이 집계에 포함됩니다.(이 TDU들은 오류가 아니라, TDU들이 모든 인접 네트워크 노드에 대한 브로드캐스트일 때의 결과입니다. 이 노드 위상 데이터베이스에 어떠한 갱신도 발생되지 않습니다.)

tg_good_high_rsns

현재 RSN 보다 큰 RSN으로 이 노드에 의해 수신된 총 위상 TG 갱신 수. 노드는 자신의 위상을 갱신하고 모든 인접 네트워크 노드에 TDU를 브로드캐스트합니다.

tg_bad_high_rsns
현재 RSN 보다 큰 홀수 RSN으로 이 노드에 의해 수신된 총 위상 TG 갱신 수. 이러한 갱신들은 APPN 네트워크 노드들 중 하나에 의해 감지된 위상 불일치를 나타냅니다. 노드는 자신의 위상을 갱신하고 모든 인접 네트워크 노드에 TDU를 브로드캐스트합니다.

tg_state_updates
APPN 위상이나 경로지정에 영향을 미치는 내부적으로 감지된 노드 상태 변경의 결과로 구축된 총 위상 TG 갱신 수. 갱신들이 TDU들에 의해 모든 인접 네트워크 노드로 송신됩니다.

tg_errors
이 노드에 의해 감지된 총 위상 TG 갱신 불일치 수. 이는 이 노드가 위상 데이터베이스를 갱신하려 하고 데이터 불일치를 감지할 경우 발생됩니다. 이 노드는 다음 홀수 번호로 점증된 현재 RSN으로 TDU를 작성하여 이를 모든 인접 네트워크 노드에 브로드캐스트합니다.

tg_timer_updates
타이머 갱신으로 인해 이 노드의 자원에 대해 구축된 총 위상 TG 갱신 수. 갱신들이 TDU들에 의해 모든 인접 네트워크 노드로 송신됩니다. 이러한 갱신은 다른 네트워크 노드들이 그들의 위상 데이터베이스에서 이 노드를 삭제하지 않게 합니다.

tg_purges
이 노드의 위상 데이터베이스로부터 일소된 총 위상 TG 레코드 수. 이는 지정된 시간 내에 노드 레코드가 갱신되지 않았을 경우에 발생됩니다. 소유 노드는 네트워크 위상에 보존하려는 자원에 대한 갱신의 브로드캐스트에 책임이 있습니다.

total_route_calcs
마지막 이래 모든 서비스 클래스(COS)에 대해 계산된 전송경로 수.

total_route_rejs
최종 초기화 이래 계산될 수 없었던 모든 서비스 클래스(COS)에 대한 전송경로 요청 수.

total_tree_cache_hits
캐쉬(cache)된 경로지정 트리에 의해 충족된 전송경로 계산 수. 각 전송경로가 여러 가지 트리의 검열을 요구할 수 있으므로, 총 계산된 전송경로 수보다 이 수가 클수도 있다는 데 주의하십시오.

total_tree_cache_misses
캐쉬(cache)된 경로지정 트리에 의해 충족되지 않았으며 따라서 새로운 경로지정 트리가 구축되어야 했던 전송경로 계산 수.

total_tdu_wars
국지 노드가 감지하고 방해한 TDU war들의 수.

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NN_TOPOLOGY_TG

이 명령은 통신 서버에만 적용됩니다.

각 네트워크 노드는 네트워크에 있는 네트워크 노드, VRN 및 네트워크 노드 대 네트워크 노드 TG들에 관한 정보를 갖고 있는 네트워크 위상 데이터베이스를 유지보수합니다. QUERY_NN_TOPOLOGY_TG는 이 데이터베이스에 있는 TG 입력항목들에 대한 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **owner**, **owner_type**, **dest**, **dest_type**, **tg_num** 및 **frsn** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드들이 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **owner**, **owner_type**, **dest**, **dest_type**, **tg_num** 및 **frsn**에 의한 것입니다. **owner** 이름과 **dest** 이름은 먼저 이름 길이로 정렬되고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). **owner_type** 그리고 **dest_type**은 다음 정렬을 따릅니다: AP_NETWORK_NODE, AP_VRN. **tg_num**과 **frsn**은 번호에 의해 정렬됩니다.

AP_LIST_INCLUSIVE가 선택되면, 반환되는 리스트가 그 이름의 첫번째 유효 레코드부터 시작합니다.

AP_LIST_FROM_NEXT가 선택되면, 지정된 것 다음의 이름으로 첫번째 유효 레코드에서 리스트가 시작할 것입니다.

frsn 필드(흐름 순서 번호)가 0이 아닌 값으로 설정되면, 이것보다 높은 FRSN을 가진 데이터베이스 입력항목들만이 반환됩니다. 이렇게 하면 노드의 현재 FRSN을 먼저 확보함으로써 많은 “chunks”에서 일관된 위상 데이터베이스가 반환됩니다. 이는 다음과 같이 작동합니다.

1. 노드의 현재 FRSN이 반환되는 QUERY_NODE를 발행합니다.
2. “chunks.”에 있는 모든 데이터베이스 입력항목을 확보하는 데 필요한 만큼의 QUERY_NN_TOPOLOGY_TG(FRSN이 0으로 설정된 상태로)를 발행합니다.
3. QUERY_NODE를 다시 발행하여 새로운 FRSN을 단계 1에서 반환된 것과 비교합니다.
4. 두 개의 FRSN이 서로 다르면 데이터베이스가 변경된 것이며, 그래서 FRSN을 단계 1에서 제공된 FRSN 보다 1만큼 크게 설정한 상태로 QUERY_NN_TOPOLOGY_TG를 발행합니다.

VCB 구조

typedef struct query_nn_topology_tg { unsigned short opcode; /* verb operation code */ unsigned char reserv2; /* reserved */ unsigned char format; /* format */ unsigned short primary_rc; /* primary return code */ unsigned long secondary_rc; /* secondary return code */ unsigned char *buf_ptr; /* pointer to buffer */ unsigned long buf_size; /* buffer size */ unsigned long total_buf_size; /* total buffer size required */ unsigned short num_entries; /* number of entries */ unsigned short total_num_entries; /* total number of entries */ unsigned char list_options; /* listing options */ unsigned char reserv3; /* reserved */ unsigned char owner[17]; /* node that owns the TG */ unsigned char owner_type; /* type of node that owns the TG */ unsigned char dest[17]; /* TG destination node */ unsigned char dest_type; /* TG destination node type */ unsigned char tg_num; /* TG number */ unsigned char reserv1; /* reserved */ unsigned long frsn; /* flow reduction sequence num */ } QUERY_NN_TOPOLOGY_TG;

```
typedef struct topology_tg_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
        unsigned char   owner[17];     /* node that owns the TG         */
        unsigned char   owner_type;    /* type of node that owns the TG */
        unsigned char   dest[17];      /* TG destination node           */
        unsigned char   dest_type;     /* TG destination node type      */
        unsigned char   tg_num;        /* TG number                     */
   unsigned char   reserv3[1];         /* reserved                      */
        unsigned long   frsn;          /* flow reduction sequence num   */
} TOPOLOGY_TG_SUMMARY;
```

```
typedef struct topology_tg_detail
{
   unsigned short  overlay_size;           /* size of this entry          */
        unsigned char   owner[17];         /* node that owns the TG       */
        unsigned char   owner_type;        /* type of node that owns the TG*/
        unsigned char   dest[17];          /* TG destination node         */
        unsigned char   dest_type;         /* TG destination node type    */
        unsigned char   tg_num;            /* TG number                   */
   unsigned char   reserv3[1];             /* reserved                    */
        unsigned long   frsn;              /* flow reduction sequence num */
        unsigned short  days_left;         /* days left until entry purged */
        LINK_ADDRESS    dlc_data           /* DLC signalling data         */
        unsigned long   rsn;               /* resource sequence number    */
        unsigned char   status;            /* node status                 */
        TG_DEFINED_CHARS tg_chars;         /* TG characteristics          */
        unsigned char   subarea_number[4];
                                           /* subarea number              */
        unsigned char   tg_type;           /* TG type                     */
        unsigned char   intersubnet_tg;    /* intersubnet TG?             */
        unsigned char   cp_cp_session_active;
                                           /* CP-CP session is active     */
        unsigned char   branch_tg;         /* TG is a branch TG           */
   unsigned char   reserva[12];            /* reserved                    */
} TOPOLOGY_TG_DETAIL;
```

```
typedef struct link_address
{
        unsigned short  length;        /* length                          */
        unsigned short  reserve1;      /* reserved                        */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                       /* address                         */
} LINK_ADDRESS;
```

주: **frsn** 필드가 0이 아닌 값으로 설정되면, 그 FRSN을 가진 노드 입력항목들만이 반환됩니다. 0으로 설정되면 모든 노드 입력항목이 반환됩니다.

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_NN_TOPOLOGY_TG

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **owner**, **owner_type**, **dest**, **dest_type**, **tg_num** 및 **frsn**의 조합(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

owner TG의 근원지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로

구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

owner_type
TG를 소유하는 노드의 유형. 이는 다음 값 중 하나일 수 있습니다.

AP_NETWORK_NODE
AP_VRN

owner_type을 알 수 없는 경우에는, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dest TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)**list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dest_type
이 TG에 대한 목적지 노드의 유형. 이는 다음 값 중 하나일 수 있습니다.

AP_NETWORK_NODE
AP_VRN

dest_type을 알 수 없는 경우에는, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

tg_num
TG와 연관된 번호. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

frsn 흐름 감소 순서 번호. 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 노드들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

topology_tg_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

topology_tg_summary.owner
TG의 근원지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

topology_tg_summary.owner_type
TG를 소유하는 노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_summary.dest
TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

topology_tg_summary.dest_type
이 TG에 대한 목적지 노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_summary.tg_num
TG와 연관된 번호.

topology_tg_summary.frsn
흐름 감소 순서 번호. 국지 노드에서 이 자원이 최종 갱신된 시간을 나타냅니다.

topology_tg_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

topology_tg_detail.owner
TG의 근원지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

topology_tg_detail.owner_type

TG를 소유하는 노드의 유형. 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_detail.dest

TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를

topology_tg_detail.tg_chars
TG 특성.

topology_tg_detail.subarea_number
TG의 소유자나 목적지 노드가 부속영역 가능하다면, 이 필드에 부속영역 가능 노드상의 이 TG와 연관된 링크 스테이션을 소유하는 유형 4나 유형 5의 부속영역 번호가 포함됩니다. 그렇지 않으면, 이 필드가 모드 2진 0으로 설정됩니다.

topology_tg_detail.tg_type
TG 유형. 이 필드는 다음 값 중 하나를 취합니다.

AP_APPN_OR_BOUNDARY_TG
APPN TG 또는 경계 기능 기반의 TG

AP_INTERCHANGE_TG
상호교환 TG

AP_VIRTUAL_ROUTE_BASED_TG
가상 경로지정 기반의 TG

AP_UNKNOWN
위상에서 보고된 이 TG의 TG 유형을 알 수 없습니다.

topology_tg_detail.intersubnet.tg
이 TG가 서브네트워크 간의 TG입니까?

AP_YES
AP_NO

topology_tg_detail.cp_cp_session_active
소유 노드의 회선경합 성공 세션 CP-CP 세션이 활동중인지의 여부를 지정합니다(AP_NO 또는 AP_YES).

branch_tg
TG가 분기 TG인지의 여부를 지정합니다.

AP_NO
TG가 분기 TG가 아닙니다.

AP_YES
TG가 분기 TG입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TG

AP_INVALID_ORIGIN_NODE
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_NODE

QUERY_NODE는 노드 특정 정보 및 통계를 반환합니다. 실행 동안 동적으로 결정된 정보를 반환하는 것외에, QUERY_NODE는 노드 초기화 동안 설정되는 매개변수 역시 반환합니다.

VCB 구조

형 D 2

```
typedef struct query_node
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        CP_CREATE_PARMS cp_create_parms;    /* create parameters            */
        unsigned long   up_time;            /* time since node started      */
        unsigned long   mem_size;           /* size of memory available     */
        unsigned long   mem_used;           /* size of memory used          */
        unsigned long   mem_warning_threshold;
                                            /* memory constrained           */
                                            /* threshold                    */
        unsigned long   mem_critical_threshold;
                                            /* memory critical threshold    */
        unsigned char   nn_functions_supported;
                                            /* NN functions supported       */
        unsigned char   functions_supported;
                                            /* functions supported          */
        unsigned char   en_functions_supported;
                                            /* EN functions supported       */
        unsigned char   nn_status;          /* node status.  One or more of */
        unsigned long   nn_frsn;            /* NN flow reduction            */
                                            /* sequence number              */
        unsigned long   nn_rsn;             /* Resource sequence number     */
        unsigned short  def_ls_good_xids;   /* Good XIDs for defined        */
                                            /* link stations                */
        unsigned short  def_ls_bad_xids;    /* Bad XIDs for defined         */
                                            /* link stations                */
        unsigned short  dyn_ls_good_xids;   /* Good XIDs for dynamic        */
                                            /* link stations                */
        unsigned short  dyn_ls_bad_xids;    /* Bad XIDs for dynamic         */
                                            /* link stations                */
        unsigned char   dlur_release_level; /* Current DLUR release level   */
        unsigned char   nns_dlus_served_lu_reg_supp;
                                            /* NNS support for registration */
                                            /* of DLUS-served LUs reserved  */
        unsigned char   reserva[19];        /* reserved                     */
       unsigned char   fq_nn_server_name[17];
                                            /* FQ name of NN server         */
       unsigned long   current_isr_sessions;/* current ISR sessions         */
       unsigned char   nn_functions2;       /* NN functions continued       */
       unsigned char   branch_ntwk_arch_version;
                                            /* branch network architecture  */
                                            /* version supported            */
       unsigned char   reservb[28];         /* reserved                     */
} QUERY_NODE;

typedef struct cp_create_parms
{
        unsigned short  crt_parms_len;      /* length of CP_CREATE_PARMS    */
   unsigned char   description[RD_LEN];
                                            /* resource description         */
        unsigned char   node_type;          /* node type                    */
        unsigned char   fqcp_name[17];      /* fully qualified CP name      */
        unsigned char   cp_alias[8];        /* CP alias                     */
```

```
        unsigned char   mode_to_cos_map_supp;
                                        /* mode to COS mapping support  */
        unsigned char   mds_supported;  /* MDS and MS capabilities      */
        unsigned char   node_id[4];     /* node ID                      */
        unsigned short  max_locates;    /* max locates node can process */
        unsigned short  dir_cache_size; /* directory cache size         */
                                        /* (reserved) if not NN)        */
        unsigned short  max_dir_entries;/* max directory entries        */
        unsigned short  locate_timeout; /* locate timeout in seconds    */
        unsigned char   reg_with_nn;    /* register resources with NNS  */
        unsigned char   reg_with_cds;   /* resource registration with   */
                                        /* CDS                          */
        unsigned short  mds_send_alert_q_size;
                                        /* size of MDS send alert queue */
        unsigned short  cos_cache_size; /* number of COS definitions    */
        unsigned short  tree_cache_size;/* Topology Database routing    */
                                        /* tree cache size              */
        unsigned short  tree_cache_use_limit;
                                        /* num times tree can be used   */
        unsigned short  max_tdm_nodes;  /* max num nodes that can be    */
                                        /* stored in Topology Database  */
        unsigned short  max_tdm_tgs;    /* max num TGs that can be      */
                                        /* stored in Topology Database  */
        unsigned long   max_isr_sessions;/* max ISR sessions            */
        unsigned long   isr_sessions_upper_threshold;
                                        /* upper threshold for ISR sess */
        unsigned long   isr_sessions_lower_threshold;
                                        /* lower threshold for ISR sess */
        unsigned short  isr_max_ru_size;/* max RU size for ISR          */
        unsigned short  isr_rcv_pac_window; /* ISR rcv pacing window size   */
        unsigned char   store_endpt_rscvs;  /* endpoint RSCV storage        */
        unsigned char   store_isr_rscvs;/* ISR RSCV storage             */
        unsigned char   store_dlur_rscvs;/* DLUR RSCV storage           */
        unsigned char   dlur_support;   /* is DLUR supported?           */
        unsigned char   pu_conc_support;/* is PU conc supported?        */
        unsigned char   nn_rar;         /* Route additional resistance  */
        unsigned char   hpr_support;    /* level of HPR support         */
        unsigned char   mobile;         /* HPR path-switch controller?  */
        unsigned char   discovery_support;  /* Discovery function utilized  */
        unsigned char   discovery_group_name[8];
                                        /* Group name for Discovery     */
        unsigned char   implicit_lu_0_to_3;
                                        /* Implicit LU 0 to 3 support   */
        unsigned char   default_preference;
                                        /* Default routing preference   */
        unsigned char   anynet_supported;
                                        /* level of AnyNet support      */
        unsigned short  max_ls_exception_events;
                                        /* maximum LS Exception events  */
        unsigned char   comp_in_series; /* compression in series allowed*/
        unsigned char   max_compress_lvl;/* maximum compression level   */
        unsigned char   node_spec_data_len; /* length of node specific data */
        unsigned char   ptf[64];        /* program temporary fix array  */
} CP_CREATE_PARMS;
```

형 D 1(이전 레벨)

```
typedef struct query_node
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        CP_CREATE_PARMS cp_create_parms;    /* create parameters           */
        unsigned long   up_time;            /* time since node started     */
        unsigned long   mem_size;           /* size of memory available    */
        unsigned long   mem_used;           /* size of memory used         */
        unsigned long   mem_warning_threshold;
```

```
                                        /* memory constrained           */
                                        /* threshold                    */
        unsigned long   mem_critical_threshold;
                                        /* memory critical threshold    */
        unsigned char   nn_functions_supported;
                                        /* NN functions supported       */
        unsigned char   functions_supported;
                                        /* functions supported          */
        unsigned char   en_functions_supported;
                                        /* EN functions supported       */
        unsigned char   nn_status;      /* node status.  One or more of */
        unsigned long   nn_frsn;        /* NN flow reduction            */
                                        /* sequence number              */
        unsigned long   nn_rsn;         /* Resource sequence number     */
        unsigned short  def_ls_good_xids;   /* Good XIDs for defined    */
                                        /* link stations                */
        unsigned short  def_ls_bad_xids;    /* Bad XIDs for defined     */
                                        /* link stations                */
        unsigned short  dyn_ls_good_xids;   /* Good XIDs for dynamic    */
                                        /* link stations                */
        unsigned short  dyn_ls_bad_xids;    /* Bad XIDs for dynamic     */
                                        /* link stations                */
        unsigned char   dlur_release_level; /* Current DLUR release level */
        unsigned char   reserva[19];    /* reserved                     */
} QUERY_NODE;
```

형 D 0(이전 레벨)

```
typedef struct query_node
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
        CP_CREATE_PARMS cp_create_parms;    /* create parameters          */
        unsigned long   up_time;            /* time since node started    */
        unsigned long   mem_size;           /* size of memory available   */
        unsigned long   mem_used;           /* size of memory used        */
        unsigned long   mem_warning_threshold;
                                            /* memory constrained         */
                                            /* threshold                  */
        unsigned long   mem_critical_threshold;
                                            /* memory critical threshold  */
        unsigned char   nn_functions_supported;
                                            /* NN functions supported     */
        unsigned char   functions_supported;
                                            /* functions supported        */
        unsigned char   en_functions_supported;
                                            /* EN functions supported     */
        unsigned char   nn_status;          /* node status.  One or more of */
        unsigned long   nn_frsn;            /* NN flow reduction          */
                                            /* sequence number            */
        unsigned long   nn_rsn;             /* Resource sequence number   */
        unsigned short  def_ls_good_xids;   /* Good XIDs for defined      */
                                            /* link stations              */
        unsigned short  def_ls_bad_xids;    /* Bad XIDs for defined       */
                                            /* link stations              */
        unsigned short  dyn_ls_good_xids;   /* Good XIDs for dynamic      */
                                            /* link stations              */
        unsigned short  dyn_ls_bad_xids;    /* Bad XIDs for dynamic       */
                                            /* link stations              */
        unsigned char   dlur_release_level; /* Current DLUR release level */
        unsigned char   reserva[19];        /* reserved                   */
} QUERY_NODE;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_NODE

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

이 필드가 0으로 설정되는 경우, 다음의 네개 필드는 Unsigned_COUNTER라기 보다는 Unsigned short입니다. **def_Is_good_xids**, **def_Is_bad_xids**,**dyn_Is_good_xids**, **dyn_Is_bad_xids**.

이 필드가 2로 설정되면, 기술된 대로 다음 필드들이 사용됩니다. **fq_nn_server_name** 그리고 **current_isr_sessions**.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

cp_create_parms.crt_parms_len
매개변수 작성 구조체의 길이.

cp_create_parms.description
자원 설명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

cp_create_parms.node_type
이는 항상 다음과 같습니다.

AP_END_NODE
AP_NETWORK_NODE
AP_LEN_NODE
AP_BRANCH_NETWORK_NODE

cp_create_parms.fqcp_name
노드의 17 바이트 전체 정식 제어점(CP) 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가집니다.)

cp_create_parms.cp_alias
국지로 사용되는 제어점(CP) 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

cp_create_parms.mode_to_cos_map_supp
노드에 의해 COS 매핑에 대한 모드가 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). AP_YES로 설정되면 DEFINE_MODE 명령상에서 지정된 COS가 SNA에 의해 정의된 COS이거나 DEFINE_COS 명령을 발행하여 정의되었어야 합니다.

cp_create_parms.mds_supported
관리 서비스가 다중 도메인 지원 및 관리 서비스 기능을 지원하는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

cp_create_parms.node_id
XID 교환에서 사용된 노드 식별자. 이는 4 바이트의 16진 문자열입니다.

cp_create_parms.max_locates
노드가 처리할 수 있는 최대 위치 수.

cp_create_parms.dir_cache_size
네트워크 노드만: 디렉토리 캐쉬(cache)의 크기.

cp_create_parms.max_dir_entries
최대 디렉토리 입력항목 수. 이 필드가 0으로 설정되면 무제한입니다.

cp_create_parms.locate_timeout
네트워크 탐색이 시간종료하기 전까지의 시간(초 단위)을 지정합니다. 0의 값은 탐색에 시간종료가 없음을 나타냅니다.

cp_create_parms.reg_with_nn
자원이 네트워크 노드 서버에 등록될 것인지의 여부를 지정합니다. 등록이 실패해도 성공적인 노드 초기화 완료에는 영향을 미치지 않습니다. 자세한 것은 622페이지의 『REGISTRATION_FAILURE』를 보십시오. 이 필드는 EN과 BrNN에 의해 서로 다르게 해석됩니다.

끝 노드:

AP_NO
노드가 NN 서버와 어떠한 LU도 등록하지 않습니다. NNS는 모든 브로드캐스트 탐색을 끝 노드에 이송합니다.

AP_YES
노드가 모든 국지 종속 LU(NNS가 옵션 세트 1116을 지원하면) 및 모든 국지 독립 LU를 NNS에 등록합니다. NNS는 자기에게 지정된 위치를 이송만 합니다(등록될 수 없었던 종속 LU를 소유하지 않는 한).

분기 네트워크 노드:

AP_REGISTER_NONE
노드가 NN 서버와 어떠한 LU도 등록되지 않습니다.

AP_REGISTER_ALL
노드가 모든 국지 종속 LU(DLUR 완전 다중 서브네트를 지원하고 NNS가 옵션 세트 1116을 지원하면) 및 모든 도메인 독립 LU를 NNS에 등록합니다.

AP_REGISTER_LOCAL_ONLY
노드가 모든 국지 종속 LU(DLUR 완전 다중 서브네트를 지원하고 NNS가 옵션 세트 1116을 지원하면) 및 모든 국지 독립 LU를 NNS에 등록합니다.

cp_create_parms.reg_with_cds
자원이 중앙 디렉토리 서버(CDS)에 등록되도록 허용되는지의 여부를 지정합니다. 이 필드는 EN, NN 또는 BrNN에 의해 서로 다르게 해석됩니다.

끝 노드: NNS가 CDS 끝 노드 자원에 등록하도록 허용되는지의 여부를 지정합니다. **reg_with_nn**이 AP_NONE으로 설정되면, 이 필드가 무시됩니다.

AP_NO
EN 자원이 CDS에 등록될 수 없습니다.

AP_YES
EN 자원이 CDS에 등록될 수 있습니다.

네트워크 노드: 국지 자원 및 도메인 자원(소유 EN이 CDS에 등록되도록 허용하는)이 CDS에 등록될 수 있는지의 여부를 지정합니다.

AP_NO
국지나 도메인 자원이 CDS에 등록될 수 없습니다.

AP_YES
국지나 도메인 자원이 CDS에 등록될 수 있습니다. 등록이 실패해도 성공적인 START_NODE 명령 완료에는 영향을 미치지 않습니다.

분기 네트워크 노드: NNS가 CDS BrNN 자원(BrNN에 국지이거나 BrNN의 도메인으로부터)에 등록되도록 허용되는지의 여부를 지정합니다. **reg_with_nn**이 AP_NO로 설정되면, 이 필드가 무시됩니다.

AP_REGISTER_NONE
노드가 NN 서버와 어떠한 LU도 등록되지 않습니다.

AP_REGISTER_ALL
노드가 모든 국지 종속 LU(DLUR 완전 다중 서브네트를 지원하고 NNS가 옵션 세트 1116을 지원하면) 및 모든 도메인 독립 LU를 NNS에 등록됩니다.

AP_REGISTER_LOCAL_ONLY
노드가 모든 국지 종속 LU(DLUR 완전 다중 서브네트를 지원하고 NNS가 옵션 세트 1116을 지원하면) 및 모든 국지 독립 LU를 NNS에 등록됩니다.

cp_create_parms.mds_send_alert_q_size
MDS 경고 송신 대기행렬의 크기. 이 한계에 도달할 때, MDS 구성 요소는 대기행렬에서 가장 오래된 입력항목을 삭제합니다.

cp_create_parms.cos_cache_size
COS 데이터베이스 가중치 캐쉬(cache)의 크기.

cp_create_parms.tree_cache_size
위상 데이터베이스 경로지정 캐쉬(cache)의 크기.

cp_create_parms.tree_cache_use_limit
캐쉬(cache)된 트리의 최대 사용 수. 이 수에 도달하면 트리가 버려지고 재연산됩니다. 이는 노드가 동일한 가중치 전송경로간에 세션의 균형을 맞추도록 허용합니다. 낮은 값은 증가된 활성화 지연 시간의 비용으로 더 나은 로드 균형을 제공합니다.

cp_create_parms.max_tdm_nodes
위상 데이터베이스에서 보관될 수 있는 최대 노드 수(0은 무제한을 의미합니다).

cp_create_parms.max_tdm_tgs
위상 데이터베이스에서 보관될 수 있는 최대 TG 수(0은 무제한을 의미합니다).

cp_create_parms.max_isr_sessions
노드가 한번에 참여할 수 있는 최대 ISR 세션 수.

cp_create_parms.isr_sessions_upper_threshold
cp_create_parms.isr_sessions_lower_threshold를 보십시오.

cp_create_parms.isr_sessions_lower_threshold
노드의 과잉 상태를 제어하는 상위 및 하위 최소 임계값. ISR 세션 수가 상위 최소 임계값을 초과하면 노드 상태가 비과잉에서 과잉으로 바뀝니다. 일단 ISR 세션이 하위 최소 임계값 밑으로 내려가면 노드 상태가 다시 비과잉으로 바뀝니다.

cp_create_parms.isr_max_ru_size
중간 세션을 위해 지원되는 최대 RU 크기.

cp_create_parms.isr_rcv_pac_window
중간 세션을 위해 제안되는 수신 페이싱 창 크기. 이 값은 인접 노드가 적합 페이싱을 지원하지 않는 경우에 중간 세션의 2차 홉상에서만 사용됩니다.

cp_create_parms.store_endpt_rscvs
RSCV들이 진단용으로 보관되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

cp_create_parms.store_isr_rscvs
RSCV들이 진단용으로 보관되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

cp_create_parms.store_dlur_rscvs
노드가 진단용으로 RSCV들을 지원하는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드가 AP_YES로 설정되면, QUERY_DLUR_LU 명령에서 RSCV가 반환됩니다.

cp_create_parms.dlur_support
노드가 제공하는 DLUR에 대한 지원 레벨을 지정합니다. 이는 한 비트 필드이며 다음 값을 취할 수도 있습니다.

AP_NO
DLUR이 지원되지 않습니다.

AP_YES
DLUR 완전 다중 서브네트가 지원됩니다.

(AP_YES | AP_LIMITED_DLUR_MULTI_SUBNET)
DLUR 제한, DLUR 다중 서브네트가 지원됩니다. 이는 노드가 끝 노드인 경우에만 유효합니다.

cp_create_parms.pu_conc_support
PU 집중(concentration)이 지원되는지의 여부를 지정합니다(항상 AP_NO).

cp_create_parms.nn_rar
네트워크 노드의 전송경로 추가 저항.

cp_create_parms.hpr_support
노드가 제공하는 HPR에 대한 지원 레벨을 지정합니다(AP_NONE, AP_BASE 또는 AP_RTP).

cp_create_parms.mobile
노드가 HPR 경로 전환 제어기인지의 여부를 지정합니다(AP_YES 또는 AP_NO). **cp_create_parms.hpr_support** 필드가 AP_RTP로 설정되지 않으면, 이 필드가 예약됩니다.

cp_create_parms.discovery_support
이 노드에 의해 발견(Discovery) 기능이 활용되는지의 여부를 지정합니다.

AP_DISCOVERY_CLIENT
이 노드에 의해 발견(Discovery) 클라이언트 기능이 사용됩니다.

AP_DISCOVERY_SERVER
이 노드에 의해 발견(Discovery) 서버 기능이 사용됩니다.

cp_create_parms.discovery_group_name
노드가 활용하는 발견(Discovery) 기능에서 사용된 그룹 이름을 지정합니다. 이 필드가 모두 0으로 설정되면, 생략시 그룹 이름이 사용됩니다.

cp_create_parms.implicit_lu_0_to_3
노드가 ACTLU에 의해 유형 0에서 3의 암시적 LU 정의를 지원하는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

cp_create_parms.default_preference
이 노드로부터 세션 초기화시 선호 경로지정 방식을 지정합니다.

주: 이는 DEFINE_PARTNER_LU 명령을 사용하여 LU 당 기준으로 대체될 수 있습니다.

이 필드는 다음 값들을 취할 수 있습니다.

AP_NATIVE
원시(APPN) 경로지정 프로토콜만을 사용합니다.

AP_NONNATIVE
비원시(AnyNet) 경로지정 프로토콜만을 사용합니다.

AP_NATIVE_THEN_NONNATIVE
원시(APPN) 프로토콜을 시도하며, 상대방 LU의 위치를 찾을 수 없으면 비원시(AnyNet) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

AP_NONNATIVE_THEN_NATIVE
비원시(AnyNet) 프로토콜을 시도하며, 상대방 LU의 위치를 찾을 수 없으면 원시(APPN) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

주: 나중 세 개의 값은 노드 연산자 기능에 AnyNet DLC가 사용가능하고 정의된 AnyNet 링크 스테이션이 있을 때에만 의미가 있습니다.

cp_create_parms.anynet_supported
AnyNet DLC를 위한 지원을 지정합니다. 이 필드는 다음 중 하나일 수 있습니다.

AP_NONE
어떤 ANYNET 기능도 지원되지 않을 것입니다. 필드 **default_preference**는 값 AP_NATIVE를 취해야 합니다.

AP_ACCESS_NODE
비원시(AnyNet) 경로지정 프로토콜만을 사용합니다.

AP_NATIVE_THEN_NONNATIVE
이 노드는 ANYNET 액세스 노드 기능을 지원할 것입니다.

AP_GATEWAY
이 노드는 ANYNET 게이트웨이 기능을 시작할 것입니다. 이 값은 **node_type**이 AP_NETWORK_NODE인 경우에만 유효합니다.

cp_create_parms.comp_in_series
RLE 압축이 앞에 붙어있는 LZ 압축의 사용이 허용되는지의 여부를 지정합니다.

AP_YES

AP_NO

cp_create_parms.max_ls_exception_events
노드에 의해 기록된 최대 LS_EXCEPTION 입력항목 수를 지정합니다. 0 - 200의 범위.

cp_create_parms.max_compress_lvl
노드에 의해 지원되는 최대 압축 레벨.

AP_NONE
노드가 압축을 지원하지 않습니다.

AP_RLE_COMPRESSION
노드가 LU 6.2 세션에서 RLE 압축이나 압축해제를 지원하며 전통적 LU 세션에서 RLE 압축이나 LZ9 압축해제를 지원합니다.

AP_LZ9_COMPRESSION
노드가 LZ9와 RLE 압축이나 압축해제를 지원할 수 있습니다.

AP_LZ10_COMPRESSION
노드가 LZ10, LZ9 그리고 RLE 압축이나 압축해제를 지원할 수 있습니다.

AP_LZ12_COMPRESSION
노드가 LZ12, LZ10, LZ9 그리고 RLE 압축이나 압축해제를 지원할 수 있습니다.

cp_create_parms.node_spec_data_len
이 필드는 항상 0으로 설정되어야 합니다.

cp_create_parms.ptf
향후 프로그램 임시 수정(PTF) 작업 구성이나 제어를 위한 배열.

cp_create_parms.ptf[0]
REQDISCONT 지원. 퍼스널 통신이나 통신 서버는 보통 REQDISCONT를 사용하여 더 이상 세션 트래픽이 필요로 하지 않는 제한된 자원 호스트 링크를 활성종료합니다. 퍼스널 통신이나 통신 서버의 REQDISCONT 사용을 억제하거나, 퍼스널 통신이나 통신 서버에 의해 송신된 REQDISCONT에서 사용된 설정값을 수정하는 데 이 바이트가 사용될 수 있습니다.

AP_SUPPRESS_REQDISCONT
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 REQDISCONT를 사용하지 않습니다.(이 바이트의 모든 다른 비트들은 무시됩니다.)

AP_OVERRIDE_REQDISCONT
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 다음 두 비트에 기초하여 REQDISCONT에 있는 일반 설정값을 대체합니다.

AP_REQDISCONT_TYPE
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 REQDISCONT에서 “immediate”의 유형을 지정합니다. 그렇지

않으면, 퍼스널 통신이나 통신 서버가 “normal”의 유형을 지정합니다.(AP_OVERRIDE_REQDISCONT가 설정되지 않으면 이 비트가 무시됩니다.)

AP_REQDISCONT_RECONTACT
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 REQDISCONT에서 “immediate recontact”의 유형을 지정합니다. 그렇지 않으면, 퍼스널 통신이나 통신 서버가 “no immediate recontact”를 지정합니다.(AP_OVERRIDE_REQDISCONT가 설정되지 않으면 이 비트가 무시됩니다.)

cp_create_parms.ptf[1]
ERP 지원.

퍼스널 통신이나 통신 서버는 보통 ACTPU(ERP)를 ERP로 처리합니다.(ACTPU(ERP)는 PU-SSCP 세션이 재설정될 것을 요청하지만, ACTPU(cold)와는 달리, 공헌이 되는 LU-SSCP 및 PLU-SLU 세션들의 암시적 활성종료를 요청하지 않습니다.) SNA 구현은 ACTPU(ERP)를 마치 ACTPU(cold)인 것처럼 적법하게 처리할 수 있습니다.

AP_OVERRIDE_ERP
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 모든 ACTPU 요청을 ACTPU(cold)로 처리합니다.

cp_create_parms.ptf[2]
BIS 지원.

퍼스널 통신이나 통신 서버는 보통 제한된 자원 LU 6.2 세션을 활성종료하기 전에 BIS 프로토콜을 사용합니다. 이 바이트는 BIS의 사용이 대체되게 허용합니다.

AP_SUPPRESS_BIS
이 비트가 설정되면, 퍼스널 통신이나 통신 서버가 BIS 프로토콜을 사용하지 않습니다. 제한된 자원 LU 6.2 세션이 UNBIND(cleanup)를 사용하여 즉시 활성종료됩니다.

up_time
노드가 시작(또는 재시작)된 이래의 시간(100분의 1초 단위).

mem_size
하부 운영체제로부터 기억영역 관리에 의해 확보된 대로의 가용 기억영역 크기.

mem_used
프로세스에 현재 할당되는 기억영역의 바이트 수.

mem_warning_threshold
기억영역 관리를 넘어서 기억영역 자원이 한정된 것으로 여기는 할당 임계값.

mem_critical_threshold
기억영역 관리를 넘어서 기억영역 자원이 심각하게 한정된 것으로 여기는 할당 임계값.

nn_functions_supported
예약됨.

functions_supported
어느 기능이 지원되는지를 지정합니다. 이는 다음 값 중 하나 또는 그 이상일 수 있습니다.

AP_NEGOTIABLE_LS
AP_SEGMENT_REASSEMBLY
AP_BIND_REASSEMBLY
AP_PARALLEL_TGS
AP_CALL_IN
AP_ADAPTIVE_PACING
AP_TOPOLOGY_AWARENESS

en_functions_supported
지원되는 끝 노드 기능을 지정합니다.

AP_SEGMENT_GENERATION
노드가 세그멘트 생성을 지원합니다.

AP_MODE_TO_COS_MAP
노드가 COS 명 매핑에 대한 모드명을 지원합니다.

AP_LOCATE_CDINIT
노드가 원격 LU의 위치를 찾기 위한 위치 및 도메인에 걸친 초기화 GDS 변수의 생성을 지원합니다.

AP_REG_WITH_NN
노드가 자신의 LU를 인접 서비스 제공 네트워크 노드에 등록할 것입니다.

AP_REG_CHARS_WITH_NN
노드가 송신 레지스터 특성을 지원합니다.(송신 레지스터 이름 역시 지원되는 경우에만 지원될 수 있습니다.)

nn_status
예약됨.

nn_frsn
예약됨.

nn_rsn
예약됨.

def_ls_good_xids
노드가 마지막으로 시작된 이래 모든 정의된 링크 스테이션에서 발생한 총 성공 XID 교환 수.

def_ls_bad_xids
노드가 마지막으로 시작된 이래 모든 정의된 링크 스테이션에서 발생한 총 비성공 XID 교환 수.

dyn_ls_good_xids
노드가 마지막으로 시작된 이래 모든 동적 링크 스테이션에서 발생한 총 비성공 XID 교환 수.

dyn_ls_bad_xids
노드가 마지막으로 시작된 이래 모든 동적 링크 스테이션에서 발생한 총 비성공 XID 교환 수.

dlur_release_level
현재의 DLUR 릴리스 레벨을 지정합니다.

nns_dlus_served_lu_reg_supp
끝 노드만. 끝 노드의 네트워크 노드 서버가 DLUS의 서비스를 받는 LU 등록을 지원하는지의 여부를 지정합니다.

AP_NO
DLUS의 서비스를 받는 LU 등록이 네트워크 노드 서버에 의해 지원되지 않습니다.

AP_YES
DLUS의 서비스를 받는 LU의 등록이 네트워크 노드 서버에 의해 지원됩니다.

AP_UNKNOWN
끝 노드가 네트워크 노드 서버를 가지지 않습니다.

NN 만: 이 필드가 AP_NO로 설정됩니다.

fq_nn_server_name
현재 네트워크 노드 서버의 전체 정식 17 바이트 길이의 이름. 이는 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

이 노드가 끝 노드가 아니거나 활동중 네트워크 노드 서버를 가지지 않으면, 이 필드가 널(null)로 설정됩니다.

current_isr_sessions

이 노드를 통해 현재 경로지정되는 활동중인 ISR 세션 수. 이 노드가 네트워크 노드가 아니면, 이 필드가 0으로 설정됩니다.

nn_functions2
지원되는 네트워크 노드 기능을 지정합니다.

AP_BRANCH_AWARENESS
노드가 ″분기를 인식합니다(branch aware)″.

branch_ntwk_arch_version
지원되는 분기 네트워크 구조의 버전을 지정하거나 노드가 분기 네트워크 구조를 지원하지 않으면 0입니다.

AP_BRANCH_AWARENESS
노드가 ″분기를 인식합니다(branch aware)″.

QUERY_NODE

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PARTNER_LU

QUERY_PARTNER_LU는 국지 LU에 의해 사용되어온 상대방 LU에 관한 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 상대방 LU에 관한 정보를 확보하거나 여러 개의 『청크(chunks)』로 리스트 정보를 확보하려면, **plu_alias** 필드가 설정되어야 합니다(또는 **plu_alias**가 모두 0으로 설정되면 **fqplu_name**). **list_options** 필드가 AP_FIRST_IN_LIST로 설정되면, 두 필드 모두 무시됩니다. **lu_name**이나 **lu_alias** 필드가 항상 설정되어야 합니다. **lu_name**이 0이 아니면, **lu_alias**에 우선하여 사용될 것입니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **fqplu_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

plu_alias가 모두 0으로 설정되면, **fqplu_name** 값이 사용될 것이고 그렇지 않으면 **plu_alias**가 항상 사용되고 **fqplu_name**은 무시됩니다.

반환된 상대방 LU의 리스트는 현재 활동중인 세션이 있는지의 여부에 따라 필터링될 수 있습니다. 필터링하려면, **active_sessions** 필드를 AP_YES로 설정해야 합니다.(그렇지 않으면, 이 필드를 AP_NO로 설정해야 합니다.)

이 명령은 최소한 하나의 세션이 상대방 LU와 형성될 때 결정되는 정보를 반환합니다.

QUERY_PARTNER_LU_DEFINITION 명령은 정의 정보만을 반환합니다.

VCB 구조

```
typedef struct query_partner_lu
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   lu_name[8];         /* LU name                       */
   unsigned char   lu_alias[8];        /* LU alias                      */
   unsigned char   plu_alias[8];       /* partner LU alias              */
```

```
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   active_sessions;    /* active sessions only filter   */
} QUERY_PARTNER_LU;

typedef struct plu_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   reserv1;            /* reserved                      */
   unsigned char   description[RD_LEN];
                                       /* resource description          */
   unsigned short  act_sess_count;     /* curr active sessions count    */
   unsigned char   partner_cp_name[17]; /* partner LU CP name              */
   unsigned char   partner_lu_located; /* CP name resolved?             */
} PLU_SUMMARY;

typedef struct plu_detail
{
   unsigned short  overlay_size;        /* size of this entry           */
   unsigned char   plu_alias[8];        /* partner LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner      */
                                        /* LU name                      */
   unsigned char   reserv1;             /* reserved                     */
   unsigned char   description[RD_LEN];
                                        /* resource description         */
   unsigned short  act_sess_count;      /* curr active sessions count   */
   unsigned char   partner_cp_name[17]; /* partner LU CP name           */
   unsigned char   partner_lu_located;  /* CP name resolved?            */
   unsigned char   plu_un_name[8];      /* partner LU uninterpreted name*/
   unsigned char   parallel_sess_supp;  /* parallel sessions supported? */
   unsigned char   conv_security;       /* conversation security        */
   unsigned short  max_mc_ll_send_size; /* max send LL size for mapped  */
                                        /* conversations                */
   unsigned char   implicit;            /* implicit or explicit entry   */
   unsigned char   security_details;    /* conversation security detail */
   unsigned char   duplex_support;      /* full-duplex support          */
   unsigned char   preference;          /* routing preference           */
   unsigned char   reserva[16];         /* reserved                     */
} PLU_DETAIL;
```

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_PARTNER_LU

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY

요약 정보만을 반환합니다.

AP_DETAIL

상세 정보를 반환합니다.

지정된 **lu_name**(또는 **lu_name**이 모두 0으로 설정된 경우 **lu_alias**) 그리고 **plu_alias**(또는 **plu_alias**가 모두 0으로 설정된 경우 **fqplu_name**)의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

plu_alias 및 **fqplu_name** 필드가 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name

LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다.

lu_alias

국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 이 필드는 **lu_name** 필드가 모두 0으로 설정된 경우에만 유효하지만, 이 경우 8 바이트 모두가 유효하며 설정되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용됩니다.

plu_alias

상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 이 필드가 모두 0으로 설정되면, **fqplu_name** 필드가 색인 값으로 사용될 것입니다.

fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 17 바이트의 길이이며 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

active_sessions
활동중인 세션 필터. 반환된 상대방 LU가 현재 활동중인 세션을 가지고 있는지에 따라 필터링되어야 하는지의 여부를 지정합니다 (AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

plu_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

plu_summary.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

plu_summary.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 17 바이트의 길이이며 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

plu_summary.description
자원 설명(DEFINE_PARTNER_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

plu_summary.act_sess_count
국지 LU와 상대방 LU간의 총 활동중인 세션 수. **active_sessions** 필터가 AP_YES로 설정되었으면, 이 필드가 항상 0보다 클 것입니다.

plu_summary.partner_cp_name
상대방 LU의 제어점(CP)에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

plu_summary.partner_lu_located
상대방 LU에 대한 제어점(CP) 이름이 해석되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

plu_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

plu_detail.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

plu_detail.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

plu_detail.description
자원 설명(DEFINE_PARTNER_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

plu_detail.act_sess_count
국지 LU와 상대방 LU간의 총 활동중인 세션 수. **active_sessions** 필터가 AP_YES로 설정되었으면, 이 필드가 항상 0보다 클 것입니다.

plu_detail.partner_cp_name
상대방 LU의 제어점(CP)에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

plu_detail.partner_lu_located
상대방 LU에 대한 제어점(CP) 이름이 해석되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

plu_detail.plu_un_name
상대방 LU의 미해석 이름. 이는 8 바이트의 유형 A EBCDIC 문자열입니다.

plu_detail.parallel_sess_supp

병렬 세션이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

plu_detail.conv_security

대화 보안 정보가 이 상대방 LU로 송신될 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO). AP_NO로 설정되면, 트랜잭션 프로그램(TP)에 의해 제공된 어떤 보안 정보도 상대방 LU로 송신되지 않습니다. 이 상대방 LU에 대한 현재 활동중인 세션이 없으면, AP_UNKNOWN으로 설정됩니다.

plu_detail.max_mc_ll_send_size

상대방 LU로 송신될 수 있는 논리적 길이(LL) 레코드의 최대 크기. 이보다 큰 데이터 레코드는 여러 개의 LL 레코드로 나누어져 상대방 LU로 송신됩니다. **max_mc_ll_send_size**가

주: 세션이 암호 변환을 지원하지 않으면, AP_PGM_STRONG의 보안 유형을 가진 ALLOCATE나 SEND_CONVERSATION은 실패할 것입니다.

AP_UNKNOWN
이 상대방 LU에 대한 현재 활동중인 세션이 없습니다.

plu_detail.duplex_support
BIND에서 조정된 대로의 대화 양방향(전송) 지원을 반환합니다. 이는 다음 값 중 하나입니다.

AP_HALF_DUPLEX
비동시 양방향(전송) 대화만이 지원됩니다.

AP_FULL_DUPLEX
비동시 양방향(전송)뿐만 아니라 동시 양방향(전송) 대화도 지원됩니다.

AP_UNKNOWN
상대방 LU에 대한 활동중인 세션이 없으므로, 대화 양방향(전송) 지원이 알려지지 않습니다.

plu_detail.preference
DEFINE_PARTNER_LU 명령에서 지정된 것과 같이의 경로지정 프로토콜 선택사항을 반환합니다.

AP_NATIVE
원시(APPN) 경로지정 프로토콜만을 사용합니다.

AP_NONNATIVE
비원시(AnyNet) 프로토콜을 사용하며, 상대방 LU의 위치를 찾을 수 없으면 비원시(AnyNet) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

AP_NATIVE_THEN_NONNATIVE
원시(APPN) 프로토콜을 시도하며, 상대방 LU의 위치를 찾을 수 없으면 원시(APPN) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

AP_USE_DEFAULT_PREFERENCE
노드가 시작될 때 정의된 생략시 선택사항을 사용합니다.(이는 START_NODE에서 설정되며 QUERY_NODE에 의해 재호출될 수 있습니다.)

비원시 경로지정은 Anynet DLC가 프로그램에 사용가능하며 정의된 AnyNet 링크 스테이션이 있을 경우에만 의미있음에 주의하십시오. 더 자세한 정보는 83페이지의 『DEFINE_LS』를 참조하십시오.

START_NODE에서 제공된 필드 **anynet_supported**가 AP_NO로 설정되었으면, 이 필드가 값 AP_NATIVE 또는 AP_USE_DEFAULT_PREFERENCE를 취해야 합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PARTNER_LU_DEFINITION

QUERY_PARTNER_LU_DEFINITION은 DEFINE_PARTNER_LU 명령에서 이전에 건네졌던 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 상대방 LU에 관한 정보를 확보하거나 여러 개의 『청크(chunks)』로 리스트 정보를 확보하려면, **plu_alias** 필드(또는 **plu_alias**가 모두 0으로 설정된 경우 **fqplu_name**)가 설정되어야 합니다. **plu_alias** 필드가 0이 아니면 색인을 결정하는 데 사용되고 **fqplu_name**은 무시됩니다. **plu_alias** 필드가 모두 0으로 설정되면, **fqplu_name**이 색인을 결정하는 데 사용될 것입니다. **list_options** 필드가 AP_FIRST_IN_LIST로 설정되면, 이 필드들 둘다 무시될 것입니다.(이 경우 반환된 리스트는 AP_LIST_BY_ALIAS **list_options**가 설정된 경우에는 **plu_alias**에 의해 정렬될 것이고, 그렇지 않으면 **fqplu_name**에 의해 정렬될 것입니다.) 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 지정된 옵션에 따라 **plu_alias** 또는 **fqplu_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(일반 MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 정의된 정렬하기에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

이 명령은 정의 정보만을 반환됨에 주의하십시오. QUERY_PARTNER_LU 명령은 최소한 하나의 세션이 상대방 LU와 형성될 때 결정되는 정보를 반환합니다.

VCB 구조

```
typedef struct query_partner_lu_definition
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
} QUERY_PARTNER_LU_DEFINITION;

typedef struct partner_lu_def_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   plu_alias[8];       /* partner LU alias              */
```

```
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   description[RD_LEN];
                                       /* resource description          */
} PARTNER_LU_DEF_SUMMARY;

typedef struct partner_lu_def_detail
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   reserv1;            /* reserved                      */
   PLU_CHARS       plu_chars;          /* partner LU characteristics    */
} PARTNER_LU_DEF_DETAIL;

typedef struct plu_chars
{
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   description[RD_LEN];/* resource description          */
   unsigned char   plu_un_name[8];     /* partner LU uninterpreted name */
   unsigned char   preference;         /* routing preference            */
   unsigned short  max_mc_ll_send_size;
                                        /* max MC send LL size           */
   unsigned char   conv_security_ver;   /* already_verified accepted     */
   unsigned char   parallel_sess_supp;  /* parallel sessions supported?  */
   unsigned char   reserv2[8];         /* reserved                      */
} PLU_CHARS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_PARTNER_LU_DEFINITION

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정되는 **plu_alias**(또는 **plu_alias**가 모두 0으로 설정되면 **fqplu_name**)(다음 매개변수를 보십시오)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

AP_LIST_BY_ALIAS
반환되는 리스트는 **plu_alias**에 의해 정렬됩니다. 이 옵션은 AP_FIRST_IN_LIST가 지정될 때에만 유효합니다. AP_LIST_FROM_NEXT 또는 AP_LIST_INCLUSIVE가 지정되는 경우, 리스트 정렬하기는 **plu_alias**이나 **fqplu_alias**가 시작점으로 제공되었는지의 여부에 따라 다를 것입니다.

plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 이 필드가 모두 0으로 설정되면, **fqplu_name** 필드가 필요한 상대방 LU를 지정하는 데 사용될 것입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) **plu_alias** 필드가 모두 0으로 설정되면, 이 필드만이 유효합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수 있습니다.

partner_lu_def_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

partner_lu_def_summary.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

partner_lu_def_summary.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

partner_lu_def_summary.description
자원 설명(DEFINE_PARTNER_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

partner_lu_def_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

partner_lu_def_detail.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

partner_lu_def_detail.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

partner_lu_def_detail.plu_chars.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

partner_lu_def_detail.plu_chars.plu_alias
상대방 LU 별명.

partner_lu_def_detail.plu_chars.description
자원 설명(DEFINE_PARTNER_LU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

partner_lu_def_detail.plu_chars.plu_un_name
상대방 LU의 미해석 이름. 이는 8 바이트의 유형 A EBCDIC 문자열입니다.

plu_chars.preference
이 상대방 LU에 대한 세션 활성화에 선호되는 경로지정 프로토콜의 세트. 이 필드는 다음 값을 취할 수 있습니다.

AP_NATIVE
원시(APPN) 경로지정 프로토콜만을 사용합니다.

AP_NONNATIVE
비원시(AnyNet) 경로지정 프로토콜만을 사용합니다.

AP_NATIVE_THEN_NONNATIVE
원시(APPN) 프로토콜을 시도하며, 상대방 LU의 위치를 찾을 수 없으면 비원시(AnyNet) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

AP_NONNATIVE_THEN_NATIVE
비원시(AnyNet) 프로토콜을 시도하며, 상대방 LU의 위치를 찾을 수 없으면 원시(APPN) 프로토콜을 사용하여 세션 활성화를 재시도합니다.

AP_USE_DEFAULT_PREFERENCE
노드가 시작될 때 정의된 생략시 선택사항을 사용합니다

주: 비원시 경로지정은 노드 연산자 기능에 AnyNet DLC가 사용가능하고 정의된 AnyNet 링크 스테이션이 있을 때에만 의미있습니다.

partner_lu_def_detail.plu_chars.max_mc_ll_send_size
상대방 LU로 송신될 수 있는 논리적 길이(LL) 레코드의 최대 크기. 이보다 큰 데이터 레코드는 여러 개의 LL 레코드로 나누어져 상대방 LU로 송신됩니다. **max_mc_ll_send_size**가 취할 수 있는 최대값은 32 767입니다.

partner_lu_def_detail.plu_chars.conv_security_ver
국지 LU를 대신해서 상대방 LU에 **user_ids**를 확인할 권한이 있는지의 여부, 즉 상대방 LU가 접속(Attach) 요청에서 이미 검증된 표시기를 설정할 수 있는지의 여부를 지정합니다.

AP_YES
AP_NO

partner_lu_def_detail.plu_chars.parallel_sess_supp
병렬 세션이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PLU_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PORT

QUERY_PORT는 노드의 포트에 관한 정보의 리스트를 반환합니다. 이 정보는 『결정된 데이터』(실행동안 동적으로 모아진 데이터)와 『정의된 데이터』(DEFINE_PORT에서 응용프로그램에 의해 제공된 데이터)로 구성됩니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 포트에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **port_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **port_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

반환되는 포트들의 리스트는 그들이 속해 있는 DLC의 이름에 의해 필터링될 수 있습니다. 이 경우, **dlc_name** 필드가 설정되어야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.)

VCB 구조

```
typedef struct query_port
{
   unsigned short  opcode;             /* verb operation code          */
   unsigned char   attributes;         /* Verb attributes              */
   unsigned char   format;             /* format                       */
   unsigned short  primary_rc;         /* primary return code          */
   unsigned long   secondary_rc;       /* secondary return code        */
   unsigned char   *buf_ptr;           /* pointer to buffer            */
   unsigned long   buf_size;           /* buffer size                  */
   unsigned long   total_buf_size;     /* total buffer size required   */
   unsigned short  num_entries;        /* number of entries            */
   unsigned short  total_num_entries;  /* total number of entries      */
   unsigned char   list_options;       /* listing options              */
   unsigned char   reserv3;            /* reserved                     */
   unsigned char   port_name[8];       /* port name                    */
   unsigned char   dlc_name[8];        /* DLC name filter              */
} QUERY_PORT;

typedef struct port_summary
{
   unsigned short  overlay_size;       /* size of this entry           */
   unsigned char   port_name[8];       /* port name                    */
   unsigned char   description[RD_LEN];
                                       /* resource description         */
   unsigned char   port_state;         /* port state                   */
   unsigned char   reserv1[1];         /* reserved                     */
   unsigned char   dlc_name[8];        /* name of DLC                  */
} PORT_SUMMARY;
```

```
typedef struct port_detail
{
   unsigned short  overlay_size;       /* size of this entry           */
   unsigned char   port_name[8];       /* port name                    */
   unsigned char   reserv1[2];         /* reserved                     */
   PORT_DET_DATA   det_data;           /* determined data              */
   PORT_DEF_DATA   def_data;           /* defined data                 */
} PORT_DETAIL;

typedef struct port_det_data
{
   unsigned char   port_state;         /* port state                   */
   unsigned char   dlc_type;           /* DLC type                     */
   unsigned char   port_sim_rim;       /* port initialization options  */
   unsigned char   reserv1;            /* reserved                     */
   unsigned short  def_ls_good_xids;   /* number of successful XIDs    */
   unsigned short  def_ls_bad_xids;    /* number of unsuccessful XIDs  */
   unsigned short  dyn_ls_good_xids;   /* successful XIDs on dynamic   */
                                       /* LS count                     */
   unsigned short  dyn_ls_bad_xids;    /* failed XIDs on dynamic       */
   unsigned short  num_implicit_links; /* number of implicit links     */
                                       /* active on this port          */
   unsigned char   neg_ls_supp;        /* are negotiable LSs supported? */
                                       /* LS count                     */
   unsigned char   abm_ls_supp;        /* are ABM LSs supported?       */
   unsigned long   start_time          /* start time                   */
   unsigned char   reserva[12];        /* reserved                     */
} PORT_DET_DATA;

typedef struct port_def_data
{
   unsigned char   description;        /* resource description          */
   unsigned char   dlc_name[8];        /* DLC name associated with port */
   unsigned char   port_type;          /* port type                     */
   unsigned char   port_attributes[4]; /* port attributes               */
   unsigned char   implicit_uplink_to_en;
                                       /* implicit links to EN are uplink*/
   unsigned char   reserv3[2];         /* NB_BYTE                       */
   unsigned long   port_number;        /* port number                   */
   unsigned short  max_rcv_btu_size;   /* max receive BTU size          */
   unsigned short  tot_link_act_lim;   /* total link activation limit   */
   unsigned short  inb_link_act_lim;   /* inbound link activation limit  */
   unsigned short  out_link_act_lim;   /* outbound link activation limit */
   unsigned char   ls_role;            /* initial link station role     */
   unsigned char   retry_flags;        /* conditions for automatic retrys*/
                                       /* retries                       */
   unsigned short  max_activation_attempts;                           */
                                       /* how many automatic retries    */
   unsigned short  activation_delay_timer;
                                       /* delay between automatic retries*/
   unsigned char   reserv1[10];        /* reserved                     */
   unsigned char   implicit_dspu_template[8];
                                       /* implicit DSPU template        */
   unsigned short  implicit_ls_limit   /* max number of implicit links  */
   unsigned char   reserv2             /* reserved                      */
   unsigned char   implicit_dspu_services;
                                       /* implicit links support DSPUs  */
   unsigned short  implicit_deact_timer;
                                       /* Implicit link HPR link        */
                                       /* deactivation timer            */
   unsigned short  act_xid_exchange_limit;
                                       /* activation XID exchange limit  */
```

```
   unsigned short  nonact_xid_exchange_limit;
                                    /* non-act. XID exchange limit    */
   unsigned char   ls_xmit_rcv_cap;    /* LS transmit-rcv capability     */
   unsigned char   max_ifrm_rcvd;      /* max number of I-frames that    */
                                    /* can be received                */
   unsigned short  target_pacing_count;
                                    /* target pacing count            */
   unsigned short  max_send_btu_size;  /* max send BTU size              */
   LINK_ADDRESS    dlc_data;           /* DLC data                       */
   LINK_ADDRESS    hpr_dlc_data;       /* HPR DLC data                   */
   unsigned char   implicit_cp_cp_sess_support;
                                    /* Implicit links allow CP-CP     */
                                    /* sessions                       */
   unsigned char   implicit_limited_resource;
                                    /* Implicit links are             */
                                    /* limited resource               */
   unsigned char   implicit_hpr_support;
                                    /* Implicit links support HPR     */
   unsigned char   implicit_link_lvl_error;
                                    /* Implicit links support         */
                                    /* HPR link-level error recovery  */
   unsigned char   retired1;           /* reserved                       */
   TG_DEFINED_CHARS default_tg_chars;  /* Default TG chars               */
   unsigned char   discovery_supported;
                                    /* Discovery function supported?  */
   unsigned short  port_spec_data_len; /* length of port spec data       */
   unsigned short  link_spec_data_len; /* length of link spec data       */
} PORT_DEF_DATA;

typedef struct link_address
{
        unsigned short  length;        /* length                         */
        unsigned short  reserve1;      /* reserved                       */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                       /* address                        */
} LINK_ADDRESS;

typedef struct tg_defined_chars
{
        unsigned char   effect_cap;    /* effective capacity             */
        unsigned char   reserve1[5];   /* reserved                       */
        unsigned char   connect_cost;  /* connection cost                */
        unsigned char   byte_cost;     /* byte cost                      */
        unsigned char   reserve2;      /* reserved                       */
        unsigned char   security;      /* security                       */
        unsigned char   prop_delay;    /* propagation delay              */
        unsigned char   modem_class;   /* modem class                    */
        unsigned char   user_def_parm_1;
                                       /* user_defined parameter 1       */
        unsigned char   user_def_parm_2;
                                       /* user_defined parameter 2       */
        unsigned char   user_def_parm_3;
                                       /* user_defined parameter 3       */
} TG_DEFINED_CHARS;

typedef struct port_spec_data
{
       unsigned char  port_data[SIZEOF_PORT_SPEC_DATA];

}  PORT_SPEC_DATA;
```

```
typedef struct link_spec_data
{
       unsigned char  link_data[SIZEOF_LINK_SPEC_DATA];

}  LINK_SPEC_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_PORT

attributes
명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **port_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

port_name
조회되고 있는 포트의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dlc_name
DLC 명 필터. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 이 필드가 설정되면 이 DLC에 속하는 포트만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

port_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

port_summary.port_name
이 링크 스테이션과 연관된 포트의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

port_summary.description
자원 설명(DEFINE_PORT에서 지정된 것과 같이). 이는 국지로 표시 가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

port_summary.port_state
포트의 현재 상태를 지정합니다.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

port_summary.dlc_name
DLC의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

port_detail.overlay_size
이 입력항목에 있는 바이트의 수(**link_spec_data**를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

port_detail.port_name
이 링크 스테이션과 연관된 포트의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

port_detail.det_data.port_state
포트의 현재 상태를 지정합니다.

AP_NOT_ACTIVE
AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE

port_detail.det_data.dlc_type
DLC의 유형. 퍼스널 통신이나 통신 서버는 다음 유형을 지원합니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

port_detail.det_data.port_sim_rim
설정 초기화 모드(SIM)와 수신 초기화 모드(RIM)가 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

port_detail.det_data.def_ls_good_xids
이 포트가 마지막으로 시작된 이래 이 포트상의 모든 정의된 링크 스테이션에서 발생한 총 성공 XID 교환 수.

port_detail.det_data.def_ls_bad_xids
이 포트가 마지막으로 시작된 이래 이 포트상의 모든 정의된 링크 스테이션에서 발생한 총 비성공 XID 교환 수.

port_detail.det_data.dyn_ls_good_xids
이 포트가 마지막으로 시작된 이래 이 포트상의 모든 동적 링크 스테이션에서 발생한 총 성공 XID 교환 수.

port_detail.det_data.dyn_ls_bad_xids
이 포트가 마지막으로 시작된 이래 이 포트상의 모든 동적 링크 스테이션에서 발생한 총 비성공 XID 교환 수.

port_detail.det_data.num_implicit_links
이 포트상에서 현재 활동중인 총 암시적 링크 수. 여기에는 Discovery의 사용에 따라 생성된 동적 링크나 암시적 링크가 포함됩니다. 이 포트에서 허용되는 그러한 링크들의 수는 PORT_DEF_DATA의 **implicit_ls_limit** 필드에 의해 제한됩니다.

port_detail.def_data.neg_ls_supp
조정가능 링크 스테이션에 대한 지원, AP_YES 또는 AP_NO.

port_detail.det_data.abm_ls_supp
ABM 링크 스테이션에 대한 지원. 이는 DLC가 시작되고 나서야 알 수 있습니다.

AP_NO
AP_YES
AP_UNKNOWN

port_detail.det_data.start_time
노드가 시작된 시간과 마지막으로 이 포트가 시작된 시간간의 경과 시간으로 100분의 1초 단위. 이 포트가 시작되었으면, 이 필드에 0이 반환됩니다.

port_detail.def_data.description
자원 설명(DEFINE_PORT에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

port_detail.def_data.dlc_name
연관된 DLC의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

port_detail.def_data.port_type
포트에 의해 사용된 회선의 유형을 지정합니다. 다음 값 중 하나에 해당합니다.

AP_PORT_NONSWITCHED
AP_PORT_SWITCHED
AP_PORT_SATF

port_detail.def_data.port_attributes[0]
이는 비트 필드입니다. 값 AP_NO를 취하거나 다음을 취할 수 있습니다.

AP_RESOLVE_BY_LINK_ADDRESS
이는 수신된 XID 3상에서 운반된 CP명(노드 ID)을 사용하여 입력 호출을 해석하기 전에 CONNECT_IN상의 링크 주소를 사용하여 입력 호출의 해석을 시도하라고 지정합니다. **port_type** 필드가 AP_PORT_SWITCHED로 설정되지 않는 한, 이 비트가 무시됩니다.

port_detail.def_data.implicit_uplink_to_en
BrNN 만: 인접 노드가 끝 노드인 경우 이 포트에서 분리된 암시적 링크 스테이션이 업링크인지 또는 다운링크인지를 지정합니다. 이 필드의 값은 동일한 상대방에 대한 기존 링크가 없고 그러한 링크들이 링크 유형을 결정하는 데 첫번째로 사용되는 경우에만 고려됩니다.

AP_NO
암시적 링크들이 다운링크입니다.

AP_YES
암시적 링크들이 업링크입니다.

기타 노드 유형: AP_NO로 설정됩니다.

port_detail.def_data.port_number
포트 번호.

port_detail.def_data.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

port_detail.def_data.tot_link_act_lim
총 링크 활성화 한계.

port_detail.def_data.inb_link_act_lim
인바운드 링크 활성화 한계.

port_detail.def_data.out_link_act_lim
아웃바운드 링크 활성화 한계.

port_detail.def_data.ls_role
링크 스테이션 역할. 이는 조정가능(AP_LS_NEG), 1차(AP_LS_PRI) 또는 2차(AP_LS_SEC)일 수 있습니다. **implicit_hpr_support**가 AP_NO로 설정되는 경우에는 예약됩니다.

port_detail.def_data.implicit_dspu_template
DEFINE_DSPU_TEMPLATE 명령으로 정의되고 이 포트에서 활성화된 암시적 링크를 위한 PU 집중(concentration)을 제공하는 데 국지 노드

가 사용되게 되어 있는 경우 정의에 사용되는 DSPU 템플리트를 지정합니다. 링크가 활성화될 때 지정된 템플리트가 존재하지 않거나 이미 그 인스턴스 한계에 있으면 활성화가 실패합니다. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다.

def_data.implicit_dspu_services 필드가 AP_PU_CONCENTRATION으로 설정되지 않으면, 이 필드가 예약됩니다.

port_detail.def_data.implicit_ls_limit

동적 링크나 발견을 위해 활성화된 링크를 포함하여 이 포트에서 동시에 활동중일 수 있는 최대 암시적 링크 스테이션 수를 지정합니다. 0의 값은 제한이 없음을 의미하며, AP_NO_IMPLICIT_LINKS의 값은 암시적 링크가 허용되지 않음을 의미합니다.

def_data.implicit.dspu_services

국지 노드가 이 포트상에서 활성화된 암시적 링크를 거쳐 다운스트림 PU에 제공할 서비스들을 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_DLUR

국지 노드가 다운스트림 PU를 위한 DLUR 서비스를 제공할 것입니다(DEFINE_DLUR_DEFAULTS 명령을 통해 구성된 생략시 DLUS). 이 설정은 국지 노드가 네트워크 노드인 경우에만 유효합니다.

AP_PU_CONCENTRATION

국지 노드가 다운스트림 PU를 위한 PU 집중(concentration)을 제공할 것입니다.(그리고 **def_data.implicit_dspu_template** 필드에서 지정된 DSPU 템플리트에 의해 지정된 것과 같이 정의를 배치할 것입니다.)

AP_NONE

국지 노드가 다운스트림 PU를 위한 어떤 서비스도 제공하지 않을 것입니다.

port_detail.def_data.retry_flags

이 필드는 **def_data.retry_flags**의 DEFINE_LS에서 플래그 AP_INHERIT_RETRY가 설정되는 경우 어떤 조건하에서 이 포트의 활성화가 자동 재시도를 하게 되는지 지정합니다. 이는 비트 필드로 다음 값 중 어느 것이든 취할 수 있습니다.

AP_RETRY_ON_START

활성화 시도시 원격 노드로부터 응답이 수신되지 않으면 링크 활성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 활동중이 아니면, 프로그램이 이를 활성화시키려고 할 것입니다.

AP_RETRY_ON_FAILURE

활동중이거나 대기 활동중인 동안 링크가 실패하면 링크 활

성화가 재시도될 것입니다. 활성화 시도시 하부 포트가 실패했으면, 프로그램이 이를 활성화시키려고 할 것입니다.

AP_RETRY_ON_DISCONNECT
링크가 원격 노드에 의해 정상적으로 중단되면 링크 활성화가 재시도될 것입니다.

AP_DELAY_APPLICATION_RETRIES
응용프로그램에 의해 초기화되는(START_LS 또는 요구가 있을때 링크 활성화를 사용하여) 링크 활성화 재시도는 **activation_delay_timer**를 사용하여 속도 조정될 것입니다.

AP_DELAY_INHERIT_RETRY
이 필드에서 플래그에 의해 지정된 재시도 조건 외에, 하부 포트 정의의 **retry_flags** 필드에서 정의된 조건들 역시 사용될 것입니다.

port_detail.def_data.max_activation_attempts
def_data.retry_flags의 DEFINE_LS에서 최소한 하나의 플래그가 설정되고, DEFINE_LS상의 **def_data.max_activation_attempts**가 AP_USE_DEFAULTS로 설정되지 않는 한, 이 필드는 아무런 영향도 미치지 않습니다.

이 필드는 원격 노드가 응답하고 있지 않거나 하부 포트가 활동중이 아닐 때 프로그램이 허용하는 재시도 회수를 지정합니다. 여기에는 자동 재시도와 응용프로그램 지향 활성화 시도가 모두 포함됩니다.

이 한도에 도달하게 되면, 더 이상 자동 재시도가 이루어지지 않습니다. 이 조건은 STOP_LS, STOP_PORT, STOP_DLC 또는 성공적 활성화에 의해 재설정됩니다. START_LS 또는 OPEN_LU_SSCP_SEC_RQ는 활성화가 실패하면 더 이상 재시도하지 않는 단일 활성화가 시도됩니다.

0은 '제한 없음'을 의미합니다. 값 AP_USE_DEFAULTS는 DEFINE_PORT에서 제공된 **max_activiation_attempts**가 사용되게 합니다.

ls_detail.def_data.activation_delay_timer
def_data.retry_flags의 DEFINE_LS에서 최소한 하나의 플래그가 설정되고, DEFINE_LS상의 **def_data.max_activation_attempts**가 AP_USE_DEFAULTS로 설정되지 않는 한, 이 필드는 아무런 영향도 미치지 않습니다.

이 필드는 **def_data.retry_flags**에서 AP_DELAY_APPLICATION_RETRIES 비트가 설정되는 경우 프로그램가 자동 재시도간에 그리고 응용프로그램 지향 활성화 시도간에 대기하는 초 수를 지정합니다.

값 AP_USE_DEFAULTS는 DEFINE_PORT에서 제공된 **activiation_delay_timer**가 사용되게 합니다.

0이 지정되면, 프로그램이 30초의 생략시 타이머 기간을 사용합니다.

def_data.implicit_dspu_template

DEFINE_DSPU_TEMPLATE 명령으로 정의되고 이 포트에서 활성화된 암시적 링크를 위한 PU 집중(concentration)을 제공하는 데 국지 노드가 사용되게 되어 있는 경우 정의에 사용되는 DSPU 템플리트를 지정합니다. 링크가 활성화될 때 지정된 템플리트가 존재하지 않거나 이미 그 인스턴스 한계에 있으면 활성화가 실패합니다. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다.

def_data.implicit_dspu_services 필드가 AP_PU_CONCENTRATION으로 설정되지 않으면, 이 필드가 예약됩니다.

def_data.implicit.dspu_services

국지 노드가 이 포트상에서 활성화된 암시적 링크를 거쳐 다운스트림 PU에 제공할 서비스들을 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_DLUR

국지 노드가 다운스트림 PU를 위한 DLUR 서비스를 제공할 것입니다(DEFINE_DLUR_DEFAULTS 명령을 통해 구성된 생략시 DLUS).

AP_PU_CONCENTRATION

국지 노드가 다운스트림 PU를 위한 PU 집중(concentration)을 제공할 것입니다.(그리고 **def_data.implicit_dspu_template** 필드에서 지정된 DSPU 템플리트에 의해 지정된 것과 같이 정의를 배치할 것입니다.)

AP_NONE

국지 노드가 다운스트림 PU를 위한 어떤 서비스도 제공하지 않을 것입니다.

def_data.implicit_deact_timer

제한된 자원 링크 활성종료 타이머(초 단위). **implicit_limited_resource**가 AP_YES나 AP_NO_SESSIONS로 설정되면, 이 타이머의 지속시간 동안 링크를 통과하는 데이터가 없고 링크를 사용하는 세션이 없는 경우 HPR 가능 암시적 링크가 자동으로 활성종료됩니다.

implicit_limited_resource가 AP_INACTIVITY로 설정되면, 이 타이머의 지속시간 동안 링크를 통과하는 데이터가 없는 경우 암시적 링크가 자동으로 활성종료됩니다.

0이 설정되면 30의 생략시 값이 사용됩니다. 그렇지 않으면 최소 값이 5입니다.(더 낮은 값으로 설정되면, 지정된 값이 무시되고 5가 사용될 것입니다.) **implicit_limited_resource**가 AP_NO로 설정되지 않는 한 이 매개변수가 예약됨에 주의하십시오.

port_detail.def_data.act_xid_exchange_limit

활성화 XID 교환 한계.

port_detail.def_data.nonact_xid_exchange_limit
비활성화 XID 교환 한계.

port_detail.def_data.ls_xmit_rcv_cap
링크 스테이션 전송/수신 능력을 지정합니다. 이는 2중 동시(AP_LS_TWS) 또는 2중 변경(AP_LS_TWA)중 하나입니다.

port_detail.def_data.max_ifrm_rcvd
승인이 보내지기 전에 국지 링크 스테이션에 의해 수신될 수 있는 최대 I-Frames 수. 범위: 1—127

port_detail.def_data.target_pacing_count
이 TG상의 BIND들을 위한 적절한 페이싱 창 크기를 나타내는 1과 32 767을 포함하여 이들 사이에 있는 숫자 값. 수는 고정 바인드 페이싱이 수행되고 있는 경우에만 유효합니다. 퍼스널 통신이나 통신 서버는 현재 이 값을 사용하고 있지 않음에 주의하십시오.

port_detail.def_data.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

port_detail.def_data.dlc_data.length
포트 주소 길이.

port_detail.def_data.dlc_data.address
포트 주소.

port_detail.def_data.hpr_dlc_data.length
HPR 포트 주소 길이.

port_detail.def_data.hpr_dlc_data.address
HPR 포트 주소. 이는 현재 THPR 링크 지원시 사용됩니다. 이 포트를 사용하는 링크 스테이션에서 교환된 XID3상의 X'61' 제어 벡터의 X'80' 서브필드에서 퍼스널 통신이나 통신 서버에 의해 송신된 정보를 지정합니다.

port_detail.def_data.implicit_cp_cp_sess_support
이 포트에서 분리된 암시적 링크 스테이션에 대해 CP-CP 세션이 허용되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

port_detail.def_data.implicit_limited_resource
링크를 사용하고 있는 세션이 없을 때 이 포트에서 분리된 암시적 링크 스테이션이 활성종료되어야 하는지의 여부를 지정합니다. 다음 값 중 하나로 설정됩니다.

AP_NO
암시적 링크는 제한된 자원이 아니며 자동으로 활성종료되지 않을 것입니다.

AP_YES 또는 AP_NO_SESSIONS
암시적 링크가 제한된 자원이며 그것을 사용하는 활동중인 세션이 없을 때 자동으로 활성종료될 것입니다.

AP_INACTIVITY
암시적 링크가 제한된 자원이며 그것을 사용하는 활동중인 세션이 없거나 **link_deact_timer** 필드에 의해 지정된 시간 동안 링크상에서 흐른 데이터가 없을 때에 자동으로 활성종료될 것입니다.

port_detail.def_data.implicit_hpr_support
암시적 링크에서 HPR이 지원되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

port_detail.def_data.implicit_link_lvl_error
링크 레벨 오류 복구를 사용하는 암시적 링크에서 HPR 트래픽이 송신되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

port_detail.def_data.default_tg_chars
TG 특성(39페이지의 『DEFINE_COS』 참조). 이 포트에서 분리된 암시적 링크 스테이션을 위해 사용되며 **use_default_tg_chars**를 지정하는 정의된 링크 스테이션을 위해서도 사용됩니다.

port_detail.def_data.discovery_supported
발견(Discovery) 탐색 기능이 이 포트상에서 수행되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

port_detail.def_data.port_spec_data_len
ACTIVATE_PORT 신호상에서 변경되지 않은 상태로 포트로 건네진 데이터의 채워지지 않은 바이트 단위 길이. 데이터는 PORT_DETAIL 구조체로 연결됩니다.

port_detail.def_data.link_spec_data_len
초기화 동안 변경되지 않은 상태로 링크 스테이션 구성요소에 건네진 데이터. 데이터는 포트 특정 데이터에 뒤따라 즉시 PORT_DETAIL 구조체에 연결됩니다. 포트 특정 데이터와 링크 특정 데이터는 4 바이트 경계에서 끝으로 채워지게 될 것입니다. 포트 특정 데이터와 링크 특정 데이터간에는 어떠한 명시적 채워넣기도 없을 것입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PORT_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

QUERY_PORT

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_PU

QUERY_PU는 국지 PU들 및 그들과 연관된 링크들의 리스트를 반환합니다.

정보는 리스트로 반환됩니다. 특정 PU에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **pu_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

명령은 국지 PU들이 호스트 시스템에 직접 연결될지 또는 DLUR을 통해 연결될지 지정합니다. 지정된 연결에 관한 정보만이 반환될 수 있도록 **host_attachment** 필드가 필터로 사용될 수 있습니다.

VCB 구조

```
typedef struct query_pu
{
   unsigned short  opcode;             /* verb operation code          */
   unsigned char   attributes;         /* Verb attributes              */
   unsigned char   format;             /* format                       */
   unsigned short  primary_rc;         /* primary return code          */
   unsigned long   secondary_rc;       /* secondary return code        */
   unsigned char   *buf_ptr;           /* pointer to buffer            */
   unsigned long   buf_size;           /* buffer size                  */
   unsigned long   total_buf_size;     /* total buffer size required   */
   unsigned short  num_entries;        /* number of entries            */
   unsigned short  total_num_entries;  /* total number of entries      */
   unsigned char   list_options;       /* listing options              */
   unsigned char   reserv3;            /* reserved                     */
   unsigned char   pu_name[8];         /* PU name                      */
   unsigned char   host_attachment;    /* Host Attachment              */
} QUERY_PU;

typedef struct pu_data
{
   unsigned short  overlay_size;      /* size of this entry           */
   unsigned char   pu_name[8];        /* PU name                      */
   unsigned char   description[RD_LEN];
                                      /* resource description         */
   unsigned char   ls_name[8];        /* LS name                      */
   unsigned char   pu_sscp_sess_active;
                                      /* Is PU-SSCP session active    */
   unsigned char   host_attachment;   /* Host attachment              */
   SESSION_STATS   pu_sscp_stats;     /* PU-SSCP session statistics   */
   unsigned char   sscp_id[6];        /* SSCP ID                      */
   unsigned char   conventional_lu_compression;
                                      /* Data compression requested   */
                                      /* for conventional LU sessions */
   unsigned char   conventional_lu_cryptography;
                                      /* Cryptography required for    */
                                      /* conventional LU sessions     */
   unsigned char   reserva[12];       /* reserved                     */
} PU_DATA;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;       /* session receive RU size      */
```

```
   unsigned short  send_ru_size;      /* session send RU size          */
   unsigned short  max_send_btu_size; /* max send BTU size             */
   unsigned short  max_rcv_btu_size;  /* max rcv BTU size              */
   unsigned short  max_send_pac_win;  /* max send pacing window size   */
   unsigned short  cur_send_pac_win;  /* curr send pacing window size  */
   unsigned short  max_rcv_pac_win;   /* max recv pacing window size   */
   unsigned short  cur_rcv_pac_win;   /* current receive pacing        */
                                      /* window size                   */
   unsigned long   send_data_frames;  /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                      /* num of FMD data frames sent   */
   unsigned long   send_data_bytes;   /* number of data bytes sent     */
   unsigned long   rcv_data_frames;   /* num data frames received      */
   unsigned long   rcv_fmd_data_frames;
                                      /* num of FMD data frames rcvd   */
   unsigned long   rcv_data_bytes;    /* number of data bytes received */
   unsigned char   sidh;              /* session ID high byte          */
                                      /* (from LFSID)                  */
   unsigned char   sidl;              /* session ID low byte           */
                                      /* (from LFSID)                  */
   unsigned char   odai;              /* ODAI bit set                  */
   unsigned char   ls_name[8];        /* Link station name             */
   unsigned char   pacing_type;       /* type of pacing in use         */
} SESSION_STATS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_PU

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

지정된 **pu_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

pu_name
나열될 첫번째 PU의 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

host_attachment
호스트 연결을 위한 필터.

AP_NONE
모든 국지 PU들에 대한 정보를 반환합니다.

AP_DLUR_ATTACHED
DLUR에 의해 지원되는 모든 국지 PU들에 대한 정보를 반환합니다.

AP_DIRECT_ATTACHED
호스트 시스템에 직접 연결되는 PU들에 대한 정보만을 반환합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

pu_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

pu_data.pu_name
PU 명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

pu_data.description
자원 설명(DEFINE_LS 또는 DEFINE_INTERNAL_PU에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

pu_data.ls_name
이 PU와 연관된 링크 스테이션의 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

pu_data.pu_sscp_sess_active
PU-SSCP 세션이 활동중인지의 여부를 지정합니다(AP_YES 또는 AP_NO).

pu_data.host_attachment
국지 PU 호스트 연결 유형.

AP_DLUR_ATTACHED
PU가 DLUR을 사용하여 호스트 시스템에 연결됩니다.

AP_DIRECT_ATTACHED
PU가 직접 호스트 시스템에 연결됩니다.

pu_data.pu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

pu_data.pu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

pu_data.pu_sscp_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

pu_data.pu_sscp_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

pu_data.pu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

pu_data.pu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정될 것입니다.

pu_data.pu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

pu_data.pu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정될 것입니다.

pu_data.pu_sscp_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

pu_data.pu_sscp_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

pu_data.pu_sscp_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

pu_data.pu_sscp_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

pu_data.pu_sscp_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

pu_data.pu_sscp_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

pu_data.pu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

pu_data.pu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

pu_data.pu_sscp_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 ACTPU의 송신자가 이 필드를 0으로 설정하며, ACTPU 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

pu_data.pu_sscp_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

pu_data.pu_sscp_stats.pacing_type
PU-SSCP 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE을 취할 것입니다.

sscp_id
PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하고 있는 6 바이트의 필드입니다.

pu_sscp_sess_active가 AP_YES가 아니면, 이 필드가 0으로 될 것입니다.

pu_data.conventional_lu_compression
이 PU를 사용하는 세션에 데이터 압축이 요청되는지의 여부를 지정합니다.

AP_NO
국지 노드가 이 PU를 사용하는 세션상에서 흐르는 데이터를 압축하거나 압축해제해서는 안됩니다.

AP_YES
호스트가 압축을 요청하면 이 PU에 의존적인 세션에 대해 데이터 압축이 사용가능해야 합니다.

pu_data.conventional_lu_cryptography
이 PU에 의존적인 전통적 LU 세션을 위해 세션 레벨 암호화가 요청되는지의 여부를 지정합니다.

AP_NONE
프로그램에 의해 세션 레벨 암호화가 수행되지 않습니다.

AP_MANDATORY
LU에 반입 키가 사용가능하면 프로그램에 의해 필수 세션 레벨 암호화가 수행됩니다. 그렇지 않으면, LU를 사용하는 응용프로그램에 의해 수행되어야 합니다.(PU 집중(concentration)이면, 다운스트림 LU에 의해 수행됩니다.)

AP_OPTIONAL
이 값은 사용된 암호화가 세션당 기준으로 호스트 응용프로그램에 의해 가동되게 합니다. 호스트가 이 PU에 의존적인 세션을 위한 암호화를 요청하는 경우, 프로그램의 행동은 AP_MANDATORY에 관한 것입니다. 호스트가 암호화를 요청하지 않으면, 행동이 AP_NONE과 동일합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_PU_TYPE
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_RTP_CONNECTION

QUERY_RTP_CONNECTION은 네트워크 노드나 끝 노드에서 사용되며 노드가 끝점인 해당 고속 전송 프로토콜(RTP)에 관한 리스트 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 RTP 연결에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **rtp_name** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **rtp_name**에 의해 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(일반 MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 정의된 정렬하기에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct query_rtp_connection
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   rtp_name[8];        /* name of RTP connection         */
} QUERY_RTP_CONNECTION;

typedef struct rtp_connection_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   rtp_name[8];        /* RTP connection name            */
   unsigned char   first_hop_ls_name[8];
                                       /* LS name of first hop           */
   unsigned char   dest_node_name[17]; /* fully qualified name of        */
                                       /* destination node               */
   unsigned char   reserv1;            /* reserved                       */
   unsigned char   cos_name[8];        /* class-of-service name          */
   unsigned short  num_sess_active;    /* number of active sessions      */
} RTP_CONNECTION_SUMMARY;

typedef struct rtp_connection_detail
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   rtp_name[8];        /* RTP connection name            */
   unsigned char   first_hop_ls_name[8];
```

```
                                        /* LS name of first hop          */
   unsigned char   dest_node_name[17]; /* fully qualified name of       */
                                        /* destination node              */
   unsigned char   isr_boundary_fn;    /* connection provides ISR BF    */
   unsigned char   reserv1[3];         /* reserved                      */
   unsigned char   cos_name[8];        /* class-of-service name         */
   unsigned short  max_btu_size;       /* max BTU size                  */
   unsigned long   liveness_timer;     /* liveness timer                */
   unsigned char   local_tcid[8];      /* local TCID                    */
   unsigned char   remote_tcid[8];     /* remote TCID                   */
   RTP_STATISTICS  rtp_stats;          /* RTP statistics                */
   unsigned short  num_sess_active;    /* number of active sessions     */
   unsigned char   reserv2[16];        /* reserved                      */
   unsigned short  rscv_len;           /* length of appended RSCV       */
} RTP_CONNECTION_DETAIL;

typedef struct rtp_statistics
{
   unsigned long   bytes_sent;         /* total number of bytes sent    */
   unsigned long   bytes_received;     /* total number of bytes received */
   unsigned long   bytes_resent;       /* total number of bytes resent  */
   unsigned long   bytes_discarded;    /* total number bytes discarded  */
   unsigned long   packets_sent;       /* total number of packets sent  */
   unsigned long   packets_received;   /* total number packets received */
   unsigned long   packets_resent;     /* total number of packets resent */
   unsigned long   packets_discarded;  /* total number packets discarded */
   unsigned long   gaps_detected;      /* gaps detected                 */
   unsigned long   send_rate;          /* current send rate             */
   unsigned long   max_send_rate;      /* maximum send rate             */
   unsigned long   min_send_rate;      /* minimum send rate             */
   unsigned long   receive_rate;       /* current receive rate          */
   unsigned long   max_receive_rate;   /* maximum receive rate          */
   unsigned long   min_receive_rate;   /* minimum receive rate          */
   unsigned long   burst_size;         /* current burst size            */
   unsigned long   up_time;            /* total uptime of connection    */
   unsigned long   smooth_rtt;         /* smoothed round-trip time      */
   unsigned long   last_rtt;           /* last round-trip time          */
   unsigned long   short_req_timer;    /* SHORT_REQ timer duration      */
   unsigned long   short_req_timeouts; /* number of SHORT_REQ timeouts  */
   unsigned long   liveness_timeouts;  /* number of liveness timeouts   */
   unsigned long   in_invalid_sna_frames;
                                        /* number of invalid SNA frames  */
                                        /* received                      */
   unsigned long   in_sc_frames;       /* number of SC frames received  */
   unsigned long   out_sc_frames;      /* number of SC frames sent      */
   unsigned char   reserve[40];        /* reserved                      */
} RTP_STATISTICS;
```

주: **rtp_connection_detail** 중복 뒤에는 SNA에 의해 정의된 대로 전송경로 선택 제어 벡터(RSCV)가 이어질 것입니다. RTP 연결 설정 후와 임의의 경로 전환 전에, 다음과 같이 RTP 연결을 위한 RSCV가 각 노드에서 보관되고 표시될 것입니다.

- RSCV에는 국지 노드로부터 상대방 RTP 노드로의 모든 홉이 포함되어 있을 것입니다.
- 상대방 RTP가 RTP 연결이 활성화되게 하는 세션의 끝점이 아닌 경우에는, RSCV가 상대방 RTP 노드로부터 끌고 오는 하나의 “경계 기능 홉” 역시 보관할 것입니다.

- 국지 노드가 세션 끝점을 포함하지 않을 지라도 RSCV는 절대로 국지 노드로 이끄는 경계 기능 홉을 포함하지 않을 것입니다.

경로 전환이 발생한 후에는, 보관되고 표시된 RSCV들이 국지 노드로부터 상대방 RTP 노드로의 홉만을 포함할 것입니다.(경계 기능 홉을 절대로 포함하지 않습니다.)

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_RTP_CONNECTION

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

rtp_name은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
rtp_name이 무시되며 반환되는 리스트가 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

rtp_name
RTP 연결 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

rtp_connection_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

rtp_connection_summary.rtp_name
RTP 연결 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

rtp_connection_summary.first_hop_ls_name
RTP 연결의 첫번째 홉의 링크 스테이션명. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

rtp_connection_summary.dest_node_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 RTP 연결의 목적지 노드의 전체 정식 17 바이트의 이름(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다).

rtp_connection_summary.cos_name
RTP 연결에 대한 서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

rtp_connection_summary.num_sess_active
RTP 연결에서 현재 활동중인 세션 수.

rtp_connection_detail.overlay_size
이 입력항목에 있는 바이트의 수(RSCV를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

rtp_connection_detail.rtp_name
RTP 연결 이름. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

rtp_connection_detail.first_hop_ls_name
RTP 연결의 첫번째 홉의 링크 스테이션명. 이는 국지로 표시가능한 문자 세트로 된 8 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

rtp_connection_detail.dest_node_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 RTP 연결의 목적지 노드의 전체 정식 17 바이트의 이름(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다).

rtp_connection_detail.isr_boundary_fn
국지 노드가 HPT-APPN 경계 기능을 제공하고 있는 임의의 ISR 세션에 대해 RTP 연결이 사용되고 있으면 AP_YES, 그렇지 않으면 AP_NO.

rtp_connection_detail.cos_name
RTP 연결에 대한 서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

rtp_connection_detail.max_btu_size
바이트 단위로 측정되는 RTP 연결에 대한 최대 BTU 크기.

rtp_connection_detail.liveness_timer
초 단위로 측정되는 RTP 연결에 대한 수명 타이머.

rtp_connection_detail.local_tcid
RTP 연결에 대한 국지 TCID.

rtp_connection_detail.remote_tcid
RTP 연결에 대한 원격 TCID.

rtp_connection_detail.rtp_stats.bytes_sent
국지 노드가 이 RTP 연결에서 송신한 총 바이트 수.

rtp_connection_detail.rtp_stats.bytes_received
국지 노드가 이 RTP 연결에서 수신한 총 바이트 수.

rtp_connection_detail.rtp_stats.bytes_resent
전송중 유실로 인해 국지 노드에 의해 재송신된 총 바이트 수.

rtp_connection_detail.rtp_stats.bytes_discarded
RTP 연결의 다른쪽에 의해 송신되고 이미 수신된 데이터의 중복으로 버려진 총 바이트 수.

rtp_connection_detail.rtp_stats.packets_sent
국지 노드가 이 RTP 연결에서 송신한 총 패킷 수.

rtp_connection_detail.rtp_stats.packets_received
국지 노드가 이 RTP 연결에서 수신한 총 패킷 수.

rtp_connection_detail.rtp_stats.packets_resent
전송중 유실로 인해 국지 노드에 의해 재송신된 총 패킷 수.

rtp_connection_detail.rtp_stats.packets_discarded
RTP 연결의 다른쪽에 의해 송신되고 이미 수신된 데이터의 중복으로 버려진 총 패킷 수.

rtp_connection_detail.rtp_stats.gaps_detected
국지 노드에 의해 감지된 총 갭 수. 각 갭은 하나 이상의 유실 프레임에 해당됩니다.

rtp_connection_detail.rtp_stats.send_rate
이 RTP 연결상의 현재 송신 비율(초당 키로비트 단위로 측정). 이는 ARB 알고리즘에 의해 계산된 대로의 최대 허용 송신 비율입니다.

rtp_connection_detail.rtp_stats.max_send_rate
이 RTP 연결상의 최대 송신 비율(초당 키로비트 단위로 측정).

rtp_connection_detail.rtp_stats.min_send_rate
이 RTP 연결상의 최소 송신 비율(초당 키로비트 단위로 측정).

rtp_connection_detail.rtp_stats.receive_rate
이 RTP 연결상의 현재 수신 비율(초당 키로비트 단위로 측정). 이는 최종 측정 간격 동안 계산된 실제 수신 비율입니다.

rtp_connection_detail.rtp_stats.max_receive_rate
이 RTP 연결상의 최대 수신 비율(초당 키로비트 단위로 측정).

rtp_connection_detail.rtp_stats.min_receive_rate
이 RTP 연결상의 최소 수신 비율(초당 키로비트 단위로 측정).

rtp_connection_detail.rtp_stats.burst_size
바이트 단위로 측정되는 RTP 연결상의 현재 버스트 크기.

rtp_connection_detail.rtp_stats.up_time
RTP 연결이 활동중이었던 총 시간(초 단위).

rtp_connection_detail.rtp_stats.smooth_rtt
국지 노드와 상대방 RTP 노드간의 유연하게 측정된 왕복 시간(1,000분의 1초 단위).

rtp_connection_detail.rtp_stats.last_rtt
국지 노드와 상대방 RTP 노드간의 최종 측정된 왕복 시간(1,000분의 1초 단위).

rtp_connection_detail.rtp_stats.short_req_timer
SHORT_REQ 타이머에 사용된 현재 지속 시간(1,000분의 1초 단위).

rtp_connection_detail.rtp_stats.short_req_timeouts
이 RTP 연결에 대해 SHORT_REQ 타이머가 만료된 총 횟수.

rtp_connection_detail.rtp_stats.liveness_timeouts
이 RTP 연결에 대해 수명 타이머가 만료된 총 횟수. 연결이 **rtp_connection_detail.liveness_timer**에서 지정된 기간 동안 유휴상태였을 때 수명 타이머가 만료됩니다.

rtp_connection_detail.rtp_stats.in_invalid_sna_frames
이 RTP 연결상에서 유효하지 않은 것으로 수신되고 버려진 총 SNA 프레임 수.

rtp_connection_detail.rtp_stats.in_sc_frames
이 RTP 연결상에서 수신된 총 세션 제어 프레임 수.

rtp_connection_detail.rtp_stats.out_sc_frames
이 RTP 연결상에서 송신된 총 세션 제어 프레임 수.

rtp_connection_detail.num_sess_active
RTP 연결에서 현재 활동중인 세션 수.

rtp_connection_detail.rscv_len
RTP 연결에 대한 첨부된 전송경로 선택 제어 벡터의 길이(아무것도 첨부되지 않으면, 길이가 0입니다). 다음 상세 입력항목의 정확한 정렬을 보장하기 위해 4 바이트 경계상에서 RSCV가 끝으로 채워지게 될 것이며, **rscv_len**은 이 채워넣기를 포함하지 않습니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_RTP_CONNECTION

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_SESSION

QUERY_SESSION은 노드가 끝점인 해당 세션에 대한 리스트 정보를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 모드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **session_id** 필드가 설정되어야 합니다. 그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. **lu_name**(또는 **lu_alias**) 그리고 **plu_alias**(또는 **fqplu_name**) 필드가 항상 설정됨에 주의하십시오. **lu_name**이 0이 아니면, **lu_alias**에 우선하여 사용될 것입니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

반환되는 세션들의 리스트는 상대방 LU 이름에 의해 필터링될 수도 있습니다. 이렇게 하려면, **fqplu_name**이나 **plu_alias** 필드가 설정되어야 합니다. **plu_alias**가 모두 0으로 설정되면, **fqplu_name** 값이 사용될 것이고, 그렇지 않으면 **plu_alias**가 항상 사용되고 **fqplu_name**은 무시됩니다.

반환되는 세션들의 리스트는 그들이 연관되어 있는 모드의 이름에 의해 필터링될 수 있습니다. 이 경우, **mode_name** 필드가 설정되어야 합니다.(그렇지 않으면, 이 필드가 모두 0으로 설정되어야 합니다.)

각 세션에 대한 상세 정보 외에, 이것이 START_NODE 매개변수에 지정되는 경우에는 전송경로 선택 제어 벡터(RSVC)가 반환될 것입니다(키 길이 형식으로). 이 RSCV(*Sna Formats*에서 지정된)는 세션이 hop-by-hop 형태에서 취하는 네트워크를 통한 전송경로를 정의합니다.

VCB 구조

```
typedef struct query_session
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   lu_name[8];         /* LU name                       */
   unsigned char   lu_alias[8];        /* LU alias                      */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   mode_name[8];       /* mode name                    */
   unsigned char   session_id[8];      /* session ID                    */
} QUERY_SESSION;
```

```
typedef struct session_summary
{
   unsigned short  overlay_size;       /* size of this entry           */
   unsigned char   plu_alias[8];      /* partner LU alias             */
   unsigned char   fqplu_name[17];    /* fully qualified partner      */
                                       /* LU name                      */
   unsigned char   reserv3[1];        /* reserved                     */
   unsigned char   mode_name[8];      /* mode name                    */
   unsigned char   session_id[8];     /* session ID                   */
   FQPCID          fqpcid;             /* fully qualified procedure    */
                                       /* correlator ID                */
} SESSION_SUMMARY;

typedef struct session_detail
{
   unsigned short  overlay_size;       /* size of this entry           */
   unsigned char   plu_alias[8];       /* partner LU alias             */
   unsigned char   fqplu_name[17];     /* fully qualified partner      */
                                       /* LU name                      */
   unsigned char   reserv3[1];         /* reserved                     */
   unsigned char   mode_name[8];       /* mode name                    */
   unsigned char   session_id[8];      /* session ID                   */
   FQPCID          fqpcid;            /* fully qualified procedure    */
                                       /* correlator ID                */
   unsigned char   cos_name[8];        /* Class-of-service name        */
   unsigned char   trans_pri;          /* Transmission priority:       */
   unsigned char   ltd_res;            /* Session spans a limited      */
                                       /* resource                     */
   unsigned char   polarity;           /* Session polarity             */
   unsigned char   contention;         /* Session contention           */
   SESSION_STATS   sess_stats;         /* Session statistics           */
   unsigned char   duplex_support;    /* full-duplex support          */
   unsigned char   sscp_id[6];         /* SSCP ID of host              */
   unsigned char   reserva[20];        /* reserved                     */
   unsigned long   session_start_time;/* start time of the session    */
   unsigned short  session_timeout;   /* session timeout              */
   unsigned char   reservb[7];         /* reserved                     */
   unsigned char   plu_slu_comp_lvl;  /* PLU to SLU compression level */
   unsigned char   slu_plu_comp_lvl;  /* SLU to PLU compression level */
   unsigned char   rscv_len;           /* Length of following RSCV     */
} SESSION_DETAIL;

typedef struct fqpcid
{
   unsigned char   pcid[8];            /* pro correlator identifier    */
   unsigned char   fqcp_name[17];      /* orig's network qualified     */
                                       /* CP name                      */
   unsigned char   reserve3[3];        /* reserved                     */
} FQPCID;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;        /* session receive RU size      */
   unsigned short  send_ru_size;       /* session send RU size         */
   unsigned short  max_send_btu_size; /* Maximum send BTU size        */
   unsigned short  max_rcv_btu_size;  /* Maximum rcv BTU size         */
   unsigned short  max_send_pac_win;  /* Max send pacing window size  */
   unsigned short  cur_send_pac_win;  /* Curr send pacing window size */
   unsigned short  max_rcv_pac_win;   /* Max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;   /* Curr rec pacing window size  */
   unsigned long   send_data_frames;  /* Number of data frames sent   */
   unsigned long   send_fmd_data_frames;
```

```
                                        /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;     /* Number of data bytes sent    */
   unsigned long   rcv_data_frames;     /* Num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                        /* num of FMD data frames recvd */
   unsigned long   rcv_data_bytes;      /* Num data bytes received      */
   unsigned char   sidh;                /* Session ID high byte         */
   unsigned char   sidl;                /* Session ID low byte          */
   unsigned char   odai;                /* ODAI bit set                 */
   unsigned char   ls_name[8];          /* Link station name            */
   unsigned char   pacing_type;         /* type of pacing in use        */
} SESSION_STATS;
```

주: 세션 상세 중복 뒤에는 *Sna Formats*에 의해 정의된 대로 전송경로 선택 제어 벡터(RSCV)가 이어질 것입니다. 이 제어 벡터는 네트워크를 통한 세션 전송경로를 정의하며 BIND상에서 운반됩니다. START_NODE 명령 상의 필드가 AP_YES로 설정되면 RSCV가 포함됩니다(키 길이 형식으로). START_NODE이면 **rscv_len**이 0으로 설정됩니다.

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_SESSION

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **lu_name**(또는 **lu_name**이 모두 0으로 설정된 경우 **lu_alias**), **plu_alias**(또는 **plu_alias**가 모두 0으로 설정된 경우

fqplu_name, **mode_name** 그리고 **session_id**(다음 매개변수 참조)의 조합은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
session_id가 무시되며 반환되는 리스트가 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다.

lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 이 필드는 **lu_name** 필드가 모두 0으로 설정된 경우에만 유효하지만, 이 경우 8 바이트 모두가 유효하며 설정되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용됩니다.

plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 이 필드가 모두 0으로 설정되면, **fqplu_alias** 필드가 색인 결정에 사용될 것입니다.

fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

mode_name
모드명 필터. 이는 모두 0으로 설정되거나 오른쪽은 EBCDIC 공백으로 채워지면서 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로 설정되어야 합니다. 이 필드가 설정되면 이 모드에 연관된 세션들만이 반환됩니다. 모두 0으로 설정되면, 이 필드가 무시됩니다.

session_id
세션의 8 바이트 식별자(ID). **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

session_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

session_summary.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

session_summary.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다

session_summary.mode_name
모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

session_summary.session_id
세션의 8 바이트 식별자(ID).

session_summary.fqpcid.pcid
절차 상관자 ID. 이는 8 바이트의 16진 문자열입니다.

session_summary.fqpcid.fqcp_name
전체 정식 제어점(CP) 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

session_detail.overlay_size
이 입력항목에 있는 바이트의 수(RSCV를 포함하여)로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

session_detail.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

session_detail.fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다

session_detail.mode_name
모드명. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

session_detail.session_id
세션의 8 바이트 식별자(ID).

session_detail.fqpcid.pcid
절차 상관자 ID. 이는 8 바이트의 16진 문자열입니다.

session_detail.fqpcid.fqcp_name
전체 정식 제어점(CP) 이름. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백없이 최대 8 바이트의 길이를 가질 수 있습니다.)

session_detail.cos_name
서비스 클래스(COS) 이름. 이는 8 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

session_detail.trans_pri
전송 우선순위. 다음 값 중 하나로 설정됩니다.

AP_LOW
AP_MEDIUM
AP_HIGH
AP_NETWORK

session_detail.ltd_res
세션이 자원 링크를 사용할지의 여부(AP_YES 또는 AP_NO)를 지정합니다.

session_detail.polarity
세션의 극성을 지정합니다(AP_PRIMARY 또는 AP_SECONDARY).

session_detail.contention
세션 회선경합 극성을 지정합니다. 이는 국지 LU가 이 세션의 '우선적 선택권'을 얻는지(AP_CONWINNER) 또는 세션 사용 전에 명령(bid)해야 하는지(AP_CONLOSER)를 나타냅니다.

session_detail.sess_stats.rcv_ru_size
최대 수신 RU 크기.

session_detail.sess_stats.send_ru_size
최대 송신 RU 크기.

session_detail.sess_stats.max_send_btu_size
송신될 수 있는 최대 BTU 크기.

session_detail.sess_stats.max_rcv_btu_size
수신될 수 있는 최대 BTU 크기.

session_detail.sess_stats.max_send_pac_win
이 세션상의 최대 송신 페이싱 창 크기.

session_detail.sess_stats.cur_send_pac_win
이 세션상의 현재 송신 페이싱 창 크기.

session_detail.sess_stats.max_rcv_pac_win
이 세션상의 최대 수신 페이싱 창 크기.

session_detail.sess_stats.cur_rcv_pac_win
이 세션상의 현재 수신 페이싱 창 크기.

session_detail.sess_stats.send_data_frames
송신된 정상 흐름 데이터 프레임의 갯수.

session_detail.sess_stats.send_fmd_data_frames
송신된 정상 흐름 FMD 데이터 프레임의 갯수.

session_detail.sess_stats.send_data_bytes
송신된 정상 흐름 데이터 바이트의 갯수.

session_detail.sess_stats.rcv_data_frames
수신된 정상 흐름 데이터 프레임의 갯수.

session_detail.sess_stats.rcv_fmd_data_frames
수신된 정상 흐름 FMD 데이터 프레임의 갯수.

session_detail.sess_stats.rcv_data_bytes
수신된 정상 흐름 데이터 바이트의 갯수.

session_detail.sess_stats.sidh
세션 ID 상위(high) 바이트.

session_detail.sess_stats.sidl
세션 ID 하위(low) 바이트.

session_detail.sess_stats.odai
원래 목적지 주소 표시기. 세션을 일으킬 때, 국지 노드가 1차 링크 스테이션을 포함하면 BIND의 송신자가 이 필드를 0으로 설정합니다. BIND 송신자가 2차 링크 스테이션을 포함하고 있는 노드라면, 이 필드를 1로 설정합니다.

session_detail.sess_stats.ls_name
통계와 연관된 링크 스테이션 이름. 이는 국지로 표시가능한 문자 세

트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다. 세션 통계를 세션 데이터가 흐르는 링크와 상관시키는 데 이 필드가 사용될 수 있습니다.

session_detail.sess_stats.pacing_type
이 세션상에서 사용중인 수신 페이싱 유형. 이는 값 AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE를 취할 수도 있습니다.

session_detail.duplex_support
BIND에서 조정된 대로의 대화 양방향(전송) 지원을 반환합니다. 이는 다음 값 중 하나입니다.

AP_HALF_DUPLEX
비동시 양방향(전송) 대화만이 지원됩니다.

AP_FULL_DUPLEX
비동시 양방향(전송)뿐만 아니라 동시 양방향(전송) 대화도 지원됩니다. 촉구된 데이터 역시 지원됩니다.

session_detail.sscp_id
종속 LU 세션의 경우, 이 필드는 국지 LU가 매핑되는 PU에 대한 호스트로부터의 ACTPU에서 수신된 SSCP ID를 포함합니다. 독립 LU 세션의 경우에는, 이 필드가 모두 2진 0으로 설정될 것입니다.

session_detail.session_start_time
시작하고 있는 CP와 활동중인 세션간에 경과한 시간으로 100분의 1초 단위로 측정됩니다. 조회가 처리될 때 세션이 완전히 활동중이지 않으면, 이 필드에 0이 반환됩니다.

session_detail.session_timeout
세션과 연관된 시간종료를 지정합니다. 이는 다음 값으로부터 파생됩니다.

국지 LU와 연관된 LU6.2 시간종료
원격 LU와 연관된 LU6.2 시간종료
모드 시간종료
글로벌 시간종료
이 세션이 제한된 자원 링크를 통해 실행중이라면 제한된 자원 시간종료

session_detail.plu_slu_comp_lvl
PLU에서 SLU로 송신된 데이터의 압축 레벨을 지정합니다.

AP_NONE
압축이 사용되지 않습니다.

AP_RLE_COMPRESSION
RLE 압축이 사용됩니다.

AP_LZ9_COMPRESSION
이 노드는 LZ9 압축을 지원합니다.

AP_LZ10_COMPRESSION
노드가 LZ10 압축을 지원할 수 있습니다.

AP_LZ12_COMPRESSION
노드가 LZ12 압축을 지원할 수 있습니다.

session_detail.slu_plu_comp_lvl
SLU에서 PLU로 송신된 데이터의 압축 레벨을 지정합니다.

AP_NONE
압축이 사용되지 않습니다.

AP_RLE_COMPRESSION
RLE 압축이 사용됩니다.

AP_LZ9_COMPRESSION
이 노드는 LZ9 압축을 지원합니다.

AP_LZ10_COMPRESSION
노드가 LZ10 압축을 지원할 수 있습니다.

AP_LZ12_COMPRESSION
노드가 LZ12 압축을 지원할 수 있습니다.

session_detail.rscv_len
session_detail 구조체로 추가되는 RSCV의 길이(아무것도 추가되지 않으면, 길이가 0입니다). RSCV는 4 바이트 경계에서 끝으로 채워질 것입니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_SESSION_ID

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_SIGNED_ON_LIST

QUERY_SIGNED_ON_LIST는 특정 LU로 사인 온한 사용자들에 대한 정보를 검색합니다.

국지 LU는 **lu_name**이나 **lu_alias**에 의해 지정됩니다. **Buf_ptr, buf_size, total_buf_size, num_entries, total_num_entries** 및 **overlay_size**는 QUERY 명령에 대한 보통의 의미를 지닙니다.

입력항목들은 **buf_ptr**에 의해 포인트되거나, **buf_ptr**이 NULL인 경우에는 QUERY_SIGNED_ON_LIST VCB에 첨부된 SIGNED_ON_LIST_ENTRY 구조체의 리스트로 반환됩니다. 리스트는 **plu_alias/fqplu_name**에 의해 정렬된 후, **user_id**에 의해 정렬되고, 그런 후 **profile**로 정렬됩니다. **plu_alias**가 지정되면, **fqplu_name**이 무시됩니다.

list_options는 값 AP_FIRST_IN_LIST, AP_LIST_FROM_NEXT, AP_LIST_INCLUSIVE를 취할 수 있습니다. **list_options**가 AP_FIRST_IN_LIST이면, **plu_alias, fqplu_name, user_id** 및 **profile**이 무시됩니다. **list**는 어느 리스트로부터 입력항목을 반환될지를 지정하는데, 이는 AP_SIGNED_ON_TO_LIST이어야 합니다.

VCB 구조

```
typedef struct query_signed_on_list
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   lu_name[8];         /* LU name                       */
   unsigned char   lu_alias[8];        /* LU alias                      */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   fqplu_name[17];     /* fully qualified partner       */
                                       /* LU name                       */
   unsigned char   user_id[10];        /* User ID                       */
   unsigned char   profile[10];        /* Profile                       */
   unsigned char   list;               /* Signed-on list type           */
} QUERY_SIGNED_ON_LIST;

typedef struct signed_on_list_entry
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   plu_alias[8];       /* partner LU alias              */
   unsigned char   user_id[10];        /* fully qualified partner       */
   unsigned char   profile[10];        /* profile                       */
} SIGNED_ON_LIST_ENTRY;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_QUERY_SIGNED_ON_LIST

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

지정된 **lu_name**(또는 **lu_name**이 모두 0으로 설정된 경우 **lu_alias**), **plu_alias**(또는 **plu_alias**가 모두 0으로 설정된 경우 **fqplu_name**), **user_id** 그리고 **profile**(다음 매개변수 참조)의 조합은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
pu_alias, fqplu_name, user_id 및 **profile**이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다.

lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 이 필드는 **lu_name** 필드가 모두 0으로 설정된 경우에만 유효하지만, 이 경우 8 바이트 모두가 유효하며 설정

되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용됩니다.

plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 이 필드가 모두 0으로 설정되면, **fqplu_alias** 필드가 색인 결정에 사용될 것입니다.

fqplu_name
상대방 LU에 대한 17 바이트의 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

user_id
사용자 ID. 이는 10 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다. 이 필드가 설정되면 이 모드에 연관된 세션들만이 반환됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

profile
10 바이트의 영숫자 EBCDIC 문자열. 프로그램은 현재 공백 프로파일(10 EBCDIC 공백)만을 지원함에 주의하십시오. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

list 사인온 리스트 유형. 이는 AP_SIGNED_ON_TO_LIST로 설정되어야 합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

signed_on_list_entry.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

signed_on_list_entry.plu_alias
상대방 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효합니다.

signed_on_list_entry.user_id
사용자 ID. 이는 10 바이트 영숫자 유형 A EBCDIC 문자열(글자로 시작)로, 오른쪽은 EBCDIC 공백으로 채워집니다.

signed_on_list_entry.profile
10 바이트의 영숫자 EBCDIC 문자열. 프로그램은 현재 공백 프로파일(10 EBCDIC 공백)만을 지원함에 주의하십시오.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_PLU_NAME
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_LIST
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료되었으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

QUERY_STATISTICS

QUERY_STATISTICS는 링크 스테이션 및 포트 통계를 조회합니다. 퍼스널 통신이나 통신 서버는 이 조회를 직접 DLC로 건넵니다. 통계의 형식은 DLC 구현에 따라 다릅니다.

VCB 구조

```
typedef struct query_statistics
{
   unsigned short  opcode;          /* verb operation code        */
   unsigned char   reserv2;         /* reserved                   */
   unsigned char   format;          /* format                     */
   unsigned short  primary_rc;      /* primary return code        */
   unsigned long   secondary_rc;    /* secondary return code      */
   unsigned char   name[8];         /* LS or port name            */
   unsigned char   stats_type;      /* LS or port statistics?     */
   unsigned char   table_type;      /* statistics table requested */
   unsigned char   reset_stats;     /* reset the statistics?      */
   unsigned char   dlc_type;        /* type of DLC                */
   unsigned char   statistics[256]; /* current statistics         */
   unsigned char   reserva[20];     /* reserved                   */
} QUERY_STATISTICS;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_STATISTICS

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

name 링크 스테이션이나 포트에 대해 정의된 이름(**stats_type** 매개변수의 설정에 따라). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 8 바이트 모두 유효하며 설정되어야 합니다. 퍼스널 통신이나 통신 서버는 이를 사용히여 응답을 정확한 링크 스테이션이나 포트에 상관시킵니다.

stats_type

통계가 요청된 해당 자원의 유형. 이는 다음 값 중 하나로 설정되어야 합니다.

AP_LS
AP_PORT

table_type

요청된 통계 테이블의 유형. 이는 다음의 정보 범주 중 하나로 설정되어야 합니다.

AP_STATS_TBL

통계 정보가 반환될 것이라고 지정합니다.

AP_ADMIN_TBL
관리 정보가 반환될 것이라고 지정합니다.

AP_OPER_TBL
운영 정보가 반환될 것이라고 지정합니다. 각 범주에 대해 반환된 정보의 형식은 DLC 구현에 따라 고유합니다.

reset_stats
통계가 재설정되어야 하는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

dlc_type
DLC 유형. 이 필드의 값은 DLC 구현에 따라 고유합니다. 이 값은 다음과 같습니다.

AP_ANYNET
AP_LLC2
AP_OEM_DLC
AP_SDLC
AP_TWINAX
AP_X25

statistics
링크 스테이션이나 포트의 현재 통계.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LINK_NAME

AP_INVALID_PORT_NAME
AP_INVALID_STATS_TYPE
AP_INVALID_TABLE_TYPE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_LINK_DEACTIVATED

AP_PORT_DEACTIVATED

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_TP

QUERY_TP는 국지 LU에 의해 현재 사용되고 있는 트랜잭션 프로그램(TP)에 관한 정보를 반환합니다.

정보는 리스트로 반환됩니다. 특정 트랜잭션 프로그램(TP)에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **tp_name** 필드가 설정되어야 합니다. **list_options** 필드가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시될 것입니다. **lu_name**이나 **lu_alias** 필드가 항상 설정되어야 함을 주의하십시오. **lu_name** 필드가 0이 아니면, **lu_alias** 필드에 우선하여 사용될 것입니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 동일한 길이의 이름에 대해 EBCDIC 사전식 정렬하기를 사용하여 **tp_name**에 의해 정렬됩니다. 이 명령은 일단 트랜잭션 프로그램(TP)이 국지 LU에 의해 사용되기 시작한 후 결정되는 정보를 반환합니다. QUERY_TP_DEFINITION 명령은 정의 정보만을 반환합니다.

VCB 구조

```
typedef struct query_tp
{
   unsigned short  opcode;             /* Verb operation code           */
   unsigned char   attributes;         /* verb attributes               */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   lu_name[8];         /* LU name                       */
   unsigned char   lu_alias[8];        /* LU alias                      */
   unsigned char   tp_name[64];        /* TP name                       */
} QUERY_TP;

typedef struct tp_data
{
   unsigned short  overlay_size;        /* size of this entry           */
   unsigned char   tp_name[64];        /* TP name                       */
   unsigned char   description[RD_LEN];
                                       /* resource description         */
   unsigned short  instance_limit;      /* max instance count           */
   unsigned short  instance_count;      /* current instance count       */
   unsigned short  locally_started_count;
                                       /* locally started instance     */
                                       /* count                        */
   unsigned short  remotely_started_count;
                                       /* remotely started instance    */
                                       /* count                        */
   unsigned char   reserva[20];         /* reserved                     */
} TP_DATA;
```

```
typedef struct tp_spec_data
{
  unsigned char pathname[256];         /* path and TP name              */
  unsigned char parameters[64];        /* parameters for TP             */
  unsigned char queued;                /* queued TP (AP_YES)            */
  unsigned char load_type;             /* type of load-DETACHED/CONSOLE */
  unsigned char dynamic_load;          /* dynamic loading of TP enabled */
  unsigned char reserved[5];           /* max size is 120 bytes         */
} TP_SPEC_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_TP

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

리스트 정보에서 무엇이 반환되어야 하는지를 나타냅니다. 지정된 **lu_name**(**lu_name**이 모두 0으로 설정되면, **lu_alias**) 및 **tp_name**의 조합(다음 매개변수 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

lu_name
LU명. 이 이름은 8 바이트의 유형 A EBCDIC 문자열입니다. 이 필드가 모두 0으로 설정되면, **lu_alias** 필드가 색인 결정에 사용될 것입니다.

lu_alias
국지로 정의된 LU 별명. 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 이 필드는 **lu_name** 필드가 모두 0으로 설정된 경우에만 유효하지만, 이 경우 8 바이트 모두가 유효하며 설정되어야 합니다. **lu_name**과 **lu_alias** 필드 둘다 모두 0으로 설정되면 제어점(CP)과 연관된 LU(생략시 LU)가 사용될 것입니다.

tp_name
트랜잭션 프로그램(TP) 명. 이는 오른쪽이 공백으로 채워지는 64 바이트의 문자열입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

tp_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

tp_data.tp_name
트랜잭션 프로그램(TP) 명. 이는 오른쪽이 공백으로 채워지는 64 바이트의 문자열입니다.

tp_data.instance.description
자원 설명(DEFINE_TP에서 지정된 것과 같이). 이는 국지로 표시가 능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

tp_data.instance_limit
지정된 트랜잭션 프로그램(TP)의 현재 활동중인 최대 인스턴스 수.

tp_data.instance_count
현재 활동중인 지정된 트랜잭션 프로그램(TP)의 인스턴스 수.

tp_data.locally_started_count
국지로 시작된(TP_STARTED 명령을 발행하는 트랜잭션 프로그램에 의해) 지정된 트랜잭션 프로그램(TP)의 인스턴스 수.

tp_data.remotely_started_count
원격으로 시작된(수신된 Attach 요청에 의해) 지정된 트랜잭션 프로그램(TP)의 인스턴스 수.

tp_chars.tp_data.pathname
경로나 트랜잭션 프로그램명을 지정합니다.

tp_chars.tp_data.parameters
트랜잭션 프로그램(TP)에 대한 매개변수를 지정합니다.

tp_chars.tp_data.queued
트랜잭션 프로그램(TP)이 대기행렬에 대기될지의 여부를 지정합니다.

tp_chars.tp_data.load_type
트랜잭션 프로그램(TP)이 로드될지의 여부를 지정합니다.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TP_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

QUERY_TP

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

QUERY_TP_DEFINITION

QUERY_TP_DEFINITION은 DEFINE_TP 명령에서 이전에 건네진 정보와 퍼스널 통신이나 통신 서버 정의 트랜잭션 프로그램(TP)에 관한 정보 둘다를 반환합니다.

정보는 요약이나 상세 정보, 이렇게 두 개의 형식 중 하나로 된 리스트로 반환됩니다. 특정 트랜잭션 프로그램(TP)에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **tp_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **tp_name**에 의해 정렬됩니다. AP_LIST_FROM_NEXT를 선택하는 경우, 반환되는 리스트는 정의된 정렬하기에 따라 다음 입력항목으로부터 시작합니다(지정된 입력항목이 있든 없든 간에).

이 명령은 정의 정보만을 반환합니다. QUERY_TP 명령은 일단 트랜잭션 프로그램(TP)이 국지 LU에 의해 사용되기 시작한 후 결정되는 정보를 반환합니다.

VCB 구조

```
typedef struct query_tp_definition
{
   unsigned short  opcode;             /* Verb operation code           */
   unsigned char   attributes;         /* verb attributes               */
   unsigned char   format;             /* format                        */
   unsigned short  primary_rc;         /* Primary return code           */
   unsigned long   secondary_rc;       /* Secondary return code         */
   unsigned char   *buf_ptr;           /* pointer to buffer             */
   unsigned long   buf_size;           /* buffer size                   */
   unsigned long   total_buf_size;     /* total buffer size required    */
   unsigned short  num_entries;        /* number of entries             */
   unsigned short  total_num_entries;  /* total number of entries       */
   unsigned char   list_options;       /* listing options               */
   unsigned char   reserv3;            /* reserved                      */
   unsigned char   tp_name[64];        /* TP name                       */
} QUERY_TP_DEFINITION;

typedef struct tp_def_summary
{
   unsigned short  overlay_size;       /* size of this entry            */
   unsigned char   tp_name[64];        /* TP name                       */
   unsigned char   description[RD_LEN];
                                       /* resource description          */
} TP_DEF_SUMMARY;
```

```
typedef struct tp_def_detail
{
   unsigned short  overlay_size;       /* size of this entry           */
   unsigned char   tp_name[64];        /* TP name                      */
   TP_CHARS        tp_chars;           /* TP characteristics           */
} TP_DEF_DETAIL;

typedef struct tp_chars
{
   unsigned char   description[RD_LEN];
                                       /* resource description         */
   unsigned char   conv_type;          /* conversation type            */
   unsigned char   security_rqd;       /* security support             */
   unsigned char   sync_level;         /* synchronization level support */
   unsigned char   dynamic_load;       /* dynamic load                 */
   unsigned char   enabled;            /* is the TP enabled?           */
   unsigned char   pip_allowed;        /* program initialization       */
                                       /* parameters supported         */
   unsigned char   duplex_support;     /* duplex supported             */
   unsigned char   reserv3[9];         /* reserved                     */
   unsigned short  tp__instance_limit; /* limit on currently active TP  */
                                       /* instances                    */
   unsigned short  incoming_alloc_timeout;
                                       /* incoming allocation timeout   */
   unsigned short  rcv_alloc_timeout;  /* receive allocation timeout    */
   unsigned short  tp_data_len;        /* TP data length               */
   TP_SPEC_DATA    tp_data;            /* TP data                      */
} TP_CHARS;

typedef struct tp_spec_data
{
  unsigned char pathname[256];         /* path and TP name             */
  unsigned char parameters[64];        /* parameters for TP            */
  unsigned char queued;                /* queued TP (AP_YES)           */
  unsigned char load_type;             /* type of load-DETACHED/CONSOLE */
  unsigned char dynamic_load;          /* dynamic loading of TP enabled */
  unsigned char reserved[5];           /* max size is 120 bytes        */
} TP_SPEC_DATA;
```

제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_QUERY_TP_DEFINITION

attributes

명령의 속성. 이 필드는 한 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램

은 VCB의 끝에 데이터를 첨부할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다.

AP_SUMMARY
요약 정보만을 반환합니다.

AP_DETAIL
상세 정보를 반환합니다.

지정된 **tp_name**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

tp_name
정의된 트랜잭션 프로그램(TP)의 이름. 이는 오른쪽이 공백으로 채워넣기되는 64 바이트의 문자열입니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

tp_def_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

tp_def_summary.tp_name
정의된 트랜잭션 프로그램(TP) 명. 이는 오른쪽이 공백으로 채워지는 64 바이트의 문자열입니다.

tp_def_summary.description
자원 설명(DEFINE_TP에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

tp_def_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

tp_def_detail.tp_name
정의된 트랜잭션 프로그램(TP) 명. 이는 오른쪽이 공백으로 채워넣기되는 64 바이트의 문자열입니다.

tp_def_detail.tp_chars.description
자원 설명(DEFINE_TP에서 지정된 것과 같이). 이는 국지로 표시가능한 문자 세트로 된 16 바이트의 문자열입니다. 16 바이트 모두 유효합니다.

tp_def_detail.tp_chars.conv_type
트랜잭션 프로그램(TP)에 의해 지원되는 대화의 유형을 지정합니다.

AP_BASIC
AP_MAPPED
AP_EITHER

tp_def_detail.tp_chars.security_rqd
트랜잭션 프로그램(TP)을 시작하는 데 대화 보안 정보가 필요한지의 여부를 지정합니다(AP_NO 또는 AP_YES).

tp_def_detail.tp_chars.sync_level
트랜잭션 프로그램(TP)에 의해 지원되는 동기화 레벨을 지정합니다.

AP_NONE
트랜잭션 프로그램(TP)이 없음(None)의 동기화 레벨을 지원합니다.

AP_CONFIRM_SYNC_LEVEL
트랜잭션 프로그램(TP)이 확약(Confirm)의 동기화 레벨을 지원합니다.

AP_EITHER
트랜잭션 프로그램(TP)이 없음 또는 확약의 동기화 레벨을 지원합니다.

AP_SYNCPT_REQUIRED
트랜잭션 프로그램(TP)이 동기점의 동기화 레벨을 지원합니다.

AP_SYNCPT_NEGOTIABLE
트랜잭션 프로그램(TP)이 없음, 확약 또는 동기점의 동기화 레벨을 지원합니다.

tp_def_detail.tp_chars.dynamic_load
트랜잭션 프로그램(TP)이 동적으로 로드될 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

tp_def_detail.tp_chars.enabled
트랜잭션 프로그램(TP)이 성공적으로 접속될 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 생략시는 AP_NO입니다.

tp_def_detail.tp_chars.pip_allowed
트랜잭션 프로그램(TP)이 프로그램 초기화(PIP) 매개변수를 수신할 수 있는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

tp_def_detail.tp_chars.duplex_support
트랜잭션 프로그램(TP)이 동시 양방향(전송)인지 비동시 양방향(전송)인지를 지시합니다.

AP_FULL_DUPLEX
트랜잭션 프로그램(TP)이 동시 양방향(전송)임을 지정합니다.

AP_HALF_DUPLEX
트랜잭션 프로그램(TP)이 비동시 양방향(전송)임을 지정합니다.

AP_EITHER_DUPLEX
트랜잭션 프로그램(TP)이 비동시 양방향(전송) 또는 동시 양방향(전송) 중 하나일 수 있다고 지정합니다.

tp_def_detail.tp_chars.tp_instance_limit
현재 활동중인 트랜잭션 프로그램(TP) 인스턴스의 수에 대한 제한.

tp_def_detail.tp_chars.incoming_alloc_timeout
입력 Attach가 RECEIVE_ALLOCATE를 기다리면서 대기행렬에 대기중인 시간(초 단위)을 지정합니다. 0은 시간종료가 없음을 의미하며, 따라서 무한정 대기중일 것입니다.

tp_def_detail.tp_chars.rcv_alloc_timeout
Attach를 기다리는 동안 RECEIVE_ALLOCATE 명령이 대기행렬에 대

기중일 시간(초 단위)을 지정합니다. 0은 시간종료가 없음을 의미하며, 따라서 무한정 대기중일 것입니다.

tp_def_detail.tp_chars.tp_data_len
구현 의존적 트랜잭션 프로그램(TP) 데이터의 길이.

tp_def_detail.tp_chars.tp_data
DYNAMIC_LOAD_INDICATION상에서 변경되지 않은 상태로 전달되는 구현 의존적 트랜잭션 프로그램(TP) 데이터.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TP_NAME

AP_INVALID_LIST_OPTION

노드가 아직 시작되지 않았으므로 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

제7장 안전 저장 명령

이 장에서는 네트워크 노드에서 발행되는 명령에 대해 설명합니다.

SAFE_STORE_TOPOLOGY

SAFE_STORE_TOPOLOGY는 네트워크 노드에서만 사용되며 노드 재시작시 나중에 액세스할 수 있도록 위상 정보를 안전하게 보관합니다. 정보가 저장되고 있는지(AP_NO) 또는 액세스되고 있는지(AP_YES)를 나타내는 데 **restore** 플래그가 사용됩니다.

노드 저장 정보는 형식된 리스트로 반환됩니다. 특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **index** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **index_node_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). 다음으로는 숫자 값에 의해 **index_node_type**상에서 리스트가 정렬됩니다. TG들이 저장 또는 복원되고 있으면, **index.tg_dest_node_name**에서 정렬이 행해지고(MIB 정렬하기), 그 후에는 **index.tg_dest_node_type** (숫자 값에 의해)에서 정렬이 행해지며, 세 번째로는 **index.tg_number** (숫자 값에 의해)에서 정렬이 행해집니다.

SAFE_STORE_TOPOLOGY 명령은 SFS_ADJACENT_NN, SFS_NN_TOPOLOGY_NODE 및 SFS_NN_TOPOLOGY_TG 명령을 망라합니다. 이는 조회 중복으로(부터) 변환하는 대신에, 위상에 나타나는 대로 제어 벡터를 사용하여 위상 정보를 저장합니다. 알 수 없는 제어 벡터가 저장 및 복원되며, 손상된 데이터가 위상안으로 도입되는 것을 막기 위해 검사합계가 제공됩니다.

VCB 구조

```
typedef struct safe_store_topology
{
        unsigned short  opcode;             /* verb operation code           */
        unsigned char   reserv2;            /* reserved                      */
        unsigned char   format;             /* format                        */
        unsigned short  primary_rc;         /* primary return code           */
        unsigned long   secondary_rc;       /* secondary return code         */
        unsigned char   buf_ptr;            /* pointer to buffer             */
        unsigned long   buf_size;           /* buffer size                   */
        unsigned long   total_buf_size;     /* total buffer size required    */
                                            /* to hold all information       */
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  total_num_entries;  /* total number of entries       */
        unsigned char   list_options;       /* listing options               */
        unsigned char   restore;            /* store or restore;             */
        unsigned char   resource_types;     /* resource types (nodes, TGs...)*/
        RESOURCE_INDEX  index;              /* resource index                */
        unsigned long   frsn;               /* flow-reduction sequence       */
                                            /* number                        */
        unsigned char   reserv3[16];        /* reserved                      */
      } SAFE_STORE_TOPOLOGY;
```

```
typedef struct resource_index
{
        unsigned char   node_name[17];      /* FQ node name                  */
        unsigned char   node_type;          /* node type                     */
        unsigned char   tg_dest_node_name[17];
                                            /* FQ name of TG destination node*/
        unsigned char   tg_dest_node_type;  /* TG destination node type      */
        unsigned char   tg_number;          /* TG number                     */
        unsigned char   reserv1[3];         /* reserved                      */
} RESOURCE_INDEX;

typedef struct safe_store_data
{
        unsigned short  overlay_size;       /* overalllength of safe         */
                                            /* store data                    */
        unsigned short  sub_overlay_size;   /* offset to first appended      */
                                            /* resource                      */
       RESOURCE_INDEX  index;               /* index of appended resource    */
        unsigned char   checksum[16];       /* reserved                      */
} RESOURCE_INDEX;


<!-

 -->

typedef struct safe_store_node_data
{
        unsigned short  overlay_size;       /* overalllength of safe         */
                                            /* store data                    */
        unsigned short  sub_overlay_size;   /* offset to first appended      */
        unsigned char   adjacent;           /* is this NNCP and adjacent     */
                                            /* NNCP?                         */
        unsigned char   reserv1;            /* reserved                      */
        unsigned long   last_frsn_sent;     /* last flow reduction sequence  */
                                            /* num sent (if node is adjacent)*/
                                            /* resource                      */
       unsigned long   last_frsn_rcvd;      /* last flow reduction sequence  */
                                            /* num rcvd (if node is adjacent)*/
        unsigned long   frsn;               /* flow reduction sequence number*/
        unsigned short  days_left           /* days left in database         */
       RESOURCE_INDEX  index;               /* index of appended resource    */
} SAFE_STORE_NODE_DATA;

typedef struct safe_store_tg_data
{
        unsigned short  overlay_size;       /* overalllength of safe         */
                                            /* store data                    */
        unsigned short  sub_overlay_size;   /* offset to first appended      */
                                            /* resource                      */
        unsigned long   frsn;               /* flow reduction sequence number*/
        unsigned short  days_left           /* days left in database         */
        unsigned short  vector_len;         /* length of appended vector(s)  */
} SAFE_STORE_TG_DATA;
```

제공되는 매개변수

restore = AP_NO일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SAFE_STORE_TOPOLOGY

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

SAFE_STORE_TOPOLOGY

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **resource_types** 및 **index**(다음 매개변수 참조)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

restore

정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

resource_types

이 비트 필드는 저장될 위상 데이터를 제어합니다. 다음 값들의 어떠한 조합이든 이 필드에서 가능합니다.

AP_SFS_NODES

위상 노드 저장

AP_SFS_ADJ_NODES

인접 노드 저장

AP_SFS_TGS

TG 저장

주: 이들 세 플래그 중 최소한 하나는 설정되어야 합니다. 인접 노드 및 위상 노드는 APPN내에서 별도의 엔티티이며, 첫번째 두 플래그는 어느 조합으로 든지 설정될 수 있습니다.

index.node_name

네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이

이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면 이 필드가 무시됩니다. AP_SFS_NODES 또는 AP_SFS_ADJ_NODES 어느 것도 **resource_types**에서 설정되지 않으면 이 필드 역시 무시됩니다.

index.node_type

노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN
AP_LEARN_NODE

node_type을 알 수 없으면, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다. AP_SFS_NODES 또는 AP_SFS_ADJ 어느 것도 **resource_types**에서 설정되지 않으면 이 필드 역시 무시됩니다.

index.tg_dest_node_name

TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다. AP_SFS_NODES 또는 AP_SFS_ADJ_NODES 어느 것도 **resource_types**에서 설정되지 않으면 이 필드 역시 무시됩니다.

index.tg_dest_node_type

이 TG에 대한 목적지 노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

tg_dest_node_type을 알 수 없으면, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면 이 필드가 무시됩니다. AP_SFS_TGS가 **resource_types**에서 설정되지 않으면 이 필드 역시 무시됩니다.

index.tg_number

TG와 연관된 번호. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다. AP_SFS_TGS가 **resource_types**에서 설정되지 않으면 이 필드 역시 무시됩니다.

frsn 흐름 감소 순서 번호(frsn). 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 위상 자원들만이 반환됩니다.

safe_store_data.overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 다음 SAFE_STORE_DATA 중복(있다면)에 대한 오프셋입니다.

safe_store_data.sub_overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 첨부된 SAFE_STORE_DATA 또는 SAFE_STORE_TG_DATA에 대한 오프셋입니다. 첨부된 데이터를 액세스할 때에는 이 필드가 항상 사용되어야 합니다.

safe_store_data.index
이 입력항목을 위한 색인. 후속 입력항목들을 나열하기 위해 후속 SAFE_STORE_TOPOLOGY 명령에 이 구조체가 제공될 수 있습니다. **dest_tg_name**이 모두 2진 0으로 설정되면, SAFE_STORE_NODE_DATA 중복이 이어집니다. 그렇지 않으면, SAFE_STORE_TG_DATA 중복이 이어집니다.

safe_store_data.checksum
첨부된 중복 및 벡터를 위한 128 비트의 검사합계. 이 검사합계나 이어지는 데이터가 손상되면, 손상이 발견되고 명령이 거부될 가능성이 매우 높습니다.

safe_store_node_data.overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 추가된 SAFE_STORE_DATA 또는 SAFE_STORE_TG_DATA에 대한 오프셋입니다.

safe_store_node_data.sub_overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 추가된 SAFE_STORE_DATA 또는 SAFE_STORE_TG_DATA에 대한 오프셋입니다. 추가된 데이터를 액세스하는 데 이 필드가 항상 사용되어야 합니다.

safe_store_node_data.adjacent
AP_YES 또는 AP_NO.AP_YES이면, 이 입력항목이 인접 네트워크 노드에 해당합니다.

safe_store_node_data.last_frsn_sent
adjacent가 AP_YES로 설정되면, 이 필드는 인접 네트워크 노드로 송신된 최종 FRSN을 보존합니다. 그렇지 않으면, 이 필드가 0으로 설정됩니다.

safe_store_node_data.last_frsn_rcvd
adjacent가 AP_YES로 설정되면, 이 필드는 인접 네트워크 노드로 송신된 최종 FRSN을 보존합니다. 그렇지 않으면, 이 필드가 0으로 설정됩니다.

safe_store_node_data.frsn
이 노드가 위상(topology)에 나타나는 경우, 이 위상 자원에 대한 흐름 감소 순서 번호. 그렇지 않으면, 이 필드가 0으로 설정됩니다.

safe_store_node_data.days_left
그 존재가 확인될 수 없는 한, 제거되기 전에 위상 데이터베이스에 이 노드가 남아 있는 일 수. 0은 제한 없음을 의미합니다.

safe_store_node_data.vector_len
추가된 벡터의 길이. 0은 어떤 벡터도 추가되지 않음을 의미합니다.

safe_store_tg_data.overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 추가된 SAFE_STORE_DATA 또는 SAFE_STORE_TG_DATA에 대한 오프셋입니다.

safe_store_tg_data.sub_overlay_size
채워넣기를 포함하여 이 입력항목의 길이. 이는 추가된 벡터가 있는 경우, 이에 대한 오프셋입니다. 추가된 벡터를 액세스하는 데 이 필드가 항상 사용되어야 합니다.

safe_store_tg_data.frsn
이 TG에 대한 흐름 감소 순서 번호.

safe_store_tg_data.days_left
그 존재가 확인될 수 없는 한, 제거되기 전에 위상 데이터베이스에 TG가 남아 있는 일 수. 0은 제한 없음을 의미합니다.

safe_store_node_data.vector_len
추가된 벡터의 길이. 0은 어떤 벡터도 추가되지 않음을 의미합니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LIST_OPTION

AP_INVALID_NODE
AP_INVALID_RESOURCE_TYPES
AP_INVALID_TG

제공되는 매개변수

restore = AP_YES일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_SAFE_STORE_TOPOLOGY

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

resource_name
네트 $ v 2j W 110 1847 0.9953 0 TD (V)Tj /F99 1 47 0.9953 0 TD (B)Tj /F123

resource_type
이 비트 필드는 저장될 위상 데이터를 제어합니다. 다음 값들의 어떠한 조합이든 이 필드에서 가능합니다.

AP_SFS_NODES
위상 노드 복원

AP_SFS_ADJ_NODES
인접 노드 복원

AP_SFS_TGS
TG 복원

주: 이들 세 플래그중 최소한 하나는 설정되어야 합니다. 인접 노드 및 위상 노드는 APPN내에서 별도의 엔티티이며, 첫번째 두 플래그는 어느 조합으로든지 설정될 수 있습니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_CHECKSUM_FAILED

AP_DATA_CORRUPT
AP_INVALID_RESOURCE_TYPES

관련 START_NODE 매개변수가 설정되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

시스템에 네트워크 노드 지원이 구축되어 있지 않았으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

노드가 아직 시작되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

SAFE_STORE_TOPOLOGY

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

SFS_ADJACENT_NN

주: 이 명령은 SAFE_STORE_TOPOLOGY에 포함되었으며 프로그램의 이전 버전과의 호환성을 위해서만 보존됩니다.

SFS_ADJACENT_NN은 노드가 재시작되는 경우 나중에 액세스할 수 있도록 위상 정보를 안전하게 저장하는 데 사용됩니다. 정보가 저장되고 있는지(AP_NO) 또는 액세스되고 있는지(AP_YES)를 나타내는 데 **restore** 플래그가 사용됩니다.

restore 플래그가 AP_NO로 설정될 때, SFS_ADJACENT_NN은 인접 네트워크 노드(즉, CP-CP 세션이 활동중이거나, 활동중이었거나, 언젠가 활동중이었던 그러한 네트워크 노드)에 관한 정보를 반환합니다.

SFS 정보는 형식된 리스트로 반환됩니다. 특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **adj_nncp_name** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **adj_nncp_name**에서 정렬됩니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). AP_LIST_FROM_NEXT를 선택하면 정의된 정렬하기에 따라 다음 입력항목으로부터 리스트가 시작합니다(지정된 입력항목이 있든 없든 간에).

VCB 구조

```
typedef struct sfs_adjacent_nn
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   *buf_ptr;           /* pointer to buffer            */
        unsigned long   buf_size;           /* buffer size                  */
        unsigned long   total_buf_size;     /* total buffer size required   */
                                            /* to hold all information      */
        unsigned short  num_entries;        /* number of entries            */
        unsigned short  total_num_entries;  /* total number of entries      */
        unsigned char   list_options;       /* listing options              */
        unsigned char   restore;            /* store or restore;            */
        unsigned char   adj_nncp_name[17];  /* CP name of adj Network Node  */
} SFS_ADJACENT_NN;

typedef struct adj_nncp_data
{
        unsigned short  overlay_size;       /* size of this entry           */
        unsigned char   adj_nncp_name[17];  /* CP name of adj Network Node  */
        unsigned char   cp_cp_sess_status;  /* CP-CP session status         */
        unsigned COUNTER
                        out_of_seq_tdus;    /* out of sequence TDUs         */
```

```
        unsigned long   last_frsn_sent;     /* last FSRN sent               */
        unsigned long   last_frsn_rcvd;     /* last FRSN received           */
        unsigned char   reserva[20];        /* reserved                     */
} ADJ_NNCP_DATA;
```

제공되는 매개변수

restore = AP_NO일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_SFS_ADJACENT_NN

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **resource_types** 및 **index**(다음 매개변수 참조)는 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

adj_nncp_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두

개의 유형 A EBCDIC 문자열로 구성된 전체 정식 17 바이트의 인접 네트워크 노드 CP 명(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다). **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

adj_nncp_data.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

adj_nncp_data.adj_nncp_name
EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 인접 네트워크 노드의 17 바이트 전체 정식 CP 명(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다).

adj_nncp_data.cp_cp_sess_status
CP-CP 세션의 상태(AP_ACTIVE 또는 AP_INACTIVE).

adj_nncp_data.out_of_seq_tdus
이 노드로부터 수신된 out_of_sequence TDU들의 갯수.

adj_nncp_data.last_frsn_sent
이 노드로 송신된 마지막 흐름 감소의 순서 번호.

adj_nncp_data.last_frsn_rcvd
이 노드로부터 수신된 마지막 흐름 감소의 순서 번호.

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc

AP_INVALID_ADJ_NNCP_NAME

AP_INVALID_LIST_OPTION

제공되는 매개변수

restore = AP_YES일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SFS_ADJACENT_NN

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

num_entries

실제로 반환된 입력항목의 갯수.

restore

정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

adj_nncp_data.overlay_size

이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

adj_nncp_data.adj_nncp_name

EBCDIC 점으로 연결되고 오른쪽은 EBCDIC 공백으로 채워지는 두 개의 유형 A EBCDIC 문자열로 구성된 인접 네트워크 노드의 17 바이트 전체 정식 CP 명(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다).

adj_nncp_data.cp_cp_sess_status

restore가 AP_YES로 설정될 때에는 이 필드가 무시됩니다.

adj_nncp_data.out_of_seq_tdus

restore가 AP_YES로 설정될 때에는 이 필드가 무시됩니다.

adj_nncp_data.last_frsn_sent

이 노드로 송신된 마지막 흐름 감소의 순서 번호.

adj_nncp_data.last_frsn_rcvd

이 노드로부터 수신된 마지막 흐름 감소의 순서 번호.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

노드가 아직 시작되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

관련 START_NODE 매개변수가 송신되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

시스템에 네트워크 노드 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

SFS_DIRECTORY

QUERY_DIRECTORY_ENTRY verb외에, 네트워크 노드상의 국지 디렉토리 캐쉬(cache)를 안전하게 저장시키고 노드 재시작시 나중에 액세스될 수 있는 SFS_DIRECTORY 명령이 있습니다. 정보가 저장되고 있는지(AP_NO) 또는 액세스되고 있는지(AP_YES)를 나타내는 데 **restore** 플래그가 사용됩니다.

restore 플래그가 AP_YES로 설정될 때, SFS_DIRECTORY는 디렉토리 데이터베이스로 하여금 **directory_entry_summary** 중복을 사용하여 재구축될 수 있게 합니다. 특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **resource_name** 및 **resource_type** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12페이지의 『노드 조회』를 참조하십시오.

캐쉬(cache)된 입력항목이나 그들의 부모(parent)에 관한 자원 정보가 다음 순서로 반환됩니다.

첫번째 네트워크 노드
- 네트워크 노드에 위치한 첫번째 LU
- 네트워크 노드에 위치한 두 번째 LU
- ...
- 네트워크 노드에 위치한 n번째 LU

이 네트워크 노드가 서비스하는 첫번째 끝 노드
- 끝 노드(1)에 위치한 첫번째 LU
- 끝 노드(1)에 위치한 두 번째 LU
- ...
- 끝 노드(1)에 위치한 n번째 LU

...

이 네트워크 노드가 서비스하는 n번째 끝 노드
- 끝 노드(n)에 위치한 첫번째 LU
- 끝 노드(n)에 위치한 두 번째 LU
- ...

두 번째 네트워크 노드

...등등..

VCB 구조

```
typedef struct sfs_directory
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   *buf_ptr;           /* pointer to buffer            */
        unsigned long   buf_size;           /* buffer size                  */
        unsigned long   total_buf_size;     /* total buffer size required   */
```

```
        unsigned short  num_entries;        /* number of entries             */
        unsigned short  total_num_entries;  /* total number of entries       */
        unsigned char   list_options;       /* listing options               */
        unsigned char   restore;            /* store or restore flag         */
        unsigned char   resource_name[17];  /* network qualified res name    */
        unsigned char   reserv3;            /* reserved                      */
        unsigned short  resource_type;      /* Resource type                 */
       } SFS_DIRECTORY;

typedef struct directory_entry_summary
{
        unsigned short  overlay_size;        /* size of this entry          */
        unsigned char   resource_name[17];   /* network qualified res name  */
        unsigned char   reserve1;            /* reserved                    */
        unsigned short  resource_type;       /* Resource type               */
        unsigned short  real_owning_cp_type; /* real owning CP type         */
        unsigned char   real_owning_cp_name[17];
                                             /* real owning CP name         */
        unsigned char   description[RD_LEN]; /* resource description        */
} DIRECTORY_ENTRY_SUMMARY;
```

제공되는 매개변수

restore = AP_NO일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SFS_ADJACENT_NN

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **resource_name** 및 **resource_type** specified (다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

resource_type
자원 유형. 다음 중 하나를 참조하십시오.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

list_options가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

directory_entry_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

directory_entry_summary.resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두

개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

directory_entry_summary.resource_type
자원 유형. 다음 중 하나를 참조하십시오.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_RES_NAME

AP_INVALID_LIST_OPTION
AP_INVALID_RES_TYPE

directory_entry_summary.real_owning_cp_type
NN 및 BrNN 만: 실제 소유 CP 유형. 이는 다음 중 하나일 수 있습니다.

AP_NONE
실제 소유 CP는 부모(parent) 자원입니다.

AP_ENCP_RESOURCE
실제 소유 CP는 부모(parent) 자원이 아니고 EN입니다.

기타 노드 유형: 이 필드가 AP_NONE으로 설정됩니다.

directory_entry_summary.real_owning_cp_name
NN 및 BrNN 만: 전체 정식 소유 CP명. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

실제 소유 CP가 부모(parent)이면, 이 필드가 2진 0으로 설정됩니다.

실제 소유 CP가 부모(parent)가 아닌 경우에는 이 필드가 실제 소유 CP의 이름으로 설정됩니다.

자원을 BrNN의 도메인에 있는 EN이 소유하는 경우에는 실제 소유 CP가 BrNN의 NNS의 디렉토리에서 부모(parent)가 아닙니다. 이 경우 실제 소유 CP는 EN이지만, 부모(parent)는 BrNN입니다.

기타 노드 유형: 이 필드가 2진 0으로 설정됩니다.

제공되는 매개변수

resorte = AP_YES일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_SFS_DIRECTORY

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size
제공된 버퍼의 크기.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 응용프로그램이 디렉토리의 첫번째 『청크(chunk)』를 복원하고 있다면, 모두 0으로 설정되어야 합니다. 그렇지 않으면, 응용프로그램이 이를 이전 『청크(chunk)』에 있는 최종 항목의 자원 이름으로 설정해야 합니다.

resource_type
자원 유형. 다음 중 하나를 참조하십시오.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

응용프로그램이 디렉토리의 첫번째 『청크(chunk)』를 복원하고 있다면, 이 필드가 모두 0으로 설정되어야 합니다.

directory_entry_summary.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다. 이는 **restore**가 AP_NO로 설정될 때 반환된 **overlay_size** 값과 동일해야 합니다.

directory_entry_summary.resource_name
네트워크 정식 자원 이름. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두

개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

directory_entry_summary.resource_type
자원 유형. 다음 중 하나를 참조하십시오.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

directory_entry_summary.real_owning_cp_type
NN 및 BrNN 만: 실제 소유 CP 유형. 이는 다음 중 하나일 수 있습니다.

AP_NONE
실제 소유 CP는 부모(parent) 자원입니다.

AP_ENCP_RESOURCE
실제 소유 CP는 부모(parent) 자원이 아니고 EN입니다.

기타 노드 유형: 이 필드가 AP_NONE으로 설정됩니다.

directory_entry_summary.real_owning_cp_name
NN 및 BrNN 만: 전체 정식 소유 CP명. 이 이름의 길이는 17 바이트이며 오른쪽은 EBCDIC 공백으로 채워집니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

실제 소유 CP가 부모(parent)이면, 이 필드가 2진 0으로 설정됩니다.

실제 소유 CP가 부모(parent)가 아닌 경우에는 이 필드가 실실 소유 CP의 이름으로 설정됩니다.

자원을 BrNN의 도메인에 있는 EN이 소유하는 경우에는 실제 소유 CP가 BrNN의 NNS의 디렉토리에서 부모(parent)가 아닙니다. 이 경우 실제 소유 CP는 EN이지만, 부모(parent)는 BrNN입니다.

기타 노드 유형: 이 필드가 2진 0으로 설정됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_RES_NAME

AP_INVALID_LIST_OPTION

관련 START_NODE 매개변수가 송신되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

시스템에 네트워크 노드 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

노드가 아직 시작되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

SFS_NN_TOPOLOGY_NODE

주: 이 명령은 SAFE_STORE_TOPOLOGY에 포함되었으며 프로그램의 이전 버전과의 호환성을 위해서만 보존됩니다.

각 네트워크 노드는 네트워크에 있는 모든 네트워크 노드, VRN 및 네트워크 노드 대 네트워크 노드 TG들에 관한 정보를 보유하고 있는 네트워크 위상 데이터베이스를 유지보수합니다. SFS_NN_TOPOLOGY_NODE 명령은 노드가 재시작되는 경우 나중에 액세스될 수 있는 위상 데이터베이스 노드 입력항목을 안전하게 저장하는 데 사용됩니다. 정보가 저장되고 있는지(AP_NO) 또는 액세스되고 있는지(AP_YES)를 나타내는 데 **restore** 플래그가 사용됩니다.

특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **node_name** 및 **node_type** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **node_name**, **node_name_type** 그리고 **frsn**에 의한 것입니다. 정렬은 먼저 이름 길이로 행해지고 난 후, 동일한 길이를 가진 이름에 대해서는 ASCII 사전식 정렬에 의해 행해집니다(IBM의 6611 APPN MIB 정렬에 따라). **node_type**에 대한 정렬하기는 AP_NETWORK_NODE, 그리고 나선 AP_VRN입니다. **frsn**은 번호로 정렬됩니다.

- AP_LIST_INCLUSIVE가 선택되면, 반환되는 리스트가 그 이름의 첫번째 유효 레코드부터 시작합니다.
- AP_LIST_FROM_NEXT가 선택되면, 지정된 것 다음의 이름으로 첫번째 유효 레코드에서 리스트가 시작할 것입니다.

frsn 필드가 0이 아닌 값으로 설정되면, 이것보다 높은 흐름 감소 순서 번호(FRSN)를 가진 데이터베이스 입력항목들만이 반환됩니다. 이렇게 하면 노드의 현재 FRSN을 먼저 확보함으로써 많은 『chunks』에서 일관된 위상 데이터베이스가 반환됩니다. 이는 다음과 같이 작동합니다.

1. 노드의 현재 FRSN을 반환하는 QUERY_NODE를 발행합니다.
2. 『chunks』에 있는 모든 데이터베이스 입력항목을 확보하는 데 필요한 만큼의 SFS_NN_TOPOLOGY_NODE (FRSN이 0으로 설정된 상태로)를 발행합니다.
3. QUERY_NODE를 다시 발행하여 새로운 FRSN을 단계 1에서 반환된 것과 비교합니다.
4. 두 개의 FRSN이 서로 다르면 데이터베이스가 변경된 것이며, 그래서 FRSN을 단계 1에서 제공된 FRSN 보다 1 만큼 크게 설정한 상태로 SFS_NN_TOPOLOGY_NODE를 발행합니다.

VCB 구조

```
typedef struct sfs_nn_topology_node
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   *buf_ptr;           /* pointer to buffer            */
        unsigned long   buf_size;           /* buffer size                  */
        unsigned long   total_buf_size;     /* total buffer size required   */
        unsigned short  num_entries;        /* number of entries            */
        unsigned short  total_num_entries;  /* total number of entries      */
        unsigned char   list_options;       /* listing options              */
        unsigned char   restore;            /* store or restore;            */
        unsigned char   node_name[17];      /* network qualified            */
                                            /* node name                    */
        unsigned char   node_type;          /* node type                    */
        unsigned long   frsn;               /* flow-reduction sequence      */
                                            /* number                       */
} SFS_NN_TOPOLOGY_NODE;

typedef struct nn_topology_node_detail
{
        unsigned short  overlay_size;       /* size of this entry           */
        unsigned char   node_name[17];      /* network qualified            */
        unsigned char   node_type;          /* node type                    */
        unsigned short  days_left           /* days left in database        */
        unsigned long   frsn;               /* flow reduction sequence number*/
        unsigned long   rsn;                /* resource sequence number     */
        unsigned char   rar;                /* route additional resistence  */
        unsigned char   status;             /* node status                  */
        unsigned char   function_support;   /* function support             */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   reserva[20];        /* reserved                     */
} NN_TOPOLOGY_NODE_DETAIL;
```

제공되는 매개변수

restore = AP_NO일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SFS_NN_TOPOLOGY_NODE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options
이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **node_name, node_types** 및 **frsn** (다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

node_name
네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

node_type
노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

node_type을 알 수 없으면, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

frsn 흐름 감소 순서 번호. 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 위상 자원들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

nn_topology_node_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

nn_topology_node_detail.node_name
네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

nn_topology_node_detail.node_type
노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

nn_topology_node_detail.days_left
위상 데이터베이스에서 이 노드 입력항목을 삭제하기 전의 일 수. 국지 노드 입력항목에 대해서는 0으로 설정될 것입니다.(이 입력항목은 절대 삭제되지 않습니다.) 레코드가 복원될 때에는(예를 들어, **restore**가 AP_YES로 설정됩니다) 0으로 설정되어야 합니다.

nn_topology_node_detail.frsn
흐름 감소 순서 번호. 이는 국지 노드에서 자원이 최종 갱신된 시간을 나타냅니다.

nn_topology_node_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

nn_topology_node_detail.rar
네트워크 노드의 전송경로 추가 저항.

nn_topology_node_detail.status

이 필드는 노드의 상태를 지정하며 AP_UNCONGESTED이거나 다음 값들중 하나 또는 그 이상일 수 있습니다.

AP_CONGESTED
ISR 세션의 수가 START_NODE verb에서 지정된 **isr_sessions_upper_threshold** 보다 큽니다.

AP_IRR_DEPLETED
ISR 세션의 수가 START_NODE verb의 **max_isr_sessions** 매개변수에서 지정된 최대치에 도달했습니다.

AP_ERR_DEPLETED
끝점 세션의 수가 지정된 최대치에 도달했습니다.

AP_QUIESCING
유형 AP_QUIENCE 또는 AP_QUIENCE_ISR의 STOP_NODE가 발행되었습니다.

nn_topology_node_detail.function_support

이 필드는 어느 기능이 지원되는지를 지정합니다. 이는 다음 중 하나 또는 그 이상일 수 있습니다.

AP_BORDER_NODE
경계 노드 기능이 지원됩니다.

AP_CDS
중앙 디렉토리 서버가 지원됩니다.

AP_GATEWAY
노드가 게이트웨이 노드입니다.(기능이 아직 구조적으로 정의되지 않습니다.)

AP_ISR
이 노드가 중계 세션 경로지정(ISR)을 지원합니다.

AP_HPR
이 노드가 중계 세션 경로지정(ISR)을 지원합니다.

AP_RTP_TOWER
이 노드가 HPR의 RTP 타워를 지원합니다.

AP_CONTROL_OVER_RTP_TOWER
이 노드가 RTP 타워를 통한 제어 흐름을 지원합니다.

주: AP_CONTROL_OVER_RTP_TOWER 노드는 AP_HPR과 AP_RTP_TOWER 둘다의 설정값에 해당합니다.

명령이 성공적으로 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_LIST_OPTION

AP_INVALID_NODE
AP_INVALID_LIST_OPTIONS

제공되는 매개변수

restore = AP_YES일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SFS_NN_TOPOLOGY_NODE

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

restore

정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

nn_topology_node_detail.overlay_size

이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

nn_topology_node_detail.node_name

네트워크 위상 데이터베이스로부터의 네트워크 정식 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

nn_topology_node_detail.node_type

노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

nn_topology_node_detail.days_left
위상 데이터베이스에서 이 노드 입력항목을 삭제하기 전의 일 수. 노드가 국지 노드가 아닌 경우에는 이 필드가 0보다 큰 값으로 설정되어야 합니다.

nn_topology_node_detail.frsn
흐름 감소 순서 번호. 이는 국지 노드에서 자원이 최종 갱신된 시간을 나타냅니다.

nn_topology_node_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

nn_topology_node_detail.rar
네트워크 노드의 전송경로 추가 저항.

nn_topology_node_detail.status
이 필드는 노드의 상태를 지정하며 AP_UNCONGESTED이거나 다음 값들중 하나 또는 그 이상일 수 있습니다.

AP_CONGESTED
ISR 세션의 수가 START_NODE verb에서 지정된 **isr_sessions_upper_threshold** 보다 큽니다.

AP_IRR_DEPLETED
ISR 세션의 수가 START_NODE verb의 **max_isr_sessions** 매개변수에서 지정된 최대치에 도달했습니다.

AP_ERR_DEPLETED
끝점 세션의 수가 지정된 최대치에 도달했습니다.

AP_QUIESCING
유형 AP_QUIENCE 또는 AP_QUIENCE_ISR의 STOP_NODE가 발행되었습니다.

nn_topology_node_detail.function_support
이 필드는 어느 기능이 지원되는지를 지정합니다. 이는 다음 중 하나 또는 그 이상일 수 있습니다.

AP_BORDER_NODE
경계 노드 기능이 지원됩니다.

AP_CDS
중앙 디렉토리 서버가 지원됩니다.

AP_GATEWAY
노드가 게이트웨이 노드입니다.(기능이 아직 구조적으로 정의되지 않습니다.)

AP_ISR
이 노드가 중계 세션 경로지정(ISR)을 지원합니다.

AP_HPR
이 노드가 중계 세션 경로지정(ISR)을 지원합니다.

AP_RTP_TOWER
이 노드가 HPR의 RTP 타워를 지원합니다.

AP_CONTROL_OVER_RTP_TOWER
이 노드가 RTP 타워를 통한 제어 흐름을 지원합니다.

주: AP_CONTROL_OVER_RTP_TOWER 노드는 AP_HPR과 AP_RTP_TOWER 둘다의 설정값에 해당합니다.

node_type
노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

node_type을 알 수 없으면, AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

frsn 흐름 감소 순서 번호. 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 위상 자원들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc
AP_INVALID_DAYS_LEFT

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DAYS_LEFT

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DAYS_LEFT

관련 START_NODE 매개변수가 설정되지 않았으므로, 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

secondary_rc
AP_INVALID_DAYS_LEFT

시스템에 네트워크 노드 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

SFS_NN_TOPOLOGY_TG

주: 이 명령은 SAFE_STORE_TOPOLOGY에 포함되었으며 프로그램의 이전 버전과의 호환성을 위해서만 보존됩니다.

각 네트워크 노드는 네트워크에 있는 모든 네트워크 노드, VRN 및 네트워크 노드 대 네트워크 노드 TG들에 관한 정보를 갖고 있는 네트워크 위상 데이터베이스를 유지보수합니다. SFS_NN_TOPOLOGY_NODE 명령은 노드가 재시작되는 경우 나중에 액세스될 수 있는 위상 데이터베이스 노드 입력항목을 안전하게 저장하는 데 사용됩니다. 정보가 저장되고 있는지(AP_NO) 또는 액세스되고 있는지(AP_YES)를 나타내는 데 **restore** 플래그가 사용됩니다. 명령이 **topology_tg_detail** 중복을 사용합니다.

특정 네트워크 노드에 관한 정보를 확보하거나 여러 개의 『청크(chunk)』로 리스트 정보를 확보하려면, **owner, owner_type, dest, dest_type** 및 **tg_num** 필드가 설정되어야 합니다.

그렇지 않으면(**list_options** 필드가 AP_FIRST_IN_LIST로 설정되면), 이 필드가 무시됩니다. 리스트 형식이 어떻게 사용되는 지에 대한 백그라운드는 12 페이지의 『노드 조회』를 참조하십시오.

이 리스트는 **owner, owner_type, dest, dest_type, tg_num** 및 **frsn**에 의한 것입니다. **owner_type** 및 **dest** 이름은 먼저 이름 길이로 정렬된 후, 동일한 길이의 이름에 대해서는 ASCII 사전식 정렬이 됩니다(IBM의 6611 APPN MIB 정렬에 따라). **owner_type** 및 **dest**에 대한 정렬하기는 다음과 같습니다: AP_NETWORK_NODE, 그리고 나서 AP_VRN. **tg_num** 및 **frsn**은 번호로 정렬됩니다.

- AP_LIST_INCLUSIVE가 선택되면, 반환되는 리스트가 그 이름의 첫번째 유효 레코드부터 시작합니다.
- AP_LIST_FROM_NEXT가 선택되면, 지정된 것 다음의 이름으로 첫번째 유효 레코드에서 리스트가 시작할 것입니다.

frsn 필드가 0이 아닌 값으로 설정되면, 이것보다 높은 흐름 감소 순서 번호(FRSN)를 가진 데이터베이스 입력항목들만이 반환됩니다. 이렇게 하면 노드의 현재 FRSN을 먼저 확보함으로써 많은 『chunks』에서 일관된 위상 데이터베이스가 반환됩니다. 이는 다음과 같이 작동합니다.

1. 노드의 현재 FRSN을 반환하는 QUERY_NODE를 발행합니다.
2. 『chunks』에 있는 모든 데이터베이스 입력항목을 확보하는 데 필요한 만큼의 SFS_NN_TOPOLOGY_NODE(FRSN이 0으로 설정된 상태로)를 발행합니다.
3. QUERY_NODE를 다시 발행하여 새 FRSN을 단계 1에서 반환된 것과 비교합니다.
4. 두 개의 FRSN이 서로 다르면 데이터베이스가 변경된 것이며, 그래서 FRSN을 단계 1에서 제공된 FRSN 보다 1 만큼 크게 설정한 상태로 SFS_NN_TOPOLOGY_NODE를 발행합니다.

VCB 구조

```
typedef struct sfs_nn_topology_tg
{
        unsigned short  opcode;             /* verb operation code          */
        unsigned char   reserv2;            /* reserved                     */
        unsigned char   format;             /* format                       */
        unsigned short  primary_rc;         /* primary return code          */
        unsigned long   secondary_rc;       /* secondary return code        */
        unsigned char   *buf_ptr;           /* pointer to buffer            */
        unsigned long   buf_size;           /* buffer size                  */
        unsigned long   total_buf_size;     /* total buffer size required   */
        unsigned short  num_entries;        /* number of entries            */
        unsigned short  total_num_entries;  /* total number of entries      */
        unsigned char   list_options;       /* listing options              */
        unsigned char   restore;            /* store or restore;            */
        unsigned char   owner[17];          /* network qualified            */
                                            /* node name                    */
        unsigned char   owner_type;         /* node type                    */
        unsigned char   dest[17];           /* TG destination node          */
        unsigned char   dest_type;          /* TG destination node type     */
        unsigned char   tg_num;             /* TG number                    */
        unsigned char   reserv1;            /* reserved                     */
        unsigned long   frsn;               /* flow-reduction sequence      */
                                            /* number                       */
} SFS_NN_TOPOLOGY_TG;

typedef struct nn_topology_tg_detail
{
        unsigned short  overlay_size;       /* size of this entry           */
        unsigned char   owner[17];          /* network qualified            */
        unsigned char   owner_type;         /* node type                    */
        unsigned char   dest[17];           /* TG destination node          */
        unsigned char   dest_type           /* TG destination node type     */
        unsigned char   tg_num;             /* TG number                    */
        unsigned char   reserv3[1];         /* reserved                     */
        unsigned long   frsn;               /* flow reduction sequence number*/
        unsigned short  days_left           /* days left in database        */
        LINK_ADDRESS    dlc_data;           /* DLC signalling data          */
        unsigned long   rsn;                /* resource sequence number     */
        unsigned char   status;             /* node status                  */
        TG_DEFINED_CHAR tg_chars;           /* TG characteristics           */
        unsigned char   reserva[20];        /* reserved                     */
}TOPOLOGY_TG_DETAIL;

typedef struct link_address
{
        unsigned short  length              /* length                       */
        unsigned short  reserve1;           /* reserved                     */
        unsigned char   address[MAX_LINK_ADDR_LEN];
                                            /* address                      */
}LINK_ADDRESS;
```

제공되는 매개변수

restore = AP_NO일 때 제공되는 매개변수

응용프로그램은 다음의 매개변수를 제공합니다.

opcode

AP_SFS_NN_TOPOLOGY_TG

format

VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

SFS_NN_TOPOLOGY_TG

buf_ptr

리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

buf_size

제공된 버퍼의 크기. 반환된 데이터가 이 크기를 초과하지 않을 것입니다.

num_entries

반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

list_options

이는 리스트 정보에서 무엇이 반환되어야 하는가를 나타냅니다. 지정된 **owner, owner_type, dest, dest_type, tg_num** 및 **frsn**(다음 매개변수를 보십시오)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST

색인 값이 무시되며 반환되는 리스트는 리스트의 첫번째 입력항목으로부터 시작합니다.

AP_LIST_FROM_NEXT

반환되는 리스트가 제공된 색인 값에 의해 지정된 것 뒤에 리스트에서 다음 입력항목으로부터 시작합니다.

AP_LIST_INCLUSIVE

반환되는 리스트가 색인 값에 의해 지정된 입력항목으로부터 시작합니다.

restore

정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

owner TG의 근원지 노드 이름(항상 국지 노드 이름으로 설정됩니다. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면 이 필드가 무시됩니다.

owner_type

노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

owner_type을 알 수 없는 경우에는 AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dest TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

dest_type
노드 유형. 이 노드는 다음 값 중 하나로 설정됩니다.

AP_NETWORK_NODE
AP_VRN

dest_type을 알 수 없는 경우에는 AP_LEARN_NODE가 지정되어야 합니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면 이 필드가 무시됩니다.

tg_num
TG와 연관된 번호. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

frsn 흐름 감소 순서 번호. 이것이 0이 아니면, 이 값보다 크거나 같은 FRSN을 가진 위상 자원들만이 반환됩니다.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼의 크기를 나타내는 반환된 값. 이는 **buf_size**보다 높을 수도 있습니다.

num_entries
실제로 반환된 입력항목의 갯수.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

topology_tg_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다.

topology_detail.owner
TG의 근원지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

topology_tg_detail.owner_type
노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_detail.dest
TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면, 이 필드가 무시됩니다.

topology_tg_detail.dest_type
노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_detail.tg_num
TG와 연관된 번호.

topology_tg_detail.frsn
흐름 감소 순서 번호. 이는 국지 노드에서 자원이 최종 갱신된 시간을 나타냅니다.

topology_tg_detail.days_left
그 존재가 확인될 수 없는 한, 제거되기 전에 위상 데이터베이스에 이 노드가 남아 있는 일 수. **owner** 필드에 의해 지정된 노드가 국지 노드가 아닌 경우에는 이 필드가 0보다 큰 값으로 설정되어야 합니다.

topology_tg_detail.dlc_data.length
주소 길이.

topology_tg_detail.dlc_data.address
주소.

topology_tg_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

topology_tg_detail.status
이 필드는 TG의 상태를 지정합니다. 이는 다음 중 하나 또는 그 이상일 수 있습니다.

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

topology_tg_detail.tg_chars
TG 특성. 추가 정보는 35페이지의 『DEFINE_CN』을 참조하십시오.

반환되는 매개변수

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_TG

AP_INVALID_ORIGIN_NODE
AP_INVALID_LIST_OPTION

명령이 성공적으로 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

제공되는 매개변수

restore = AP_YES일 때 제공되는 매개변수

이 응용프로그램은 다음의 매개변수를 제공합니다.

opcode
AP_SFS_NN_TOPOLOGY_TG

format
VCB의 형식을 식별합니다. 위에서 나열된 VCB의 버전을 지정하려면 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보가 기록될 수 있는 버퍼에 대한 포인터. 응용프로그램은 VCB의 끝에 데이터를 추가할 수 있는데, 그런 경우에는 **buf_ptr**가 널(null)로 설정되어야 합니다.

num_entries
반환될 최대 입력항목 갯수. 입력항목의 갯수가 이 값을 초과하지 않을 것입니다. 0의 값은 제한 없음을 의미합니다.

buf_size
버퍼에 반환되는 정보의 길이.

restore
정보가 복원되어야 할지(AP_YES) 또는 저장되어야 할지(AP_NO)를 나타내는 플래그. 이 경우에는 AP_NO로 설정됩니다.

total_num_entries
반환될 수 있었던 총 입력항목 갯수. 이는 **num_entries**보다 높을 수도 있습니다.

topology_tg_detail.overlay_size
이 입력항목에 있는 바이트의 수로 반환되는 다음 입력항목(있다면)의 오프셋이 됩니다. 이는 **restore** = AP_NO일 때 반환된 **overlay_size** 값과 동일해야 합니다.

topology_detail.owner
TG의 근원지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.)

topology_tg_detail.owner_type
TG를 소유하는 노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_detail.dest
TG에 대한 전체 정식 목적지 노드 이름. 이 이름은 오른쪽이 EBCDIC 공백으로 채워지는 17 바이트의 인접 제어점(CP) 이름입니다. 이는 EBCDIC 점으로 연결되는 두 개의 유형 A EBCDIC 문자열로 구성됩니다.(각 이름은 내포된 공백 없이 최대 8 바이트의 길이를 가질 수 있습니다.) 이 필드는 APPN 노드에 대한 링크에 대해서만 관련되며 다른 경우는 무시됩니다. **list_options**가 AP_FIRST_IN_LIST로 설정되면 이 필드가 무시됩니다.

topology_tg_detail.dest_type
노드 유형. 이는 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

topology_tg_detail.tg_num
TG와 연관된 번호.

topology_tg_detail.frsn
흐름 감소 순서 번호. 이는 국지 노드에서 자원이 최종 갱신된 시간을 나타냅니다.

topology_tg_detail.days_left
그 존재가 확인될 수 없는 한, 제거되기 전에 위상 데이터베이스에 이 노드가 남아 있는 일 수. **owner** 필드에 의해 지정된 노드가 국지 노드가 아닌 경우에는 이 필드가 0보다 큰 값으로 설정되어야 합니다.

topology_tg_detail.dlc_data.length
주소 길이.

topology_tg_detail.dlc_data.address
주소.

topology_tg_detail.rsn
자원 순서 번호. 이는 이 자원을 소유하는 네트워크 노드에 의해 지정됩니다.

topology_tg_detail.status
이 필드는 TG의 상태를 지정합니다. 이는 다음 중 하나 또는 그 이상일 수 있습니다.

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_TG_HPR
AP_TG_RTP
AP_NONE

topology_tg_detail.tg_chars
TG 특성. 추가 정보는 35페이지의 『DEFINE_CN』을 참조하십시오.

반환되는 매개변수

명령이 성공적으로 실행되면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_DAYS_LEFT

관련 START_NODE 매개변수가 설정되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

시스템에 네트워크 노드 지원이 구축되어 있지 않으므로, 명령이 실행되지 않으면 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_INVALID_VERB

노드가 아직 시작되지 않았으므로, 명령이 실행되지 않는 경우에는 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램이 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSYEM_ERROR

제8장 세션 한계 명령

이 장에서는 세션 한계를 초기화하거나 변경하거나 재설정하는 데 사용되는 명령에 대해 설명합니다.

CHANGE_SESSION_LIMIT

CHANGE_SESSION_LIMIT 명령은 특정 모드(또는 세션 그룹)의 세션 한계를 변경하도록 요청합니다. 이 명령의 처리 결과로 세션이 활성화되거나 활성종료될 수 있습니다.

VCB 구조

```
typedef struct change_session_limit
{
   unsigned short  opcode;              /* verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short  primary_rc;          /* primary return code         */
   unsigned long   secondary_rc;        /* secondary return code       */
   unsigned char   lu_name[8];          /* local LU name               */
   unsigned char   lu_alias[8];         /* local LU alias              */
   unsigned char   plu_alias[8];        /* partner LU alias            */
   unsigned char   fqplu_name[17];      /* fully qualified partner     */
                                        /* LU name                     */
   unsigned char   reserv3;             /* reserved                    */
   unsigned char   mode_name[8];        /* mode name                   */
   unsigned char   reserv3a;            /* reserved                    */
   unsigned char   set_negotiable;      /* set max negotiable limit?   */
   unsigned short  plu_mode_session_limit;
                                        /* session limit               */
   unsigned short  min_conwinners_source;
                                        /* min source contention       */
                                        /* winner sessions             */
   unsigned short  min_conwinners_target;
                                        /* min target contention       */
                                        /* winner sessions             */
   unsigned short  auto_act;            /* auto activation limit       */
   unsigned char   responsible;         /* responsible indicator       */
   unsigned char   reserv4[3];          /* reserved                    */
   unsigned long   sense_data;          /* sense data                  */
} CHANGE_SESSION_LIMIT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_CHANGE_SESSION_LIMIT

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

세션 한계를 변경하도록 요청할 국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

세션 한계를 변경하도록 요청할 국지 LU의 별명. 국지로 표시가능

한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정된 경우에만 유효하며, 8바이트 모두 유효 바이트로 반드시 설정해야 합니다. **lu_name** 및 **lu_alias** 필드를 둘다 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

plu_alias

국지 LU에서 사용되는 상대방 LU의 별명. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 요구되는 상대방 LU를 지정하는 데 사용됩니다.

fqplu_name

상대방 LU의 전체 정식 LU 명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 **plu_alias** 필드를 모두 0으로 설정한 경우에만 유효합니다.

mode_name

구성시에 정의한 네트워킹 특성 집합의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

SNASVCMG 및 CPSVCMG 모드 한계는 변경할 수 없습니다.

Set_negotiable은 이 모드의 최대 조정가능 세션 한계가 수정되어 **plu_mode_session_limit**가 되는지의 여부를 지정합니다.

set_negotiable

이 모드의 최대 조정가능 세션 한계가 수정되어 **plu_mode_session_limit**가 되는지의 여부를 지정합니다.

AP_YES
AP_NO

plu_mode_session_limit

이 모드에 대해 요청되는 총 세션 한계. 실제 세션 한계(상대방 LU와 조정할 수 있는)는 이 모드에서의 국지 LU와 상대방 LU간에 지원되는 협의된 최대 세션 수입니다.

min_conwinners_source

국지 LU가 회선경합 성공 세션이 되는 이 모드에서의 최소 세션 수.

min_conwinners_target

상대방 LU가 회선경합 성공 세션이 되는 이 모드에서의 최소 세션 수.

auto_act

세션 한계가 변경된 후 자동으로 활성화되는 세션 수. 자동으로 활

성화되는 실제 세션 수는 이 값과 국지 LU에 대한 조정된 회선경합 성공 세션 최대 수의 최소값입니다. 세션이 정상적으로 활성종료되어(AP_DEACT_NORMAL을 지정함으로써) 이 한계값 미만이 되면, 새로운 세션이 활성화되어 이 한계값에 달하게 됩니다.

responsible
세션 한계가 변경된 후 소스(국지)와 목표(상대방) LU 중 어느 것이 세션을 활성종료해야 하는지를 지시합니다(AP_SOURCE 또는 AP_TARGET).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_LU_MODE_SESSION_LIMIT_ZERO

AP_EXCEEDS_MAX_ALLOWED
AP_INVALID_MODE_NAME
AP_INVALID_PLU_NAME
AP_INVALID_RESPONSIBLE
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_MODE_RESET

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

할당 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY

sense_data
할당 오류와 연관된 감지 데이터.

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

secondary_rc
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

INITIALIZE_SESSION_LIMIT

INITIALIZE_SESSION_LIMIT 명령은 모드 세션 한계를 초기화합니다.

VCB 구조

```
typedef struct initialize_session_limit
{
   unsigned short  opcode;               /* verb operation code         */
   unsigned char    reserv2;             /* reserved                    */
   unsigned char    format;              /* format                      */
   unsigned short  primary_rc;           /* primary return code         */
   unsigned long   secondary_rc;         /* secondary return code       */
   unsigned char   lu_name[8];           /* local LU name               */
   unsigned char   lu_alias[8];          /* local LU alias              */
   unsigned char   plu_alias[8];         /* partner                     */
   unsigned char   fqplu_name[17];       /* fully qualified partner     */
                                         /* LU name                     */
   unsigned char   reserv3;              /* reserved                    */
   unsigned char   mode_name[8];         /* mode name                   */
   unsigned char   reserv3a;             /* reserved                    */
   unsigned char   set_negotiable;       /* set max negotiable limit?   */
   unsigned short  plu_mode_session_limit;
                                         /* session limit               */
   unsigned short  min_conwinners_source;
                                         /* min source contention       */
                                         /* winner sessions             */
   unsigned short  min_conwinners_target;
                                         /* min target contention     */
                                         /* winner sessions           */
   unsigned short  auto_act;             /* auto activation limit     */
   unsigned char   reserv4[4];           /* reserved                    */
   unsigned long   sense_data;           /* sense data                  */
} INITIALIZE_SESSION_LIMIT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_INITIALIZE_SESSION_LIMIT

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

세션 한계를 초기화하도록 요청할 국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

세션 한계를 초기화하도록 요청할 국지 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모

두 유효 바이트로 반드시 설정해야 합니다. **lu_name** 및 **lu_alias** 필드를 둘다 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

plu_alias

국지 LU에서 사용되는 상대방 LU의 별명. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 요구되는 상대방 LU를 지정하는 데 사용됩니다.

fqplu_name

상대방 LU의 전체 정식 LU 명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 **plu_alias** 필드를 모두 0으로 설정한 경우에만 유효합니다.

mode_name

구성시에 정의한 네트워킹 특성 집합의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

이 필드에 모드명 SNASVCMG 또는 CPSVCMG가 지정되어 있고 한계값이 **plu_mode_session_limit 2, min_conwinners_source 1** 및 **min_conwinners target 1** 이외의 값을 취할 경우에는 이 명령이 거부(reject)됩니다.

set_negotiable

이 모드의 최대 조정가능 세션 한계가 수정되어 **plu_mode_session_limit**가 되는지의 여부를 지정합니다.

AP_YES
AP_NO

plu_mode_session_limit

이 모드에 대해 요청되는 총 세션 한계. 실제 세션 한계(상대방 LU와 조정할 수 있는)는 이 모드에서의 국지 LU와 상대방 LU간에 지원되는 협의된 최대 세션 수입니다. 이 필드는 1-32 767 범위 내의 한 값으로 설정해야 합니다.

min_conwinners_source

국지 LU가 회선경합 성공 세션이 되는 이 모드에서의 최소 세션 수. 이 필드는 0-32 767 범위 내의 한 값으로 설정해야 합니다.

min_conwinners_target

상대방 LU가 회선경합 성공 세션이 되는 이 모드에서의 최소 세션 수. 이 필드는 0-32 767 범위 내의 한 값으로 설정해야 합니다.

auto_act
세션 한계가 변경된 후 자동으로 활성화되는 세션 수. 자동으로 활성화되는 실제 세션 수는 이 값과 국지 LU에 대한 조정된 회선경합 성공 세션 최대 수의 최소값입니다. 세션이 정상적으로 활성종료되어(AP_DEACT_NORMAL을 지정함으로써) 이 한계값 미만이 되면, 새로운 세션이 활성화되어 이 한계값에 달하게 됩니다. 이 필드는 0-32 767 범위 내의 한 값으로 설정해야 합니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc
AP_AS_SPECIFIED

AP_AS_NEGOTIATED

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_CANT_CHANGE_TO_ZERO

AP_EXCEEDS_MAX_ALLOWED
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_PLU_NAME
AP_INVALID_MODE_NAME
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS
AP_INVALID_SCVMG_LIMITS

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_MODE_NOT_RESET

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

할당 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY

sense_data
할당 오류와 연관된 감지 데이터.

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

secondary_rc
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

RESET_SESSION_LIMIT

RESET_SESSION_LIMIT 명령은 모드 세션 한계를 재설정하도록 요청합니다.

VCB 구조

```
typedef struct reset_session_limit
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_name[8];          /* local LU name              */
   unsigned char   lu_alias[8];         /* local LU alias             */
   unsigned char   plu_alias[8];        /* partner LU alias           */
   unsigned char   fqplu_name[17];      /* fully qual partner LU name */
   unsigned char   reserv3;             /* reserved                   */
   unsigned char   mode_name[8];        /* mode name                  */
   unsigned char   mode_name_select;    /* select mode name           */
   unsigned char   set_negotiable;      /* set max negotiable limit?  */
   unsigned char   reserv4[8];          /* reserved                   */
   unsigned char   responsible;         /* responsible                */
   unsigned char   drain_source;        /* drain source               */
   unsigned char   drain_target;        /* drain target               */
   unsigned char   force;               /* force                      */
   unsigned long   sense_data;          /* sense data                 */
} RESET_SESSION_LIMIT;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_RESET_SESSION_LIMIT

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

세션 한계를 재설정하도록 요청할 국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias

세션 한계를 재설정하도록 요청할 국지 LU의 별명. 국지로 표시가 능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

plu_alias

국지 LU에서 사용되는 상대방 LU의 별명. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 이름

능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 요구되는 상대방 LU를 지정하는 데 사용됩니다.

fqplu_name
상대방 LU의 전체 정식 LU 명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 이 필드는 **plu_alias** 필드를 모두 0으로 설정한 경우에만 유효합니다.

mode_name
구성시에 정의한 네트워킹 특성 집합의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

mode_name_select
세션 한계를 단일 지정 모드에서 재설정하는지 또는 국지 LU와 상대방 LU 사이의 모든 모드에서 재설정하는지를 선택합니다.

AP_ONE
AP_ALL

set_negotiable
이 모드의 최대 조정가능 세션 한계가 수정되는지의 여부를 지정합니다.

AP_YES
AP_NO

responsible
세션 한계가 재설정된 후 소스(국지)와 목표(상대방) LU 중 어느 것이 세션을 활성종료해야 하는지를 지시합니다(AP_SOURCE 또는 AP_TARGET).

drain_source
세션 한계가 변경되거나 재설정될 때 소스 LU가 세션을 활성종료하기 전에 대기 중인 세션 요청을 처리하는지의 여부를 지정합니다(AP_NO 또는 AP_YES).

drain_target
세션 한계가 변경되거나 재설정될 때 목표 LU가 세션을 활성종료하기 전에 대기 중인 세션 요청을 처리하는지의 여부를 지정합니다(AP_NO 또는 AP_YES).

force CNOS 조정이 실패할 경우에도 세션 한계가 0으로 설정되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

secondary_rc
AP_FORCED

AP_AS_SPECIFIED
AP_AS_NEGOTIATED

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_EXCEEDS_MAX_ALLOWED

AP_INVALID_PLU_NAME
AP_INVALID_MODE_NAME
AP_INVALID_MODE_NAME_SELECT
AP_INVALID_RESPONSIBLE
AP_INVALID_DRAIN_SOURCE
AP_INVALID_DRAIN_TARGET
AP_INVALID_FORCE
AP_INVALID_SET_NEGOTIABLE
AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_MODE_RESET

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

할당 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY

sense_data
할당 오류와 연관된 감지 데이터.

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_CONV_FAILURE_NO_RETRY

AP_CNOS_PARTNER_LU_REJECT

secondary_rc
AP_CNOS_COMMAND_RACE_REJECT

AP_CNOS_MODE_NAME_REJECT

제9장 노드 연산자 기능 API 표시

노드 연산자 기능 API는 표시 명령을 생성하여 노드 운영요원에게 노드의 변경사항에 대해 알립니다. 표시 명령은 다음과 같은 일반 구조를 사용합니다.

```
typedef struct indication_hdr
          {
            unsigned short  opcode;         /* verb operation code      */
            unsigned char   reserv2;        /* reserved                 */
            unsigned char   format;         /* format                   */
            unsigned short  primary_rc;     /* primary return code      */
            unsigned long   secondary_rc;   /* secondary return code    */
            unsigned char   data_lost;      /* previous indication lost */
          } INDICATION_HDR;
```

DLC_INDICATION

이 표시는 DLC가 활성 상태에서 비활성 상태로 되거나 비활성 상태에서 활성 상태로 될 때 생성됩니다.

VCB 구조

```
typedef struct dlc_indication
{
   unsigned short  opcode;               /* verb operation code           */
   unsigned char   attributes;           /* verb attributes               */
   unsigned char    format;              /* format                        */
   unsigned short  primary_rc;           /* primary return code           */
   unsigned long   secondary_rc;         /* secondary return code         */
   unsigned char   data_lost;            /* previous indication lost      */
   unsigned char   deactivated;          /* has session been deactivated? */
   unsigned char   dlc_name[8];          /* link station name             */
   unsigned char   description[RD_LEN];  /* resource description          */
   unsigned char   reserva[20];          /* reserved                      */
} DLC_INDICATION;
```

제공되는 매개변수

opcode

AP_DLC_INDICATION

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습

dlc_name

DLC의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

description

자원 설명(DEFINE_DLC에 지정한 것과 동일). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

DLUR_LU_INDICATION

이 표시는 DLUR LU가 활성화되거나 활성종료될 때마다 생성됩니다. 이 표시는 등록된 응용프로그램이 현재 활동중인 DLUR LU 리스트를 유지보수할 수 있도록 합니다.

VCB 구조

```
typedef struct dlur_lu_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   reason;              /* reason for this indication   */
   unsigned char   lu_name[8];          /* LU name                      */
   unsigned char   pu_name[8];          /* PU name                      */
   unsigned char   nau_address;         /* NAU address                  */
   unsigned char   reserv5[7];          /* reserved                     */
} DLUR_LU_INDICATION;
```

제공되는 매개변수

opcode

AP_DLUR_LU_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

reason

DLUR LU가 DLUS에 의해 활성화되었으면 AP_ADDED로 설정됩니다. DLUR LU가 DLUS에 의해 명시적으로 활성종료되었거나 링크 장애 혹은 PU의 활성종료로 인해 암시적으로 활성종료되었으면 AP_REMOVED로 설정됩니다.

lu_name

LU의 이름. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

pu_name

이 LU가 사용하는 PU의 이름. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

nau_address

LU의 네트워크 지정 장치(NAU) 주소로서, 1–255 범위 내의 값이어야 합니다.

DLUR_PU_INDICATION

이 표시는 DLUR PU가 활성화되거나 활성종료될 때마다 생성됩니다. 이 표시는 등록된 응용프로그램이 현재 활동중인 DLUR PU 리스트를 유지보수할 수 있도록 합니다.

VCB 구조

```
typedef struct dlur_pu_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   reason;              /* reason for this indication   */
   unsigned char   pu_name[8];          /* PU name                      */
   unsigned char   pu_id[4];            /* PU identifier                */
   unsigned char   pu_location;         /* downstream or local PU       */
   unsigned char   pu_status;           /* status of the PU             */
   unsigned char   dlus_name[17];       /* current DLUS name            */
   unsigned char   dlus_session_status; /* status of the DLUS pipe      */
   unsigned char   reserv5[2];          /* reserved                     */
} DLUR_PU_INDICATION;
```

제공되는 매개변수

opcode

AP_DLUR_PU_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

reason

표시 원인. 다음 중 하나입니다.

AP_ACTIVATION_STARTED

PU가 활성화중입니다.

AP_ACTIVATING

PU가 활동중인 상태로 되었습니다.

AP_DEACTIVATING
PU가 활동중이 아닌 상태로 되었습니다.

AP_FAILED
PU가 실패했습니다.

AP_ACTIVATION_FAILED
PU가 활성화에 실패했습니다.

pu_name
PU의 이름. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

pu_id DEFINE_INTERNAL_PU 명령에 정의했거나 다운스트림 PU로부터의 XID에서 얻어지는 PU 식별자(ID). 4바이트 16진 문자열입니다. 비트 0–11은 블럭 번호로 설정되며 비트 12–31은 PU를 고유하게 식별하는 ID 번호로 설정됩니다.

plu_location
PU 위치. 다음 중 하나가 될 수 있습니다.

AP_INTERNAL
AP_DOWNSTREAM

dlur_pu_detail.pu_status
PU 상태(DLUR에서 볼 때). 다음 중 하나로 설정될 수 있습니다.

AP_RESET_NO_RETRY
PU가 재설정 상태이며 재시도되지 않습니다.

AP_RESET_RETRY
PU가 재설정 상태이며 재시도됩니다.

AP_PEND_ACTPU
PU가 호스트로부터의 ACTPU를 기다립니다.

AP_PEND_ACTPU_RSP
DLUR이 ACTPU를 PU로 전달한 후 PU가 응답할 때까지 기다립니다.

AP_ACTIVE
PU가 활동중입니다.

AP_PEND_DACTPU_RSP
DLUR이 DACTPU를 PU로 전달한 후 PU가 응답할 때까지 기다립니다.

AP_PEND_INOP
DLUR이 PU를 활성종료하기 전에 필요한 모든 이벤트가 완료되기를 기다립니다.

pu_id PU가 현재 사용중인(또는 사용하려고 시도 중인) DLUS 노드의 이름. EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다. (각

이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) PU 활성화에 실패한 경우, 이 필드는 모두 0으로 설정됩니다.

dlur_pu_detail.dlus_session_status
PU가 현재 사용중인 DLUS 파이프의 상태. 다음 중 하나가 될 수 있습니다.

AP_PENDING_ACTIVE
AP_ACTIVE
AP_PENDING_INACTIVE
AP_INACTIVE

DLUS_INDICATION

이 표시는 DLUS 노드로의 파이프가 비활성 상태에서 활성 상태로(또는 그 반대로) 될 때 생성됩니다. 파이프 통계는 파이프가 비활성 상태로 될 때 제공됩니다.

VCB 구조

```
typedef struct dlus_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   deactivated;         /* has session been deactivated? */
   unsigned char   dlus_name[17];       /* DLUS name                     */
   unsigned char   reserv1;             /* reserved                      */
   PIPE_STATS      pipe_stats;          /* pipe statistics               */
   unsigned char   reserva[20];         /* reserved                      */
} DLUS_INDICATION;

typedef struct pipe_stats
{
   unsigned long   reqactpu_sent;       /* REQACTPUs sent to DLUS        */
   unsigned long   reqactpu_rsp_received;
                                        /* RSP(REQACTPU)s received       */
                                        /* from DLUS                     */
   unsigned long   actpu_received;      /* ACTPUs received from DLUS     */
   unsigned long   actpu_rsp_sent;      /* RSP(ACTPU)s sent to DLUS      */
   unsigned long   reqdactpu_sent;      /* REQDACTPUs sent to DLUS       */
   unsigned long   reqdactpu_rsp_received;
                                        /* RSP(REQDACTPU)s received      */
                                        /* from DLUS                     */
   unsigned long   dactpu_received;     /* DACTPUs received from DLUS    */
   unsigned long   dactpu_rsp_sent;     /* RSP(DACTPU)s sent to DLUS     */
   unsigned long   actlu_received;      /* ACTLUs received from DLUS     */
   unsigned long   actlu_rsp_sent;      /* RSP(ACTLU)s sent to DLUS      */
   unsigned long   dactlu_received;     /* DACTLUs received from DLUS    */
   unsigned long   dactlu_rsp_sent;     /* RSP(DACTLU)s sent to DLUS     */
   unsigned long   sscp_pu_mus_rcvd;    /* MUs for SSCP-PU sess received */
   unsigned long   sscp_pu_mus_sent;    /* MUs for SSCP-PU sessions sent */
   unsigned long   sscp_lu_mus_rcvd;    /* MUs for SSCP-LU sess received */
   unsigned long   sscp_lu_mus_sent;    /* MUs for SSCP-LU sessions sent */
} PIPE_STATS;
```

제공되는 매개변수

opcode

AP_DLUS_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc
0과 동일합니다.

data_lost
데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivated
파이프가 비활성 상태로 될 때 AP_YES로 설정됩니다. 파이프가 활성 상태로 될 때는 AP_NO로 설정됩니다.

dlus_name
DLUS 이름. EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성된 17바이트 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

pipe_stats.reqactpu_sent
파이프를 통해 DLUS로 송신되는 REQACTPU 수.

pipe_stats.reqactpu_rsp_received
파이프를 통해 DLUS에서 수신되는 RSP(REQACTPU) 수.

pipe_stats.actpu_received
파이프를 통해 DLUS에서 수신되는 ACTPU 수.

pipe_stats.actpu_rsp_sent
파이프를 통해 DLUS로 송신되는 RSP(ACTPU) 수.

pipe_stats.reqdactpu_sent
파이프를 통해 DLUS로 송신되는 REQDACTPU 수.

pipe_stats.reqdactpu_rsp_received
파이프를 통해 DLUS에서 수신되는 RSP(REQDACTPU) 수.

pipe_stats.dactpu_received
파이프를 통해 DLUS에서 수신되는 DACTPU 수.

pipe_stats.dactpu_rsp_sent
파이프를 통해 DLUS로 송신되는 RSP(DACTPU) 수.

pipe_stats.actlu_received
파이프를 통해 DLUS에서 수신되는 ACTLU 수.

pipe_stats.actlu_rsp_sent
파이프를 통해 DLUS로 송신되는 RSP(ACTLU) 수.

pipe_stats.dactlu_received
파이프를 통해 DLUS에서 수신되는 DACTLU 수.

pipe_stats.dactlu_rsp_sent
파이프를 통해 DLUS로 송신되는 RSP(DACTLU) 수.

pipe_stats.sscp_pu_mus_rcvd
파이프를 통해 DLUS에서 수신되는 SSCP-PU MU 수.

pipe_stats.sscp_pu_mus_sent
파이프를 통해 DLUS로 송신되는 SSCP-PU MU 수.

pipe_stats.sscp_lu_mus_rcvd
파이프를 통해 DLUS에서 수신되는 SSCP-LU MU 수.

pipe_stats.sscp_lu_mus_sent
파이프를 통해 DLUS로 송신되는 SSCP-LU MU 수.

DOWNSTREAM_LU_INDICATION

이 명령은 통신 서버에만 적용됩니다.

이 표시는 다운스트림 LU와 호스트간의 LU-SSCP 세션이 비활동 상태에서 활성 상태로(또는 그 반대로) 되거나 PLU-SLU 세션이 비활동 상태에서 활성 상태로(또는 그 반대로) 될 때 생성됩니다. LU-SSCP 통계는 LU-SSCP 세션이 활성종료될 때 제공되고, PLU-SLU 통계는 PLU-SLU 세션이 활성종료될 때 제공됩니다.

VCB 구조

```
typedef struct downstream_lu_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char   attributes;          /* attributes                   */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   dspu_name[8];        /* PU Name                      */
   unsigned char   ls_name[8];          /* Link station name            */
   unsigned char   dslu_name[8];        /* LU Name                      */
   unsigned char   description[RD_LEN]; /* resource description         */
   unsigned char   nau_address;         /* NAU address                   */
   unsigned char   lu_sscp_sess_active;
                                        /* Is SSCP session active?      */
   unsigned char   plu_sess_active;     /* Is PLU-SLU session active?   */
   unsigned char   dspu_services;       /* DSPU services                */
   unsigned char   reserv1;             /* reserved                     */
   SESSION_STATS   lu_sscp_stats;       /* LU-SSCP session statistics   */
   SESSION_STATS   ds_plu_stats;        /* Downstream PLU-SLU sess stats */
   SESSION_STATS   us_plu_stats;        /* Upstream PLU-SLU sess stats  */
} DOWNSTREAM_LU_INDICATION;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;          /* session receive RU size      */
   unsigned short  send_ru_size;         /* session send RU size         */
   unsigned short  max_send_btu_size;    /* max send BTU  size           */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size             */
   unsigned short  max_send_pac_win;     /* max send pacing window size  */
   unsigned short  cur_send_pac_win;     /* curr send pacing window size */
   unsigned short  max_rcv_pac_win;      /* max rcv pacing window size   */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing win size */
   unsigned long   send_data_frames;     /* number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                         /* num FMD data frames sent     */
   unsigned long   send_data_bytes;      /* number of data bytes sent    */
   unsigned long   rcv_data_frames;      /* num of data frames received  */
   unsigned long   rcv_fmd_data_frames;
                                         /* num FMD data frames received */
   unsigned long   rcv_data_bytes;       /* num data bytes received      */
   unsigned char   sidh;                 /* session ID high byte         */
   unsigned char   sidl;                 /* session ID low byte          */
```

```
   unsigned char   odai;                /* ODAI bit set                */
   unsigned char   ls_name[8];          /* Link station name           */
   unsigned char   pacing_type;         /* type of pacing in use       */
} SESSION_STATS;
```

제공되는 매개변수

opcode

AP_DOWNSTREAM_LU_INDICATION

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

dspu_name

다운스트림 LU와 연관된 다운스트림 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

ls_name

링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다.

dslu_name

다운스트림 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

description

자원 설명(DEFINE_DOWNSTREAM_LU에 지정한 것과 동일).

DOWNSTREAM_LU__INDICATION

nau_address
LU의 네트워크 지정 장치(NAU) 주소로서, 1–255 범위 내의 값이어야 합니다.

lu_sscp_sess_active
다운스트림 LU에 대한 LU-SSCP 세션이 활동중인지의 여부를 지시합니다. AP_YES 또는 AP_NO로 설정됩니다.

plu_sess_active
다운스트림 LU에 대한 PLU-SLU 세션이 활동중인지의 여부를 지시합니다. AP_YES 또는 AP_NO로 설정됩니다.

dspu_services
국지 노드가 링크를 통해 다운스트림 LU에 제공하는 서비스를 지정합니다. 다음 중 하나로 설정됩니다.

AP_PU_CONCENTRATION
국지 노드가 다운스트림 PU에 PU 집중을 제공합니다.

AP_DLUR
국지 노드가 다운스트림 PU에 DLUR 지원을 제공합니다.

lu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

lu_sscp_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

lu_sscp_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, BIND 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

lu_sscp_stats.pacing_type
업스트림 LU-SSCP 세션에서 사용중인 수신 페이싱 유형. AP_NONE 값을 취합니다.

ds_plu_stats.rcv_ru_size
최대 수신 RU 크기.

ds_plu_stats.send_ru_size
최대 송신 RU 크기.

ds_plu_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

ds_plu_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

ds_plu_stats.max_send_pac_win
이 세션에서의 송신 페이싱 창 최대 크기.

ds_plu_stats.cur_send_pac_win
이 세션에서의 송신 페이싱 창 현재 크기.

ds_plu_stats.max_rcv_pac_win
이 세션에서의 수신 페이싱 창 최대 크기.

ds_plu_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

ds_plu_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

ds_plu_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

ds_plu_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

ds_plu_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

ds_plu_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

ds_plu_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

ds_plu_stats.sidh
세션 ID 상위(high) 바이트.

ds_plu_stats.sidl
세션 ID 하위(low) 바이트.

ds_plu_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, BIND 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1

us_plu_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

us_plu_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

us_plu_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

us_plu_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

us_plu_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

us_plu_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

us_plu_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

us_plu_stats.sidh
세션 ID 상위(high) 바이트. 이 필드는 **dspu_services**가 AP_PU_CONCENTRATION으로 설정된 경우에 수신됩니다.

us_plu_stats.sidl
세션 ID 하위(low) 바이트. 이 필드는 **dspu_services**가 AP_PU_CONCENTRATION으로 설정된 경우에 수신됩니다.

us_plu_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, BIND 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다. 이 필드는 **dspu_services**가 AP_PU_CONCENTRATION으로 설정된 경우에 수신됩니다.

us_plu_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 **dspu_services**가 AP_PU_CONCENTRATION으로 설정된 경우에 수신됩니다.

us_plu_stats.pacing_type
업스트림 PLU-SLU 세션에서 사용중인 수신 페이싱 유형. AP_NONE 또는 AP_PACING_FIXED 값을 취할 수 있습니다.

DOWNSTREAM_PU_INDICATION

이 명령은 통신 서버에만 적용됩니다.

이 표시는 다운스트림 PU와 호스트간의 PU-SSCP 세션이 비활성 상태에서 활성 상태로(또는 그 반대로) 될 때 생성됩니다. PU-SSCP 통계는 PU-SSCP 세션이 활성종료될 때 제공됩니다.

VCB 구조

```
typedef struct downstream_pu_indication
{
   unsigned short  opcode;               /* verb operation code           */
   unsigned char   attributes;           /* attributes                    */
   unsigned char    format;              /* format                        */
   unsigned short  primary_rc;           /* primary return code           */
   unsigned long   secondary_rc;         /* secondary return code         */
   unsigned char   data_lost;            /* previous indication lost      */
   unsigned char   dspu_name[8];         /* PU Name                       */
   unsigned char   description[RD_LEN];  /* resource description          */
   unsigned char   ls_name[8];           /* Link Station name             */
   unsigned char   pu_sscp_sess_active;
                                         /* Is PU-SSCP session active?    */
   unsigned char   dspu_services;        /* DSPU services                 */
   unsigned char   reserv1[2];           /* reserved                      */
   SESSION_STATS   pu_sscp_stats;        /* PU-SSCP session statistics    */
} DOWNSTREAM_PU_INDICATION;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;          /* session receive RU size       */
   unsigned short  send_ru_size;         /* session send RU size          */
   unsigned short  max_send_btu_size;    /* max send BTU  size            */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size              */
   unsigned short  max_send_pac_win;     /* max send pacing window size   */
   unsigned short  cur_send_pac_win;     /* curr send pacing window size  */
   unsigned short  max_rcv_pac_win;      /* max rcv pacing window size    */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing win size  */
   unsigned long   send_data_frames;     /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                         /* num FMD data frames sent      */
   unsigned long   send_data_bytes;      /* number of data bytes sent     */
   unsigned long   rcv_data_frames;      /* num of data frames received   */
   unsigned long   rcv_fmd_data_frames;
                                         /* num FMD data frames received  */
   unsigned long   rcv_data_bytes;       /* num data bytes received       */
   unsigned char   sidh;                 /* session ID high byte          */
   unsigned char   sidl;                 /* session ID low byte           */
   unsigned char   odai;                 /* ODAI bit set                  */
   unsigned char   ls_name[8];           /* Link station name             */
   unsigned char   pacing;               /* pacing_type                   */
} SESSION_STATS;
```

제공되는 매개변수

opcode

AP_DOWNSTREAM_PU_INDICATION

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc
AP_OK

secondary_rc
0과 동일합니다.

data_lost
데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

dspu_name
다운스트림 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

description
자원 설명(DEFINE_LS에 지정한 것과 동일).

ls_name
링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

pu_sscp_sess_active
다운스트림 PU에 대한 PU-SSCP 세션이 활동중인지의 여부를 지시합니다. AP_YES 또는 AP_NO로 설정됩니다.

dspu_services
국지 노드가 링크를 통해 다운스트림 PU에 제공하는 서비스를 지정합니다. 다음 중 하나로 설정됩니다.

AP_PU_CONCENTRATION
국지 노드가 다운스트림 PU에 PU 집중을 제공합니다.

AP_DLUR
국지 노드가 다운스트림 PU에 DLUR 지원을 제공합니다.

pu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

pu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

pu_sscp_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

pu_sscp_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

pu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

pu_sscp_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

pu_sscp_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

pu_sscp_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

pu_sscp_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

pu_sscp_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

pu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

pu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

pu_sscp_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, BIND 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

pu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

pu_sscp_stats.pacing_type
업스트림 PU-SSCP 세션에서 사용중인 수신 페이싱 유형. AP_NONE 값을 취합니다.

FOCAL_POINT_INDICATION

이 표시는 중재점이 얻어지거나 변경되거나 혹은 취소될 때 생성됩니다.

VCB 구조

```
typedef struct focal_point_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   ms_category[8];      /* Focal point category         */
   unsigned char   fp_fqcp_name[17];    /* Fully qualified focal        */
                                        /* point CP name                */
   unsigned char   ms_appl_name[8];     /* Focal point application name */
   unsigned char   fp_type;             /* type of current focal point  */
   unsigned char   fp_status;           /* status of focal point        */
   unsigned char   fp_routing;          /* type of MDS routing to       */
                                        /* reach FP                     */
   unsigned char   reserva[20];         /* reserved                     */
} FOCAL_POINT_INDICATION;
```

제공되는 매개변수

opcode

AP_FOCAL_POINT_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

ms_category

중재점이 얻어지거나 변경되거나 취소된 중재점의 범주. *SNA 관리 서비스*에 설명된 바와 같이 관리 서비스 범주에 대해 구조적으로 정의된 4바이트 값(오른쪽이 EBCDIC 공백으로 채워짐) 중 하나이거나, 8바이트 1134 유형 EBCDIC 설치 정의 이름일 수 있습니다.

fp_fqcp_name

현재 중재점의 전체 정식 제어점(CP)명. 길이가 17바이트이며 오른

쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) 중재점이 취소되고 아직 대체되지 않은 경우에는(따라서 현재 활동중인 중재점이 없는 경우) 이 이름이 모두 0으로 설정됩니다.

ms_appl_name

현재 중재점 응용프로그램의 이름. *SNA 관리 서비스*에 설명된 것과 같이 관리 서비스 어플리케이션에 대해 구조적으로 정의된 4바이트 값(오른쪽이 EBCDIC 공백으로 채워짐) 중 하나이거나, 8바이트 1134 유형 EBCDIC 설치 정의 이름일 수 있습니다. 중재점이 취소되고 아직 대체되지 않은 경우에는(따라서 현재 활동중인 중재점이 없는 경우) 이 이름이 모두 0으로 설정됩니다.

fp_type

중재점 유형. 자세한 내용은 *SNA 관리 서비스*를 참조하십시오.

AP_EXPLICIT_PRIMARY_FP
AP_BACKUP_FP
AP_DEFAULT_PRIMARY_FP
AP_DOMAIN_FP
AP_HOST_FP
AP_NO_FP

fp_status

중재점의 상태.

AP_NOT_ACTIVE
중재점이 활성 상태에서 비활성 상태가 되었습니다.

AP_ACTIVE
중재점이 비활성 상태나 활성화 미정 상태에서 활동 상태로 되었습니다.

fp_routing

응용프로그램이 MDS 트랜스포트를 사용하여 중재점으로 데이터를 전송할 때 지정하는 경로지정 유형(중재점 상태가 AP_ACTIVE일 경우에만 유효합니다).

AP_DEFAULT
MDS_MU를 중재점으로 전달하는 데 생략시 경로지정이 사용됩니다.

AP_DIRECT
세션에서 MDS_MU가 중재점으로 직접 경로지정됩니다.

ISR_INDICATION

이 명령은 통신 서버에만 적용됩니다.

이 표시는 ISR 세션이 활성화되거나 활성종료될 때 생성됩니다. 이 세션이 활성종료될 때, 최종 세션 통계가 반환됩니다. 이 세션이 활성화되면, **pri_sess_stats** 및 **sec_sess_stats** 필드가 예약됩니다.

VCB 구조

```
typedef struct isr_indication
{
   unsigned short  opcode;            /* verb operation code          */
   unsigned char    reserv2;          /* reserved                     */
   unsigned char    format;           /* format                       */
   unsigned short  primary_rc;        /* primary return code          */
   unsigned long   secondary_rc;      /* secondary return code        */
   unsigned char   data_lost;         /* previous indication lost     */
   unsigned char   deactivated;       /* has ISR session been         */
                                      /* deactivated?                 */
   FQPCID          fqpcid;            /* fully qualified procedure    */
                                      /* correlator ID                */
   unsigned char   fqplu_name[17];    /* fully qualified primary      */
                                      /* LU name                      */
   unsigned char   fqslu_name[17];    /* fully qualified secondary    */
                                      /* LU name                      */
   unsigned char   mode_name[8];      /* mode name                    */
   unsigned char   cos_name[8];       /* COS name                     */
   unsigned char   transmission_priority;
                                      /* transmission priority        */
   unsigned long   sense_data;        /* sense data                   */
   unsigned char   reserv2a[2];       /* reserved                     */
   SESSION_STATS   pri_sess_stats;    /* primary hop session stats    */
   SESSION_STATS   sec_sess_stats;    /* secondary hop session        */
                                      /* statistics                   */
   unsigned char   reserva[20];       /* reserved                     */
} ISR_INDICATION;

typedef struct fqpcid
{
   unsigned char   pcid[8];           /* pro correlator identifier    */
   unsigned char   fqcp_name[17];     /* orig's network qualified     */
                                      /* CP name                      */
   unsigned char   reserve3[3];       /* reserved                     */
} FQPCID;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;       /* session receive RU size      */
   unsigned short  send_ru_size;      /* session send RU size         */
   unsigned short  max_send_btu_size; /* Maximum send BTU size        */
   unsigned short  max_rcv_btu_size;  /* Maximum rcv BTU size         */
   unsigned short  max_send_pac_win;  /* Max send pacing window size  */
   unsigned short  cur_send_pac_win;  /* Curr send pacing window size */
   unsigned short  max_rcv_pac_win;   /* Max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;   /* Curr rec pacing window size  */
   unsigned long   send_data_frames;  /* Number of data frames sent   */
   unsigned long   send_fmd_data_frames;
```

```
                                        /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;     /* Number of data bytes sent    */
   unsigned long   rcv_data_frames;     /* Num data frames received     */
   unsigned long   rcv_fmd_data_frames;
                                        /* num of FMD data frames recvd */
   unsigned long   rcv_data_bytes;      /* Num data bytes received      */
   unsigned char   sidh;                /* Session ID high byte         */
   unsigned char   sidl;                /* Session ID low byte          */
   unsigned char   odai;                /* ODAI bit set                 */
   unsigned char   ls_name[8];          /* Link station name            */
   unsigned char   pacing_type;         /* type of pacing in use        */
} SESSION_STATS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_ISR_INDICATION

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

primary_rc

AP_OK

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivate

ISR 세션이 활성종료될 때 AP_YES로 설정됩니다. 이 세션이 활성화될 때 AP_NO로 설정됩니다.

fqpcid.pcid

절차 상관자 ID. 8바이트 16진 문자열입니다.

fqpcid.pcid_name

전체 정식 제어점 이름. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다).

fqplu_name

전체 정식 1차 LU 명(BIND 요청에 지정한 것과 동일). 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) **deactivated**가 AP_YES로 설정된 경우, 이 이름은 모두 0으로 설정됩니다.

fqslu_name
전체 정식 2차 LU 명(BIND 요청에 지정한 것과 동일). 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.) **deactivated**가 AP_YES로 설정된 경우, 이 이름은 모두 0으로 설정됩니다.

cos_name
서비스 클래스(COS) 명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다. **deactivated**가 AP_YES로 설정된 경우, 이 이름은 모두 0으로 설정됩니다.

transmission_priority
세션과 연관된 전송 우선순위. **deactivated**가 AP_YES인 경우, 이 필드는 예약됩니다.

sense_data
UNBIND 요청에서 송신 또는 수신되는 감지 데이터. **deactivated**가 AP_YES인 경우, 이 필드는 예약됩니다.

pri_sess_stats.rcv_ru_size
최대 수신 RU 크기.

pri_sess_stats.send_ru_size
최대 송신 RU 크기.

pri_sess_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

pri_sess_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

pri_sess_stats.max_send_pac_win
이 세션에서의 송신 페이싱 창 최대 크기.

pri_sess_stats.cur_send_pac_win
이 세션에서의 송신 페이싱 창 현재 크기.

pri_sess_stats.max_rcv_pac_win
이 세션에서의 수신 페이싱 창 최대 크기.

pri_sess_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

pri_sess_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

pri_sess_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

pri_sess_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

pri_sess_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

pri_sess_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

pri_sess_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

pri_sess_stats.sidh
세션 ID 상위(high) 바이트.

pri_sess_stats.sidl
세션 ID 하위(low) 바이트.

pri_sess_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정합니다. BIND 전송자가 2차 링크 스테이션을 포함하는 노드인 경우에는 이 필드를 1로 설정합니다.

pri_sess_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 세션 통계와 세션 트래픽이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

pri_sess_stats.pacing_type
1차 세션에서 사용중인 수신 페이싱 유형. AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE 값을 취할 수 있습니다.

sec_sess_stats.rcv_ru_size
최대 수신 RU 크기.

sec_sess_stats.send_ru_size
최대 송신 RU 크기.

sec_sess_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

sec_sess_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

sec_sess_stats.max_send_pac_win
이 세션에서의 송신 페이싱 창 최대 크기.

sec_sess_stats.cur_send_pac_win
이 세션에서의 송신 페이싱 창 현재 크기.

sec_sess_stats.max_rcv_pac_win
이 세션에서의 수신 페이싱 창 최대 크기.

sec_sess_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

sec_sess_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

sec_sess_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

sec_sess_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

sec_sess_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

sec_sess_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

sec_sess_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

sec_sess_stats.sidh
세션 ID 상위(high) 바이트.

sec_sess_stats.sidl
세션 ID 하위(low) 바이트.

sec_sess_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정합니다. BIND 전송자가 2차 링크 스테이션을 포함하는 노드인 경우에는 이 필드를 1로 설정합니다.

sec_sess_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 세션 통계와 세션 트래픽이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

sec_sess_stats.pacing_type
2차 세션에서 사용중인 수신 페이싱 유형. AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE 값을 취할 수 있습니다.

LOCAL_LU_INDICATION

이 표시는 국지 LU가 정의되거나 삭제될 때마다 생성됩니다. 이 표시를 통해 등록된 응용프로그램은 현재 정의된 모든 국지 LU의 리스트를 유지보수할 수 있습니다.

VCB 구조

```
typedef struct local_lu_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   reason;              /* reason for this indication   */
   unsigned char   lu_name[8];          /* LU name                      */
   unsigned char   description[RD_LEN];
                                        /* resource description         */
   unsigned char   lu_alias[8];         /* LU alias                     */
   unsigned char   nau_address;         /* NAU address                  */
   unsigned char   reserv4;             /* reserved                     */
   unsigned char   pu_name[8];          /* PU name                      */
   unsigned char   lu_sscp_active;      /* Is LU-SSCP session active    */
   unsigned char   reserv5;             /* reserved                     */
   SESSION_STATS   lu_sscp_stats;       /* LU-SSCP session statistics   */
   unsigned char   sscp_id[6];          /* SSCP ID                      */
} LOCAL_LU_INDICATION;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;          /* session receive RU size      */
   unsigned short  send_ru_size;         /* session send RU size         */
   unsigned short  max_send_btu_size;    /* max send BTU  size           */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size             */
   unsigned short  max_send_pac_win;     /* max send pacing window size  */
   unsigned short  cur_send_pac_win;     /* current send pacing win size */
   unsigned short  max_rcv_pac_win;      /* max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing win size */
   unsigned long   send_data_frames;     /* number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                         /* num of FMD data frames sent  */
   unsigned long   send_data_bytes;      /* number of data bytes sent    */
   unsigned long   rcv_data_frames;      /* num of data frames received  */
   unsigned long   rcv_fmd_data_frames;
                                         /* num FMD data frames received */
   unsigned long   rcv_data_bytes;       /* number of data bytes received*/
   unsigned char   sidh;                 /* session ID high byte         */
   unsigned char   sidl;                 /* session ID low byte          */
   unsigned char   odai;                 /* ODAI bit set                 */
   unsigned char   ls_name[8];           /* Link station name            */
   unsigned char   pacing_type;          /* type of pacing in use        */
} SESSION_STATS;
```

주: LU-SSCP 통계는 **nau_address**가 nonzero이고 LU-SSCP 세션이 활성 상태에서 비활성 상태로 될 경우에만 유효합니다. 그 이외의 경우에는 필드들이 예약됩니다.

제공되는 매개변수

opcode

AP_LOCAL_LU_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 경우에 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용 프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

reason

표시가 발행된 이유.

AP_ADDED

LU가 정의되었습니다.

AP_REMOVED

LU가 DELETE_LOCAL_LU에 의해 명시적으로 삭제되었거나 DELETE_LS, DELETE_PORT 또는 DELETE_DLC에 의해 암시적으로 삭제되었습니다.

AP_SSCP_ACTIVE

노드가 ACTLU를 처리한 후 LU-SSCP 세션이 활성 상태로 되었습니다.

AP_SSCP_INACTIVE

일반 DACTLU 또는 링크 장애 후 LU-SSCP 세션이 비활성 상태로 되었습니다.

lu_name

LU의 이름. 상태가 변경된 국지 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

description

자원 설명(DEFINE_LOCAL_LU에 지정한 것과 동일).

lu_alias

국지로 정의된 LU 별명. 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

LOCAL_LU__INDICATION

nau_address
LU의 네트워크 지정 장치(NAU) 주소로, 0–255 범위 내의 값이어야 합니다. non-zero 값은 LU가 종속 LU임을 의미합니다. 0은 LU가 독립 LU임을 의미합니다.

pu_name
이 LU가 사용하는 PU의 이름. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열입니다. 이 필드는 LU가 종속 LU일 경우(즉, **nau_address**가 nonzero일 경우)에만 유효하며, 독립 LU인 경우에는 모두 2진 0으로 설정됩니다.

lu_sscp_sess_active
LU-SSCP 세션이 활동중인지의 여부를 지정합니다(AP_YES 또는 AP_NO). **nau_address**가 0이면, 이 필드는 예약됩니다.

lu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

lu_sscp_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

lu_sscp_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, ACTLU 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, ACTLU 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 이 세션과 세션이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

lu_sscp_stats.pacing_type
LU-SSCP 세션에서 사용중인 수신 페이싱 유형. AP_NONE 값을 취합니다.

sscp_id
이 LU에서 사용하는 PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하는 6바이트 필드입니다.

이 필드는 종속 LU에서만 사용하며, 독립 LU이거나 **lu_sscp_sess_active**가 AP_YES로 설정되지 않은 경우에는 모두 2진 0으로 설정됩니다.

LOCAL_TOPOLOGY_INDICATION

이 표시는 노드의 국지 위상 데이터베이스에 있는 TG 입력항목이 활동 상태에서 비활성 상태로 또는 비활성 상태에서 활성 상태로 변경될 때 생성됩니다.

VCB 구조

```
typedef struct local_topology_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   status;              /* TG status                    */
   unsigned char   dest[17];            /* name of TG destination node  */
   unsigned char   dest_type;      /* TG destination node type      */
   unsigned char   tg_num;         /* TG number                     */
   unsigned char   cp_cp_session_active;
                                   /* CP-CP session is active       */
   unsigned char   branch_link_type;
                                   /* branch link type              */
   unsigned char   branch_tg;      /* TG is a branch TG             */
   unsigned char   reserva[17];    /* reserved                      */
} LOCAL_TOPOLOGY_INDICATION;
```

제공되는 매개변수

opcode

AP_LOCAL_TOPOLOGY_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

status TG의 상태를 지정합니다. 다음 값 중 하나 이상이 될 수 있습니다(둘 이상일 경우에는 OR로 연결됨).

AP_TG_OPERATIVE
AP_TG_CP_CP_SESSIONS
AP_TG_QUIESCING
AP_NONE

dest TG의 전체 정식 목적지 노드명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

dest_type
노드 유형. 다음 중 하나입니다.

AP_END_NODE
AP_NETWORK_NODE
AP_VRN

tg_num
TG와 연관된 번호.

cp_cp_session_active
국지 노드의 회선경합 성공 CP-CP 세션이 활동중인지의 여부를 지정합니다(AP_NO 또는 AP_YES).

branch_link_type
BrNN 전용. 이 TG의 분기 링크 유형. 다음 중 하나로 설정됩니다.

AP_UPLINK
링크가 업링크입니다.

AP_DOWNLINK
링크가 EN에 대해 다운링크입니다.

AP_DOWNLINK_TO_BRNN
TG는 상응하는 EN에 표시되는 BrNN 다운링크입니다.

AP_OTHERLINK
링크가 기타 링크입니다.

기타 노드 유형: 이 필드가 유효하지 않으며 항상 AP_BRNN_NOT_SUPPORTED로 설정됩니다.

branch_tg
NN 전용. TG가 분기 TG인지의 여부를 지정합니다.

AP_NO
TG가 분기 TG가 아닙니다.

AP_YES
TG가 분기 TG입니다.

기타 노드 유형: 이 필드가 유효하지 않으며 항상 AP_NO로 설정됩니다.

LS_INDICATION

이 표시는 링크를 사용하는 활동중인 세션의 수가 변경되거나 링크 스테이션의 외부 상태가 변경될 때 생성됩니다. 링크 스테이션 통계는 링크 스테이션이 비활성 상태가 될 때 제공됩니다.

VCB 구조

```
typedef struct ls_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char   attributes;          /* verb attributes               */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   deactivated;         /* has session been deactivated? */
   unsigned char   ls_name[8];          /* link station name             */
   unsigned char   description[RD_LEN]; /* resource description          */
   unsigned char   adj_cp_name[17];     /* network qualified Adj CP name */
   unsigned char   adj_node_type;       /* adjacent node type            */
   unsigned short  act_sess_count;      /* active session count on link  */
   unsigned char   indication_cause;    /* cause of indication           */
   LS_STATS        ls_stats;            /* link station statistics       */
   unsigned char   tg_num;              /* TG number                     */
   unsigned long   sense_data;          /* sense data                    */
   unsigned char   brnn_link_type;      /* branch link type              */
   unsigned char   adj_cp_is_brnn;      /* adjacent CP is a BrNN         */
   unsigned char   reserva[17];         /* reserved                      */
} LS_INDICATION;

typedef struct ls_stats
{
   unsigned long   in_xid_bytes;        /* num of XID bytes received     */
   unsigned long   in_msg_bytes;        /* num message bytes received    */
   unsigned long   in_xid_frames;       /* num XID frames received       */
   unsigned long   in_msg_frames;       /* num message frames received   */
   unsigned long   out_xid_bytes;       /* num XID bytes sent            */
   unsigned long   out_msg_bytes;       /* num message bytes sent        */
   unsigned long   out_xid_frames;      /* number of XID frames sent     */
   unsigned long   out_msg_frames;      /* num message frames sent       */
   unsigned long   in_invalid_sna_frames;
                                        /* num invalid frames recvd      */
   unsigned long   in_session_control_frames;
                                        /* number of control             */
                                        /* frames recvd                  */
   unsigned long   out_session_control_frames;
                                        /* number of control             */
                                        /* frames sent                   */
   unsigned long   echo_rsps;           /* response from adj LS count    */
   unsigned long   current_delay;       /* time taken for last           */
                                        /* test signal                   */
   unsigned long   max_delay;           /* max delay by test signal      */
   unsigned long   min_delay;           /* min delay by test signal      */
   unsigned long   max_delay_time;      /* time since longest delay      */
   unsigned long   good_xids;           /* successful XID on LS count    */
   unsigned long   bad_xids;            /* unsuccessful XID on LS count  */
} LS_STATS;
```

제공되는 매개변수

opcode

AP_LS_INDICATION

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivated

LS가 비활성 상태로 될 때는 AP_YES로 설정됩니다. LS가 활성 상태로 될 때는 AP_NO로 설정됩니다.

ls_name

링크 스테이션의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

description

자원 설명(DEFINE_LS에 지정한 것과 동일). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

adj_cp_name

17 바이트 길이의 전체 정식 인접 제어점 이름. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

adj_node_type

노드의 유형. 다음 중 하나입니다.

AP_END_NODE
AP_NETWORK_NODE
AP_LEN_NODE
AP_VRN

act_sess_count
링크를 사용하는 활동중인 세션(끝점과 중간 둘다)의 총 수.

indication_cause
표시 원인. 다음 중 하나입니다.

AP_ACTIVATION_STARTED
링크가 활성화중입니다.

AP_ACTIVATING
링크가 활성 상태로 되었습니다.

AP_DEACTIVATION_STARTED
링크가 활성종료중입니다.

AP_DEACTIVATING
링크가 비활성 상태로 되었습니다.

AP_SESS_COUNT_CHANGING
링크를 사용하는 활동중인 세션 수가 변경되었습니다.

AP_CP_NAME_CHANGING
인접 노드가 제어점 이름을 변경했습니다.

AP_FAILED
링크가 실패했습니다.

AP_ACTIVATION_FAILED
링크가 활성화에 실패했습니다.

AP_PENDING_RETRY
재시도 타이머가 시작되었습니다. 타이머가 만기될 때, 링크 활성화가 자동으로 재시도됩니다.

AP_DATA_LOST
이전 표시가 유실되었습니다. 링크 스테이션 통계는 링크 스테이션이 활성 상태에서 비활성 상태로 될 경우에만 제공됩니다(즉, deactivating이 AP_YES로 설정되어 있고 **indication_cause**가 AP_DEACTIVATING으로 설정되어 있을 때). 그 이외의 경우에는 필드들이 예약됩니다.

ls_stats.in_xid_bytes
이 링크 스테이션에서 수신되는 XID(교환 식별자) 바이트 총 수.

ls_stats.in_msg_bytes
이 링크 스테이션에서 수신되는 데이터 바이트 총 수.

ls_stats.in_xid_frames
이 링크 스테이션에서 수신되는 XID(교환 식별자) 프레임 총 수.

ls_stats.in_msg_frames
이 링크 스테이션에서 수신되는 데이터 프레임 총 수.

ls_stats.out_xid_bytes
이 링크 스테이션에서 송신되는 XID(교환 식별자) 바이트 총 수.

ls_stats.out_msg_bytes
이 링크 스테이션에서 송신되는 데이터 바이트 총 수.

ls_stats.out_xid_frames
이 링크 스테이션에서 송신되는 XID(교환 식별자) 프레임 총 수.

ls_stats.out_msg_frames
이 링크 스테이션에서 송신되는 데이터 프레임 총 수.

ls_stats.in_invalid_sna_frames
이 링크 스테이션에서 수신되는 SNA 오류 프레임 총 수.

ls_stats.in_session_control_frames
이 링크 스테이션에서 수신되는 세션 제어 프레임 총 수.

ls_stats.out_session_control_frames
이 링크 스테이션에서 송신되는 세션 제어 프레임 총 수.

ls_stats.echo_rsps
인접 노드로부터 수신되는 에코 응답 수. 에코 요청은 인접 노드로의 전달 지연을 측정하기 위해 주기적으로 송신됩니다.

ls_stats.current_delay
최종 검사 신호가 이 링크 스테이션에서 인접 링크 스테이션으로 송신되고 반환되는 데 걸리는 시간(밀리초).

ls_stats.max_delay
검사 신호가 이 링크 스테이션에서 인접 링크 스테이션으로 송신되고 반환되는 데 걸리는 최대 시간(밀리초).

ls_stats.min_delay
검사 신호가 이 링크 스테이션에서 인접 링크 스테이션으로 송신되고 반환되는 데 걸리는 최소 시간(밀리초).

ls_stats.max_delay_time
시스템 시작 후 최장(longest) 지연이 발생한 시간(1/100초 단위).

ls_stats.good_xids
이 링크 스테이션이 시작된 후 이 링크 스테이션에서 발생한 성공적 XID 교환 총 수.

ls_stats.bad_xids
이 링크 스테이션이 시작된 후 이 링크 스테이션에서 발생한 비성공적 XID 교환 총 수.

tg_num
TG와 연관된 번호.

sense_data
이 감지 데이터는 퍼스널 통신이나 통신 서버이 XID 프로토콜 오류를 검출할 때 설정됩니다. **indication_cause**가 AP_FAILED로 설정되지 않은 경우에는 이 필드가 예약됩니다.

brnn_link_type
BrNN 전용. 이 분기 링크 유형. 다음 중 하나입니다.

AP_UPLINK
링크가 업링크입니다.

AP_DOWNLINK
링크가 다운링크입니다.

AP_OTHERLINK
링크가 기타 링크입니다.

AP_UNKNOWN_LINK_TYPE
링크가 기타 링크입니다.

기타 노드 유형: 이 필드가 유효하지 않으며 항상 AP_BRNN_NOT_SUPPORTED로 설정됩니다.

adj_cp_is_brnnt
모든 노드 유형: 인접 노드가 BrNN인지의 여부를 지정합니다.

AP_UNKNOWN
인접 노드가 BrNN인지의 여부를 알 수 없습니다.

AP_NO
인접 노드가 BrNN이 아닙니다.

AP_YES
인접 노드가 BrNN입니다.

LU_0_TO_3_INDICATION

이 표시는 국지 LU(0-3 유형)의 상태가 변경될 때 생성됩니다.

VCB 구조

```
typedef struct lu_0_to_3_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char   attributes;          /* attributes                   */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   pu_name[8];          /* PU Name                      */
   unsigned char   lu_name[8];          /* LU Name                      */
   unsigned char   description[RD_LEN]; /* resource description         */
   unsigned char   nau_address;         /* NAU address                  */
   unsigned char   lu_sscp_sess_active;
                                        /* Is SSCP session active?      */
   unsigned char   appl_conn_active;    /* Is application using LU?     */
   unsigned char   plu_sess_active;     /* Is PLU-SLU session active?   */
   unsigned char   host_attachment;     /* Host attachment              */
   SESSION_STATS   lu_sscp_stats;       /* LU-SSCP session statistics   */
   SESSION_STATS   plu_stats;           /* PLU-SLU session statistics   */
   unsigned char   sscp_id[16];         /* SSCP ID                      */
} LU_0_TO_3_INDICATION;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;          /* session receive RU size       */
   unsigned short  send_ru_size;         /* session send RU size          */
   unsigned short  max_send_btu_size;    /* max send BTU  size            */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size              */
   unsigned short  max_send_pac_win;     /* max send pacing window size   */
   unsigned short  cur_send_pac_win;     /* current send pacing win size  */
   unsigned short  max_rcv_pac_win;      /* max receive pacing win size   */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing win size  */
   unsigned long   send_data_frames;     /* number of data frames sent    */
   unsigned long   send_fmd_data_frames;
                                         /* num of FMD data frames sent   */
   unsigned long   send_data_bytes;      /* number of data bytes sent     */
   unsigned long   rcv_data_frames;      /* num of data frames received   */
   unsigned long   rcv_fmd_data_frames;
                                         /* num FMD data frames received  */
   unsigned long   rcv_data_bytes;       /* number of data bytes received */
   unsigned char   sidh;                 /* session ID high byte          */
   unsigned char   sidl;                 /* session ID low byte           */
   unsigned char   odai;                 /* ODAI bit set                  */
   unsigned char   ls_name[8];           /* Link station name             */
   unsigned char   pacing_type;          /* type of pacing in use         */
} SESSION_STATS;
```

제공되는 매개변수

opcode

AP_LU_0_TO_3_INDICATION

LU_0_TO_3__INDICATION

attributes
이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

data lost가 AP_YES로 설정되어 있으면, 이 필드는 AP_EXTERNALLY_VISIBLE로 설정됩니다.

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc
AP_OK

secondary_rc
0과 동일합니다.

data_lost
데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

pu_name
국지 PU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

lu_name
상태가 변경된 국지 LU의 이름. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

description
자원 설명(DEFINE_LU_0_TO_3에 지정한 것과 동일). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

nau_address
LU의 네트워크 지정 장치(NAU) 주소(10—2554 범위 내의 값이어야 합니다).

lu_sscp_sess_active
ACTLU가 처리되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

appl_conn_active
응용프로그램이 이 LU를 사용중인지의 여부를 설정합니다(AP_YES 또는 AP_NO).

plu_sess_active
PLU-SLU 세션이 활성화되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

host_attachment
LU 호스트 접속 유형을 지정합니다.

AP_DLUR_ATTACHED
LU가 DLUR을 사용하여 호스트 시스템에 접속됩니다.

AP_DIRECT_ATTACHED
LU가 호스트 시스템에 직접 접속됩니다. LU-SSCP 및 PLU-SLU 통계는 이들 세션이 활성 상태에서 비활성 상태로 될 경우에만 유효함에 주의하십시오. 그 이외의 경우에는 필드들이 예약됩니다.

lu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

lu_sscp_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

lu_sscp_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

lu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

lu_sscp_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

lu_sscp_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

lu_sscp_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

lu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

lu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

lu_sscp_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, ACTLU 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, ACTLU 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

lu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 이 세션과 세션이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

lu_sscp_stats.pacing_type
LU-SSCP 세션에서 사용중인 수신 페이싱 유형. AP_NONE 값을 취합니다.

plu_stats.rcv_ru_size
최대 수신 RU 크기.

plu_stats.send_ru_size
최대 송신 RU 크기.

plu_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

plu_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

plu_stats.max_send_pac_win
이 세션에서의 송신 페이싱 창 최대 크기.

plu_stats.cur_send_pac_win
이 세션에서의 송신 페이싱 창 현재 크기.

plu_stats.max_rcv_pac_win
이 세션에서의 수신 페이싱 창 최대 크기.

plu_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

plu_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

plu_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

plu_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

plu_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

plu_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

plu_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

plu_stats.sidh
세션 ID 상위(high) 바이트.

plu_stats.sidl
세션 ID 하위(low) 바이트.

plu_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, ACTLU 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, ACTLU 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

plu_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 이 세션과 세션이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

plu_stats.pacing_type
PLU-SLU 세션에서 사용중인 수신 페이싱 유형. AP_NONE 또는 AP_PACING_FIXED 값을 취합니다.

sscp_id
이 LU에서 사용하는 PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하는 6바이트 필드입니다.

lu_sscp_sess_active가 AP_YES가 아닌 경우, 이 필드는 0으로 설정됩니다.

MODE_INDICATION

이 표시는 국지 LU와 상대방 LU 조합이 특정 모드를 사용하기 시작할 때와 국지 LU, 상대방 LU 및 모드 조합에 대한 현재 세션 수가 변경될 때 송신됩니다.

VCB 구조

```
typedef struct mode_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   removed;             /* is entry being removed?      */
   unsigned char   lu_alias[8];         /* LU alias                     */
   unsigned char   plu_alias[8];        /* partner LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner      */
                                        /* LU name                      */
   unsigned char   mode_name[8];        /* mode name                    */
   unsigned char   description[RD_LEN]; /* resource description         */
   unsigned short  curr_sess_count;     /* current session count        */
   unsigned char   reserva[20];         /* reserved                     */
} MODE_INDICATION;
```

제공되는 매개변수

opcode

AP_MODE_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

removed

입력항목이 제거되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 입력항목이 추가될 때가 아닌 제거될 때 설정됩니다.

lu_alias

국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

plu_alias

상대방 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

fqplu_name

상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다).

mode_name

세션 그룹의 네트워크 특성을 지정하는 모드명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

description

자원 설명(DEFINE_MODE에 지정한 것과 동일). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

curr_sess_count

이 국지 LU, 상대방 LU 및 모드 조합에 대한 현재 세션 수.

NN_TOPOLOGY_NODE_INDICATION

이 명령은 통신 서버에만 적용됩니다.

이 표시는 네트워크 노드의 위상 데이터베이스에 있는 노드 입력항목이 활성 상태에서 비활성 상태로 또는 비활성 상태에서 활성 상태로 변경될 때 생성됩니다.

VCB 구조

```
typedef struct nn_topology_node_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   deactivated;         /* has the node become inactive? */
   unsigned char   node_name[17];       /* node name                     */
   unsigned char   node_type;           /* node type                     */
   unsigned char   branch_aware;        /* node is branch aware          */
   unsigned char   reserva[19];         /* reserved                      */
} NN_TOPOLOGY_NODE_INDICATION;
```

제공되는 매개변수

opcode

AP_NN_TOPOLOGY_TG_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivated

노드가 비활성 상태로 될 때 AP_YES로 설정됩니다. 노드가 활동 상태로 될 때는 AP_NO로 설정됩니다.

node_name

네트워크 위상 데이터베이스에 있는 네트워크 정식 노드명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은

EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

node_type
노드 유형. 다음 중 하나입니다.

AP_NETWORK_NODE
AP_VRN

branch_aware
노드가 branch aware인지의 여부를 지정합니다.

AP_NO
노드가 branch aware가 아닙니다.

AP_YES
노드가 branch aware입니다.

NN_TOPOLOGY_TG_INDICATION

이 명령은 통신 서버에만 적용됩니다.

이 표시는 네트워크 노드의 위상 데이터베이스에 있는 TG 입력항목이 활성 상태에서 비활성 상태로 또는 비활성 상태에서 활성 상태로 변경될 때 생성됩니다.

VCB 구조

```
typedef struct nn_topology_tg_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   status;              /* TG status                     */
   unsigned char   owner[17];           /* name of TG owner node         */
   unsigned char   dest[17];            /* name of TG destination node   */
   unsigned char   tg_num;              /* TG number                     */
   unsigned char   owner_type;          /* Type of node that owns the TG */
   unsigned char   dest_type;           /* TG destination node type      */
   unsigned char   cp_cp_session_active;
                                        /* CP-CP session is active       */
   unsigned char   branch_tg;           /* TG is a branch TG             */
   unsigned char   reserva[16];         /* reserved                      */
} NN_TOPOLOGY_TG_INDICATION;
```

제공되는 매개변수

opcode

AP_NN_TOPOLOGY_TG_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

status TG의 상태를 지정합니다. 다음 값 중 하나 이상이 될 수 있습니다.(둘 이상일 경우에는 OR로 연결됨.)

AP_TG_OPERATIVE
AP_TG_QUIESCING
AP_TG_CP_CP_SESSIONS
AP_NONE

owner TG 근원지 노드의 이름(항상 국지 노드명으로 설정됩니다). 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다 (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다).

dest TG의 전체 정식 목적지 노드명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

tg_num
TG와 연관된 번호.

owner_type
TG를 소유하는 노드의 유형.

AP_NETWORK_NODE
AP_VRN

dest_type
노드의 유형.

AP_NETWORK_NODE
AP_VRN

cp_cp_session_active
소유 노드의 회선경합 성공 CP-CP 세션이 활동중인지의 여부를 지정합니다.(AP_NO 또는 AP_YES.)

branch_tg
TG가 분기 TG인지의 여부를 지정합니다.

AP_NO
TG가 분기 TG가 아닙니다.

AP_YES
TG가 분기 TG입니다.

PLU_INDICATION

이 표시는 국지 LU가 맨 처음 상대방 LU에 연결할 때 생성됩니다. 이러한 상황은 이 PLU에 대한 첫번째 ALLOCATE가 처리되거나 이 PLU에서 첫번째 BIND가 수신될 때 발생합니다. 이 표시는 또한 상대방 제어점 이름이 변경될 경우에도 생성됩니다.

VCB 구조

```
typedef struct plu_indication
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   data_lost;           /* has previous indication    */
                                        /* been lost?                 */
   unsigned char   removed;             /* is entry being removed?    */
   unsigned char   lu_alias[8];         /* LU alias                   */
   unsigned char   plu_alias[8];        /* partner LU alias           */
   unsigned char   fqplu_name[17];      /* fully qualified partner    */
                                        /* LU name                    */
   unsigned char   description[RD_LEN]; /* resource description       */
   unsigned char   partner_cp_name[17]; /* partner CP name            */
   unsigned char   partner_lu_located;  /* partner CP name resolved?  */
   unsigned char   reserva[20];         /* reserved                   */
} PLU_INDICATION;
```

제공되는 매개변수

,

opcode
AP_PLU_INDICATION

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc
AP_OK

secondary_rc
0과 동일합니다.

data_lost
하나 이상의 표시가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시를 송신할 수 없었을 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

removed

입력항목이 제거되는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 입력항목이 추가될 때가 아닌 제거될 때 설정됩니다.

lu_alias

국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

plu_alias

상대방 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

fqplu_name

상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다).

description

자원 설명(DEFINE_PARTNER_LU에 지정한 것과 동일). 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

partner_cp_name

상대방 LU 제어점(CP)의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다).

partner_lu_located

상대방 제어점 이름이 해결되었는지의 여부와(AP_YES 또는 AP_NO), 이로써 **partner_cp_name** 필드에 이 제어점 이름이 포함되는지의 여부를 지정합니다.

PORT_INDICATION

이 표시는 포트가 활성 상태에서 비활성 상태로(또는 그 반대로) 될 때 생성됩니다.

VCB 구조

```
typedef struct port_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char   attributes;          /* verb attributes               */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication lost      */
   unsigned char   deactivated;         /* has session been deactivated? */
   unsigned char   port_name[8];        /* link station name             */
   unsigned char   description[RD_LEN]; /* resource description          */
   unsigned char   reserva[20];         /* reserved                      */
} PORT_INDICATION;
```

제공되는 매개변수

opcode

AP_PORT_INDICATION

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivated

포트가 비활성 상태로 될 때 AP_YES로 설정됩니다. 포트가 활성 상태로 될 때에는 AP_NO로 설정됩니다.

port_name

포트의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

description

자원 설명(DEFINE_PORT에 지정한 것과 동일). 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

PU_INDICATION

이 표시는 국지 PU의 상태가 변경될 때 생성됩니다.

VCB 구조

```
typedef struct pu_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char   attributes;          /* attributes                   */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   pu_name[8];          /* PU Name                      */
   unsigned char   description[RD_LEN]; /* resource description         */
   unsigned char   pu_sscp_sess_active;
                                        /* Is SSCP session active?      */
   unsigned char   host_attachment;     /* Host attachment              */
   unsigned char   reserv1[2];          /* reserved                     */
   SESSION_STATS   pu_sscp_stats;       /* PU-SSCP session statistics   */
   unsigned char   sscp_id[6];          /* SSCP ID                      */
} PU_INDICATION;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;          /* session receive RU size      */
   unsigned short  send_ru_size;         /* session send RU size         */
   unsigned short  max_send_btu_size;    /* max send BTU  size           */
   unsigned short  max_rcv_btu_size;     /* max rcv BTU size             */
   unsigned short  max_send_pac_win;     /* max send pacing window size  */
   unsigned short  cur_send_pac_win;     /* curr send pacing window size */
   unsigned short  max_rcv_pac_win;      /* max rcv pacing window size   */
   unsigned short  cur_rcv_pac_win;      /* curr receive pacing win size */
   unsigned long   send_data_frames;     /* number of data frames sent   */
   unsigned long   send_fmd_data_frames;
                                         /* num FMD data frames sent     */
   unsigned long   send_data_bytes;      /* number of data bytes sent    */
   unsigned long   rcv_data_frames;      /* num of data frames received  */
   unsigned long   rcv_fmd_data_frames;
                                         /* num FMD data frames received */
   unsigned long   rcv_data_bytes;       /* num data bytes received      */
   unsigned char   sidh;                 /* session ID high byte         */
   unsigned char   sidl;                 /* session ID low byte          */
   unsigned char   odai;                 /* ODAI bit set                 */
   unsigned char   ls_name[8];           /* Link station name            */
   unsigned char   pacing_type;          /* type of pacing in use        */
} SESSION_STATS;
```

제공되는 매개변수

opcode

AP_PU_INDICATION

attributes

이 명령의 속성. 이 필드는 비트 필드입니다. 첫번째 비트에는 정의할 자원의 가시성이 포함되며 다음 중 하나와 일치합니다.

AP_EXTERNALLY_VISIBLE
AP_INTERNALLY_VISIBLE

data lost가 AP_YES로 설정되어 있으면, 이 필드는 AP_EXTERNALLY_VISIBLE로 설정됩니다.

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc
AP_OK

secondary_rc
0과 동일합니다.

data_lost
데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

pu_name
PU의 이름(DEFINE_LS 명령에 구성한). 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

description
자원 설명(DEFINE_LS 또는 DEFINE_INTERNAL_PU에 지정한 것과 동일). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

pu_sscp_sess_active
ACTPU가 처리되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

host_attachment
PU 호스트 접속 유형.

AP_DLUR_ATTACHED
PU가 DLUR을 사용하여 호스트 시스템에 접속됩니다.

AP_DIRECT_ATTACHED
PU가 호스트 시스템에 직접 접속됩니다.

주: PU-SSCP 통계는 세션 상태가 활성 상태에서 비활성 상태로 되었을 때에만 유효합니다.
그 이외의 경우에는 다음 필드들이 예약됩니다.

pu_sscp_stats.rcv_ru_size
이 필드는 항상 예약됩니다.

pu_sscp_stats.send_ru_size
이 필드는 항상 예약됩니다.

pu_sscp_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

pu_sscp_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

pu_sscp_stats.max_send_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.cur_send_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.max_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.cur_rcv_pac_win
이 필드는 항상 0으로 설정됩니다.

pu_sscp_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

pu_sscp_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

pu_sscp_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

pu_sscp_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

pu_sscp_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

pu_sscp_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

pu_sscp_stats.sidh
세션 ID 상위(high) 바이트.

pu_sscp_stats.sidl
세션 ID 하위(low) 바이트.

pu_sscp_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, ACTPU 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, ACTPU 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

pu_sscp_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자

세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 이 세션과 세션이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

pu_stats.pacing_type

PU-SSCP 세션에서 사용중인 수신 페이싱 유형. AP_NONE 값을 취합니다.

sscp_id

이 PU에 대한 ACTPU에서 수신된 SSCP ID를 포함하는 6바이트 필드입니다.

plu_sscp_sess_active가 AP_YES가 아닌 경우, 이 필드는 0으로 설정됩니다.

REGISTER_INDICATION_SINK

REGISTER_INDICATION_SINK는 표시가 송신되는 프로세스 및 대기행렬을 등록합니다.

REGISTER_INDICATION_SINK의 verb_signal 헤더에 있는 **orig_verb_data**는 수신되는 모든 표시의 verb_signal 헤더에서 반환됩니다.

VCB 구조

```
typedef struct port_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned PROC_ID
                   proc_id;             /* process identifier of sink   */
   unsigned QUEUE_ID
                   queue_id;            /* queue identifier where       */
                                        /* indications will be sent     */
   unsigned short  indication_opcode;   /* opcode of indication to      */
                                        /* be sunk                      */
} REGISTER_INDICATION_SINK;
```

제공되는 매개변수

opcode
AP_REGISTER_INDICATION_SINK

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

proc_id
수신 프로세스의 프로세스 ID.

queue_id
수신 프로세스 표시가 송신되는 대기행렬 ID.

indication_opcode
표시 싱크가 등록된 경우, 생성될 때마다 반환되는 표시의 Opcode.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_OP_CODE

AP_DYNAMIC_LOAD_ALREADY_REGD

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_LU_NAME

관련 START_NODE 매개변수가 송신되지 않아 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

STOP_NODE 명령을 발행했으므로 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

REGISTRATION_FAILURE

REGISTRATION_FAILURE는 네트워크 노드 서버에 자원을 등록하려는 시도가 실패했음을 지시합니다.

VCB 구조

```
typedef struct registration_failure
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   resource_name[17];   /* network qualified            */
                                        /* resource name                */
   unsigned short  resource_type;       /* resource type                */
   unsigned char   description[RD_LEN]; /* resource description         */
   unsigned char   reserv2b[2];         /* reserved                     */
   unsigned long   sense_data;          /* sense data                   */
   unsigned char   reserva[20];         /* reserved                     */
} REGISTRATION_FAILURE;
```

제공되는 매개변수

opcode

AP_REGISTRATION_FAILURE

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

resource_name

등록에 실패한 자원의 이름. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

resource_type

자원 유형. 다음 중 하나입니다.

AP_NNCP_RESOURCE
AP_ENCP_RESOURCE
AP_LU_RESOURCE

description

자원 설명(DEFINE_LOCAL_LU 또는 DEFINE_ADJACENT_NODE에 지정한 것과 동일).

sense_data

감지 데이터(*SNA* 형식에 지정된).

RTP_INDICATION

이 표시는 다음의 경우에 생성됩니다.

- RTP 연결이 연결되거나 단절될 때
- 활동중인 세션 수가 변경될 때
- 연결이 경로 전환을 수행할 때

연결이 단절될 때 최종 RTP 통계가 반환됩니다. 그 외에는 **rtp_stats** 필드가 예약됩니다.

VCB 구조

```
typedef struct rtp_indication
{
   unsigned short  opcode;              /* verb operation code           */
   unsigned char    reserv2;            /* reserved                      */
   unsigned char    format;             /* format                        */
   unsigned short  primary_rc;          /* primary return code           */
   unsigned long   secondary_rc;        /* secondary return code         */
   unsigned char   data_lost;           /* previous indication(s) lost   */
   unsigned char   connection_state;    /* the current state of the RTP  */
                                        /* connection                    */
   unsigned char   rtp_name[8];         /* name of the RTP connection    */
   unsigned short  num_sess_active;     /* number of active sessions     */
   unsigned char   indication_cause;    /* reason for this indication    */
   unsigned char   reserv3[3];           /* reserved                     */
   RTP_STATISTICS  rtp_stats;           /* RTP statistics                */
} RTP_INDICATION;

 typedef struct rtp_statistics
{
   unsigned long   bytes_sent;          /* total num of bytes sent       */
   unsigned long   bytes_received;      /* total num bytes received      */
   unsigned long   bytes_resent;        /* total num of bytes resent     */
   unsigned long   bytes_discarded;     /* total num bytes discarded     */
   unsigned long   packets_sent;        /* total num of packets sent     */
   unsigned long   packets_received;    /* total num packets received    */
   unsigned long   packets_resent;      /* total num of packets resent   */
   unsigned long   packets_discarded;   /* total num packets discarded   */
   unsigned long   gaps_detected;       /* gaps detected                 */
   unsigned long   send_rate;           /* current send rate             */
   unsigned long   max_send_rate;       /* maximum send rate             */
   unsigned long   min_send_rate;       /* minimum send rate             */
   unsigned long   receive_rate;        /* current receive rate          */
   unsigned long   max_receive_rate;    /* maximum receive rate          */
   unsigned long   min_receive_rate;    /* minimum receive rate          */
   unsigned long   burst_size;          /* current burst size            */
   unsigned long   up_time;             /* total uptime of connection    */
   unsigned long   smooth_rtt;          /* smoothed round-trip time      */
   unsigned long   last_rtt;            /* last round-trip time          */
   unsigned long   short_req_timer;     /* SHORT_REQ timer duration      */
   unsigned long   short_req_timeouts;  /* number of SHORT_REQ timeouts  */
   unsigned long   liveness_timeouts;   /* number of liveness timeouts   */
   unsigned long   in_invalid_sna_frames;
                                        /* number of invalid SNA frames  */
                                        /* received                      */
```

```
   unsigned long   in_sc_frames;       /* number of SC frames received   */
   unsigned long   out_sc_frames;      /* number of SC frames sent       */
   unsigned char   reserve[40];        /* reserved                       */
} RTP_STATISTICS;
```

제공되는 매개변수

opcode

AP_RTP_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 포함된 데이터가 이전 표시가 수신된 후 두 번 이상 변경되었을 가능성이 있습니다.

connection_state

RTP 연결의 현재 상태. 다음 중 하나입니다.

AP_CONNECTING

연결 설정이 시작되었으나 아직 완료되지는 않았습니다.

AP_CONNECTED

연결이 완전히 활동중입니다.

AP_DISCONNECTED

연결이 더이상 활동중에 있지 않습니다.

rtp_name

RTP 연결 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

num_sess_active

연결에서 현재 활동중인 세션 수.

indication_cause

표시 원인. 다음 중 하나입니다.

AP_ACTIVATED

연결이 활성 상태로 되었습니다.

AP_DEACTIVATED

연결이 비활성 상태로 되었습니다.

RTP_INDICATION

AP_PATH_SWITCHED
연결이 경로 전환을 완료하였습니다.

AP_SESS_COUNT_CHANGING
연결을 사용하는 활동중인 세션 수가 변경되었습니다.

AP_SETUP_FAILED
연결이 완전한 활성 상태가 되기 전에 실패했습니다. RTP 연결 통계는 연결이 비활성 상태로 될 때, 즉 **indication_cause**이 AP_DEACTIVATED 또는 AP_SETUP_FAILED일 때에만 유효합니다. 그 이외의 경우에는 필드들이 예약됩니다.

rtp_stats.bytes_sent
국지 노드가 이 RTP 연결에서 송신한 바이트 총 수.

rtp_stats.bytes_received
국지 노드가 이 RTP 연결에서 수신한 바이트 총 수.

rtp_stats.bytes_resent
송신시의 유실로 인해 국지 노드가 재송신한 총 바이트 수.

rtp_stats.bytes_discarded
RTP 연결의 다른쪽 끝에서 송신되었으며 이미 수신된 데이터의 중복으로 버려진 바이트의 총 수.

rtp_stats.packets_sent
국지 노드가 이 RTP 연결에서 송신한 패킷 총 수.

rtp_stats.packets_received
국지 노드가 이 RTP 연결에서 수신한 패킷 총 수.

rtp_stats.packets_resent
송신시의 유실로 인해 국지 노드가 재송신한 총 패킷 수.

rtp_stats.packets_discarded
RTP 연결의 다른쪽 끝에서 송신되었으며 이미 수신된 데이터의 중복으로서 버려진 패킷의 총 수.

rtp_stats.gaps_detected
국지 노드가 검출한 총 간격 수. 각 간격은 하나 이상의 유실된 프레임에 해당합니다.

rtp_stats.send_rate
이 RTP 연결에서의 현재 송신율(초당 킬로비트 단위로 측정). ARB 알고리즘으로 계산되는 최대 허용 송신율입니다.

rtp_stats.max_send_rate
이 RTP 연결에서의 최대 송신율(초당 킬로비트 단위로 측정).

rtp_stats.min_send_rate
이 RTP 연결에서의 최소 송신율(초당 킬로비트 단위로 측정).

rtp_stats.receive_rate
이 RTP 연결에서의 현재 수신율(초당 킬로비트 단위로 측정). 최종 측정 기간 동안에 계산된 실제 수신율입니다.

rtp_stats.max_receive_rate

이 RTP 연결에서의 최대 수신율(초당 킬로비트 단

SESSION_INDICATION

이 표시는 세션이 활성화되거나 활성종료될 때 생성됩니다. 세션이 활성종료될 때는 최종 세션 통계가 반환됩니다. 세션이 활성화될 때는 **sess_stats** 필드가 예약됩니다.

VCB 구조

```
typedef struct session_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   data_lost;           /* previous indication lost     */
   unsigned char   deactivated;         /* has session been deactivated? */
   unsigned char   lu_name[8];          /* LU name                      */
   unsigned char   lu_alias[8];         /* LU alias                     */
   unsigned char   plu_alias[8];        /* partner LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner      */
                                        /* LU name                      */
   unsigned char   mode_name[8];        /* mode name                    */
   unsigned char   session_id[8];       /* session ID                   */
   FQPCID          fqpcid;              /* fully qualified procedure    */
   unsigned long   sense_data;          /* sense_data                   */
   unsigned char   duplex_support;      /* full-duplex support          */
   SESSION_STATS   sess_stats;          /* session statistics           */
   unsigned char   sscp_id[6];          /* SSCP ID of host              */
   unsigned char   plu_slu_comp_lvl;    /* PLU to SLU compression level */
   unsigned char   slu_plu_comp_lvl;    /* SLU to PLU compressionlevel  */
                                        /* correlator ID                */
   unsigned char   reserva[12];         /* reserved                     */
} SESSION_INDICATION;

typedef struct fqpcid
{
   unsigned char   pcid[8];             /* procedure correlator         */
                                        /* identifier                   */
   unsigned char   fqcp_name[17];       /* originator's network         */
                                        /* qualified CP name            */
   unsigned char   reserve3[3];         /* reserved                     */
} FQPCID;

typedef struct session_stats
{
   unsigned short  rcv_ru_size;         /* session receive RU size      */
   unsigned short  send_ru_size;        /* session send RU size         */
   unsigned short  max_send_btu_size;   /* max send BTU  size           */
   unsigned short  max_rcv_btu_size;    /* max rcv BTU size             */
   unsigned short  max_send_pac_win;    /* max send pacing window size  */
   unsigned short  cur_send_pac_win;    /* curr send pacing window size */
   unsigned short  max_rcv_pac_win;     /* max receive pacing win size  */
   unsigned short  cur_rcv_pac_win;     /* curr receive pacing win size */
   unsigned long   send_data_frames;    /* number of data frames sent   */
   unsigned long   send_fmd_data_frame;
                                        /* num FMD data frames sent     */
   unsigned long   send_data_bytes;     /* number of data bytes sent    */
   unsigned long   rcv_data_frames;     /* num data frames received     */
   unsigned long   rcv_fmd_data_frames;
```

```
                                        /* num FMD data frames received  */
   unsigned long   rcv_data_bytes;      /* num data bytes received       */
   unsigned char   sidh;                /* session ID high byte          */
   unsigned char   sidl;                /* session ID low byte           */
   unsigned char   odai;                /* ODAI bit set                  */
   unsigned char   ls_name[8];          /* Link station name             */
   unsigned char   pacing_type;         /* type of pacing in use         */
   unsigned char   reserve;             /* reserved                      */
} SESSION_STATS;
```

제공되는 매개변수

opcode

AP_SESSION_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

secondary_rc

0과 동일합니다.

data_lost

데이터가 유실되었는지의 여부를 지정합니다(AP_YES 또는 AP_NO). 이 필드는 내부 구성요소가 이전 표시의 유실을 초래한 장애를 검출할 때 설정됩니다. **data_lost** 플래그가 AP_YES로 설정되어 있으면, 뒤에 오는 데이터 필드들이 널(null)로 설정될 수 있습니다. 응용프로그램은 QUERY 명령을 발행하여 유실된 정보를 갱신해야 합니다.

deactivated

세션이 활성화될 때 AP_NO로 설정됩니다. 세션이 활성종료될 때 AP_YES로 설정됩니다.

lu_name

LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다.

lu_alias

국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

plu_alias

상대방 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다.

fqplu_name

상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

SESSION_INDICATION

mode_name
세션 그룹의 네트워크 특성을 지정하는 모드명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

session_id
8바이트 세션 식별자(ID).

fqpcid.pcid
절차 상관자 ID. 8바이트 16진 문자열입니다.

fqpcid.fqcp_name
전체 정식 제어점 이름. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

sense_data
UNBIND 요청에서 송신 또는 수신되는 감지 데이터. **deactivated**가 AP_NO인 경우, 이 필드는 예약됩니다.

duplex_support
BIND에 조정된 대화 양방향(전송) 지원을 반환합니다. 다음 값 중 하나입니다.

AP_HALF_DUPLEX
비동시 양방향(전송)만이 지원됩니다.

AP_FULL_DUPLEX
동시 양방향(전송) 및 비동시 양방향(전송)이 지원됩니다.

AP_UNKNOWN
상대방 LU에 대해 활동중인 세션이 없으므로 대화 양방향(전송) 지원을 알 수 없습니다.

sess_stats.rcv_ru_size
최대 수신 RU 크기.

sess_stats.send_ru_size
최대 송신 RU 크기.

sess_stats.max_send_btu_size
송신가능한 최대 BTU 크기.

sess_stats.max_rcv_btu_size
수신가능한 최대 BTU 크기.

sess_stats.max_send_pac_win
이 세션에서의 송신 페이싱 창 최대 크기.

sess_stats.cur_send_pac_win
이 세션에서의 송신 페이싱 창 현재 크기.

sess_stats.max_rcv_pac_win
이 세션에서의 수신 페이싱 창 최대 크기.

sess_stats.cur_rcv_pac_win
이 세션에서의 수신 페이싱 창 현재 크기.

sess_stats.send_data_frames
송신되는 정상 흐름 데이터 프레임 수.

sess_stats.send_fmd_data_frames
송신되는 정상 흐름 FMD 데이터 프레임 수.

sess_stats.send_data_bytes
송신되는 정상 흐름 데이터 바이트 수.

sess_stats.rcv_data_frames
수신되는 정상 흐름 데이터 프레임 수.

sess_stats.rcv_fmd_data_frames
수신되는 정상 흐름 FMD 데이터 프레임 수.

sess_stats.rcv_data_bytes
수신되는 정상 흐름 데이터 바이트 수.

sess_stats.sidh
세션 ID 상위(high) 바이트.

sess_stats.sidl
세션 ID 하위(low) 바이트.

sess_stats.odai
발신 목적지 주소 표시기. 세션을 활성화할 때, BIND 전송자는 국지 노드에 1차 링크 스테이션이 있을 경우 이 필드를 0으로 설정하고, BIND 전송자가 2차 링크 스테이션이 있는 노드일 경우 이 필드를 1로 설정합니다.

sess_stats_stats.ls_name
통계와 연관된 링크 스테이션명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다. 이 필드는 세션 통계와 세션 트래픽이 흐르는 링크를 상관시키는 데 사용할 수 있습니다.

sess_stats_pacing_type
이 세션에서 사용중인 수신 페이싱 유형. AP_NONE, AP_PACING_FIXED 또는 AP_PACING_ADAPTIVE 값을 취할 수 있습니다.

sscp_id
종속 LU 세션의 경우, 이 필드에는 국지 LU가 매핑되는 PU에 대한 호스트로부터의 ACTPU에서 수신되는 SSCP ID가 포함됩니다. 독립 LU의 경우, 이 필드는 모두 2진 0으로 설정됩니다.

plu_slu_comp_lvl
PLU에서 SLU로 송신되는 데이터의 압축 레벨을 지정합니다.

AP_NONE
압축이 사용되지 않습니다.

AP_RLE_COMPRESSION
RLE 압축이 사용됩니다.

AP_LZ9_COMPRESSION
노드가 LZ9 압축을 지원할 수 있습니다.

AP_LZ10_COMPRESSION
노드가 LZ10 압축을 지원할 수 없습니다.

AP_LZ12_COMPRESSION
노드가 LZ12 압축을 지원할 수 있습니다.

slu_plu_comp_lvl
SLU에서 PLU로 송신되는 데이터의 압축 레벨을 지정합니다.

AP_NONE
압축이 사용되지 않습니다.

AP_RLE_COMPRESSION
RLE 압축이 사용됩니다.

AP_LZ9_COMPRESSION
노드가 LZ9 압축을 지원할 수 있습니다.

AP_LZ10_COMPRESSION
노드가 LZ10 압축을 지원할 수 없습니다.

AP_LZ12_COMPRESSION
노드가 LZ12 압축을 지원할 수 있습니다.

SESSION_FAILURE_INDICATION

이 표시는 세션이 활성종료될 때마다 생성됩니다. 이 표시는 보장됩니다. 즉, 반드시 생성됩니다.

VCB 구조

```
typedef struct session_failure_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned char   reserv3[3];          /* reserved                     */
   unsigned char   lu_name[8];          /* LU name                      */
   unsigned char   lu_alias[8];         /* LU alias                     */
   unsigned char   plu_alias[8];        /* partner LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner      */
                                        /* LU name                      */
   unsigned char   mode_name[8];        /* mode name                    */
   unsigned char   session_id[8];       /* session ID                   */
   unsigned long   sense_data;          /* sense_data                   */
  } SESSION_FAILURE_INDICATION;
```

제공되는 매개변수

opcode

AP_SESSION_FAILURE_INDICATION

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

primary_rc

AP_OK

lu_name

LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다.

lu_alias

국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효합니다.

plu_alias

상대방 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다.

fqplu_name

상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

SESSION_FAILURE_INDICATION

mode_name

세션 그룹의 네트워크 특성을 지정하는 모드명. 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

session_id

8바이트 세션 식별자(ID).

sense_data

세션 활성종료 원인을 설명하는 감지 데이터.

UNREGISTER_INDICATION_SINK

UNREGISTER_INDICATION_SINK는 요청되지 않은 표시를 수신하는 프로세스 및 대기행렬의 식별명(ID)을 제거합니다.

지정한 **proc_id, queue_id** 및 **indication_opcode** 조합이 한 번만 등록한 경우, 해당 입력항목이 제거됩니다. 지정한 조합을 두 번 이상 등록한 경우, UNREGISTER_INDICATION_SINK의 verb_signal 헤더에 있는 **orig_verb_data**와 일치하는 입력항목이 제거됩니다.

VCB 구조

```
typedef struct port_indication
{
   unsigned short  opcode;              /* verb operation code          */
   unsigned char    reserv2;            /* reserved                     */
   unsigned char    format;             /* format                       */
   unsigned short  primary_rc;          /* primary return code          */
   unsigned long   secondary_rc;        /* secondary return code        */
   unsigned PROC_ID
                   proc_id;             /* process identifier of sink   */
   unsigned QUEUE_ID
                   queue_id;            /* queue identifier where       */
                                        /* indications will be sent     */
   unsigned short  indication_opcode;   /* opcode of indication to      */
                                        /* be sunk                      */
} REGISTER_INDICATION_SINK;
```

제공되는 매개변수

opcode
AP_UNREGISTER_INDICATION_SINK

format
VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

proc_id
표시가 송신되는 프로세스의 프로세스 ID.

queue_id
표시가 송신되는 대기행렬의 대기행렬 ID.

indication_opcode
반환되는 표시의 Opcode.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_OP_CODE

AP_DYNAMIC_LOAD_ALREADY_REGD

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_LU_NAME

관련 START_NODE 매개변수가 송신되지 않아서 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_FUNCTION_NOT_SUPPORTED

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

STOP_NODE 명령을 발행했으므로 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

제10장 보안 명령

이 장에서는 보안 암호를 정의하고 삭제하는 데 사용되는 명령에 대해 설명합니다.

CONV_SECURITY_BYPASS

CONV_SECURITY_BYPASS는 응용프로그램으로 하여금 프로그램이 로컬 LU에 대해 대화 레벨 보안을 강제 실행하는지의 여부를 제어할 수 있도록 합니다. 일단 보안이 바이패스되면, 프로그램은 국지 LU 상의 대화에 대해 인증이나 권한 부여 작업을 수행하지 않습니다.

VCB 구조

```
typedef struct conv_security_bypass
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_name[8];          /* local LU name              */
   unsigned char   lu_alias[8];         /* local LU alias             */
   unsigned char   bypass_security;     /* should security be         */
                                        /* bypassed?                  */
   unsigned char   reserv3[3];          /* reserved                   */
} DEFINE_LU_LU_PASSWORD;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_CONV_SECURITY_BYPASS

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name
국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 로컬 LU를 판별하는 데 사용됩니다.

lu_alias
국지 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설정해야 합니다. **lu_alias** 및 **lu_name** 필드 둘다를 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

bypass_security
보안이 바이패스되는지의 여부를 지정합니다(AP_YES 또는 AP_NO).

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_LU_ALIAS
AP_INVALID_BYPASS_SECURITY

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

CREATE_PASSWORD_SUBSTITUTE

CREATE_PASSWORD_SUBSTITUTE는 지정한 세션의 대체 및 verifier를 생성하는 데 사용되는 암호 대체, 암호 verifier 및 송신 순서 번호를 반환합니다.

VCB 구조

```
typedef struct create_password_substitute
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_alias[8];        /* LU alias                   */
   unsigned char   conv_group_id[8];   /* partner LU alias           */
   unsigned char   user_id[10];        /* user ID                    */
   unsigned char   pw[10];             /* clear text password        */
   unsigned char   seq_no[8];          /* sequence number            */
   unsigned char   pw_sub[10];         /* password substitute        */
   unsigned char   pw_verifier[10];    /* password verifier          */
} CREATE_PASSWORD_SUBSTITUTE;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_CREATE_PASSWORD_SUBSTITUTE

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_alias
국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다.

conv_group_id
LU에서 사용하는 세션의 대화 그룹 식별자(CGID).

user_id
사용자 ID.

pw 암호화 알고리즘에서 사용되는 제거 텍스트 암호.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

seq_no
암호화 알고리즘에서 사용되는 송신 순서 번호. 명령이 성공적일 경우, 이 세션에 대한 송신 순서 번호의 내부 값이 증가됩니다. 반환되는 값은 증가된 후의 값입니다.

pw_sub
암호화 알고리즘에 의해 생성되는 암호 대체.

pw_verifier
암호화 알고리즘에 의해 생성되는 암호 verifier.

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_LU_ALIAS

AP_DEACT_CG_INVALID_CGID

세션이 암호 대체를 지원하지 않아서 명령이 실행되지 않을 경우, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_PW_SUB_NOT_SUPP_ON_SESS

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_LU_LU_PASSWORD

DEFINE_LU_LU_PASSWORD는 국지 LU와 상대방 LU간의 세션 레벨 검증에 사용되는 암호를 제공합니다.

VCB 구조

```
typedef struct define_lu_lu_password
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_name[8];          /* local LU name              */
   unsigned char   lu_alias[8];         /* local LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner    */
                                        /* LU name                    */
   unsigned char   verification_protocol
                                        /* LULU verification protocol */
   unsigned char   description[RD_LEN];
                                        /* resource description       */
   unsigned char   reserv3[8];          /* reserved                   */
   unsigned char   password[8];         /* password                   */
} DEFINE_LU_LU_PASSWORD;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_LU_LU_PASSWORD

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 로컬 LU를 판별하는 데 사용됩니다.

lu_alias

국지 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설정해야 합니다. **lu_alias** 및 **lu_name** 필드 둘다를 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

fqplu_name

전체 정식 상대방 LU 명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다. (각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

verification_protocol
이 상대방 LU에서 사용할 LU-LU 검증 프로토콜.

AP_BASIC_PROTOCOL
이 상대방 LU에서 기본 프로토콜만이 사용됩니다.

AP_ENHANCED_PROTOCOL
이 상대방 LU에서 확장 프로토콜만이 사용됩니다.

AP_EITHER_PROTOCOL
다음 세부사항에 따라, 이 상대방 LU에서 기본 또는 확장 프로토콜이 사용될 수 있습니다.

- 이 필드의 생략시 설정이 AP_EITHER_PROTOCOL입니다.
- 확장 프로토콜 사용으로 쉽게 이주할 수 있도록 AP_EITHER_PROTOCOL 값이 제공됩니다. 국지 LU는 상대방 LU가 확장 프로토콜을 사용하는 데 동의할 때까지 기본 프로토콜을 받아들입니다. 그리고 상대방 LU가 확장 프로토콜 사용에 동의한 이후부터는, 다음 DEFINE_LU_LU_PASSWORD를 발행하여 기본 프로토콜을 수락하도록 하지 않는 한, 국지 LU가 기본 프로토콜을 받아들이지 않습니다.

description
자원 설명.

password
암호. 8바이트 16진 문자열입니다. 암호 각 바이트의 최소 유효 비트는 세션 레벨 검증에 사용되지 않음에 주의하십시오.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DEFINE_USERID_PASSWORD

DEFINE_USERID_PASSWORD는 사용자 ID와 연관된 암호를 정의합니다.

VCB 구조

```
define_userid_password
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned short  define_type;         /* what the define type is    */
   unsigned char   user_id[10];         /* user id                    */
   unsigned char   reserv3[8];          /* reserved                   */
   USERID_PASSWORD_CHARS password_chars;
                                        /* password characteristics   */
} DEFINE_USERID_PASSWORD;

typedef struct userid_password_chars
{
   unsigned char   description[RD_LEN];
                                         /* resource description       */
   unsigned short  profile_count;        /* number of profiles         */
   unsigned short  reserv1;              /* reserved                   */
   unsigned char   password[10];         /* password                   */
   unsigned char   profiles[10][10]; /* profiles                   */
} USERID_PASSWORD_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_USERID_PASSWORD

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

define_type

정의할 사용자 암호의 유형을 지정합니다.

AP_ADD_USER

새로운 사용자를 지정하거나 기존 사용자의 암호를 변경합니다.

AP_ADD_PROFILES

기존 사용자의 프로파일에 대한 추가를 지정합니다.

user_id

사용자 식별자(ID). 10바이트 AE 유형 EBCDIC 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다.

password_chars.description
자원 설명. 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

password_chars.profile_count
프로파일 수.

password_chars.password
사용자 암호. 10바이트 AE 유형 EBCDIC 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다.

password_chars.profiles
사용자와 연관된 프로파일. 이들 각각은 10바이트 AE 유형 EBCDIC 문자열로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_NO_PROFILES

AP_UNKNOWN_USER
AP_INVALID_UPDATE_TYPE
AP_TOO_MANY_PROFILES
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_PASSWORD

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

DELETE_LU_LU_PASSWORD

DELETE_LU_LU_PASSWORD는 LU-LU 암호를 삭제합니다.

VCB 구조

```
typedef struct delete_lu_lu_password
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_name[8];          /* LU name                    */
   unsigned char   lu_alias[8];         /* local LU alias             */
   unsigned char   fqplu_name[17];      /* fully qualified partner    */
                                        /* LU name                    */
   unsigned char   reserv3;             /* reserved                   */
} DELETE_LU_LU_PASSWORD;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DELETE_LU_LU_PASSWORD

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

lu_name

국지 LU의 LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정되면, **lu_alias** 필드가 로컬 LU를 판별하는 데 사용됩니다.

lu_alias

국지 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설정해야 합니다. **lu_alias** 및 **lu_name** 필드 둘다를 0으로 설정하면, 명령이 제어점(CP)과 연관된 LU(생략시 LU)로 전달됩니다.

fqplu_name

전체 정식 상대방 LU 명. 길이가 17바이트이며 오른쪽이 EBCDIC 공백으로 채워집니다. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PLU_NAME

AP_INVALID_LU_NAME
AP_INVALID_LU_ALIAS

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_USERID_PASSWORD

DELETE_USERID_PASSWORD는 사용자 ID와 연관된 암호를 삭제합니다.

VCB 구조

```
typedef struct delete_userid_password
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned short  delete_type;         /* type of delete             */
   unsigned char   user_id[10];         /* user id                    */
   USERID_PASSWORD_CHARS password_chars;
                                        /* password characteristics   */

} DELETE_USERID_PASSWORD;

typedef struct userid_password_chars
{
   unsigned char   description[RD_LEN]; /* resource description       */
   unsigned short  profile_count;       /* number of profiles         */
   unsigned short  reserv1;             /* reserved                   */
   unsigned char   password[10];        /* password                   */
   unsigned char   profiles[10][10]; /* profiles                   */
} USERID_PASSWORD_CHARS;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_USERID_PASSWORD

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

delete_type
삭제 유형을 지정합니다.

AP_REMOVE_USER
사용자 암호 및 연관된 모든 프로파일을 삭제합니다.

AP_REMOVE_PROFILES
지정한 프로파일을 삭제합니다.

user_id
사용자 식별자(ID). 10바이트 AE 유형 EBCDIC 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다.

password_chars.description
이 명령을 처리할 때 이 필드는 무시됩니다.

password_chars.profile_count
프로파일 수.

password_chars.password
이 명령을 처리할 때 이 필드는 무시됩니다.

password_chars.profiles
사용자와 연관된 프로파일. 이들 각각은 10바이트 AE 유형 EBCDIC 문자열로서, 오른쪽이 EBCDIC 공백으로 채워집니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_NO_PROFILES

AP_UNKNOWN_USER
AP_INVALID_UPDATE_TYPE

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

SIGN_OFF

SIGN_OFF는 LU로 하여금 사인 온 리스트에서 입력항목을 제거하도록 지시합니다. 현재로서는, 사인 온 리스트에 있는 입력항목만이 제거됩니다. 이 명령은 모든 입력항목을 제거하도록 지정하거나 추가한 sign_off_data 구조 내의 입력항목만을 제거하도록 지정할 수 있습니다.

VCB 구조

```
typedef struct query_sign_off
{
   unsigned short  opcode;              /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;          /* primary return code        */
   unsigned long   secondary_rc;        /* secondary return code      */
   unsigned char   lu_name[8];         /* LU name                     */
   unsigned char   lu_alias[8];        /* LU alias                    */
   unsigned char   plu_alias[8];       /* partner LU alias            */
   unsigned char   fqplu_name[17];     /* fully qualified partner     */
                                       /* LU name                     */
   unsigned char   list;               /* signed on to/from list      */
   unsigned char   all_in_list;        /* sign off all entries in list */
   unsigned char   immediate;          /* remove entries immediately  */
   unsigned char   num_entries;        /* number of entries           */
} QUERY_SIGN_OFF;

typedef struct sign_off_data
{
   unsigned char   user_id[10];       /* user ID                      */
   unsigned char   all_profiles;      /* all profiles for this user   */
   unsigned char   profile[10];       /* specific profile             */
} SIGN_OFF_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_CREATE_PASSWORD_SUBSTITUTE

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

LU 명. 이 이름은 8바이트 A 유형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 색인을 판별하는 데 사용됩니다.

lu_alias

국지로 정의된 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설

정해야 합니다. **lu_name** 및 **lu_alias** 필드 둘다를 모두 0으로 설정하면, 제어점(CP)과 연관된 LU(생략시 LU)가 사용됩니다.

plu_alias
상대방 LU 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다. 이 필드를 모두 0으로 설정하면, **fqplu_name** 필드가 색인을 판별하는 데 사용됩니다.

fqplu_name
상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

list 사인 온 리스트 유형. 이 필드는 AP_SIGNED_ON_TO_LIST로 설정해야 합니다.

AP_SIGNED_ON_TO_LIST
국지 LU에서 원격 LU에 사인 온한 사용자 리스트. 이 리스트에서 입력항목이 제거될 때 원격 LU에는 통지되지 않습니다. 현재 지원되는 유일한 값입니다.

all_in_list
AP_YES로 설정할 경우, **list**로 지정한 리스트 내의 모든 사용자가 사인 오프됩니다.

immediate
AP_YES로 설정할 경우, 사용자가 즉시 제거됩니다. AP_NO로 설정하면, 원격 LU가 사인 오프가 완료되었음을 확인한 후에 사용자가 제거됩니다. **list**가 AP_SIGNED_ON_TO_LIST인 경우, 이 필드는 예약됩니다.

num_entries
실제 반환되는 입력항목 수.

all_in_list가 AP_NO이면, 사용자 리스트를 일련의 SIGN_OFF_DATA 구조로서 SIGN_OFF VCB에 추가해야 합니다. SIGN_OFF_DATA 구조의 매개변수는 다음과 같습니다.

user_id
사용자 ID.

all_profiles
반환가능한 입력항목 총 수. **num_entries**보다 클 수 있습니다.

profile
10바이트 영숫자 EBCDIC 문자열. 프로그램은 현재 공백 프로파일(10개의 eBCDIC 공백)만을 지원함에 주의하십시오. **list_options**을 AP_FIRST_IN_LIST로 설정할 경우에는 이 필드가 무시됩니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_PLU_NAME
AP_INVALID_USERID
AP_INVALID_PROFILE
AP_INVALID_LIST
AP_INVALID_LIST_OPTION

프로그램에 의해 성공적으로 처리되지 않는 모든 SIGN_OFF_DATA **user_id/profile** 조합은 VCB에 추가된 응용프로그램에 반환되며, **num_entries**의 반환된 값은 프로그램에 의해 반환되는 SIGN_OFF_DATA 입력항목(처리할 수 없는)의 수입니다.

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_ALIAS

AP_INVALID_LU_NAME
AP_INVALID_LU_NAME
AP_INVALID_LIST

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료되었으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

SIGN_OFF

제11장 APING 및 CPI-C 명령

이 장에서는 다른 노드를 『핑(ping)』하는 데 사용되는 명령과 CPI-C 부가 정보를 정의, 삭제 및 조회하는 데 사용되는 명령에 대해 설명합니다.

APING

APING은 관리 응용프로그램이 네트워크에서 원격 LU를 『핑(ping)』 할 수 있도록 합니다. VCB의 끝에 검증 데이터 문자열(지정한 길이의)을 추가할 수 있으며, **partner_ver_len** 필드를 0보다 큰 값으로 설정할 때 이 문자열이 반환될 수 있습니다.

퍼스널 통신이나 통신 서버 APING은 퍼스널 통신이나 통신 서버 APPC API(*퍼스널 통신 클라이언트/서버 통신 프로그래밍*에 설명되어 있음)를 사용하는 내부 『서비스 트랜잭션 프로그램(TP)』으로 구현됩니다.

VCB 구조

```
typedef struct aping
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   secondary_rc;         /* secondary return code      */
   unsigned char   lu_name[8];        /* local LU name                 */
   unsigned char   lu_alias[8];       /* local LU alias                */
   unsigned long   sense_data;        /* sense data                    */
   unsigned char   plu_alias[8];      /* partner LU alias              */
   unsigned char   mode_name[8];      /* mode name                     */
   unsigned char   tp_name[64];       /* destination TP name           */
   unsigned char   security;          /* security level                */
   unsigned char   reserv3a[3];       /* reserved                      */
   unsigned char   pwd[10];           /* password                      */
   unsigned char   user_id[10];       /* user ID                       */
   unsigned short  dlen;              /* length of data to send        */
   unsigned short  consec;            /* number of consecutive sends   */
   unsigned char   fqplu_name[17];    /* fully qualified partner       */
                                      /* LU name                       */
   unsigned char   echo;              /* data echo flag                */
   unsigned short  iterations;        /* number of iterations          */
   unsigned long   alloc_time;        /* time taken for ALLOCATE       */
   unsigned long   min_time;          /* min send/receive time         */
   unsigned long   avg_time;          /* average send/receive time     */
   unsigned long   max_time;          /* max send/receive time         */
   unsigned short  partner_ver_len;   /* size of string to receive     */
} APING;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_APING

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

lu_name

APING 명령을 송신하는 국지 LU의 LU 명. 이 이름은 8바이트 A 유

형 EBCDIC 문자열입니다. 이 필드를 모두 0으로 설정하면, **lu_alias** 필드가 국지 LU를 판별하는 데 사용됩니다.

lu_alias
APING 명령을 송신하는 국지 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 **lu_name** 필드를 모두 0으로 설정한 경우에만 유효하며, 8바이트 모두 유효 바이트로서 반드시 설정해야 합니다. **lu_name** 및 **lu_alias** 필드 둘다를 2진 0으로 설정하면, 생략시(제어점(CP) LU가 사용됩니다.

plu_alias
국지 트랜잭션 프로그램(TP)에서 사용되는 상대방 LU의 별명. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 8바이트 모두 유효 바이트이므로 반드시 설정해야 합니다. 이 이름은 구성시에 설정한 상대방 LU의 이름과 일치해야 합니다. 이 매개변수를 2진 0으로 설정하면, **fqplu_name** 매개변수가 사용됩니다.

mode_name
사용할 모드의 이름. 이 이름은 8바이트 영숫자 A 유형 EBCDIC 문자열(문자로 시작)로, 오른쪽이 EBCDIC 공백으로 채워집니다.

tp_name
기동되는 트랜잭션 프로그램(TP)의 이름. 64바이트 문자열입니다. 노드 연산자 기능은 이 문자열의 문자 세트를 점검하지 않습니다. **tp_name**의 값은 원격 LU에 구성한 값과 일치해야 합니다. 이 문자열은 주로 EBCDIC 형식의 『APINGD』로 설정되며, 오른쪽이 EBCDIC 공백으로 채워집니다.

security
기동된 트랜잭션 프로그램(TP)에 대한 액세스의 유효성을 검사하기 위해 상대방 LU가 요구하는 정보를 지정합니다.

AP_NONE
AP_PGM
AP_SAME
AP_PGM_STRONG

pwd **user_id**와 연관된 암호. 10바이트 AE 유형 EBCDIC 문자열로서, 오른쪽이 EBCDIC 공백으로 채워집니다. **security**를 AP_PGM 또는 AP_PGM_STRONG으로 설정할 경우에만 필요합니다.

user_id
상대방 트랜잭션 프로그램(TP)에 액세스하는 데 필요한 사용자 ID. 10바이트 AE 유형 EBCDIC 문자열로, 오른쪽이 EBCDIC 공백으로 채워집니다. **security**를 AP_PGM, AP_PGM_STRONG 또는 AP_SAME으로 설정할 경우에 필요합니다.

dlen APING 트랜잭션 프로그램(TP)이 전송할 데이터 길이. APING은 길이가 **dlen**인 0 문자열을 송신합니다.

consec

각 반복시에 수행되는 연속 송신 수. APING은 이러한 수만큼의 MC_SEND_DATA 명령을 발행하며, 각 명령은 **dlen**바이트의 데이터로 구성됩니다. **echo** 매개변수를 AP_YES로 설정하면, APING은 마지막 MC_SEND_DATA를 AP_SEND_DATA_P_TO_R_FLUSH(플러쉬 수신 준비)로 마크하며 상대방 APINGD 트랜잭션 프로그램(TP)으로부터 데이터를 포함하는 응답을 기다립니다(MC_RECEIVE_AND_WAIT를 발행하여). **echo** 매개변수를 AP_NO로 설정하면, APING은 데이터를 플러쉬하고 확인을 기다립니다(마지막 MC_SEND_DATA를 AP_SEND_DATA_CONFIRM으로 마크하여). 두 경우 모두 여기에 기술된 순서가 SNA 체인에 해당합니다.

fqplu_name

상대방 LU의 17바이트 전체 정식 네트워크명. 이 이름은 EBCDIC 점으로 연결된 두 개의 A 유형 EBCDIC 문자열로 구성되며, 오른쪽이 EBCDIC 공백으로 채워집니다(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다). 이 필드는 **plu_alias** 필드를 모두 0으로 설정한 경우에만 유효합니다.

echo APING 트랜잭션 프로그램(TP)이 요구된 양의 데이터 송신을 완료했을 때 응답을 기대하는지의 여부를 지정합니다.

AP_YES
AP_NO

iterations

APING이 발행하는 연속 순서 반복 수(**consec** 매개변수로 지정). SNA의 경우, 이 매개변수는 송신되는 체인의 수를 정의합니다.

partner_ver_len

관리 응용프로그램이 수신할 수 있는 상대방 트랜잭션 프로그램(TP) 검증 데이터 문자열 최대 길이.

반환되는 매개변수

명령이 실행되면, APING은 다음 매개변수를 반환합니다.

primary_rc

AP_OK

sense_data

명령이 성공적으로 반환된 경우, 이 필드는 0이 됩니다.

alloc_time

원격 트랜잭션 프로그램(TP)에 대한 MC_ALLOCATE를 완료하는 데 필요한 시간(밀리초 단위).

min_time

데이터 송신 반복에 요구되는 최소 시간(밀리초 단위). 이 매개변수에는 상대방이 응답하는 데(**echo** 매개변수의 설정에 따라 데이터를 송신하거나 확인을 발행하여) 필요한 시간이 포함됩니다.

avg_time
데이터 송신 반복에 요구되는 평균 시간(밀리초 단위). 이 매개변수에는 상대방이 응답하는 데(**echo** 매개변수의 설정에 따라 데이터를 송신하거나 확인을 발행하여) 필요한 시간이 포함됩니다.

max_time
데이터 송신 반복에 요구되는 최대 시간(밀리초 단위). 이 매개변수에는 상대방이 응답하는 데(**echo** 매개변수의 설정에 따라 데이터를 송신하거나 확인을 발행하여) 필요한 시간이 포함됩니다.

partner_ver_len
상대방 트랜잭션 프로그램(TP)이 반환하는 검증 문자열 길이. 문자열 자체가 VCB 끝에 추가됩니다.

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_LU_ALIAS

APING은 퍼스널 통신이나 통신 서버 APPC API에서 제공하는 MC_ALLOCATE, MC_SEND_DATA, MC_RECEIVE_AND_WAIT, MC_CONFIRM 및 MC_DEALLOCATE 명령을 사용합니다. 이들 명령이 실행에 실패할 때 반환하는 매개변수는 *퍼스널 통신 클라이언트/서버 통신 프로그래밍*에 설명되어 있습니다.

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

CPI-C 명령

DEFINE_CPIC_SIDE_INFO

이 명령은 메모리에 부가 정보를 추가하거나 대체합니다. CPI-C 부가 정보 입력항목은 대화 특성 집합을 기호식 목적지 이름과 연관시킵니다. 이 명령에 지정한 것과 동일한 기호식 목적지 이름을 가지는 부가 정보가 이미 메모리에 들어 있는 경우, 이것은 이 호출에 제공되는 데이터로 대체됩니다. 퍼스널 통신이나 통신 서버에서 제공하는 CPI-C 지원에 대해서는 *CPI-C 참조 사항*을 참조하십시오.

VCB 구조

```
typedef struct define_cpic_side_info
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   secondary_rc;         /* secondary return code      */
   unsigned char   reserv2a[8];         /* reserved                   */
   unsigned char   sym_dest_name[8];     /* Symbolic destination name  */
   CPIC_SIDE_INFO_DEF_DATA def_data;   /* defined data               */
} DEFINE_CPIC_SIDE_INFO;

typedef struct cpic_side_info_def_data
{
   unsigned char   description[RD_LEN];
                                       /* resource description       */
   CPIC_SIDE_INFO  side_info;          /* CPIC side info             */
   unsigned char   user_data[32];      /* User defined data          */
} CPIC_SIDE_INFO_DEF_DATA;

typedef struct cpic_side_info
{
   unsigned char   partner_lu_name[17];
                                       /* Fully qualified partner    */
                                       /* LU name                    */
   unsigned char   reserved[3];        /* Reserved                   */
   unsigned long   tp_name_type;       /* TP name type               */
   unsigned char   tp_name[64];        /* TP name                    */
   unsigned char   mode_name[8];       /* Mode name                  */
   unsigned long   conversation_security_type;
                                       /* Conversation security type */
   unsigned char   security_user_id[CPIC_SECURITY_INFO_LEN];
                                       /* User ID                    */
   unsigned char   security_password[CPIC_SECURITY_INFO_LEN];
                                       /* Password                   */
} CPIC_SIDE_INFO;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

AP_DEFINE_CPIC_SIDE_INFO

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

sym_dest_name

부가 정보 입력항목을 식별하는 기호식 목적지 이름. 국지로 표시가능한 문자 세트로 된 최대 8바이트 길이의 이름으로서, 공백으로 채워집니다. 허용되는 문자는 대문자(A - Z)와 숫자 0–9입니다.

def_data.description

자원 설명(QUERY_CPIC_SIDE_INFO에서 반환). 이 이름은 국지로 표시가능한 문자 세트로 된 16바이트 문자열입니다. 16바이트 모두 유효합니다.

def_data.side_info.partner_lu_name

상대방 LU의 전체 정식명. 국지로 표시가능한 문자 세트로 된 17바이트 길이의 이름으로서, 오른쪽이 공백으로 채워집니다. 이 이름은 점으로 연결된 두 개의 문자열로 구성됩니다.(각 이름의 최대 길이는 삽입 공백 없이 8바이트입니다.)

def_data.side_info.tp_name_type

트랜잭션 프로그램(TP) 명 유형. 이 필드는 다음 값 중 하나로 설정됩니다.

XC_APPLICATION_TP

입력한 트랜잭션 프로그램명(TPN)이 서비스 트랜잭션 프로그램(TP)이 아님을 지정합니다. 트랜잭션 프로그램명(TPN)에 지정한 모든 문자는 국지로 표시가능한 문자 세트로 된 유효한 문자라야 합니다.

XC_SNA_SERVICE_TP

입력한 트랜잭션 프로그램명(TPN)이 서비스 트랜잭션 프로그램(TP)의 이름이 되도록 지정합니다. 트랜잭션 프로그램(TP)에 지정한 모든 문자는 첫 번째 문자를 제외하곤 국지로 표시가능한 문자 세트로 된 유효한 문자라야 합니다. 첫 번째 문자는 X'01'-X'3F'(X'0E' 및 X'0F' 포함) 범위 내의 16진 숫자라야 합니다.

def_data.side_info.tp_name

국지로 표시가능한 문자 세트로 된 64바이트 문자열로 오른쪽이 공백으로 채워지는 트랜잭션 프로그램(TP) 명.

def_data.side_info.mode_name

국지로 표시가능한 문자 세트로 된 8바이트 문자열로 오른쪽이 공백으로 채워지는 모드명.

def_data.side_info.conversation_security_type

대화 보안 유형. 이 필드는 다음 값 중 하나로 설정됩니다.

XC_SECURITY_NONE
XC_SECURITY_SAME
XC_SECURITY_PROGRAM
XC_SECURITY_PROGRAM_STRONG.

def_data.side_info.security_user_id
사용자 ID. 퍼스널 통신이나 통신 서버는 대화 레벨 보안 강제 실행 시에 이 필드를 사용합니다.

def_data.side_info.security_password
암호. 퍼스널 통신이나 통신 서버는 대화 레벨 보안 강제 실행시에 이 필드를 사용합니다.

def_data.user_data
사용자 데이터. 이 데이터는 QUERY_CPIC_SIDE_INFO에서 반환되나, 퍼스널 통신이나 통신 서버에 의해 사용되거나 해석되지는 않습니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_SYM_DEST_NAME

AP_INVALID_LENGTH

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

DELETE_CPIC_SIDE_INFO

이 명령은 CPI-C 부가 정보 입력항목을 삭제합니다. 퍼스널 통신이나 통신 서버에서 제공하는 CPI-C 지원에 대해서는 *CPI-C 참조 사항*을 참조하십시오.

VCB 구조

```
typedef struct delete_cpic_side_info
{
   unsigned short  opcode;               /* verb operation code        */
   unsigned char    reserv2;            /* reserved                   */
   unsigned char    format;             /* format                     */
   unsigned short  primary_rc;           /* primary return code        */
   unsigned long   secondary_rc;         /* secondary return code      */
   unsigned char   reserv2a[8];         /* reserved                   */
   unsigned char   sym_dest_name[8];     /* Symbolic destination name  */
} DELETE_CPIC_SIDE_INFO;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DELETE_CPIC_SIDE_INFO

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

sym_dest_name
부가 정보 입력항목을 식별하는 기호식 목적지 이름. 국지로 표시 가능한 문자 세트로 된 최대 8바이트 길이의 이름으로, 공백으로 채워집니다. 허용되는 문자는 대문자(A - Z)와 숫자 0–9입니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_SYM_DEST_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

w- B

QUERY_CPIC_SIDE_INFO

이 명령은 주어진 기호식 목적지 이름의 부가 정보 입력항목을 반환합니다. 이 정보는 리스트로서 반환됩니다. 특정 부가 정보 입력항목이나 특정 입력항목 청크량키

,b

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_QUERY_CPIC_SIDE_INFO

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

buf_ptr
리스트 정보를 쓸 수 있는 버퍼에 대한 포인터.

buf_size
제공되는 버퍼의 크기. 반환되는 데이터는 이 크기를 초과하지 않습니다.

num_entries
반환할 입력항목 최대 수. 입력항목 수는 이 값을 초과하지 않습니다. 0 값은 제한 없음을 의미합니다.

list_options
이 필드는 리스트 정보에서 반환되는 내용을 지시합니다. 지정한 **sym_dest_name**(아래 참조)은 반환될 실제 정보의 시작점을 지정하는 데 사용되는 색인 값을 나타냅니다.

AP_FIRST_IN_LIST
색인 값이 무시되고, 반환되는 리스트가 리스트의 첫 번째 입력항목에서 시작됩니다.

AP_LIST_FROM_NEXT
반환되는 리스트가 입력한 색인 값이 지정하는 입력항목 다음의 입력항목부터 시작됩니다.

AP_LIST_INCLUSIVE
반환되는 리스트가 색인 값이 지정하는 입력항목부터 시작됩니다.

sym_dest_name
부가 정보 입력항목을 식별하는 기호식 목적지 이름. 국지로 표시가능한 문자 세트로 된 최대 8바이트 길이의 이름으로서, 공백으로 채워집니다. 허용되는 문자는 대문자(A - Z)와 숫자 0–9입니다.

반환되는 매개변수

명령이 실행되면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_OK

buf_size
버퍼에서 반환되는 정보의 길이.

total_buf_size
요청된 모든 리스트 정보를 반환하는 데 필요한 버퍼 크기를 지시하는 리턴 값. 이 값은 **buf_size**보다 클 수도 있습니다.

num_entries
실제 반환되는 입력항목 수.

total_num_entries
반환가능한 입력항목 총 수. **num_entries**보다 클 수도 있습니다.

cpic_side_info_data.overlay_size
이 입력항목의 바이트 수, 따라서 반환되는 다음 입력항목에 대한 오프셋(있을 경우).

cpic_side_info_data.sym_dest_name
반환되는 부가 정보 입력항목의 기호식 목적지 이름.

cpic_side_info_data.def_data
DEFINE_CPIC_SIDE_INFO에서 입력한 정의된 CPI-C 부가 정보.

주: 퍼스널 통신이나 통신 서버가 DEFINE_CPIC_SIDE_INFO를 처리한 후에는, CPIC 호출이 이 명령에서 반환되는 부가 정보를 변경할 수도 있습니다.

상태 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_INVALID_SYM_DEST_NAME

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT_STARTED

노드가 종료중이므로 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_STOPPING

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램은 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

제12장 접속 관리자 명령

퍼스널 통신이나 통신 서버 접속 관리자는 APPC 또는 CPI-C 프로그램의 시작을 관리하는 데 사용됩니다. 접속 관리자 기능에 대한 설명은 *퍼스널 통신 클라이언트/서버 통신 프로그래밍*을 참조하십시오.

퍼스널 통신이나 통신 서버 노드 연산자 기능은 접속 관리자를 제어하기 위해 세 개의 명령을 지원합니다. 이들 명령은 퍼스널 통신이나 통신 서버 노드 연산자 기능을 사용하는 모든 응용프로그램 프로그램에서 사용할 수 있습니다.

DISABLE_ATTACH_MANAGER

퍼스널 통신이나 통신 서버 접속 관리자는 노드가 시작될 때 기본적으로 사용가능화됩니다. 사용자는 이 명령을 사용하여 모든 동적 로드를 사용불능화할 수 있습니다. 이 명령은 접속 관리자가 트랜잭션 프로그램(TP)을 시작하기 전에 점검하는 글로벌 플래그를 재설정합니다.

VCB 구조

```
typedef struct disable_am
{
   unsigned short   opcode;            /* Verb operation code         */
   unsigned char    reserv2;           /* reserved                    */
   unsigned char    format;            /* format                      */
   unsigned short   primary_rc;        /* Primary return code         */
   unsigned long    secondary_rc;      /* Secondary return code       */
} DISABLE_AM;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_DISABLE_ATTACH_MGR

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 실행되면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_OK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

ENABLE_ATTACH_MANAGER

접속 관리자가 사용불가능하게 되면, 퍼스널 통신이나 통신 서버 노드 연산자 기능 명령 ENABLE_AM을 발행하여 접속 관리자를 다시 사용가능하게 할 수 있습니다. 그러면, 접속 관리자가 트랜잭션 프로그램(TP)을 시작하기 전에 점검하는 글로벌 플래그가 설정됩니다.

VCB 구조

```
typedef struct enable_am
{
   unsigned short   opcode;          /* Verb operation code        */
   unsigned char    reserv2;         /* reserved                   */
   unsigned char    format;          /* format                     */
   unsigned short   primary_rc;      /* Primary return code        */
   unsigned long    secondary_rc;    /* Secondary return code      */
} ENABLE_AM
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode
AP_ENABLE_ATTACH_MGR

format
VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 실행되면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_OK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_NODE_NOT STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_ATTACH_MANAGER

QUERY_ATTACH_MANAGER 명령은 ENABLE_ATTACH_MANAGER 및 DISABLE_ATTACH_MANAGER 명령을 사용하여 시작 및 종료할 수 있는 접속 관리자 구성요소의 상태를 발견하는 데 사용할 수 있습니다.

VCB 구조

```
typedef struct query_am
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
        unsigned short   primary_rc;      /* primary return code          */
        unsigned long    secondary_rc;    /* secondary return code        */
        unsigned short   active;          /* status of the Attach Manager */
} QUERY_AM;
```

제공되는 매개변수

opcode

AP_QUERY_ATTACH_MGR

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

반환되는 매개변수

명령이 실행되면, 다음 매개변수가 반환됩니다.

primary_rc

AP_OK

active 이 필드는 접속 관리자 구성요소의 상태를 보고합니다.

AP_YES

접속 관리자가 활동중입니다.

AP_NO

접속 관리자가 활동중이 아닙니다.

매개변수 오류로 인해 명령이 실행되지 않으면, 다음 매개변수가 반환됩니다.

primary_rc

AP_PARAMETER_CHECK

아직 노드가 시작되지 않았으므로 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc

AP_NODE_NOT STARTED

시스템 오류로 인해 명령이 실행되지 않으면, 접속 관리자는 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

제2부 퍼스널 통신이나 통신 서버 관리 서비스 API

제13장 관리 서비스 API 소개

이 부에서는 퍼스널 통신이나 통신 서버에서 제공하는 관리 서비스 API에 대해 설명합니다.

관리 서비스 명령

퍼스널 통신이나 통신 서버는 다음 관리 서비스(MS) 명령을 지원함으로써, 어플리케이션 프로그램에 SNA 네트워크에서 사용가능한 관리 서비스 중재점에 잠재적 문제점을 보고하는 방법을 제공합니다.

- ALERT_INDICATION
- FP_INDICATION
- MDS_MU_RECEIVED
- NMVT_RECEIVED
- SEND_MDS_MU
- TRANSFER_MS_DATA

진입점

퍼스널 통신이나 통신 서버는 관리 서비스 명령을 처리하는 라이브러리 파일을 제공합니다.

관리 서비스 명령은 단순 언어 인터페이스를 가집니다. 사용하는 프로그램은 *명령 제어 블럭*이라고 하는 메모리 블럭 내의 필드를 채웁니다. 그런 후, 프로그램은 진입점을 호출하여 명령 제어 블럭에 포인터를 전달합니다. 프로그램 작업이 완료되면, 관리 서비스(MS) API가 명령 제어 블럭 내의 필드를 사용하고 수정한 후 반환합니다. 이로써, 프로그램은 명령 제어 블럭에서 반환되는 매개변수를 읽을 수 있습니다. 다음은 관리 서비스 명령에 대한 진입점 리스트입니다.

- ACSSVC()
- WinMS()
- WinAsyncMS()
- WinAsyncMSEx()
- WinCSV()
- WinMSCancelAsyncRequest()
- WinMSCleanup()
- WinMSStartup()

진입점에 대한 자세한 설명은 683페이지의 『제14장 관리 서비스 진입점』를 참조하십시오.

명령 제어 블럭(VCB)

프로그램 주의사항: 기본 운영 체제(BOS)는 호출하는 응용프로그램의 주소 영역에서 몇몇 하위 시스템을 실행함으로써 성능을 최적화합니다. 이는 완전히 또는 적절히 디버깅되지 않은 응용프로그램 프로그램이 국지 설명자 테이블(LDT) 설명자를 잘못 사용할 경우, 운영상의 문제나 시스템 장애를 초래할 수 있음을 의미합니다. 따라서, 응용프로그램 프로그램은 포인터의 LDT 선택자 필드를 변경하게 되는 포인터 산술 연산을 수행해서는 안됩니다.

명령 제어 블럭(VCB)에 사용되는 세그멘트는 읽기/쓰기 데이터 세그먼트라야 합니다. 사용하는 프로그램은 VCB를 프로그램에서 하나의 변수로서 선언하거나, VCB를 할당하거나, 보다 큰 세그먼트에서 VCB를 서브할당할 수 있습니다. VCB는 프로그램이 발행하는 명령의 모든 필드를 포함할 수 있을만큼 충분히 커야 합니다.

응용프로그램 프로그램은 명령이 발행될 때부터 완료될 때까지 명령 제어 블럭의 어떠한 부분도 변경해서는 안됩니다. 관리 서비스는 명령 실행을 완료했을 때 수정된 전체 VCB를 다시 원래의 블럭으로 복사합니다. 따라서, 사용하는 프로그램이 명령 제어 블럭을 변수로 선언할 경우에는, 내부 절차의 스택이 아닌 정적 기억영역에서 선언하도록 하십시오.

각 VCB에 있는 모든 예약이나 비사용 필드를 0으로 채우십시오(X'00'). 사실상, 프로그램이 매개변수에 값을 지정하기 전에 전체 명령 제어 블럭을 0으로 설정하는 것이 시간상으로 보다 효율적일 수 있습니다. 예약 필드를 0으로 설정하는 것은 특히 중요합니다.

주: VCB가 읽기/쓰기가 아니거나 10바이트 이상이 아닌 경우(즉, 관리 서비스 1차 및 2차 리턴 코드를 보유할 수 있을만큼 크기 않을 경우), 관리 서비스는 여기에 액세스할 수 없으며 기본 운영 체제(BOS)가 프로세스를 비정상적으로 종료합니다. 이러한 종료는 *일반 보호 오류*, 프로세서 예외 트랩 D로 인식됩니다.

VCB가 너무 작거나 틀린 세그먼트 유형이 사용될 때, 관리 서비스는 INVALID_VERB_SEGMENT 1차 리턴 코드를 반환합니다.

관리 서비스(MS) 프로그램 작성

퍼스널 통신이나 통신 서버는 관리 서비스 명령을 처리하는 동적 링크 라이브러리(DLL) 파일을 제공합니다.

DLL은 재진입이 가능합니다. 즉, 복수의 응용프로그램 프로세스 및 스레드가 동시에 DLL을 호출할 수 있습니다.

관리 서비스 명령은 단순 언어 인터페이스를 가집니다. 사용하는 프로그램은 *명령 제어 블럭*이라고 하는 메모리 블럭 내의 필드를 채웁니다. 그런 다음, 프로그램은 관리 서비스 DLL을 호출하여 명령 제어 블럭에 포인터를 전

달합니다. 프로그램 작업이 완료되면, 관리 서비스가 VCB 내의 필드를 사용하고 수정한 후 반환합니다. 이로써, 프로그램은 명령 제어 블럭에서 반환되는 매개변수를 읽을 수 있습니다.

681페이지의 표 3은 관리 서비스 프로그램을 컴파일하고 링크하는 데 필요한 제공되는 헤더 파일과 라이브러리의 소스 모듈 사용법을 보여줍니다. 일부 헤더 파일에는 요구되는 다른 헤더 파일이 포함될 수도 있습니다.

표 3. 관리 서비스용 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL 명
WINNT & WIN95	WINMS.H	WINMS32.LIB	WINMS32.DLL
WIN3.1	WINCSV.H	WINCSV.LIB	WINCSV.DLL
OS/2	ACSSVCC.H	ACSSVC.LIB	ACSSVC.DLL

SNA API 클라이언트 지원

통신 서버에는 Windows 95, Windows NT, Windows 3.1 및 OS/2 운영체제용 클라이언트 세트가 포함됩니다. 이들 클라이언트(이 책에서는 SNA API 클라이언트라고 함)는 전체 관리 서비스 명령의 서브세트만을 지원합니다. 구체적으로는 다음과 같습니다.

- **WINMS**는 Windows 95 및 NT 클라이언트에서 지원하는 유일한 API입니다(자세한 내용은 686페이지의 『WinMS()』 참조).
- **WINCSV**는 Windows 3.1 클라이언트에서만 지원됩니다(자세한 내용은 685페이지의 『WinCSV()』 참조).
- **ACSSVC**는 SNA API OS/2 클라이언트에서만 지원됩니다(자세한 내용은 684페이지의 『ACSSVC()』 참조).

다음은 지원되는 관리 서비스 명령 리스트입니다.

- TRANSFER_MS_DATA
- SEND_MDS_MU

제14장 관리 서비스 진입점

이 장에서는 관리 서비스 명령의 진입점에 대해 설명합니다.

ACSSVC()

이 함수는 SNA API OS/2 클라이언트에 대해 지원되는 유일한 진입점입니다.

이 함수는 OS/2 SNA API 클라이언트에서 다음 관리 서비스 API 명령을 발행하기 위한 동기 진입점입니다.

구문

```
void ACSSVC(long);
```

입력은 명령 제어 블럭 포인터입니다.

리턴 값

1차 및 2차 리턴 코드에서 리턴 값을 점검하십시오.

WinCSV()

이 함수는 Windows 3.1 클라이언트에서 지원되는 유일한 진입점입니다.

이 함수는 CSV API에 동기 진입점을 제공합니다.

구문

```
void WINAPI WinCSV(long vcb)
```

매개변수

	설명
vcb	명령 제어 블럭에 대한 포인터.

리턴 값

리턴 값이 없습니다. 명령 제어 블럭에 있는 **primary_rc** 및 **secondary_rc** 필드는 오류를 지시합니다.

WinMS()

WinMSCleanup()

이 함수는 관리 서비스 API에서 응용프로그램을 종료하고 등록취소합니다.

구문

```
BOOL WINAPI WinMSCleanup(void);
```

리턴 값

리턴 값은 등록이 취소되었는지의 여부를 지정합니다. 값이 0이 아닐 경우, 응용프로그램 등록이 취소된 것입니다. 0 값이 반환되면, 응용프로그램 등록이 취소되지 않은 것입니다.

사용시 주의사항

관리 서비스 API로부터의 관리 서비스 응용프로그램 등록취소를 지시하려면, **WinMSCleanup()**을 사용하십시오.

WinMSCleanup은 **WinMSGetIndication** 에서 대기중인 스레드를 블럭해제합니다. 이들은 WMSNOTREG(응용프로그램이 표시를 수신하도록 등록되어 있지 않습니다)와 함께 반환됩니다. **WinMSCleanup**은 응용프로그램을 모든 표시에 대해 등록해제합니다. **WinMSCleanup**은 모든 미결 명령(동기 또는 비동기)과 함께 AP_CANCELLED 오류를 반환합니다. 단, 이 명령은 노드 내부에서 완료됩니다.

WinMSStartup 및 **WinMSCleanup**을 반드시 사용해야 할 필요는 없습니다. 단, 응용프로그램이 이들 호출을 사용하는 데 있어서 일관성을 유지해야 합니다. 이들 둘다를 사용하거나 둘다를 사용하지 말아야 합니다.

주: **WinMSStartup()** 역시 참조하십시오.

WinMSStartup()

이 함수는 응용프로그램이 요구되는 관리 서비스 API의 버전을 지정하고 제품이 지원하는 API의 버전을 검색할 수 있도록 합니다. 이 함수는 응용프로그램이 다른 관리 서비스 API 호출을 발행하여 자신을 등록하기 전에 호출할 수 있습니다.

구문

```
int WINAPI WinMSStartup(WORD wVersionRequired,
                        LPWMSDATA msdata);
```

매개변수

설명

wVersionRequired

요구되는 관리 서비스 API 지원의 버전을 지정합니다. 고차(high-order) 바이트는 부 버전(개정판) 번호를 지정하고, 하위(low-order) 바이트는 주 버전 번호를 지정합니다.

msdata

관리 서비스 API 버전 및 관리 서비스 구현 설명을 반환합니다.

리턴 값

리턴 값은 응용프로그램이 등록되었는지의 여부와 관리 서비스 API 구현이 지정된 버전 번호를 지원할 수 있는지의 여부를 지정합니다. 값이 0이면, 응용프로그램이 등록되었고 지정된 버전을 지원할 수 있음을 의미합니다. 값이 0이 아닌 경우, 리턴 값은 다음 값 중 하나입니다.

WMSSYSERROR

기본 네트워크 하위 시스템이 네트워크 통신 준비가 되어 있지 않습니다.

WMSVERNOTSUPPORTED

이 특정 관리 서비스 API 구현에서 요구되는 관리 서비스 API 지원 버전을 제공하지 않습니다.

WMSBADPOINTER

msdata 매개변수가 틀립니다.

사용시 주의사항

WinMSStartup은 앞으로 나오게 될 API 버전과의 호환성을 위한 것입니다. 지원되는 현재 버전은 1.0입니다.

WinMSStartup 및 **WinMSCleanup**을 반드시 사용해야 할 필요는 없습니다. 단, 응용프로그램이 이들 호출을 사용하는 데 있어서 일관성을 유지해야 합니다. 이들 둘다를 사용하거나 둘다를 사용하지 말아야 합니다.

주: **WinMSCleanup()**도 참조하십시오.

WinMSRegisterApplication()

이 함수는 응용프로그램을 NMVT 레벨 응용프로그램, MDS 레벨 어플리케이션 또는 경고 처리기로 등록합니다. 이러한 등록은 응용프로그램이 수신하는 요청되지 않은 표시를 결정합니다.

- NMVT 레벨 응용프로그램은 NMVT_RECEIVED 표시를 수신합니다.
- MDS 레벨 응용프로그램은 MDS_MU_RECEIVED 표시를 수신하며, 중재점이 변경될 때 FP_NOTIFICATION 표시를 수신합니다.
- 경고 처리기는 ALERT_INDICATION 표시를 수신합니다.

주: 또한, MDS MU로 변환되는 NMVT를 수신하도록 등록할 수도 있습니다.

이러한 표시를 처리하지 않는 응용프로그램은 **WinMSRegisterApplication**을 호출할 수 없습니다.

구문

```
BOOL WINAPI WinMSRegisterApplication(unsigned short reg_type,
                                     unsigned char  *ms_appl_name,
                                       unsigned short vector_key,
                                       unsigned char  mds_conv_reqd,
                                       unsigned char  *ms_category,
                                       unsigned short max_rcv_size,
                                       unsigned char  alert_dest,
                                     unsigned short *primary_rc,
                                     unsigned long  *secondary_rc);
```

매개변수

설명

reg_type

등록 유형

```
WMSNMVTAPP          NMVT-level application
                   (or MDS-level application
                    registering to receive NMVTs)
WMSMDSAPP           MDS-level application
WMSALERTHANDLER     Alert handler
```

ms_appl_name

관리 서비스 응용프로그램명. 유효한 이름은 필요시 후미 공백((X'40') 문자로 채워지는 8바이트 영숫자 1134 유형 EBCDIC 문자열이거나, *SNA 관리 서비스 참조 사항*의 부록 D에 지정된 관리 서비스 discipline-specific 응용프로그램 프로그램 중 하나(필요시, 후미 공백(X'40') 문자로 채워짐)가 될 수 있습니다.

이 이름은 **reg_type**이 WMSNMVTAPP 또는 WMSMDSAPP일 때 사용됩니다. **reg_type**이 WMSALERTHANDLER일 경우에는 이 이름을 사용할 수 없습니다.

vector_key

응용프로그램 허용값에서 받아들이는 관리 서비스 주 벡터 키는 다음과 같습니다.

```
X'YYYY'             specific major vector key
AP_SPCF_KEYS        major vector keys X'8061'
                    through X'8064'
AP_ALL_KEYS         all major vector keys
```

이 키는 **reg_type**이 WMSNMVTAPP일 때 사용됩니다. **reg_type**이 WMSMDSAPP 또는 WMSALERTHANDLER일 경우에는 이 이름을 사용할 수 없습니다.

mds_conv_reqd

등록 응용프로그램이 MDS 레벨인지의 여부와 이 응용프로그램으로 송신되는 NMVT를 MDS MU로 변환해야 하는지의 여부를 지정합니다.

(AP_YES or AP_NO)

이 매개변수는 **reg_type**이 WMSNMVTAPP일 때 사용됩니다. **reg_type**이 WMSMDSAPP 또는 WMSALERTHANDLER일 경우에는 이 매개변수를 사용할 수 없습니다.

ms_category

응용프로그램이 관리 서비스 범주의 중재점과 관련된 정보를 원할 때 그 범주를 지정합니다. 관리 서비스 범주는 *SNA 관리 서비스 참조사항* 부록 D의 관리 서비스 discipline-specific 응용프로그램 프로그램 표에 지정된 범주 코드 중 하나(필요시, 후미 공백(X'40') 문자로 채워짐)이거나 사용자 정의 범주일 수 있습니다. 사용자 정의 범주명은 8바이트 영숫자 1134 유형 EBCDIC 문자열로서, 필요시 후미 공백(X'40') 문자로 채워집니다.

이 매개변수는 **reg_type**이 WMSMDSAPP일 때 사용됩니다. **reg_type**이 WMSNMVTAPP 또는 WMSALERTHANDLER일 경우에는 이 매개변수를 사용할 수 없습니다.

max_rcv_size

응용프로그램이 하나의 청크(chunk)에서 수신할 수 있는 최대 바이트 수. 이 크기보다 큰 MDS MU가 세그먼트화되며, 각 세그멘트는 개별 MDS_MU_RECEIVED 표시에서 전달됩니다.

이 매개변수는 **reg_type**이 WMSMDSAPP일 때 사용됩니다. **reg_type**이 WMSNMVTAPP 또는 WMSALERTHANDLER일 경우에는 이 매개변수를 사용할 수 없습니다.

alert_dest

응용프로그램이 모든 경고의 유일한 목적지가 되고자 하는지의 여부를 지정합니다. 이 필드가 AP_YES로 설정되면, 모든 경고가 해당 어플리케이션으로 경로지정되고 다른 곳으로는 경로지정되지 않습니다. 이 필드가 AP_NO로 설정되면, 경고는 평상시와 동일한 방식으로 해당 어플리케이션과 SNA 네트워크 전부분으로 경로지정됩니다.

이 매개변수는 **reg_type**이 WMSALERTHANDLER일 때 사용됩니다. **reg_type**이 WMSNMVTAPP 또는 WMSMDSAPP일 경우에는 이 매개변수를 사용할 수 없습니다.

primary_rc
반환됨: 1차 리턴 코드

secondary_rc
반환됨: 2차 리턴 코드

리턴 값

이 함수는 등록이 완료되었는지의 여부를 지시하는 값을 반환합니다. 값이 0이 아닐 경우, 이는 등록이 완료되었음을 의미합니다. 값이 0일 경우, 이는 등록에 실패했음을 의미합니다.

사용시 주의사항

응용프로그램은 복수의 호출을 수행하여 둘 이상의 표시 클래스를 등록할 수 있습니다.

WinMSRegisterApplication을 호출하는 어플리케이션은 그들에 대해 대기행렬화되는 표시를 수신하기 위해 **WinMSGetIndication**을 호출해야 합니다.

주: **WinMSUnregisterApplication** 및 **WinMSGetIndication**도 참조하십시오.

WinMSUnregisterApplication()

이 함수는 응용프로그램을 등록취소하여 이전의 **WinMSRegisterApplication** 호출 결과를 원상태로 되돌리고, 응용프로그램에 대한 표시가 더이상 대기 행렬화되지 않도록 합니다.

구문

```
BOOL WINAPI WinMSUnregisterApplication(unsigned short reg_type,
                                       unsigned char  *ms_appl_name,
                                       unsigned short *primary_rc,
                                       unsigned long  *secondary_rc);
```

매개변수

설명

reg_type

등록 유형. 다음 값 중 하나가 될 수 있습니다.

WMSNMVTAPP

NMVT 레벨 응용프로그램

WMSMDSAPP

MDS 레벨 응용프로그램

WMSALERTHANDLER

경고 처리기

ms_appl_name

MS 응용프로그램명. 유효한 이름은 필요시 후미 공백((X'40') 문자로 채워지는 8바이트 영숫자 1134 유형 EBCDIC 문자열이거나, *SNA 관리 서비스 참조 사항*의 부록 D에 지정된 관리 서비스 discipline-specific 응용프로그램 프로그램 중 하나(필요시, 후미 공백(X'40') 문자로 채워짐)가 될 수 있습니다.

이 매개변수는 **reg_type**이 WMSNMVTAPP 또는 WMSMDSAPP일 때 사용됩니다. **reg_type**이 WMSALERTHANDLER일 경우에는 이 매개변수를 사용할 수 없습니다.

primary_rc

반환됨: 1차 리턴 코드

secondary_rc

반환됨: 2차 리턴 코드

리턴 값

이 함수는 등록이 해제되었는지의 여부를 지시하는 값을 반환합니다. 값이 0이 아닐 경우, 이는 등록이 해제되었음을 의미합니다. 값이 0일 경우, 이는 등록해제에 실패했음을 의미합니다.

사용시 주의사항

WinMSUnregisterApplication을 호출하면, **WinMSRegisterApplication** 호출에 의한 이전의 등록이 종료됩니다. **WinMSRegisterApplication**을 여러 번 호출한 응용프로그램이 모든 등록을 취소하기 위해서는 여러 번 **WinMSUnregisterApplication**을 호출해야 합니다.

WinMSUnregisterApplication과 **WinMSCleanup**의 차이점은 다음과 같습니다.

- **WinMSUnregisterApplication**은 표시를 수신하기 위한 이전의 등록을 종료하지만, 응용프로그램이 다른 관리 서비스 API 호출(예를 들면, WinMS)을 수행하지 못하도록 하지는 않습니다.
- **WinMSCleanup**은 관리 서비스 API 사용을 종료합니다.

응용프로그램이 **WinMSUnregisterApplication**을 호출할 때, 그 응용프로그램에 대한 표시가 이미 대기행렬화되어 있을 수도 있습니다. 이러한 표시가 대기행렬화된 상태로 있으면, 응용프로그램은 **WinMSGetIndication**을 호출하여 표시를 수신하고 처리해야 합니다. 응용프로그램이 등록해제된 후에는, 그 어플리케이션에 대해 새로운 표시가 대기행렬화되지 않습니다.

주: **WinMSRegisterApplication** 및 **WinMSGetIndication**도 참조하십시오.

WinMSGetIndication()

이 함수는 응용프로그램이 요청되지 않은 표시를 수신할 수 있도록 합니다.

구문

```
int WINAPI WinMSGetIndication(long buffer,
                              unsigned short *buffer_size,
                              unsigned long  timeout);
```

매개변수

설명

buffer 표시를 수신할 버퍼에 대한 포인터.

buffer_size
버퍼 크기. 반환됨: 표시 크기.

timeout
표시를 기다리는 시간(밀리초 단위).

리턴 값

이 함수는 표시가 수신되었는지의 여부를 지시하는 값을 반환합니다.

0 표시가 반환되었습니다.

WMSTIMEOUT
표시 대기 시간종료.

WMSSYSNOTREADY
기본 네트워크 부속시스템이 네트워크 통신 준비가 되어 있지 않습니다.

WMSNOTREG
응용프로그램이 표시를 수신하도록 등록되어 있지 않습니다.

WMSBADSIZE
버퍼가 너무 작아 표시를 수신할 수 없습니다. 충분한 크기의 버퍼로 **WinMSGetIndication** 호출을 다시 발행하십시오. 표시 크기는 **buffer_size** 매개변수에서 반환됩니다.

WMSBADPOINTER
버퍼 또는 **buffer_size** 매개변수가 유효하지 않습니다.

WMSSYSERROR
예기치 못한 시스템 오류가 발생했습니다.

사용시 주의사항

이것은 방해 호출로서, 다음과 같은 상황에서 반환됩니다.

- 표시가 반환됩니다.
- 시간종료가 만기됩니다.
- 응용프로그램이 **WinMSCleanup** 호출을 발행합니다.

- 제품이 종료됩니다.
- 시스템 오류가 발생합니다.

주: **WinMSRegisterApplication** 및 **WinMSUnregisterApplication**도 참조하십시오.

제15장 관리 서비스 명령

퍼스널 통신이나 통신 서버에서 제공하는 관리 서비스 API 명령은 응용프로그램으로 하여금 경고나 MDS MU를 송신하고, 노드가 MDS 또는 NMVT 데이터를 수신하거나 경고를 발행할 때 표시를 수신할 수 있도록 합니다.

TRANSFER_MS_DATA

이 명령은 NMVT 레벨 응용프로그램이 요청되지 않은 경고를 송신하고 이전에 수신된 NMVT 요청에 응답하는 데 사용합니다.

TRANSFER_MS_DATA는 또한 MDS 레벨 응용프로그램이 요청되지 않은 경고를 송신하는 데에도 사용됩니다. 이 명령은 WinMS 호출을 사용하는 어플리케이션이 사용할 수 있습니다.

VCB 구조

```
typedef struct ms_transfer_ms_data
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    data_type;            /* Data type supplied by app */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned char    options;            /* Verb options                */
   unsigned char    reserv3;            /* reserved                    */
   unsigned char    originator_id[8];   /* Originator ID               */
   unsigned char    pu_name[8];         /* Physical unit name          */
   unsigned char    reserv4[4];         /* reserved                    */
   unsigned short   dlen;               /* Length of data              */
   unsigned char    *dptr;              /* Data                        */
} MS_TRANSFER_MS_DATA;
```

제공되는 매개변수

응용프로그램은 다음 매개변수를 제공합니다.

opcode

SV_TRANSFER_MS_DATA

data_type

포함시킬 데이터 유형을 지정합니다. 관리 서비스는 아래 설명된 것과 같이 데이터를 처리합니다. 허용되는 값은 다음과 같습니다.

SV_NMVT

데이터에 전체 NMVT 요청 단위(RU)가 포함됩니다. 데이터에 경고가 포함되고 경고가 MDS 레벨 또는 이주 레벨 중재점으로 송신될 경우, 관리 서비스는 데이터를 MDS_MU 또는 CP_MSU 형식으로 변환합니다. 응용프로그램이 NMVT_RECEIVED 신호에 응답할 때 요구되는 유형입니다.

SV_ALERT_SUBVECTORS

데이터에 경고 주 벡터에 대한 SNA 정의 형식의 관리 서비스 서브벡터가 포함됩니다. 관리 서비스는 NMVT 헤더 및 경고 주 벡터 헤더를 추가합니다. 결과적으로, 경고가 MDS 레벨 또는 이주 레벨 중재점으로 송신될 경우, 관리 서비스는 데이터를 MDS_MU 또는 CP_MSU 형식으로 변환됩니다.

SV_USER_DEFINED

데이터에 전체 NMVT 요청 단위(RU)가 포함됩니다. 관리 서비스가 항상 데이터를 로그하나 송신하지는 않습니다.

SV_PDSTATS_SUBVECTORS

데이터에 문제점 진단 통계가 포함됩니다. 관리 서비스는 항상 데이터를 로그하며, 경고 처리기가 등록되어 있으면, 관리 서비스가 ALERT_INDICATION에 데이터를 포함시켜 경고 처리기로 송신합니다.

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

options

이 명령에서 제공되는 데이터에 대한 선택적 처리를 지정합니다. **data_type**과 지정한 옵션이 상충될 경우, 관리 서비스는 주로 지정한 **type**에 따라 데이터를 처리합니다. 이 매개변수는 1바이트 값이며, 개별 비트 설정은 선택된 옵션을 지시합니다. 모든 옵션을 지정한 경우에는, 이 바이트를 0으로 설정하십시오.

비트 0은 최대 유효 비트이고, 비트 7은 최소 유효 비트입니다.

(비트 1–3은 **data_type**이 SV_USER_DEFINED로 설정될 경우 무시됩니다.)

비트 0: 0으로 설정될 경우, 데이터에 날짜/시간(X'01') 서브벡터를 추가합니다.

비트 1: 0으로 설정될 경우, 데이터에 제품 설정 ID(X'10') 서브벡터를 추가합니다. 응용프로그램이 이미 제품 설정 ID가 포함된 데이타를 제공할 경우, 관리 서비스는 기존의 제품 설정 ID 서브벡터 바로 앞에 퍼스널 통신이나 통신 서버의 제품 설정 ID 서브벡터를 추가합니다.

비트 2: 0으로 설정될 경우, SNA 세션에서 데이터를 송신합니다. 데이터에 경고가 포함되지 않을 경우, 관리 서비스는 생략시 SSCP-PU 세션에서 데이터를 송신합니다. 데이터에 경고가 포함될 경우, 관리 서비스는 퍼스널 통신이나 통신 서버가 경고 중재점에 경고를 전송할 때 사용하는 세션 유형에 따라 SSCP-PU 세션, CP-CP 세션 또는 LU-LU 세션에서 데이터를 송신합니다.

비트 3: 0으로 설정될 경우, 퍼스널 통신이나 통신 서버 문제점 진단 기능을 통해 데이타를 로그합니다.

주: 다음 상수는 관리 서비스 헤더 파일에서 제공되며, 위에 지정된 개별 비트를 참조합니다.

- SV_TIME_STAMP_SUBVECTOR
- SV_PRODUCT_SET_ID_SUBVECTOR
- SV_SEND_ON_SESSION
- SV_LOCAL_LOGGING

비트 4–7: 예약됩니다.

originator_id
명령을 발행한 구성요소의 이름. 이 이름은 국지로 표시가능한 문자 세트로 된 8바이트 문자열입니다. 이 필드는 관리 서비스가 TRANSFER_MS_DATA를 로그할 때에만 사용됩니다.

pu_name
데이터를 송신할 물리 장치(PU)의 이름. 오른쪽이 EBCDIC 공백으로 채워지는 8바이트 영숫자 A 유형 EBCDIC 문자열로 설정하거나, **pu_name**을 지정하지 않은 경우, 모두 2진 0으로 설정해야 합니다. TRANSFER_MS_DATA를 사용하여 NMVT_RECEIVED 신호에 응답하는 응용프로그램은 NMVT_RECEIVED 신호에서 수신되는 **pu_name**을 지정해야 합니다. **pu_name**을 지정하지 않는 SV_NMVT 유형의 TRANSFER_MS_DATA 신호에 포함된 데이터는 생략시 PU 세션(사용가능한 경우)을 통해 송신됩니다. 경고를 포함하는 TRANSFER_MS_DATA 신호는 응용프로그램이 경고 데이터가 특정 PU로 송신되도록 명시적으로 지정하지 않는 한 **pu_name**을 지정하지 않습니다. 이로 인해, 일반 관리 서비스 경고 경로지정 알고리즘이 바이패스됩니다.

dlen 데이터 길이.

dptr 데이터에 대한 포인터. 널(null)로 설정하면, 관리 서비스는 데이타가 VCB에 연결되는(그리하여 VCB 바로 다음부터 시작되는) 것으로 간주합니다.

반환되는 매개변수

명령이 실행되면, 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
SV_INVALID_DATA_TYPE

SV_DATA_EXCEEDS_RU_SIZE
AP_INVALID_PU_NAME

상태 오류로 인해 명령이 실행되지 않으면, 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc

SV_SSCP_PU_SESSION_NOT_ACTIVE

시스템 오류로 인해 명령이 실행되지 않으면, 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

MDS_MU_RECEIVED

다음 경우에 관리 서비스는 이 명령 표시를 등록된 MDS 레벨 어플리케이션으로 송신합니다.

- 피어 MDS 레벨 응용프로그램에서 MDS_MU가 수신되었을 때
- NMVT가 수신되었을 때
  - 해당 NMVT 레벨 응용프로그램이 등록되어 있지 않을 때
  - MDS 레벨 응용프로그램이 입력 NMVT의 관리 서비스 주 벡터 키에서 전송되는 이름과 일치하는 이름으로 등록되어 있을 때(관리 서비스는 NMVT를 MDS_MU로 변환합니다).

VCB 구조

```
typedef struct ms_mds_mu_received
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned char    first_message;    /* First message for curr MDS_MU */
   unsigned char    last_message;     /* Last message for curr MDS_MU  */
   unsigned char    pu_name[8];         /* Physical unit name          */
   unsigned char    reserv3[8];       /* reserved                    */
   unsigned short   mds_mu_length;    /* Length of incoming MDS_MU     */
   unsigned char    *mds_mu;          /* MDS_MU data                 */
} MS_MDS_MU_RECEIVED;
```

제공되는 매개변수

opcode

AP_MDS_MU_RECEIVED

format

VCB의 형식을 식별합니다. 이 필드는 0으로 설정되어 위에 나열된 VCB의 버전을 지정합니다.

first_message

MDS_MU에 대한 첫번째 메세지인지의 여부를 지시하는 플래그(AP_YES 또는 AP_NO). **WinMSRegisterApplication** 호출에 지정한 **max_rcv_size**가 전달되는 MDS_MU의 길이보다 작을 경우, MDS_MU는 청크(chunk) 단위로 응용프로그램에 송신됩니다.

last_message

MDS_MU에 대한 마지막 메세지인지의 여부를 지시하는 플래그(AP_YES 또는 AP_NO).

pu_name

NMVT(MDS_MU로 변환된)의 발신지인 물리 장치(PU)의 이름. 응용프로그램이 입력 NMVT에 응답해야 합니다. 응용프로그램은 SEND_MDS_MU를 사용하여 응답을 송신합니다. 응답을 송신할 때, 어플리케이션은 SEND_MDS_MU의 **pu_name** 필드를

MDS_MU_RECEIVED 신호에서 제공되는 **pu_name**으로 설정해야 합니다. MDS_MU가 MDS 레벨 전송 메카니즘에서 수신된 경우, **pu_name**은 모두 2진 0으로 설정됩니다.

mds_mu_length

신호에 포함되는 MDS_MU 부분의 길이.

mds_mu

MDS_MU 데이터. 데이터 포인터는 널(null)로 설정되며, 데이터는 VCB에 연결됩니다(VCB 바로 다음에 시작됩니다).

SEND_MDS_MU

이 명령은 MDS 레벨 응용프로그램이 **WinMS** 집입점을 사용하여 경고 이외의 네트워크 관리 데이터를 송신할 때 사용합니다. MDS_MU를 목적지 응용프로그램에 송신할 때 오류가 발생하면, 두 가지 방법 중 하나로 오류가 발신 응용프로그램에 보고됩니다. 국지 노드에서 오류가 검출된 경우, SEND_MDS_MU 응답의 리턴 코드를 통해 응용프로그램에 통지됩니다. 원격 노드에서 오류가 검출된 경우에는, MDS_MU_RECEIVED VCB에서 전송되는 MDS_MU 오류를 통해 오류가 보고됩니다. SSCP-PU 세션을 통해 목적지 노드에 도달해야 할 경우, 관리 서비스는 전송 MDS_MU를 NMVT로 변환할 수 있습니다. 응용프로그램은 국지 노드의 ID를 몰라도 됩니다. 응용프로그램이 **netid**나 **nau**에, 또는 MDS 경로지정 정보 GDS 변수의 발신 위치명 서브벡터의 두 서브필드에 8개의 EBCDIC 공백을 제공하면, 퍼스널 통신이나 통신 서버가 해당 값을 제공합니다. 응용프로그램이 **netid** 또는 **nau**를 채우지 않고 8개 미만의 공백을 제공하면, 퍼스널 통신이나 통신 서버는 AP_INVALID_MDS_MU_FORMAT의 2차 리턴 코드를 반환합니다.

VCB 구조

```
typedef struct ms_send_mds_mu
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned char    options;            /* Verb options                */
   unsigned char    reserv3;            /* reserved                    */
   unsigned char    originator_id[8];   /* Originator ID               */
   unsigned char    pu_name[8];         /* Physical unit name          */
   unsigned char    reserv4[4];         /* reserved                    */
   unsigned short   dlen;               /* Length of data              */
   unsigned char    *dptr;              /* Data                        */
} MS_SEND_MDS_MU;
```

제공되는 매개변수

opcode

AP_SEND_MDS_MU

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

options

이 명령에서 제공되는 데이터에 대한 선택적 처리를 지정합니다. 이 매개변수는 1바이트 값이며, 개별 비트 설정은 선택된 옵션을 지시합니다. 모든 옵션을 지정한 경우에는, 이 바이트를 0으로 설정하십시오.

비트 0은 최대 유효 비트이고, 비트 7은 최소 유효 비트입니다.

비트 0: 0으로 설정될 경우, 데이터에 날짜/시간(X'01') 서브벡터를 추가합니다.

비트 1: 0으로 설정될 경우, 데이터에 제품 설정 ID(X'10') 서브벡터를 추가합니다. 응용프로그램이 이미 제품 설정 ID 서브벡터가 포함된 데이타를 제공할 경우, 관리 서비스는 기존의 제품 설정 ID 서브벡터 바로 앞에 퍼스널 통신이나 통신 서버의 제품 설정 ID 서브벡터를 추가합니다.

비트 2: 예약됩니다.

비트 3: 0으로 설정될 경우, 퍼스널 통신이나 통신 서버 문제점 진단 기능을 통해 데이타를 로그합니다.

주: 다음 상수는 관리 서비스 헤더 파일에서 제공되며, 위에 지정된 비트 0, 1 및 3을 참조합니다.

- SV_TIME_STAMP_SUBVECTOR
- SV_PRODUCT_SET_ID_SUBVECTOR
- SV_LOCAL_LOGGING

비트 4: 관리 서비스가 목적지 응용프로그램에 관리 서비스 데이타를 송신할 때 생략시 경로지정을 사용하는지 또는 직접 경로지정을 사용하는지를 지정합니다(AP_DEFAULT 또는 AP_DIRECT).

주: 비트 4를 설정하려면, 적절히 시프트된 AP_DEFAULT 또는 AP_DIRECT를 사용하십시오(예를 들면, AP_DIRECT<<3).

비트 5–7: 예약됩니다.

originator_id

명령을 발행한 구성요소의 이름. 이 필드는 관리 서비스가 SEND_MDS_MU를 로그할 때에만 사용됩니다.

pu_name

데이터를 송신할 물리 장치(PU)의 이름. 오른쪽이 EBCDIC 공백으로

primary_rc
AP_OK

매개변수 오류로 인해 명령이 실행되지 않으면, 프로그램 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_PU_NAME

AP_INVALID_MDS_MU_FORMAT
SV_INVALID_DATA_SIZE

상태 오류로 인해 명령이 실행되지 않으면, 프로그램 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_SSCP_PU_SESSION_NOT_ACTIVE

시스템 오류로 인해 명령이 실행되지 않으면, 프로그램 관리 서비스는 다음 매개변수를 반환합니다.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

ALERT_INDICATION

이 명령 표시는 관리 서비스가 경고 주 벡터를 그들을 처리하는 등록된 경고 처리기나 등록된 held 경고 처리기에 송신할 때 사용합니다.

VCB 구조

```
typedef struct ms_alert_indication
{
   unsigned short   opcode;             /* AP_AlERT_INDICATION         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned short   alert_length;       /* Length of alert             */
   unsigned char    reserv3[6];         /* reserved                    */
   unsigned char    *alert;             /* Alert data                  */
} MS_ALERT_INDICATION;
```

제공되는 매개변수

opcode

AP_ALERT_INDICATION

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

alert_length

경고 데이터의 길이.

alert 경고 데이터에 대한 포인터. 데이터 포인터는 널(null)로 설정되며, 데이터는 VCB에 연결됩니다(VCB 바로 다음에 시작됩니다).

FP_NOTIFICATION

특정 관리 서비스 범주와 이러한 범주가 변경될 때의 중재점 상태에 대해 MDS 레벨 응용프로그램이 등록되어 있으면, 관리 서비스는 이러한 응용프로그램에 이 명령 신호를 송신합니다.

VCB 구조

```
typedef struct ms_fp_notification
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned char    fp_routing;         /* Type of routing to focal pt */
   unsigned char    reserv1;            /* reserved                    */
   unsigned short   fp_data_length;     /* Length of incoming focal    */
                                        /* point data                  */
   unsigned char    *fp_data;           /* focal point data            */
} MS_FP_NOTIFICATION;
```

제공되는 매개변수

opcode

AP_FP_NOTIFICATION

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

fp_routing

중재점에 메세지를 송신할 때 SEND_MDS_MU에 지정되는 경로지정 유형(AP_DEFAULT 또는 AP_DIRECT).

fp_data_length

중재점 데이터의 길이.

fp_data

중재점 통지(X'E1') 서브벡터와 중재점 식별명(ID)(X'21') 서브벡터가 있는 중재점 데이터. 이 데이터 포인터는 널(null)로 설정되며, 데이터는 VCB에 연결됩니다(VCB 바로 다음에 시작됩니다).

NMVT_RECEIVED

이 명령 신호는 원격 노드에서 NMVT가 수신될 때 관리 서비스가 등록된 NMVT 레벨 응용프로그램에 송신합니다.

관리 서비스는 입력 NMVT의 경로지정에 다음 규칙을 적용합니다.

1. 입력 NMVT에서 전송되는 주 벡터 키로 등록된 NMVT 레벨 어플리케이션으로 경로지정합니다.
2. 주 벡터 키가 X'8061'-X'8064' 중 하나일 경우, 등록된 NMVT 레벨 AP_SPCF_KEYS 응용프로그램으로 경로지정합니다.
3. 등록된 NMVT 레벨 AP_ALL_KEYS 응용프로그램으로 경로지정합니다.
4. NMVT(MDS_MU로 변환한 후)를 입력 NMVT에서 전송되는 주 벡터 키로 등록된 MDS 레벨 응용프로그램으로 경로지정합니다.
5. 주 벡터 키가 X'8061'-X'8064' 중 하나일 경우, NMVT(MDS_MU로 변환한 후)를 등록된 MDS 레벨 응용프로그램으로 경로지정합니다.
6. 등록된 AP_ALL_KEYS MDS 레벨 응용프로그램으로 경로지정합니다 (MDS_MU로 변환한 후).
7. NMVT에 부정적으로 응답합니다.

VCB 구조

```
typedef struct ms_nmvt_received
{
   unsigned short   opcode;             /* Verb operation code         */
   unsigned char    reserv2;            /* reserved                    */
   unsigned char    format;             /* format                      */
   unsigned short   primary_rc;         /* Primary return code         */
   unsigned long    secondary_rc;       /* Secondary return code       */
   unsigned char    pu_name[8];         /* Physical unit name          */
   unsigned char    reserv3[6];         /* reserved                    */
   unsigned short   nmvt_length;        /* Length of incoming NMVT     */
   unsigned char    *nmvt;              /* NMVT data                   */
} MS_NMVT_RECEIVED;
```

제공되는 매개변수

opcode

AP_NMVT_RECEIVED

format

VCB의 형식을 식별합니다. 위에 나열된 VCB의 버전을 지정하려면, 이 필드를 0으로 설정하십시오.

pu_name

NMVT를 송신한 물리 장치(PU)의 이름. 응용프로그램이 입력 NMVT에 응답해야 합니다. 응용프로그램은 TRANSFER_MS_DATA를 사용하여 응답을 송신합니다. 응답을 송신할 때, 응용프로그램은 TRANSFER_MS_DATA의 **pu_name** 필드를 NMVT_RECEIVED 신호에서 제공되는 **pu_name**으로 설정해야 합니다.

NMVT_RECEIVED

nmvt_length
NMVT 데이터의 길이.

nmvt REGISTER_NMVT_APPLICATION에 지정된 유형의 관리 서비스 주 벡터가 있는 전체 NMVT. 이 데이터 포인터는 널(null)로 설정되며, 데이터는 VCB에 연결됩니다(VCB 바로 다음에 시작됩니다).

부록A. IBM APPN MIB 표

다음 표는 RFC1593에 정의된 바와 같이 IBM APPN 관리 정보 블럭(MIB)의 표 구현에 대한 세부사항을 제공합니다. 이 표에서는 다음을 다음을 정의합니다.

- 각 MIB 표를 구현하는 데 사용되는 노드 연산자 기능 QUERY 명령.
- 입력 매개변수 설정
- 요구되는 필터링 작업

(반환되는 매개변수와 MIB 표 변수 사이의 매핑은 노드 연산자 기능 QUERY 명령 정의에서 추출할 수 있습니다). 퍼스널 통신이나 통신 서버는 현재 ibmappnNodePortDlcTraceTable 및 ibmappnLsStatusTable MIB 표를 지원하지 않습니다.

IBM MIB 표	**노드 연산자 기능 명령 및 MIB 표 변수**	**입력 매개변수 설정**
ibmappnNodePortTable	QUERY_PORT	**port_name** ibmappnNodePortName
ibmappnNodePortIpTable	(주 1)	
ibmappnNodePortDlsTable	QUERY_PORT (**dlc_type**이 AP_SDLC인입력항목 선택)	**port_name**ibmappnNodePortDlsName
ibmappnNodePortTrTable	QUERY_PORT	**port_name** ibmappnNodePortTrName
ibmappnNodeLsTable	QUERY_LS	**ls_name** ibmappnNodeLsName
ibmappnNodeLsIpTable	(주 1)	
ibmappnNodeLsDlsTable	QUERY_LS (**dlc_type**이 AP_SDLC인입력항목 선택)	**ls_name** ibmappnNodeLsDlsName
ibmappnNodeLsTrTable	QUERY_LS	**ls_name** ibmappnNodeLsTrName
ibmappnNnTopoRouteTable	QUERY_COS	**cos_name** ibmappnNnTopoRouteCos
ibmappnNnAdjNodeTable	QUERY_ADJACENT_NN	**a d j _ n n c p _ n a m e** ibmappnNnAdjNodeAdjName
ibmappnNnTopologyTable	QUERY_NN_TOPOLOGY_NODE	**node_name** ibmappnNnNodeName **node_type** AP_LEARN_NODE **frsn** 0

IBM MIB 표	**노드 연산자 기능 명령 및 MIB 표 변수**	**입력 매개변수 설정**
ibmappnNnTgTopologyTable	QUERY_NN_TOPOLOGY_TG	**owner** ibmappnNnTgOwner **owner_type** AP_LEARN_NODE **dest** ibmappnNnTgDest **dest_type** AP_LEARN_NODE **tg_num** ibmappnNnTgNum **frsn** 0
ibmappnNnTopologyFRTable	QUERY_NN_TOPOLOGY_NODE	**node_name** ibmappnNnFRNode **node_type** AP_LEARN_NODE **frsn** ibmappnNnFRFrsn
ibmappnNnTgTopologyFRTable	QUERY_NN_TOPOLOGY_TG	**owner** ibmappnNnTgFROwner **owner_type** AP_LEARN_NODE **dest** ibmappnNnTgFRDest **dest_type** AP_LEARN_NODE **tg_num** ibmappnNnTgFRNum **frsn** ibmappnNnTgFRFrsn
ibmappnLocalTgTable	QUERY_LOCAL_TOPOLOGY	**dest** ibmappnLocalTGDest **dest_type** AP_LEARN_NODE **tg_num** ibmappnLocalTgNum

IBM MIB 표	노드 연산자 기능 명령 및 MIB 표 변수	입력 매개변수 설정
ibmappnLocalEnTable	QUERY_LOCAL_TOPOLOGY (고유한 **dest**를 가지는입력항목 선택)(주 2)	**dest** ibmappnLocalEnName **dest_type** AP_END_NODE **dest_type** AP_LEARN_NODE
ibmappnLocalEnTgTable	QUERY_LOCAL_TOPOLOGY(주 3)	**dest** ibmappnLocalEnTgOrigin **dest_type** AP_LEARN_NODE **tg_num** ibmappnLocalEnTgNum
ibmappnDirTable	QUERY_DIRECTORY_LU	**lu_name** ibmappnDirLuName
ibmappnCosModeTable	QUERY_MODE_TO_COS_MAPPING	**mode_name** ibmappnCosModeName
ibmappnCosNameTable	QUERY_COS	**cos_name** ibmappnCosName

주:

1. 퍼스널 통신이나 통신 서버는 DLC 유형으로 IP를 지원하지 않습니다.
2. **dest**가 동일한 입력항목은 QUERY_LOCAL_TOPOLOGY를 기준으로 연속적으로 순서지정됩니다.
3. ibmappnLocalEnTgTable은 접속된 끝 노드의 관점에서 TG를 봅니다(즉, 끝 노드의 TG로서). 단, 현재 레벨의 APPN 구조와 호환되는 네트워크 노드만이 직접 접속된 끝 노드와의 TG에 대한 끝 노드 TG 정보를 저장합니다. 따라서, 이 표에 있는 모든 입력항목의 ibmappnLocalEnTgDest는 국지 노드의 이름(ibmappnNodeCpName)으로 설정됩니다.

부록B. 주의사항

이 책은 미국에서 제공되는 제품과 서비스에 대한 것입니다. 다른 나라에서는 이 책에 설명된 제품이나 서비스 또는 기능들이 제공되지 않을 수도 있습니다. 해당 지역에서 현재 구입이 가능한 제품 및 서비스에 대해서는 해당 지역의 IBM 영업대표에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM 제품, 프로그램 또는 서비스만을 사용해야 한다는 의미는 아닙니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 타사의 제품, 프로그램 또는 서비스를 IBM 제품, 프로그램 또는 서비스 대신 사용할 수 있습니다. 단, 타사 제품, 프로그램 또는 서비스와의 조작에 대한 평가나 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 주제에 대한 특허를 보유하고 있거나 현재 출원중일 수 있습니다. 이 책자를 제공했다고 해서 이러한 특허권을 부여하는 것은 아닙니다. 특허 사용권에 대한 문의는 한국 IBM 지적 재산권부(02-781-6028)로 하시기 바랍니다.

다음 사항은 영국이나 지역법에 위배되는 국가에는 적용되지 않@니다. IBM사는 본 출판물을 매매성 또는 특정 목적에의 적합성 등(단, 이에 국한되지 않음) 명시적이거나 상정적인 어떠한 보증 없이 『그 자체로서』 제공합니다. 특정 거래에 있어서 명시적이거나 상정적인 보증의 거부를 인정하지 않는 일부 국가에서는 이러한 사항이 적용되지 않습니다.

이 책에는 기술상으로 정확하지 않은 정보나 인쇄상의 오류가 있을 수 있습니다. 이 책의 내용은 주기적으로 변경되며, 변경사항은 개정판에 수록됩니다. IBM은 이책에 기술된 제품이나 프로그램을 아무런 통보 없이 언제라도 수정할 수 있습니다.

(1) 독자적으로 작성된 프로그램과 기타 프로그램(이 프로그램을 포함하여) 간의 정보 교환이나 (2) 교환된 정보의 항호 이용을 목적으로 이 프로그램에 대한 정보를 필요로 하는 사용권자는 한국 IBM 소프트웨어 사업본부(02-781-7777)로 문의하십시오.

이러한 정보는 해당 조건에 따라 사용이 가능하며, 특정 경우에는 요금을 지불해야 합니다.

이 책에 기술된 공인 프로그램과 이 공인 프로그램에 사용할 수 있는 공인된 모든 제품은 IBM 고객 동의서, 국제 프로그래밍 사용권 동의서 또는 그와 동등한 협약 조건 하에 IBM에서 제공합니다.

타사의 제품에 관한 정보는 제품 공급자나 출판된 책자 또는 사용가능한 기타 소스로부터 얻은 것입니다. IBM은 이들 제품을 검사하지 않았으므로, 성능의 정확성이나 호환성 또는 타사 제품과 관련한 기타 여러 가지 요구들을 확인할 수 없습니다. 타사 제품의 호환성에 대해서는 제품 공급자에게 문의하십시오.

저작권 사용권

이 책에는 다양한 운영 플랫폼에서의 프로그래밍 방법을 설명하는 원본 언어로 된 샘플 응용프로그램 프로그램이 들어 있습니다. 사용자는 샘플 프로그램이 작성된 운영 플랫폼의 응용프로그램 프로그램 인터페이스(API)에 일치하는 응용프로그램 프로그램을 개발, 사용, 판매 또는 분배할 목적으로, IBM에 비용을 지불하지 않고 사용자가 원하는 형식으로 이들 샘플 프로그램을 복사, 수정 및 분배할 수 있습니다. 이들 예제는 모든 조건 하에서 철저하게 검사된 것이 아닙니다. 따라서 IBM은 이들 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 함축할 수 없습니다. 사용자는 IBM의 응용프로그램 프로그래밍 인터페이스(API)에 일치하는 어플리케이션 프로그램을 개발, 사용, 판매 또는 분배할 목적으로, IBM에 비용을 지불하지 않고 사용자가 원하는 형식으로 이들 샘플 프로그램을 복사, 수정 및 분배할 수 있습니다.

샘플 프로그램의 각 사본이나 특정 부분 또는 여기서 파생된 내용은 다음과 같은 저작권 알림사항을 포함시켜야 합니다.

부록C. 등록상표

다음 용어는 미국과 그 밖의 나라에서 사용되는 IBM의 등록상표입니다.

ACF/VTAM
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
APPN
AS/400
AT
CICS
Common User Access
CUA
IBM
IMS
MVS
MVS/ESA
MVS/XA
NetView
Operating System/2
OS/2
OS/400
RACF
System/370
Virtual Machine/Enterprise Systems Architecture
VM/ESA
VTAM

기타 회사, 제품 및 서비스명(이중 별표(**)가 표시될 수도 있음)은 타사의 등록상표이거나 서비스 표시입니다.

C-bus는 Corollary사의 등록상표입니다.

ActionMedia, LANDesk, MMX, Pentium 및 ProShare는 미국과 그 밖의 나라에서 사용되는 Intel사의 등록상표입니다.

Java 및 HotJava는 Sun Microsystems사의 등록상표입니다.

Microsoft, Windows, Windows NT 및 Windows 95 로고는 Microsoft사의 등록상표입니다.

UNIX는 미국과 그 밖의 나라에서 사용되며 X/Open사를 통해 독점 공인된 등록상표입니다.

PC Direct는 Ziff Communications사의 등록상표이며 사용권 하에 IBM사가 사용합니다.

색인

[하]

A

C

D

E

F

H

I

L

M

N

P

Q

R

S

T

W

X

SA30-0537-01