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

IBM

클라이언트/서버 통신 프로그래밍

SA30-0536-01

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

IBM

클라이언트/서버 통신 프로그래밍

SA30-0536-01

주의!

이 책z 이 책이 지x하는 제품을 사k하b 전! 423페이지의 『부록F. 주의사항』을 읽으십시@.

제2판(1998년 7월)

후속판!서 특별히 명시하b 전n지 이 책은 IBM eNetwork 통신 서버의 버전 6.0, Windows 95 및 Windows NTk 퍼스널 통신의 버전 4.2, W리m 모든 후속판 및 3정판! 적k됩니다.

목차

그림

표

이 책에 관하여

이 책은 Windows NT k IBM eNetwork 통신 서버 및 Windows 95 및 Windows NT k IBM eNetwork 퍼스널 통신 ! 서 제 x 하는 클라이 p 트 및 서버 n 플리케이션의 사 k 자를 위한 M 입니다. 클라이 p 트 API는 Windows 95 and Windows NT, Windows 3.1 및 OS/2 플랫폼 ! 제 x 됩니다.

Windows NT k IBM eNetwork 통신 서버(이 책 ! 서는 *통신 서버*로 지칭됨)는 통신 서비스 플랫폼입니다. 이 플랫폼은 호스트 컴퓨터 및 b 타 v 크스테이션 z 통신하는 Windows NT v 크스테이션 ! 다 g 한 서비스를 제 x 합니다. 통신 서버 사 k 자는) 러 ! 지 x] , a 성 I 션 중 ! 서 x 하는 M 을 선택할 수 있습니다.

Windows 95 및 Windows NT k IBM eNetwork 퍼스널 통신(이 책 ! 서는 *퍼스널 통신*으로 지칭됨)은 O 전한 b 능을 . 춘 ! 뮬레이터입니다. 호스트 단말 b ! 뮬레이션 ! 더하) 다음 z 0 은 유 k 한 b 능을 제 x 합니다.

- 파일 전송
- 동적 8 성
- 사 k 하 b 쉬 n W 래프 인터페이스
- SNA를 b 본으로 하는 클라이 p 트 n 플리케이션 ! 대한 API
- TCP/IP를 b 본으로 하는 n 플리케이션이 SNA를 b 본으로 하는 네트 v 크 ! 서 통신하도록 허 k 하는 API

대부분의 f l 퍼스널 통신 및 통신 서버 k 프로 W 램 z 이의 클라이 p 트를 3 발하는 M 은 이들 " " 이 동일한 명령 n 를 다수 지 x 한다는 점 ! 서 F 주 유사합니다. W 러나 ` # 의 차이 ! 있습니다. 이러한 차이점은 이 책 ! 서 특별한 F 이콘을 사 k 하) 표시되 n 있습니다. 자세한 내 k 은 xiii페이지의 『F 이콘』을 참조하십시 @ . 이 책 ! 서 퍼스널 통신 및 통신 서버 프로 W 램명은 정보 ! 이 둘 모두 ! 서 적 k 될 때 사 k 됩니다. 퍼스널 통신 프로 W 램 또는 통신 서버 프로 W 램 ! 만 적 k 될 f l ! 는 특정 프로 W 램명이 사 k 됩니다.

이 책은 다음의 네 부분으로 나뉘 n 있습니다.

- 제1부 APPC API ! 서는 퍼스널 통신 및 통신 서버 m ^ 프로 W 램 # 통신(APPC) 인터페이스를 사 k 하는 프로 W 램의 3 발 방법 ! 대해 설명합니다. APPC는 논리 장치(LU) 유형 6.2를 위한 시스템 네트 v 크 체 h (SNA)의 8 현을 ! 리킵니다. 이 책 ! 서 특별히 명시되지 J 는 한 APPC는 퍼스널 통신 및 통신 서버로 8 현된 APPC를 나타냅니다.

  APPC는 n 떤 처리 b 능을 수행하 b 위해 협력하는 두 3 이상의 프로 W 램 ! 서 분산 트랜잭션 프로세싱 b 능을 제 x 합니다. 이 b 능 ! 는 프로세서 사이클, 데이타베이스, 작 w 대 b 행렬, 키보드 및 표시장치 M 0 은 물리적 인터페이스 M 0 은 자 x 을 x 유할 수 있도록 하는 프로 W 램들 # 의 통신이 | 련됩니다.

- 제2부 LUA API ! 서는 SNA LU 유형 0, 1, 2 및 3 ! 대한 W 세스를 제 x 하는 IBM | k 일반 LU n 플리케이션(LUA) 인터페이스(이 책 ! 서 LUA 는 d 청 단위(RU) 인터페이스(RUI)를 지칭하 b 도 함)를 사 k 하는 프로 W 램의 3 발 방법 ! 대해 설명합니다.
- 제3부 Common Services API ! 는 Common Services API를 8 성하는 명령 n ! 나 M 있습니다.
- 제4부 EHNAPPC API ! 는 확장 APPC 인터페이스 ! 대한 함수, 8 조 및 리턴 코드 ! 나 M 있습니다.

이 책 ! 서 *Windows*는 Windows 95 및 Windows NT를 ! 리킵니다. 이 책 전반 ! I 쳐 *windows*는 지 x 되는 모든 3 인 k 컴퓨터를 ! 리킵니다. 단 한 ! 지 모델이나 8 조의 3 인 k 컴푸터를 ! 리킬 때 ! 는 W 유형만이 지정됩니다.

통신 서버의 f l b 본 n 5 체제(BOS)로 NT V4.0을 사 k 하 m 있는 M 으로 # 주합니다. 퍼스널 통신의 f l ! 는 b 본 n 5 체제(BOS)로 Windows 95 또는 Windows NT를 사 k 하 m 있는 M 으로 # 주합니다.

이 책의 대상 독자

이 책은 APPC 또는 LUA n 플리케이션을 작성하는 프로 W 래머나 3 발자들을 위한 M 입니다.

이 책 ! 서는 독자 ! *SNA Transaction Programmer's Reference Manual for LU Type 6.2*를 숙지하 m 있는 M 으로 # 주합니다.

이 책의 이용 방법

- 제1장 APPC의 소 3 ! 서는 m ^ 프로 W 램 # 통신(APPC) ! 대해 설명합니다.
- 제2장 APPC의 b 본 3 념 ! 서는 APPC 트랜잭션 프로 W 램(TP) ! 대해 설명합니다.
- 제3장 접속 | 리자 사 k 법 ! 서는 접속 | 리자의 사 k 방법 ! 대해 설명합니다.
- 제4장 트랜잭션 프로 W 램 작성 ! 서는 트랜잭션 프로 W 램(TP)의 작성 방법 ! 대해 설명합니다.
- 제5장 APPC 트랜잭션 프로 W 램의 8 현 ! 서는 APPC 확장 ! 대해 설명합니다.
- 제6장 CPI-C 프로 W 램 8 현 ! 서는 CPI-C 프로 W 램 ! 대해 설명합니다.
- 제7장 APPC # 트리 포인트 ! 서는 APPC API ! 대한 프로시듀 n # 트리 포인트 ! 대해 설명합니다.
- 제8장 APPC 명령 n ! 서는 " APPC 명령 n 의 8 문 ! 대해 설명합니다.) b ! 는 " 명령 n ! 대한 정보 ! 들 n 있는 8 조 ! 포함되 n 있으며 " 항목 ! 대한 설명이 나 B 다음 W 뒤 ! ! 능한 리턴 코드 목록이 나 - 됩니다.

- 제9장 IBM | k LU n플리케이션의 b본 3념!서는 이 책!서 사k되는 b본적인 LUA 프로W래밍 3념! 대해 설명합니다.
- 제10장 RUI LUA 명령n의 특징!서는 LUA 명령n의 특징! 대해 설명합니다.
- 제11장 LUA 프로W램 8현!서는 LUA n플리케이션 프로W램 작성의 몇!지 |점! 대해 설명합니다.
- 제12장 RUI LUA #트리 포인트!서는 LUA! 대한 프로시듀n #트리 포인트! 대해 설명합니다.
- 제13장 RUI 명령n!서는 " LUA 명령n! 대해 상세히 설명합니다.
- 제16장 x통 서비스 #트리 포인트!서는 프로시듀n #트리 포인트! 대해 설명합니다.
- 제17장 x통 서비스 명령n (CSV)!서는 x통 서비스 명령n! 대해 설명합니다.
- 제18장 EHNAPPC n플리케이션 프로W램 인터페이스!서는 EHNAPPC API! 대해 설명합니다.
- 제19장 데이타 변환 Windows n플리케이션 프로W램 인터페이스!서는 데이타 변환 창 API! 대해 설명합니다.
- 제20장 JAVAk 호스트 W세스 등^ 라이브러리 소3!서는 자바를 위한 호스트 W세스 등^ 라이브러리M 자바 등^을 사k하는 3270 및 5250z의 |h! 대해 설명합니다.
- 제21장 Javak CPIC-C 사k하b!서는 자바 API를 위한 CPI-C! 대해 설명합니다.
- 부록A. APPC x통 리턴 코드!는 x통 리턴 코드! 대한 설명이 들n 있습니다.
- 부록B. LUA 명령n 리턴 코드!는 LUA x통 리턴 코드! 대한 설명이 들n 있습니다.
- 부록C. APPC 대화 상태 전이!서는 APPC 명령n를 실행할 수 있는 대화 상태 및 명령n 0료시 발생하는 상태 변f! 대해 설명합니다.
- 부록D. 통신 서버 서비스 위치 프로토콜!서는 n플리케이션 프로W램 3발자! 서비스를 찾은 다음 TCP/IP 프로토콜을 사k하) 서비스들#의 U형을 유지하는 방법! 대해 설명합니다.
- 부록E. DLL 버전 정보!는 32비트 Windows DLL 버전 정보! 들n 있습니다.

아이콘

이 책!서는 *F이콘*을 사k하))러 !지 유형의 정보를 쉽T 찾을 수 있도록 도M줍니다.

이 F이콘은 b본 APPC 명령n! 적k되는 정보를 나타냅니다. b본 명령n! 대해서는 제8장 APPC 명령n를 참조하십시@.

이 F이콘은 직접 APPC 명령n ! 적k되는 정보를 나타냅니다. 직접 명령n ! 대해서는 제8장 APPC 명령n를 참조하십시@.

이 F이콘은 참m사항, 퍼스널 통신이나 통신 서버의 n5 ! 5향을 줄 수 있는 중d 정보 또는 타스크의 O료를 나타냅니다.

이 F이콘은 퍼스널 통신 프로W램!만 해당하는 정보일 때 나타납니다.

이 F이콘은 통신 서버 프로W램!만 해당하는 정보일 때 나타납니다.

이 책에 - 사용되는 규약

퍼스널 통신 및 통신 서버 라이브러리 전반! I쳐 다음z 0은 T`이 사k됩니다. 나-된 T` 중!는 이 책!서 사k되지 J는 M도 있습니다.

텍스트 규약

굵은체	=은체는 프로W램이나 명령 프롬프트!서 사k할 수 있는 명령n, 함수 및 매3변수를 나타냅니다. 이러한 *은 대소문자! 8분되며 텍스트! 표시된 대로 정확하T 입력해_합니다.
b o임체	b o임체는 다음을 나타냅니다. • *을 지정할 수 있는 변수. • 리스트, 체크 박스, 입력 필드, 누름 버튼 및 메뉴 선택사항 z 0은 창 제n의 이름. 이들은 창! 표시되는 대로 텍스트! 표시됩니다. • 책 제목. • 문자! W대로 사k되E나 단n ! W대로 사k되m 있습니다. (예: *a*! 나@면 이M을 *an*으로 #주하지 마십시@.)
굵은 기울임체	=은 b o임체는 단n를 -조하는 데 사k됩니다.
대문자	대문자는 명령이나 명령 프롬프트!서 사k할 수 있는 상수, 파일명, 키v드 및 I션을 나타냅니다. 이러한 *은 대문자 또는 소문자로 입력할 수 있습니다.
큰 따H표	큰 따H표는 창! 표시되는 메세지를 나타냅니다. 이의 9로 !뮬레이터 세션의 n5dx 정보 5*(OIA)! 표시되는 메세지를 들 수 있습니다.
예제 유형	9제 유형은 명령 프롬프트나 창! 입력하도록 지시되는 정보를 나타냅니다.

수 규약

2진수	BX'xxxx xxxx' 또는 BX'x'로 표시되며, 단 텍스트M 함께 표시되는 f l(『2진 xxxx xxxx의 * 은...』)는 9 \ 로 함.
비트 위치	! 장 @른쪽 위치(최하위 비트) ! 서 0으로 시작됨.
십진수	4자리 이상의 십진수는 미터 g식으로 표시됩니다. 3자리를 8분하는 데 콤마 대신 x백이 사k됩니다. 9를 들n 16,147은 16 147로 2) 집니다.
16진수	텍스트 ! 서 16진 xxxx 또는 X'xxxx'로 표시됩니다. (『인접 노드의 주소는 16진 5D이며 X'5d'로 지정됩니다. 』)

2바이트 문자 <트 지원

퍼스널 통신 및 통신 서버는 2바이트 문자 세트(DBCS)를 지x하며) b서 " 문자는 2바이트로 표시됩니다. 256 코드로 더 많은 b호를 표시할 수 있는 한[, 일본n 및 중9 n M 0은 p n ! 는 2바이트 문자 세트! 필d합니다. " 문자 ! 2바이트! 필d하므로 DBCS 문자의 입력, 표시 및 인쇄 ! 는 DBCS를 지x하는 하드~n 및 프로W램이 필d합니다.

특별히 DBCS ! 만 적k되는 정보일 f l 이 정보 단위 ! 서 명시됩니다.

이 책 ! 서 ASCII는 PC 1바이트 코드를 지칭합니다. 일본n ! 서 ASCII는 JISCII로 #주해_ 합니다.

상< 정보

자세한 내k을 K려면 통신 서버 라이브러리 및 |련 책자!
대해 자세히 설명되n 있는 *빠*른 *시작*을 참조하십시@.

통신 서버를 설치한 후 특정 책자를 보려면 데스크탑!서 다음 f로를 이k하십시@.

1. 프로W램
2. IBM 통신 서버
3. 문서
4. 책 리스트!서 선택

통신 서버 책자는 PDF(Portable Document Format)로 되n 있으며 Adobe Acrobat Reader로 -람이 !능합니다. 이 프로W램이 사k자 b h! 설치되지 J은 f l 문서 리스트!서 이를 설치할 수 있습니다.

인터넷상의 통신 서버 홈 페이지!는 APAR 및 수정사항! 대한 서비스 정보뿐 F니라 일반 제품 정보도 들n 있습니다. IBM Web ExplorerM 0은 인터넷 브라l저를 사k하) 홈 페이지! 들n!려면 다음 URL로 !십시@.

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

자세한 내k을 K려면 퍼스널 통신 라이브러리 및 |련 책자!
대해 자세히 설명되n 있는 *빠*른 *시작*을 참조하십시@.

퍼스널 통신을 설치한 후 특정 책자를 보려면, 데스크탑!서 다음 f로를 이k하십시@.

1. 프로W램
2. IBM 통신 서버
3. 문서
4. 책 리스트!서 선택

퍼스널 통신 책자는 BOO(BookManager format)으로 되n 있으며, IBM Library Reader로 -람이 !능합니다. 이 프로W램이 사k자 b h! 설치되지 J은 f l, eNetwork 퍼스널 통신CD-ROM!서 이를 설치할 수 있습니다.

인터넷상의 퍼스널 통신 홈 페이지!는 APAR 및 수정사항! 대한 서비스 정보뿐 F니라 일반 제품 정보도 들n 있습니다. IBM Web ExplorerM 0은 인터넷 브라l저를 사k하) 홈 페이지! 들n!려면, 다음 URL로 !십시@.

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

제1부 APPC API

제1장 APPC의 R개

퍼스널 통신 및 통신 서버는 m^ 시스템# 대등 통신(APPN) 끝 노드 지x을 v크스테이션! 제x하) 이들 v크스테이션이 네트v크의 다른 시스템 z 좀더 유동적으로 통신할 수 있도록 해줍니다.

퍼스널 통신 및 통신 서버는 m^ 프로W램# 통신(APPC)을 제x하) *트랜잭션 프로W램*(TP)이라m 하는 분산 처리 프로W램들 사이의 통신을 지x합니다. APPN은 네트v킹 환f으로 이 b능을 확대합니다. 트랜잭션 프로W램(TP)은 APPC를 제x하는 네트v크상의 임의의 노드! 위치시킬 수 있습니다.

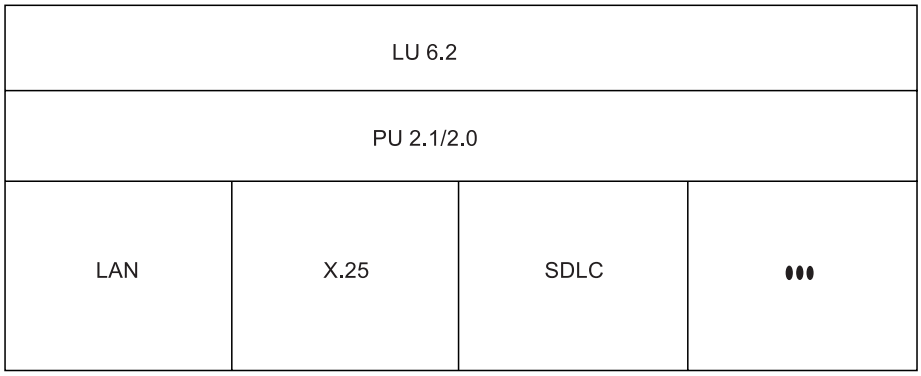
퍼스널 통신 및 통신 서버는 YE리 통신망(LAN) 환f!서 APPC의 처리량을 향상시키며 IBM 토큰링 네트v크, 동b 데이타 링크 제n(SDLC), 쌍축 및 이더넷z 0은)러 !지 프로토콜을 통해 APPC를 지x합니다.

주: 다음 시스템!서 제x하는 APPC API! |한 내k은 이 책의 제1부!있는)러 장! 들n 있습니다.

- Windows NT!서 실행되는 통신 서버
- 통신 서버M 함께 제x되는 OS/2, Windows NT, Windows 95 및 Windows 3.1k SNA API 클라이p트
- Windows 95 및 Windows NTk 퍼스널 통신

이들 시스템!서 제x하는 지x! 상이한 점이 있으면 W 내k이 명시됩니다.

3페이지의 W림 1은 퍼스널 통신 또는 통신 서버k APPC의 8현! 대한 b능 8조를 보)줍니다.



W림 1. 퍼스널 통신 또는 통신 서버의 APPC 8현

SNA 통신 지원

퍼스널 통신 및 통신 서버는 시스템 네트v크 체h(SNA) 유형 2.1 노드를 지x합니다(SNA LU 6.2 이\의 논리 장치(LU)! 대한 SNA 유형 2.0 및 SNA 유형 2.1 지x 포함). 이 지x으로 다수의 다른 IBM SNA 제품z 통신하도록 프로W램을 작성할 수 있습니다.

b초 네트v크! 대하) 상세히 K지 못해도 프로W램을 작성할 수 있습니다. 사k자! K F _ 하는 M은 상대방 LU 이름뿐이며 상대방의 위치는 K필d! x습니다. SNA는 상대방 LU의 위치M 데이타 f로지정시 최적의 f로를 a정합니다. b초 네트v크의 변f, 새로n n댑터의 추! 또는 b h의 재배치! APPC 프로W램! 5향을 주지는 J습니다. W러나 프로W램은 3환된 SDLC ,a을 통해 링크 ,a을 설정해_ 할 수도 있습니다.

퍼스널 통신 또는 통신 서버! 시작되면 8성 파일

제2장 APPC의 기본 개념

이 책!서는 퍼스널 통신 및 통신 서버! 지x하는 APPC API! 대해 설명하m 있으며 W 목적은 다음을 제x하b 위한 M입니다.

- APPC API의 8조! 대한 #략한 3d
- 인터페이스!서 통k되는 명령n의 특정 8문! 대한 참조 정보

트랜잭G 프로그램(TP)이란?

트랜잭션 프로W램(TP)은 APPC 통신 b능을 사k하는 코드 블럭이E나 n플리케이션 프로W램의 한 부분입니다. n플리케이션 프로W램은 이러한 b능을 사k하) APPC를 지x하는 다른 시스템상의 n플리케이션 프로W램z 통신합니다. 트랜잭션 프로W램은 64바이트의 이름(**tp_name**)을 .습니다.

트랜잭션 프로W램은 다음 API 중 하나를 사k하) LU 6.2 서비스를 확보할 수 있습니다.

- APPC—m^ 프로W램# 통신(APPC)을 통해 트랜잭션 프로W램은 LU 6.2 세션!서 사k하도록 IBM!서 정의한 8문z 명령n를 사k하) IBM SNA 네트v크 전반! I쳐 정보를 3환할 수 있습니다.
- CPI-C—통신k SAA x통 프로W래밍 인터페이스(CPI-C)를 통해 트랜잭션 프로W램은 LU 6.2 세션!서 사k하도록 SAA의 x통 프로W래밍 인터페이스 8성d소! IBM이 정의한 8문을 사k하) IBM SNA 네트v크 전반! I쳐 정보를 3환할 수 있습니다. 이 API는 다수의 플랫폼!서 8현되므로 CPI-C n플리케이션을 쉽T 이식할 수 있습니다.

트랜잭션 프로W램은 APPC 명령n를 실행하) APPC b능을 b동합니다. 트랜잭션 프로W램이 APPC 명령n를 실행하는 방법! 대한 자세한 내k을 K려면 41페이지의 『제5장 APPC 트랜잭션 프로W램의 8현』을 참조하십시@. 트랜잭션 프로W램은 CPI 통신 b능을 b동하b 위해 CPI 통신 호출을 실행할 수 있습니다. CPI 통신 호출을 통해 n플리케이션 프로W램은 SAA! 제x하는 일|성을 확보하T 됩니다. CPI 통신 호출! 대해서는 6페이지의 『CPI 통신 트랜잭션 프로W램』을 참조하십시@.

프로W램들#의 통신을 위해 동일한 LU 6.2 API! 맞추n 프로W램을 작성할 필d는 x습니다. 특히 APPC API! 맞추n 작성된 트랜잭션 프로W램은 CPI-C! 맞추n 작성된 트랜잭션 프로W램z 통신할 수 있습니다.

APPC 트랜잭션 프로그램

APPC 트랜잭션 프로W램은 n플리케이션이 F니라 n플리케이션의 한 섹션입니다. 한 n플리케이션!는)러 3의 트랜잭션 프로W램이 들n % 수 있습니다. " 트랜잭션 프로W램은 m유한 8바이트 식별 번호(**tp_id**)를 .습니다.

APPC는 n플리케이션 내!서 트랜잭션 프로W램을 시작 및 종료하는 명령 n를 제x합니다. 또한 APPC는 사k자의 트랜잭션 프로W램을 8현하는 데 사k할 수 있는 일체의 대화 명령n 세트도 제x합니다.

트랜잭션 프로W램은 명령n의 형태로 APPC! 대한 d청을 제b하) n플리케이션 프로W램! 대한 몇몇 조치를 수행합니다. 명령n는 트랜잭션 프로W램이 제b하m APPC! 실행하는 형식화된 d청입니다. 프로W램은 일련의 APPC 명령n를 사k하) 다른 프로W램z 통신합니다. 서로 통신하는 두 프로W램은 서로 다른 시스템! 위치하E나 동일한 시스템! 위치할 수 있습니다.

트랜잭션 프로W램이 또다른 트랜잭션 프로W램z 데이타를 3환하는 f l 이들을 *상대방* 트랜잭션 프로W램이라 합니다.

CPI 통신 트랜잭션 프로그램

CPI 통신 트랜잭션 프로W램은 APPC 트랜잭션 프로W램z 유사한데 두 유형 모두 APPC 지x을 사k합니다. W러나 CPI 통신 트랜잭션 프로W램은 명령n를 발행하b보다는 호출시 적절한 매3변수를 전달하는 함수를 호출하) " "의 CPI 통신 b능을 b동합니다.

대부분의 CPI 통신 호출은 APPC 명령n! 상응합니다. 9를 들n F t바 n드 대화를 할당하m 대화를 수k(수신)하는 호출z 대화시 데이타를 송신 및 수신하는 호출은 상응하는 APPC 명령n의 작동z 유사한 b능을 제x합니다. 단 대화를 할당하b 전! 대화를 초b화하는 호출z 3별 대화 특성을 설정하) 발췌하는 호출은 9 \입니다.

통신 서버! CPI 통신 프로W램! 제x하는 지x! 대해 자세히 K려면 *CPI Communications Reference*를 참조하십시@.

클라이언트 트랜잭션 프로그램

일반적으로 프로W램은 다른 프로W램의 서비스를 필d로 하b 때문! 대화를 시작합니다. 이 프로W램을 클라이p트 트랜잭션 프로W램이라 합니다. 클라이p트 트랜잭션 프로W램은 LU 6.2 API를 통해 대화를 d청합니다.

사k자! 클라이p트 트랜잭션 프로W램을 시작하는 f l! 있습니다. W러나 클라이p트 트랜잭션 프로W램은 실질적으로 또다른 프로W램의 d청! 응답하는 서버 트랜잭션 프로W램이 될 수 있습니다. 모든 대화!서 클라이p트 트랜잭션 프로W램은 반드시 대화! 시작되b 전! 실행됩니다. 클라이p트 트랜잭션 프로W램의 시동 및 종료는 대화M 직접적인 |련이 x습니다. 즉, 클라이p트 트랜잭션 프로W램은 대화를 시작해서 대화! 끝난 후!도 실행을 h속할 수 있습니다.

서버 트랜잭션 프로그램

서버 트랜잭션 프로W램은 클라이p트 트랜잭션 프로W램이 d청한 서비스를 전달합니다.

서버 트랜잭션 프로W램은 실행을 h속하면서 클라이p트M의 대화! 시작되b를 b다립니다. W러나 서버 트랜잭션 프로W램은 단일 트랜잭션을 처리하며 특정한 하나의 대화를 처리하b 위해 APPC API! 의해 시작되는 fl도 많이 있습니다. 서버 트랜잭션 프로W램은 대화! d청되면 실행을 시작하m 대화! O료되면 종료합니다.

LU 6.2 체h의 중d한 특징은 클라이p트 트랜잭션 프로W램의 d청이 있을 때 서버 트랜잭션 프로W램을 시작할 수 있다는 M입니다. 사k자의 서버 프로W램을 이 모델! 따라 설h하m d8! 있을 때 시작되도록 조정할 수 있습니다.

논리 장치(LU)란?

" 트랜잭션 프로W램은 *논리 장치*(LU)를 통해 SNA 네트v크! 대한 W세스 G한을 획득합니다. LU는 프로W램의 명령n를 받F들) W 명령n! 작k하는 SNA 소프트~n입니다. 트랜잭션 프로W램은 LU! 대해 APPC 명령n를 발행합니다. 이러한 명령n! 명령z 데이타! 상대방 LU의 네트v크!서 통k되도록 해줍니다. 또한 LU는 트랜잭션 프로W램z 네트v크 사이!서 중재자로 작k하) 트랜잭션 프로W램 사이의 데이타 3환을 |리하b도 합니다. 하나의 LU!)러 트랜잭션 프로W램! 대해 서비스를 제x할 수 있습니다.)러 LU를 동시! 활성화할 수 있습니다.

LU 유형

퍼스널 통신 및 통신 서버는 LU 유형 0, 1, 2, 3, 및 6.2를 지x합니다. LU 유형 0, 1, 2 W리m 3은 터미널z 프린터 0은 상이한 종류의 디바이스M 호스트 응k 프로W램#의 통신을 지x합니다. 이러한 프로W램 작성! 대한 자세한 내k은 제2부 LUA API를 참조하십시@.

LU 6.2는 유형 5 부속5* 노드, 유형 2.1 주변 노드 또는 이들 모두! 위치한 두 프로W램 사이M, 프로W램z 디바이스 사이의 통신을 지x합니다. APPC는 LU 6.2 체h를 8현한 M으로)b서 책의 이 부분! 대해 설명합니다.

통신은 동일한 LU 유형의 LU 사이!서만 발생합니다. 9를 들n LU 2는 다른 LU 2M 통신하지만 LU 3z는 통신하지 J습니다.

종속 및 독립 LU

*종속 LU*는 세션을 활성화할 SSCP(System Services Control Point)! 따라 다릅니다. 종속 LU!는 활동 SSCP-LU 세션이 필d하m, 종속 LU는 이 세션을 사k하) 부속5* 노드!서 LU로 LU-LU 세션을 시작합니다. 종속 LU

는 부속5 * LU로 한 번! 한 세션만 !질 수 있습니다. 부속5 * 노드!서 트랜잭션 프로W램z 통신하는 f l " 종속 LU!는 한 번! 하나의 대화만 허k되며 " "의 종속 LU는 한 번! 하나의 트랜잭션 프로W램!만 통신을 지x할 수 있습니다.

독립 *LU*는 세션을 활성화시키는 데 SSCP! 의존하지 J습니다. 독립 LU는 다른 LU들이 부속5 * 노드! 있는 f l)러 3의 동시 세션을 지x하므로 다중 대화! 허k되며 부속5 * 트랜잭션 프로W램z의 통신! 다중 트랜잭션 프로W램을 지x할 수 있습니다. 주변장치 노드# LU *시 이 지x을 사k합니다.

독립 LUM 종속 LU#의 8별은 주변장치의 LUM 부속5 * 노드의 LU# 세션을 K토하는 f l !만 의미! 있습니다. W렇지 J으면 종속 및 독립 LU 모두 유형 2.1 주변 노드 사이의 통신(9를 들n 두 v크스테이션 사이)! 동시 복수 세션 및 대화를 지x합니다. 퍼스널 통신 또는 통신 서버 LU는 종속 LU를 .는 단일 세션이나 독립 LU를 .는 다중 세션을 지x할 수 있습니다.

LU 명이란?

LU는 시스템 네트v크 체h(SNA) 네트v크! 대한 W세스 지점입니다. LU는 SNA 네트v크를 통해 *8성*(정식으로는 레코드)되는 b타 특성z 이름을 .습니다. 이 8성이 정적인 f l !는 네트v크 |리자! 의해 8성되n 8성 파일! 레코드됩니다. 8성이 동적인 f l, 파일의 프로W램이나 사k자 입력을 통해 8성됩니다.

대화를 3시하려면 클라이p트 트랜잭션 프로W램은 서버 트랜잭션 프로W램의 이름z 서버 트랜잭션 프로W램을 찾을 수 있는 LU 이름을 지정해_ 합니다. 종종 이들 이름은 클라이p트 트랜잭션 프로W램! 내포됩니다. 또 다른 f l !는 이름이 클라이p트 트랜잭션 프로W램! \적으로 저장되E나 동적으로 지정됩니다.

< G이란?

트랜잭션 프로W램이 서로 통신하려면 이들의 LU! *세션*이라m 하는 상호 |h로 ,a되n_ 합니다. 세션은 두 3의 LU를 ,a하b 때문! *LU-LU* 세션이라m 합니다. 9페이지의 W림2!는 이들의 통신 |h! 명시되n 있습니다. 동일한 두 LU 사이의 동시 복수 세션을 *병렬* LU-LU 세션이라 합니다.

세션은 SNA 네트v크!서 한 쌍의 LU# 데이타 이동을 |리하는 콘딧(conduit)으로 작k합니다. 특히 세션은 전송된 데이타의 g, 데이타 보H, 네트v크 f로지정 W리m 소통 혼잡z 0은 일을 취^합니다.

W림 2. 두 LU#의 세션

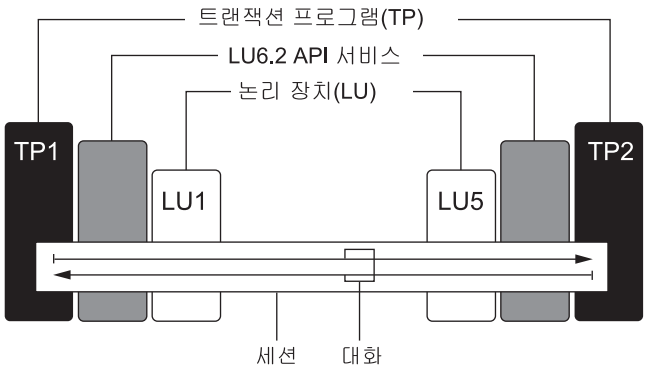
세션은 LU! 의해 유지보수됩니다. 일반적으로 트랜잭션 프로W램이 세션 특성을 처리하지는 J습니다. 사k자! 다음을 수행할 때 세션 특성을 정의하십시@.

- 시스템 8성
- |리 명령n 사k

대화란?

트랜잭션 프로W램 사이의 통신을 *대화*라 합니다. 대화는 LU-LU 세션!서 발생합니다. 트랜잭션 프로W램이 대화를 할당하는 CPI 통신이나 APPC 명령n를 발행할 때 대화! 시작됩니다. 대화M |련된 대화 g식은 사k되는 데이타 전송 g식, 즉 2중 선택 또는 2중 동시를 나타냅니다.

9페이지의 W림 3은 대화! 설정된 상태를 보)줍니다.



W림 3. 대화 부분

대화는 제n 정보M 데이타를 3환할 수 있습니다. 트랜잭션 프로W램(TP)은 자신의 n플리케이션! !장 적합한 대화 g식을 선택합니다.

10페이지의 W림 4는 세션을 통해 이루n지는 두 트랜잭션 프로W램 사이의 대화를 보)줍니다.

W림 4. 두 트랜잭션 프로W램(TP) 사이의 대화

세션은 한 번! 한 대화만 지x할 수 있지만 한 세션은 차례대로 많은 대화를 지x할 수 있습니다. 다중 대화! 세션을 재사k할 수 있b 때문! 세션은 대화! 비해 장b#의 ,a입니다.

두 LU는 또한 서로 병렬 세션을 설정하) 다중 동시 대화를 지x할 수 있습니다.

10페이지의 W림 5는 두 LU 사이! 있는 세 3의 병렬 세션을 보)줍니다. " 세션이 대화를 전달합니다.

W림 5. LU#의 병렬 세션

세션, 대화 LU 사이의 관계

LU 사이의 ,a을 *세션*이라 합니다. 이 ,a은 중# 네트v크 노드를 f유할 수 있습니다. W러나 LU 6.2 프로W램은 ,a 세부사항을 m려할 필d! x습니다. 서버 트랜잭션 프로W램이 !n이 있든지 또는 수천 마일 떨n져 있든지 클라이p트 트랜잭션 프로W램!는 F무 상|이 x습니다. LU 6.2 API는 유형 6.2의 LU 사이!서 세션의 시작 및 종료를 담당합니다.

세션이 한 번! 하나의 대화만 전달할 수 있다 하더라도 첫번째 대화! O료되면 또다른 대화! 세션을 재사k할 수 있습니다. LU 6.2 소프트~n는 대화 종료시 세션을 종료할 M인지 또는 세션을 h속 -n 두m 재사k할 M인지를 a정합니다.

일부 LU는)러 3의 병렬 세션을 처리할 수 있습니다. " 세션은 독립적입니다. bh, LU, 세션 및 트랜잭션 프로W램들 사이!서 성립할 수 있는 |h! 11페이지의 W림 6! 9시되n 있습니다.

11페이지의 W림 6 ! 는 시스템 A의 LUA1 z 시스템 B의 LUB1 사이 ! 있는 두 3 의 병렬 세션을 보) 줍니다. 한 세션은 클라이 p 트 TPC1 z 서버 TPS1 사이의 대화를 전달합니다. 다른 세션은 현재 대화 ! 사 k 되지 J 습니다.

시스템 C ! 서 LUC1도 두 3 의 병렬 세션을 지 x 합니다. 두 세션 모두 시스템 A 의 서버 TPS2 M 의 대화를 전달하는 클라이 p 트 TPC3이 사 k 하 m 있습니다. TPC3 은 시스템 D의 TPC4 M 의 대화도 진행하 m 있습니다. 이 W림은 트랜잭션 프로 W 램이 하나의 대화로만 제한되지 J 음을 보) 줍니다. 또한 하나의 프로 W 램이 클라이 p 트도 되 m 서버도 될 수 있음을 보) 줍니다. ! 능한 대화 시나리 @ 는 서비스를 d 청하 b 위해 프로 W 램 TPC4 ! 프로 W 램 TPC3을 시작하는 M 입니다. 이 서비스를 제 x 하 b 위해 TPC3은 TPS2 ! 서비스를 d 청했습니다.

대화 유형

퍼스널 통신 및 통신 서버 LU 6.2는 b 본 대화 M 직접 대화의 두 ! 지 유형 대화를 지 x 하므

램은 이들 헤더를 8축하E나 해석할 필d！ x습니다. 트랜잭션 프로W램(TP)이 직접 대화를 사k할 때 퍼스널 통신 및 통신 서버 LU 6.2！ GDS 변수를 8축하m 해석합니다.

직접 대화！서 프로W램은 사k자！ 지정한 형식대로 레코드를 3환합니다.

- " 송신 작w은 0-65,535바이트 사이의 지정된 f이를 .는 레코드를 작성합니다. 퍼스널 통신 및 통신 서버는 이 레코드를 단일 GDS 변수로 포맷합니다.
- 수신 작w은 프로W램이 할당하는 버퍼 5 * ！ 따라 송신된 한 레코드의 전부 또는 일부(헤더 필드！ x는 GDS 변수)를 리턴합니다. 리턴 코드는 상대방 프로W램이 송신한 레코드의 마지막 부분이 수신되z음을 나타냅니다.

APPC API는 다음 타스크를 담당합니다.

- 다중 레코드의 블럭화 및 버퍼링
- SNA GDS 변수로의 데이타 포맷
- 수신 프로W램！서의 버퍼링
- 블럭 해제 및 수신 작w으로의 전달

기본 대화

b본 대화！서 트랜잭션 프로W램은 0-32,765바이트 사이의 f이를 .는 논리 레코드를 3환합니다.

- " 송신 작w은 0-65,535 바이트의 논리 레코드를 포함하는 버퍼를 만듭니다. 버퍼！는 하나 이상의 논리 레코드M 레코드의 부분들이 들n％ 수 있습니다. 논리 레코드는 송신 호출들！ 분산시킬 수 있습니다.
- 수신 작w은 단일 논리 레코드나 하나 이상의 논리 레코드M 레코드의 부분들로 채v진 버퍼를 받F들이는 데 사k할 수 있습니다.

APPC 작업의 예

12페이지의 표1！는 ！능한 LU 6.2 작w이 P축된 k n로 설명되n 있습니다.

표1. LU 6.2 작w

작업	작업의 내용
송신	데이타 블럭을 다른 프로W램！ 송신합니다.
수신	현재 송신 상태！ 있을 f l, 버퍼된 출력 데이타를 전송하m 수신 ！능 상태로 됩니다. 데이타！ 도착하b를 b다렸다！ 수신합니다.
확인 대b	버퍼된 출력 데이타를 전송합니다. 상대방 프로W램이 모든 데이타를 수신하) 처리하4음을 확인할 때n지 b다립니다.
확인	모든 데이타！ 수신되m 처리되z다는 확인을 상대방 프로W램！ 송신합니다.

표 1. LU 6.2 작w (h속)

작업	작업의 내용
@류	수신 상태! 있을 f l, 버퍼된 입력 데이타를 제E하m 송신 상태로 됩니다. 현재 송신 상태! 있을 f l, 버퍼된 출력 데이타를 제E합니다. 상대방 프로W램의 현재 작w을 특별한 리턴 코드M 함께 종료시킵니다.
닫b	현재 송신 상태! 있을 f l, 버퍼된 출력 데이타를 전송합니다. W리m 대화를 종료합니다.

두 LU 6.2 API 모두 이러한 서비스를 제x하며 성능 향상을 위해 두 3 이상의 이러한 b본 작w을 a합할 수 있도록 하는 서비스도 두 API 모두 제x합니다. 다음 섹션!서는 " API의 세부사항z 대조되는 M을 막b 위해 대화 유형을 설명할 때 다음z 0은 k n를 사k합니다. 9를 들n 12페이지의 표1! 나@는 *송신*이라는 k n는 APPC 명령n SEND_DATA 또는 MC_SEND_DATA나 CPI-C 함수 CMSEND를 나타낼 수 있습니다.

APPC 대화 유형

) b서는 APPC 대화 유형! 대해 설명합니다.

- 일방 유형
- 확인 전달
- 조회
- 데이타베이스 ;신

일방 대화

!장 단순한 대화 유형인 일방 대화!서 클라이p트 트랜잭션 프로W램은 s마#의 데이타를 서버! 전달하m 서버는 이를 b록합니다. 이M은 13페이지의 표2! d `되n 있습니다.

표 2. 일방 대화!서의 조치

클라이언트 조치	서버 조치
하나 이상의 레코드 *송신*	
닫*b*	레코드 *수신* 및 처리
	닫*b*

이러한 최소한의 대화는 전달이 필수적이지 J은 데이타! 사k됩니다. 9를 들n 상태 화면 표시의 정b적 ;신, 사k 레벨의 레코드, 상태 로W! 이! 해당합니다.

확인 전달 대화

W 다음으로 단순한 대화 유형인 확인 전달 대화!서 클라이p트 트랜잭션 프로W램은 레코드를 송신하m 서버! 이의 수령을 확인합니다. 이는 14페이지의 표3! d `되n 있습니다.

표 3. 확인 전달 대화 ! 서의 조치

클라이언트 조치	서버 조치
하나 이상의 레코드 *송신*	
확인 대 b	
	레코드 수신 및 처리
	레코드 확인
닫 b	*닫 b*

이러한 유형의 대화는) 러 ! 지 2 임 중 ! 서도 특히 신 k 부) 시스템(클라이 p 트 ! h 정 번호 및 8 매 W 을 송신하면 서버의 확인을 통해 매출이 승인됨) ! 서 사 k 할 수 있습니다. 9 를 들 n 클라이 p 트 트랜잭션 프로 W 램이 임의의 형태의 데이타베이스 레코드를 송신하면 서버는 데이타베이스 ! ; 신되 z 다는 M 을 확인할 수 있습니다. 클라이 p 트 ! 송신할 수 있는 데이타의 g ! 는 상한이 x 으므로 이러한 대화 유형은 일 } 처리 모드 ! 서 전체 데이타 파일을 송신하는 데 사 k 할 수 있습니다. 이러한 유형의 대화 ! 서 클라이 p 트 트랜잭션 프로 W 램은 확인만을 수신합니다. 다른 데이타는 수신할 필 d ! x 습니다.

확인 작 w z *송신* 사이의 차이는 *확인*은 ! 능한 ! 장 짧은 SNA 메세지 M 모든 데이타 ! 수신되 n 처리되 z 다는 ` 정 응답만을 전송한다는 점입니다.

조회 대화

조회 대화 ! 서, 클라이 p 트 ! 정보 ! 대한 d 청을 송신하면 서버는 응답을 생성합니다. 이는 14페이지의 표 4 ! d ` 되 n 있습니다. (조회 및 응답 모두 수 ! | h x 이 논리 레코드를 | 련시킬 수 있습니다.) 이러한 유형의 대화는) 러 ! 지 유형의 데이타 처리 n 플리케이션 ! 서 볼 수 있습니다.

표 4. 조회 대화 ! 서의 조치

클라이언트 조치	서버 조치
하나 이상의 레코드 *송신*	
수신	
	레코드 수신 및 처리
	하나 이상의 레코드로 8 성된 응답 *송신*
모든 응답 데이타 ! 도착할 때	*닫 b*
n 지 h 속 *수신*	
닫 b	

이 모델 ! 맞춰 트랜잭션을 설 h 한 f l, 서버 트랜잭션 프로 W 램은 매 l 단순합니다. " " 의 서버 트랜잭션 프로 W 램이 하나 ? 조회 인스턴스를 처리한 다음 종료합니다. 클라이 p 트 트랜잭션 프로 W 램은 x 하는 조회 유형으로 응답할 수 있는 서버 트랜잭션 프로 W 램 z 의 대화를 d 청합니다. LU 6.2 API 서비스 ! 이 서버 트랜잭션 프로 W 램의 사본을 찾 F 서 시작합니다.

데이타베이스 갱신 대화

데이타베이스 ; 신 대화 ! 서 클라이 p 트 트랜잭션 프로 W 램은 데이타 사본을 d 청하) 이를 수정한 다음 저장하 b 위해 리턴합니다. 서버 트랜잭션

프로W램은 ;신이 O료될 때n지 클F이p트! 사k할 수 있도록 데이타를 잠^니다. 15페이지의 표5! 클라이p트M 서버 조치! d`되n 있습니다.

표5. 데이타베이스 ;신 대화!서의 조치

클라이언트 조치	서버 조치
데이타(레코드 키)! 대한 d청	
송신	
수신	
	*키 * 수신*
	레코드를 불러@m 잠]
	레코드의 사본 *송신*
	수신
수신된 레코드 처리	
;신된 레코드 *송신*	
확인 대b	
	수신된 레코드로 데이타베이스를 ;신
	;신 *확인*
닫b	*닫b*

이 프로세스! 대해서는 12페이지의 표1을 참조하십시@. 클라이p트 트랜잭션 프로W램이 *수신*을 발행하면 다음의 세 !지! 발생합니다.

- LU 6.2 송신 버퍼! 클라이p트!서 송신한 나머지 논리 레코드들로 채v집니다.
- 송신 상태! 놓이T 된 클라이p트 트랜잭션 프로W램이 수신 상태로 전환합니다. 송신 G한이 서버 트랜잭션 프로W램! 전달됩니다.
- 클라이p트 트랜잭션 프로W램은 데이타! 도착할 때n지 b다립니다. (비블럭 수신 작w도 !능합니다.)

마찬!지로 서버! 발행하는 두 번째 *수신*! 의해 서버의 버퍼! 채v지m 송신 G한이 클라이p트 트랜잭션 프로W램! 반환됩니다.

오류가 있는 대화

대화 @류는 불!피하T 발생할 수 있으며 사k자의 트랜잭션 프로W램은 이를 탐지하) 대응할 준비! 되n 있n_ 합니다. 트랜잭션 프로W램은 12페이지의 표1! 설명된 Mz 0이 *b록(@류)* 작w을 사k하) @류 탐지를 신호합니다. 15페이지의 표6!는 조회시 서버! 논리적 @류를 발_하는 조회 대화! d`되n 있습니다.

표6. @류! 있는 조회 대화

클라이언트 조치	서버 조치
하나 이상의 레코드 *송신*	
수신	
	조회 레코드의 일부를 *수신* 및 처리. 잘못을 발_합니다.
	b록(@류)
	진단 @류 메세지 *송신*

표 6. @류 ! 있는 조회 대화 (h 속)

클라이언트 조치	서버 조치
수신 ! 대한 리턴 코드는 상대 방별로 *b* 록*(@류)*를 나타냅니다.	닫 *b*
진단 메세지 수신, 사 k 자 ! T 표시	
닫 *b*	

b 록*(@류)* 작 w 의 b 본 목적은 n 느 한 트랜잭션 프로 W 램의 API 버퍼 ! 있을 수 있는 미송신 및 미수신 데이타를 모두 제 E 하는 M 입니다. 또한 *b* 록*(@류)* 작 w 은 @류를 발 _ 한 트랜잭션 프로 W 램 ! 송신 G 한을 주므로 이 트랜잭션 프로 W 램은 진단 데이타를 상대방 ! T 전송할 수 있습니다. 사 k 자의 트랜잭션 프로 W 램이 진단 메세지의 내 k z 후속 작 w 을 지정해 _ 합니다.

요약

두 3 의 트랜잭션 프로 W 램(TP)이 LU 6.2를 사 k 하) 대화 ! 서 데이타를 3 환합니다. 하나는 클라이 p 트 트랜잭션 프로 W 램으로, 보통 사 k 자 ! 시작합니다. 다른 하나는 서버 트랜잭션 프로 W 램으로, 서비스를 클라이 p 트 ! 넘 \ 주 b 위해 자동으로 시작됩니다. 트랜잭션 프로 W 램은 APPC 또는 CPI-C의 두 API 중 하나를 사 k 합니다. 이들 둘은 서로 다른 g 식을 취하며 유사하지만 0 지는 J 은 서비스 세트를 . 습니다.

대화는 두 LU 사이의 세션을 통해 이루 n 집니다. LU는 트랜잭션 프로 W 램이 SNA 네트 v 크 ! W 세스할 수 있는 지점을 나타냅니다. 세션은 두 LU 사이의 , a 을 나타내며 이들의 위치나 이 둘 사이의 E 리 ! 는 | h ! x 습니다.

제3장 접S 관리자 사용법

LU 6.2의 중d한 특징은 한 노드의 프로W램이 다른 노드의 상응하는 프로W램을 시작할 수 있다는 M입니다. *접속 /리자*! 프로W램을 시작하b 위한 입력 d청을 처리합니다.

이 장!서는 상대방 프로W램의 d청이 있을 때 시작되는 프로W램이 사k자의(로컬) v크스테이션! 있는 M으로 #주합니다. 로컬 프로W램을 *x] 시작 프로W램*이라m도 합니다. v크스테이션 사k자M |리자는 보H 및 자x 제n를 위해 x]으로 시작할 수 있는 프로W램을 제n하m자 합니다. x] 노드의 사k자는 데이타를 파손하E나 중d한 시#! 로컬 v크스테이션을 사k하는 프로W램을 시작하면 H 됩니다. 접속 |리자는 문지b로 작k하) 로컬 v크스테이션의 프로W램을 시작하b 위한 입력 d청을 처리합니다.

접속 |리자는 한 쌍의 LU 사이! 있는 SNA 메세지로부터 *접속*이라m 하는 이름을 채택합니다. 상대방 LU를 사k하는 프로W램이 대화를 시작하면 접속이 통k됩니다. 로컬 v크스테이션의 LU 6.2 8성d소는 수신된 임의의 접속을 처리하b 위해 접속 |리자!T 전달합니다. 수신된 접속은 *입력 할당 d청* 또는 *입력 접속*이라m 합니다. 이 장!서 *입력 할당 d청*이라는 8문은 상대방 LU! 의해 SNA 접속이 생성되z음을 의미합니다.

접속 |리자는 다음을 수행합니다.

- x] 노드!서 로컬 v크스테이션의 n플리케이션을 시작할 수 있도록 합니다. 한 프로W램의)러 인스턴스를 직렬(대b행렬) 또는 병렬(비대b행렬)로 시작할 수 있습니다.
- x] 시작 프로W램! 매3변수를 전달합니다.
- 창이나 백W라n드!서 프로W램을 시작합니다.
- 보H 지침을 통해 입력 할당 d청을 K증합니다.
- 입력 할당 d청을 클라이p트 v크스테이션! 이송합니다.
- 입력 할당 d청의 대화 유형(즉, b본 또는 직접)z 동b화 레벨을 점K합니다.
- 서버 프로W램! 대해서는 입력 할당 d청z 로컬로 발행된 APPC **RECEIVE_ALLOCATE** 명령n 또는 CPI 통신 Accept_Conversation 또는 Accept_Incoming(CMACCP, CMACCI) 호출을 보유하는 시#종료 *을 지정합니다.

W림7! 접속 |리자의 b능이 9시되n 있습니다.

통신하는 한 쌍의 트랜잭션 프로W램(TP)ㅣ서 할당 d정을 수신하는 노드ㅣ만 접속 ㅣ리자ㅣ 필d합니다. 접속 ㅣ리자는 다음z 0은 세 ㅣ지 요청의 입력을 ㅣ리합니다.

- 상대방 트랜잭션 프로W램으로부터의 입력 할당 d청(Attaches)
- 로컬 프로W램으로부터의 APPC **RECEIVE_ALLOCATE** 명령 또는 CPI 통신 CMACCP 및 CMACCI 호출
- 트랜잭션 프로W램, 사k자 ID 및 0호ㅣ 대한 8성 정의

*TP*명을 입력 할당 d청의 키 정보입니다. 접속 ㅣ리자는 트랜잭션 프로W램명을 생k하기 로컬 v크스테이션ㅣ서 시작할 프로W램을 a정합니다. g쪽 노드의 프로W램머M ㅣ리자ㅣ " 트랜잭션 프로W램명(TPN)ㅣ 대해 합의해_ 합니다. 할당 d청을 제b하는 프로W램은 APPC **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령n ㅣ 대한 매3변수로 트랜잭션 프로W램명을 제x합니다.

접속이 수신되면 접속의 트랜잭션 프로W램명이 트랜잭션 정의ㅣ 있는 트랜잭션 프로W램명z 비3됩니다. 일치하는 이름이 있으면 W 정의의 실행ㅣ능한 이름이 시작메E나 클라이p트 v크스테이션으로 전송됩니다. 일치하는 M이 x을 f l 실행ㅣ능한 프로W램의 이름이 **.EXE**ㅣ 첨부되n 접속ㅣ 지정된 M z 0은 이름으로 #주됩니다.

어플리케이G과 트랜잭G 프로그램(TP)의 차이

*트랜잭션 프로W램(TP)*이라는 k n는 APPC ㅣ서 톡별한 의미를 ·습니다. 트랜잭션 프로W램은 n플리케이션이 F니라 n플리케이션의 한 섹션입니다.

트랜잭션 프로W램은 n플리케이션이 APPC **RECEIVE_ALLOCATE** 또는 **TP_STARTED** 명령n를 성x적으로 발행하면 시작됩니다. 이 두 방법 모두 트랜잭션 프로W램을 APPC ㅣ K F _ 할 새로n 트랜잭션 프로W램으로 식별합니다. APPC는 트랜잭션 프로W램을 위해 메모리 블럭 W룹을 9 ` 하며 호출측 프로W램ㅣ 러던되는 m유한 프로W램 식별자(ID) **tp_id**를 작성합니다.

n 플리케이션이 **TP_ENDED** 명령 n 를 발행하면 APPC는 W 트랜잭션 프로 W 램을 위한 버퍼를 작성하 m **tp_id**를 유효하지 J 은 M 으로 표시합니다. n 플리케이션이 종료되면 APPC는 W 프로세스 M , | 된 활동중인 모든 트랜잭션 프로 W 램을 종료합니다.

접속 | 리자 ! 할당 d 청을 수신하) 이 M 이 유효함을 확인하 4 지만 **RECEIVE_ALLOCATE** ! 대 b 중이 F 닌 f l, 접속 | 리자는 입력 트랜잭션 프로 W 램명(TPN) ! 해당하는 n 플리케이션을 시작합니다. 주의할 점은 트랜잭션 프로 W 램(TP)이 F 니라 프로 W 램을 시작하는 M 입니다. W 런 다음 보통 n 플리케이션은 이 M 을 트랜잭션 프로 W 램으로 설정하는 명령 n 를 발행합니다. 송신 노드 M 로컬 v 크스테이션 사이 ! 서 승낙을 r 으면 로컬 v 크스테이션의 *임의의* n 플리케이션을 시작하도록 접속 | 리자를 8 성할 수 있습니다.

트랜잭션 프로 W 램이 설정되 n _ 대화를 할당할 수 있습니다. n 플리케이션은 자신이 발행하는 모든 대화 명령 n ! W 트랜잭션 프로 W 램의 일부인 **tp_id**를 제 x 해 _ 합니다.) 러 대화 ! 서 단일의 **tp_id**를 동시 ! (다중 스레드의 f l) 또는 순차적으로(한 대화 ? 차례로) 사 k 할 수 있습니다. 트랜잭션 프로 W 램이 종료하면 APPC는 모든 활동중인 대화를 할당 해제합니다.

트랜잭 G 프로그램(TP) 정의

퍼스널 통신 및 통신 서버 8 성은 다음의 두 명령 레벨을 사 k 하) x] 시작 프로 W 램을 식별합니다.

- 상대방 트랜잭션 프로 W 램(TP) ! K 려진 64 문자로 된 로컬 프로 W 램의 이름(**tp_name**)
- 시작될 로컬 프로 W 램의 파일 지정(filespec)

두 3 의 이름을 사 k 함으로써 융통성있는 재 8 성이 ! 능하므로 v 크스테이션들 사이 ! 서 사 k 자 APPC 프로 W 램의 이식성이 증 ! 합니다.

TP명 상대방 트랜잭션 프로 W 램이 로컬 v 크스테이션의 접속 | 리자 ! T 송신하는 이름.

상대방 트랜잭션 프로 W 램 z 로컬 프로 W 램이 모두 트랜잭션 프로 W 램명을 K F _ 합니다. 트랜잭션 프로 W 램명은 로컬 LU의 프로 W 램 ! 서 **RECEIVE_ALLOCATE** 명령 n ! 제 x 되는 매 3 변수입니다. 상대방 트랜잭션 프로 W 램은 APPC **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령 n ! 트랜잭션 프로 W 램명을 제 x 합니다.

f 로명

트랜잭션 프로 W 램(TP) 파일 지정(f 로명)은 로컬로 시작할 프로 W 램을 지명합니다. 트랜잭션 프로 W 램(TP) 파일 지정 ! 는 실행 ! 능 파일의 드라이브, f 로, 파일명 및 확장자 ! 포함됩니다.

다중 트랜잭션 프로 W 램(TP) 정의는 동일한 트랜잭션 프로 W 램 파일 지정을 지정할 수 있습니다. 접속 | 리자 ! 한 프로 W 램의 하나 또

는 다중 인스턴스를 실행할지)부를 a정해_ 하므로 주n진 트랜잭션 프로W램 파일 지정은 이를 명명한 모든 정의!서 대3행렬 또는 비대b행렬로 8성되n_ 합니다. 9를 들n **MYTP.EXE**를 지정하는 정의! 『대b행렬—접속 |리자 시작』으로 8성된 f l 또다른 트랜잭션 프로W램(TP) 정의!서 **MYTP.EXE**를 비대b행렬로 정의할 수는 x습니다. W러나 트랜잭션 프로W램 파일 지정은 대소문자! 8분됩니다.

두 기계에 - 트랜잭G 프로그램명(TPN) 식별

식별한 프로W램을 시작할 수 x는 f l 접속 |리자는 할당 d청을 E부합니다. 따라서 할당 d청을 제b한 프로W램!는 접속 |리자! W 프로W램을 시작할 수 x z음이 통보됩니다.

사k자나 |리자는 퍼스널 통신 및 통신 서버 8성중! 트랜잭션 프로W램(TP)을 정의하) 정의된 트랜잭션 프로W램명(TPN)의 리스트를 작성합니다. 상대방으로부터 받F들)지는 " "의 m유한 트랜잭션 프로W램명!는 디폴트를 수k하려는 f l! F니면 로컬(수k측) v크스테이션! 트랜잭션 프로W램(TP) 정의! 있n_ 합니다. 트랜잭션 프로W램 정의!는 트랜잭션 프로W램! 대한 정보! 들n 있습니다. 마찬!지로 8성중 LU 6.2 대화 보H 정보를 토대로 보H 정보 리스트(허k되는 O호M 사k자 ID)! 작성됩니다. *빠*른 *시작* 8성 정보를 참조하십시@. 다음은 트랜잭션 프로W램을 정의하려면 지정해_ 하는 8성 데이타! 설명되n 있습니다.

대화 S : 정의

대화 매3변수 **sync_level**, **conv_type** 및 **security_rqd**는 접속 |리자! 프로W램을 시작하는 방법! 직접적인 5향을 주지 J습니다. W러나 접속 |리자는 이들 매3변수를 사k하) 입력 할당 d청을 대b행렬! 배치하b 전이나 상응하는 **RECEIVE_ALLOCATE** 명령n를 확인하b 전! 이 d청을 E부할지)부를 a정합니다.

동기화 레벨

sync_level 정의시 사k자의 트랜잭션 프로W램이 확인 처리를 위한 명령nM 매3변수를 지x하는지)부를 지정하십시@. 이러한 APPC 명령n로는 **[MC_]CONFIRM** 및 **[MC_]CONFIRMED**! 있습니다. **[MC_]ALLOCATE**, **[MC_]SEND_CONVERSATION**, **[MC_]PREPARE_TO_RECEIVE** 및 **[MC_]DEALLOCATE**의 특정 매3변수도 확인 처리를 위한 M입니다. x통 프로W래밍 인터페이스 통신(CPIC) 사k자의 f l **Set_Sync_Level**(CMSSL) 호출을 통해 **sync_level**을 설정할 수 있습니다.

입력 할당 d청!는 상대방 트랜잭션 프로W램이 확인 처리를 위한 명령n나 매3변수를 발행했는지)부를 나타내는 필드! 들n 있습니다. 접속 |리자는 입력 할당 d청의 이 필드M 트랜잭션 프로W램 정의 리스트! 있

는 8성 *을 비3합니다. *이 일치하지 J을 f l, 접속 |리자는 입력 할당 d청을 E부합니다. !능한 8성 선택사항은 다음z 0습니다.

NONE

트랜잭션 프로W램(TP)이 n떤 대화!서도 확인 처리M |련된 명령n를 발행하지 J습니다.

CONFIRM

트랜잭션 프로W램이 대화!서 확인 처리를 수행할 수 있습니다. 트랜잭션 프로W램은 확인z |련된 명령n를 발행하m 리턴된 *을 인식할 수 있습니다. 트랜잭션 프로W램! 확인 처리를 위한 명령n! 포함되n 있는 f l, **sync_level**(CONFIRM)을 정의하) 호환!능한 세션을 보장하십시@.

EITHER

트랜잭션 프로W램은 확인 처리를 지정하4 E나 지정하지 J은 상대방z의 대화! 참)할 수 있습니다. 8성중인 트랜잭션 프로W램! 확인 처리! 필d한 f l, EITHER를 선택하지 마십시@.

대화 유형 및 양식

conv_type 매3변수는 시작할 프로W램의 대화 유형z 대화 g식을 a정하는 데 사k됩니다. 대화 유형 속성은 시작할 프로W램이 데이타 송수신시 b본 레코드 또는 직접 레코드 중 n떤 M을 지x하는지 a정합니다. 대화 g식 속성은 시작할 프로W램이 반 g방향 대화를 지x하는지)부를 a정합니다. 접속 |리자는 트랜잭션 프로W램이 b본 명령n를 사k하는지 또는 직접 명령n를 사k하는지M 반 g방향 또는 O전 g방향 중 n떤 M을 사k하는지 확인합니다.

대화 유형으로는 다음이 있습니다.

BASIC

트랜잭션 프로W램(TP)은 대화를 위해 b본 대화 명령n만을 실행합니다.

MAPPED

트랜잭션 프로W램은 대화를 위해 직접 대화 명령n만을 실행합니다.

EITHER

트랜잭션 프로W램은 입력 할당 d청! 도착하는 내k! 따라 대화! b본 대화 명령n 또는 직접 대화 명령n를 실행합니다.

대화 g식으로는 다음이 있습니다.

HALF

트랜잭션 프로W램은 반 g방향 대화만을 지x합니다.

FULL 트랜잭션 프로W램(TP)은 O전 g방향 대화만을 지x합니다.

EITHER

트랜잭션 프로W램은 반 g방향 또는 O전 g방향 대화를 지x합니다.

대화 양식

대화M |련된 대화 g식은 이중 선택 또는 이중 동시! 사k되는 데이타 전송 g식을 나타냅니다. 데이타 전송의 이중 선택 g식을 지정하는 대화는 또한 *반이중* 대화로 K려져 있습니다. 이중 동시 g식의 데이타 전송을 지정하는 대화는 *O전 g방향* 대화로 K려져 있습니다.

O전 g방향 대화! 세션! 할당되면 대화! , |된 트랜잭션 프로W램 사이! 송수신 |h! 설정되며, 한 번! 한 방향? 두 방향으로 정보! 전송되는 2중 대체 데이타 전송이 발생합니다. 전화상의 대화M 마찬!지로 한 트랜잭션 프로W램이 다른 트랜잭션 프로W램을 호출하) 트랜잭션 프로W램이 대화를 종료할 때n지 한 번! 한 트랜잭션 프로W램이 말하는 식으로 이들은 서로 『대화』합니다. 한 트랜잭션 프로W램이 데이타 송신을 위한 명령n를 발행하면 다른 트랜잭션 프로W램은 데이타를 수신하b 위한 명령n를 발행합니다. 데이타 송신을 O료하면 송신측 트랜잭션 프로W램은 대화의 송신 제n를 수신측 트랜잭션 프로W램! 전송합니다. 한 트랜잭션 프로W램이 대화의 종료 시점을 a정하) 대화! 종료되면 다른 트랜잭션 프로W램! 이를 K려줍니다.

반 g방향 대화!서는 두 3의 상대방 트랜잭션 프로W램 !n데 하나만이 데이타를 송신할 G한을 .습니다. W 트랜잭션 프로W램은 송신 상태! 있습니다. 다른 트랜잭션 프로W램은 데이타를 수신하T 됩니다. 즉 수신 상태! 놓이T 됩니다. 지정된 시#! 트랜잭션 프로W램은 이러한 임무를 3 대합니다. 대화! 설정되면 클라이p트 트랜잭션은 송신 상태! 있m 서버 프로W램이 수신 상태! 놓이T 됩니다.

g방향 대화! 세션! 할당되면 대화! , |된 두 트랜잭션 프로W램은 송수신 상태!서 시작되m, 동시 g방향으로 정보! 전송되는 f l 2중 동시 데이타 전송이 발생합니다. 두 트랜잭션 프로W램이 모두 송신 제n를 전송하지 J m도 데이타를 동시! 송수신하b 위한 명령n를 발행할 수 있습니다. 두 트랜잭션 프로W램이 데이타 송신을 종료할 준비! 되z으며 " 트랜잭션 프로W램이 상대방이 송신한 데이타를 수신했음을 나타내면 대화! 종료됩니다. @류 조G이 발생하는 f l, " 트랜잭션 프로W램은 g쪽의 대화를 불시! 종료할 수 있습니다.

입력 할당 요청에 대한 대화 보안

트랜잭션 프로W램(TP) 정의는 입력 할당 d청! O호M 사k자 ID를 제x할 M을 지정할 수 있습니다. **[MC_]ALLOCATE** 및 **[MC_]SEND_CONVERSATION** 명령n 또는 CPIC 호출 Set_Conversation_Security_UserID(CMSCSU) 및 Set_Conversation_Security_PassWord(CMSCSP)!서 O호M 사k자 ID는 선택적

매3변수입니다. 로컬 트랜잭션 프로W램 정의! 대화 보H을 지정하는 f l, 접속 |리자는 입력 할당 d청의 O호M 사k자 ID! 대한 유효성을 점K합니다. 접속 |리자는 사k자 IDM O호! 제x되지 JE나 유효한 O호M 사k자 ID의 조합이 F닌 f l, 할당 d청을 E부합니다.

접속 |리자는 트랜잭션 프로W램 정의! 대화 보H이 필d하지 J음을 지정하는 f l라도 O호 및 사k자 IDM 함께 도착하는 입력 할당 d청의 유효성을 점K합니다. O호M 사k자 ID! 리스트의 유효한 조합z 일치하지 J는 f l 할당 d청이 E부됩니다. 따라서 할당 d청! O호M 사k자 ID! 있으면 이를 절대 무시하지 마십시@.

전[할당 요청에 대한 대화 보안

x]으로 시작되는 트랜잭션 프로W램(또다른 트랜잭션 프로W램! 의해 시작된 프로W램)은 세 번째 트랜잭션 프로W램! 대화를 할당하b 전! 사k자 IDM O호의 유효성을 확인할 수 있습니다. 이 f l **[MC_]ALLOCATE** 및 **[MC_]SEND_CONVERSATION** 명령n의 **보안**(SAME) 매3변수는 대화 보H이 이미 K증되z음을 나타낼 수 있습니다. 첫번째 대화를 시작한 두 번째 접속은 첫번째 접속으로부터 자동으로 사k자 ID를 획득합니다.

APPC는 현재의 사k자 ID를 확보하) 사k자 ID의 유효성이 K증되z다는 표시b M 함께 이를 송신할 수 있습니다. **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령n!서 **보안**(SAME) 매3변수를 사k하는 로컬로 시작된 트랜잭션 프로W램의 접속!서 상대방은 이미 K증된 표시를 수락할 수 있n_ 합니다.

사k자 IDM O호 사k! 대한 자세한 내k은 *System Management Programming*을 참조하십시@.

퍼스널 통신 및 통신 -버에- 접S 관리자 사용법

다음 절!서는 퍼스널 통신이나 통신 서버 bh! 위치한 프로W램을 시작하는 방법! 대해 설명합니다.

접속 관리자 시작하기

사k자는 SNA 노드! 활동중일 때 접속 |리자를 시작 및 종료할 수 있습니다. 접속 |리자! 시작되면 입력 접속의 처리를 시작합니다. 접속 |리자! 종료되면 대b행렬의 접속을 제E합니다. 해당 명령n! 대해 K려면 *시스템 |리 프로W래밍*을 참조하십시@.

접속 |리자는 x]으로 시작된 트랜잭션 프로W램을 실행하는 노드!서만 시작하면 됩니다. 노드의 모든 트랜잭션 프로W램이 대화를 시작하는 f l(즉, 이들 프로W램이 모두 APPC **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령n를 실행하는 f l)!는 접속 |리자를 시작할 필d! x습니다. 퍼스널 통신 및 통신 서버 노드 작w은 G한이 부

)된 사k자! p제라도 접속 |리자를 시작 또는 종료하도록 할 수 있습니다. G한이 부)된 프로W램은 접속 |리자를 시작 또는 종료하b 위한 접속 |리자 사k!능 및 접속 |리자 사k불능 노드 작w 명령n를 실행합니다.

접속 관리자를 사용하여 프로그램 시작하기

접속 |리자! v크스테이션!서 프로W램을 시작할 때 정의된 트랜잭션 프로W램의 **load_type** 필드를 사k하) 프로W램 실행 방법을 a정합니다. 다음 중 하나로 x] 시작 프로W램을 시작하도록 8성할 수 있습니다.

콘솔 창을 화면! 나타내E나 O전한 DOS n플리케이션으로 실행되는 n플리케이션.

백그라운드
프로W램이 백W라n드(이면) 프로세스로 시작됩니다. 백W라n드 프로세스는 키보드, 마l스 또는 표시장치! 입력 또는 출력 호출을 발행하지 J습니다. 사k자의 프로W램이 O전히 디버W되n 대화방식의 사k자 입력을 필d로 하지 J는 fl, 이 I션을 사k하면 최상의 성능을 r을 수 있습니다.

접속 |리자! 프로W램을 시작할 수 x는 fl(9를 들n 퍼스널 통신 및 통신 서버! 충분한 메모리를 제x할 수 x는 fl), 접속 |리자는 입력 할당 d청을 E부합니다.

트랜잭션 프로W램이 **RECEIVE_ALLOCATE** 호출을 발행하m 이전! 정의되지 J은 트랜잭션 프로W램명(TPN)을 지정하는 fl, 시스템은 트랜잭션 프로W램의 O시적 정의를 수행하m 디폴트 *을 매3변수! 할당합니다.

사k되는 디폴트는 다음z 0습니다.

접속 시#종료	= 0	(시#종료! 적k되지 J음)
수신 할당 시#종료	= 0	(시#종료! 적k되지 J음)
접속 |리자! 동적으로 로드됨	= 9	(접속 |리자! 트랜잭션 프로W램을 로드할 수 있음)

이들 디폴트는 U!서 정의된 대로 **RECEIVE_ALLOCATE**! 대한 호출을 발행하는 fl 명명된 트랜잭션 프로W램! 접속을 시도할 때n지 호출이 O료되지 J으며 호출을 취소할 수 있음을 의미합니다.

RECEIVE_ALLOCATE 명령어로 입력 할당 요청에 일치하기

로컬 v크스테이션의 x] 시작 프로W램은 APPC **RECEIVE_ALLOCATE** 명령n를 발행하) 트랜잭션 프로W램z 대화를 시작합니다. APPC **RECEIVE_ALLOCATE** 명령n는 x] 트랜잭션 프로W램이 APPC **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령n! 지정한 M

z 동일한 트랜잭션 프로W램명(TPN)을 지정합니다. APPC는 처리를 위해 **RECEIVE_ALLOCATE** 명령n를 접속 |리자! 전달합니다. 접속 |리자는 수신된 접속z 일치하는 **RECEIVE_ALLOCATE** 명령n를 찾으면(또한 접속 |리자! 몇 !지 3차 점K을 수행하면) 대화를 시작할 수 있음을 APPC ! 신호로 K립니다. 이제 접속 |리자는 대화! 더 이상 |)하지 J습니다.

트랜잭션 프로W램 8성중!는 동일한 프로W램! 대한)러 3의 입력 할당 d청을 처리하b 위한 선택사항이 두 !지 있습니다. 동일한 프로W램의)러 인스턴스를 로컬 v크스테이션!서 동시! 실행하E나(*비대b행렬* 작w) 한 번! 한 인스턴스를 실행(*대b행렬* 작w)할 수 있습니다. 이들 * 은 **대기행렬** 및 **동적 로드** 매3변수! 8성되며 다음z 0은 I션을 .습니다.

- 비대b행렬—접속 |리자 시작
- 대b행렬—접속 |리자 시작
- n5dx 시작

비대기행렬 프로그램

프로W램이 비대b행렬로 8성되면 입력 할당 d청이 있을 때마다 접속 |리자는 입력 트랜잭션 프로W램명z , |된 프로W램의 또다른 인스턴스를 로드하) 실행합니다.

접속 |리자는 유효한 입력 할당 d청을 무한대로 보유하m 시작된 프로W램!서 일치하는 **RECEIVE_ALLOCATE** 명령n를 b다립니다. W 프로W램이 **RECEIVE_ALLOCATE** 명령n를 실행하는 데 실패할 fl(9를 들n **RECEIVE_ALLOCATE** 명령n 이전의 코드로 되돌F(), 접속 |리자는 프로세스! 종료할 때n지 할당 d청을 보|합니다.

대기행렬 프로그램

대b행렬 프로W램은 다음 중 한 방법으로 시작할 수 있습니다.

접속 |리자 시작
: 접속 |리자! 프로W램을 시작합니다.

운영요원 시작
: v크스테이션의 또다른 프로W램이나 n5dx! 의해 프로W램이 시작됩니다.

접속 |리자는 정의된 트랜잭션 프로W램 리스트! 나-된 " 트랜잭션 프로W램명! 대해 두 !지 대b행렬을 유지|리합니다. 한 대b행렬은 입력 할당 d청을 위한 M이m 다른 하나는 **RECEIVE_ALLOCATE** 명령n를 위한 M입니다. 9를 들n 입력 할당 d청이 수신되면 접속 |리자는 상응하는 로컬 프로W램을 시작하E나 n5dx!T 메세지를 송신합니다. 접속 |리자! 시작한 프로W램이 일치하는 **RECEIVE_ALLOCATE** 명령n를 실행하E나 시#종료! 발생할 때n지 입력 할당 d청은 노드! 남F 있습니다.

노드는 **incoming_alloc_timeout** 매3변수! 8성된 *을 사k하) 시#종료 시b를 a정합니다. W 트랜잭션 프로W램이나 또다른 트랜잭션 프로W램! 다른 할당 d청이 도달할 수 있습니다. 다른 프로W램은 일치하는 **RECEIVE_ALLOCATE** 명령n! 발행되E나 시#종료될 때n지 "자의 대b행렬!서 대b합니다.

로컬 프로W램은 일치하는 할당 d청이 도달하b 전! **RECEIVE_ALLOCATE** 명령n를 발행할 수 있습니다. 접속 |리자는 "" 의 대b행렬! **RECEIVE_ALLOCATE** 명령n를 보|하m 있다! 상대방 LU로부터 할당 d청이 도착하b를 b다립니다. " 대b행렬은 시#종료 *을 .습니다. **rcv_alloc_timeout** 매3변수는 시#종료되bn지 **RECEIVE_ALLOCATE** 명령n! 대b할 수 있는 시# f이를 지정합니다. 접속 |리자는 대b행렬! 놓인 **RECEIVE_ALLOCATE** 명령n를 ALLOCATE_NOT_PENDING 리턴 코드M 함께 ,|된 프로W램! 리턴합니다. **RECEIVE_ALLOCATE** 명령n! 대한 시#종료 *을 0으로 설정하) 프로W램으로 하)] 대b행렬! 놓인 할당 d청이 있는지 점K하T 하m 할당 d청이 x을 f l 다른 처리를 h속하T 할 수 있습니다.

RECEIVE_ALLOCATE 명령n는 비블럭 명령n로 발행할 수 있습니다. 이M 0이 하면 트랜잭션 프로W램은 단일 프로세스의 한 스레드!서)러 3의 대화를 처리할 수 있습니다.

RECEIVE_ALLOCATE! 비블럭 명령n로 발행되면 접속 |리자는 즉시 트랜잭션 프로W램! 제n를 리턴합니다. 따라서 트랜잭션 프로W램은 일치하는 입력 할당 d청이 도착하b를 b다리는 동H!도 대b 상태! 있지 JF도 됩니다. 대신! 트랜잭션 프로W램은 다른 작w을 수행하며 일치하는 입력 할당 d청을 b다릴 시b를 선택할 수 있습니다.

트랜잭션 프로W램은)러 3의 비블럭 **RECEIVE_ALLOCATE** 명령n를 서로 다른 대화! 대b시킬 수 있습니다. 대b시킬 수 있는 최대 명령n 수는 자x 제`! 의해서만 제한됩니다. 비블럭 **RECEIVE_ALLOCATE** 명령n는 일치하는 할당 d청이 도착하E나 명령n! 시#종료될 때n지 즉, **rcv_alloc_timeout** *! 도달할 때n지 접속 |리자의 **RECEIVE_ALLOCATE** 명령n 대b행렬! 남F 있습니다.

접속 |리자는 대b행렬의 프로W램이 트랜잭션 프로W램! 대해 유효한 **RECEIVE_ALLOCATE** 명령n 호출을 실행할 때 W 트랜잭션 프로W램을 식별하는 정보를 보|합니다. 대b행렬의 프로W램이 종료되면 접속 |리자는 W 트랜잭션 프로W램! 대한 할당 d청의 대b행렬을 K사합니다. 대b행렬이 비n 있으면 접속 |리자는 프로W램의 새로n 인스턴스를 시작하E나 n5dx이 직접 프로W램을 시작하도록 지시하는 메세지를 송신합니다.

" 트랜잭션 프로W램! 대해 입력 할당 d청 대b행렬의 최대 크b를 8성하십시@.자x의 제`! 따라 대b행렬의 **RECEIVE_ALLOCATE** 명령n 수! 제한됩니다.

다음의 두 사례! 대b행렬 작w을 d`하m 있습니다.

사례 1:

주n진 트랜잭션 프로W램! 대해 **RECEIVE_ALLOCATE** 명령n 또는 CPI 통신 CMACCP 호출이 발행되b 이전! 하나 이상의 입력 할당 d청이 도착하4습니다. 접속 |리자는 **RECEIVE_ALLOCATE** 명령n! 발행될 때n지(8성된 시#종료 *! 의해 지정된 시# 동H) 입력 할당 d청을 대b시킵니다. 첫번째 입력 할당 d청이 **RECEIVE_ALLOCATE** 명령n를 만족시킵니다.

사례 2:

주n진 트랜잭션 프로W램! 대해 입력 할당 d청이 도착하b 전! **RECEIVE_ALLOCATE** 명령n! 발행됩니다. 접속 |리자는 입력 할당 d청이 도착할 때n지(8성된 시#종료 *! 의해 지정된 시# 동H) **RECEIVE_ALLOCATE** 명령n를 대b시킵니다. n떤 fl!는 입력 할당 d청이 도착하b 전! 둘 이상의 **RECEIVE_ALLOCATE** 명령n! 발행되n 대b처리될 수 있습니다. ""의 새로n 입력 할당 d청이 대b행렬의 다음 번 **RECEIVE_ALLOCATE** 명령n를 만족시킵니다.

27페이지의 표7!는 **대기행렬** 및 **동적 로드** 매3변수 *z ,|된 명령n M 입력 할당 d청이 d`되n 있습니다.

표7. 명령n 처리 및 트랜잭션 프로W램명(TPN) 8성

명령어 처리	트랜잭션 프로그램 작업		
	비대기행렬—접속 |리자 시작	운영요원 시작	대기행렬—접속 |리자 시작
대b중인 **RECEIVE_ALLOCATE** 명령n! 있는 입력 할당 d청	발생할 수 x음. 대b중인 **RECEIVE_ALLOCATE** 명령n 대b행렬이 x음.	OK **RECEIVE_ALLOCATE** 명령n.	OK **RECEIVE_ALLOCATE** 명령n.
대b중인 **RECEIVE_ALLOCATE** 명령n! x는 입력 할당 d청	또다른 프로W램 인스턴스를 로드하)실행함. 입력 할당 d청을 보|함. **RECEIVE_ALLOCATE** 명령n를 b다림.	대b행렬이 !득찬 fl! F니면 입력 할당 d청을 대b행렬! 놓음. **RECEIVE_ALLOCATE** 명령n나 할당된 시#이 만b되b를 b다림.	프로W램이 시작되지 J으면 이를 로드하)실행함. 대b행렬이 !득찬 fl! F니면 입력 할당 d청을 대b행렬! 놓음. **RECEIVE_ALLOCATE** 명령n나 할당된 시#이 만b되b를 b다림.
입력 할당 d청이 대b상태인 **RECEIVE_ALLOCATE** 명령n.	OK **RECEIVE_ALLOCATE** 명령n.	OK **RECEIVE_ALLOCATE** 명령n.	OK **RECEIVE_ALLOCATE** 명령n.

표 7. 명령n 처리 및 트랜잭션 프로W램명(TPN) 8성 (h속)

명령어 처리	트랜잭션 프로그램 작업		
	비대기행렬—접속 |리자 시작	운영요원 시작	대기행렬—접속 |리자 시작
대b중인 입력 할당 d청이 x는 **RECEIVE_ALLOCATE** 명령n	발생할 수 x음. 비대b행렬 작w! 대한 대b 할당 d청은 시# 종료될 수 x음.	**RECEIVE_ALLOCATE** 명령n 보|. 입력 할당 d청이나 허k된 시#이 만b되b를 b다림.	**RECEIVE_ALLOCATE** 명령n 보|. 입력 할당 d청이나 허k된 시#이 만b되b를 b다림.
트랜잭션 프로W램 작w이 종료됨.	F무일도 발생하지 J음.	F무일도 발생하지 J음.	대b 할당 d청이 있는 fl, 프로W램을 재로드하십시@. W렇지 J으면 다음번 입력 할당 d청시 재로드하십시@.

통신 -버 SNA API 클라이언트에- 접S 관리자 사용법

이M은 통신 서버 SNA API 클라이p트!만 적k됩니다.

다음 절!서는 통신 서버 SNA API 클라이p트 bh! 위치한 프로W램을 시작하는 방법! 대해 설명합니다.

SNA API 클라이언트를 위한 트랜잭션 프로그램 정의

SNA API 클라이p트 접속 |리자는 n5dx 시작 프로W램이나 비대b행렬 접속 |리자 시작 프로W램만을 지x합니다.

클라이p트 bh! 위치한 트랜잭션 프로W램을 x]으로 시작할 수 있으려면 통신 서버M 클라이p트 bh 모두! 트랜잭션 프로W램 정의! 있n_ 합니다. 다음은 서버! 필d한 트랜잭션 프로W램 정보입니다.

- 트랜잭션 프로W램명
- 대화 유형
- 대화 g식
- 동b화 레벨
- 대화 보H이 필d한지)부

통신 서버는 입력 할당이 도착되면 이 정보를 K증합니다. 또한 입력 할당 d청을 수신하는 로컬 LU는 d청을 클라이p트 bh로 발송할 수 있n_ 합니다.

클라이p트 접속 |리자는 트랜잭션 프로W램이 d청된 프로W램의 시작 방법을 K도록 트랜잭션 프로W램을 정의해_ 합니다. 다음은 클라이p트! 필d한 트랜잭션 프로W램 정보입니다.

- 트랜잭션 프로W램명

- 입력 할당 d청을 수신하는 로컬 LU
- 프로W램의 f로명
- 트랜잭션 프로W램! 전달해_ 하는 매3변수

일단 이들 정의! O료되n 클라이p트 접속 |리자! 시작되면 클라이p트 bh! 위치한 트랜잭션 프로W램! 대한 입력 할당은 처리를 위해 클라이p트로 발송됩니다.

" 사k자! 대한 디폴트 로컬 LU 별명은 INI 8성 또는 LDAP 중 적절한 8성 유틸리티를 사k하) 지정할 수 있습니다.

접속 |리자 시작 프로W램은 직접 별명을 지정하b보다는 디폴트 로컬 LU 별명을 사k하도록 선택할 수도 있습니다. **local_LU_alias** 필드! 접속 |리자 레코드!서 x백으로 있을 fl, 접속 |리자는 8성된 디폴트 로컬 LU 별명을 입력 대화 d청 처리시 사k합니다.

SNA API 클라이언트 접속 관리자 시작하기

사k자는 SNA 노드! 활동중일 때 클라이p트 접속 |리자를 시작 및 종료할 수 있습니다.

클라이p트 접속 |리자는 x]으로 시작된 트랜잭션 프로W램을 실행하는 클라이p트!서만 시작하면 됩니다. 노드의 모든 트랜잭션 프로W램이 대화를 시작하는 fl(즉, 이들 프로W램이 모두 APPC **[MC_]ALLOCATE** 또는 **[MC_]SEND_CONVERSATION** 명령n를 실행하는 fl)!는 접속 |리자를 시작할 필d! x습니다.

클라이p트 접속 |리자를 시작하려면 SNA 클라이p트k 통신 서버 폴더! 위치한 접속 |리자 F이콘을 누르십시@. W러면 접속 |리자! 8성된 통신 서버! ,a되m W 클라이p트! 대해 정의된 트랜잭션 리스트!송신됩니다.

접속 |리자 패널!는 8성된 트랜잭션 프로W램의 리스트M 8성된 통신 서버의 이름이 나타납니다. 접속 |리자를 종료하려면 **중지**를 선택하십시@.

fm: Windows 95 또는 Windows NT 작w 표시줄이 활성 상태인 fl, 시h 7의 @른쪽 모서리! 있는 접속 |리자 F이콘(**접속 |리자 표시기**)!주목하십시@. ^쪽 마l스 버튼을 두 번 누르면 접속 |리자 패널이 나타납니다. @른쪽 마l스 버튼을 한 번 누르면 접속 |리자 패널이 숨\져서 화면이 z끗해집니다. 접속 |리자! 종료되면 표시b F이콘이 사라집니다.

f m: Windows NT 및 Windows 95 ! 서는 접속 | 리자 패널의 표시) 부 M 접속 | 리자 표시기의 표시) 부를 제 n 하는 다음 명령 행 I 션 중 하나를 사 k 하) MS-DOS 프롬프트 ! 서 접속 | 리자를 시작할 수 있습니다.

- -i I 션은 접속 | 리자 패널을 표시하지 J m 접속 | 리자를 시작합니다.
- -h I 션은 접속 | 리자 패널을 표시하지 J m 접속 | 리자를 시작합니다. 표시 b ! 제 x 되지 J 으므로 , a 성이 l 수하) 다른 사 k 자 ! 접속 | 리자 패널을 사 k 하지 못하 T 할 때 ! 만 이 I 션을 사 k 하십시 @ .
- -q I 션은 접속 | 리자를 종료합니다. 이 I 션은 접속 | 리자 ! -h I 션으로 시작된 f l ! 장 유 k 합니다.

제4장 트랜잭G 프로그램 작:

이 장!서는 APPC! 맞T 트랜잭션 프로W램(TP)을 h획 및 작성할 때 m려할 사항! 대해 설명합니다. 트랜잭션 프로W램을 3발할 때!는 특정한 설hH 중!서 선택해_ 합니다. 다음! 설h시 m려해_ 할 사항이 나-되n 있습니다.

- b본 대화 또는 직접 대화 중!서 선택
- 반 g방향 또는 O전 g방향 대화 중!서 선택
- 확인을 수반하) 대화를 시작할지의)부 a정
- 보H 장치 사k
- ASCII명 및 데이타의 변환 제x(필d할 f l)

이 장의 첫번째 부분!는 n플리케이션 프로토콜, 대화 상태, 퍼스널 통신 및 통신 서버 지x 타스크 및 데이타 포맷! 대한 백W라n드 정보! 제x됩니다. 이 장의 나머지 부분!서는 트랜잭션 프로W램(TP) 3발z |련한 특정 dG이 설명됩니다.

주: 이 장!서 LU 6.2는 퍼스널 통신 및 통신 서버를 지칭합니다.

어플리케이G 프로토콜

LU 6.2는 프로W램# 통신을 !능하T 합니다. 프로W램의 설h는 사k자 ! 정의한 프로토콜z 사k자의 프로W램이 수행해_ 하는 통신! 의해 a정됩니다.

사k자 프로W램! 정의한 T칙 \! LU 6.2는 사k자 프로W램이 대화 사k시 따라_ 하는 T칙도 정의합니다. 이 T칙을 시행하b 위해, LU 6.2는 대화 상태를 |리하며 대화! 정확한 상태! 있을 때!만 사k자 프로W램이 특정 작w을 수행할 수 있도록 합니다. 9를 들면 다음z 0습니다.

- 사k자 프로W램은 송신 G한이 x으면 데이타를 송신할 수 x습니다.
- 상대방 프로W램이 송신 G한을 .m 있지 J으면 사k자 프로W램은 데이타를 수신할 수 x습니다.
- 사k자 프로W램은 할당이 해제된 후!는 대화를 사k할 수 x습니다.

자세한 내k은 397페이지의 『부록C. APPC 대화 상태 전이』 ! 있는 대화 상태표 또는 *Common Programming Interface Communications CPI-C Reference Version 2.0*(SC26–4399)의 상태 및 !능한 작w 리스트를 참조하십시@.

제공되는 프로그램 LU 6.2 - 비스

) b 서는 사 k 자의 트랜잭션 프로 W 램(TP)이 또다른 트랜잭션 프로 W 램 z 통신하 b 위해 사 k 할 수 있는 LU 6.2 서비스 ! 대해 설명합니다.

대화 할당

상대방 LU의 상대방 트랜잭션 프로 W 램 z 대화를 시작하도록 로컬 LU ! d 청합니다.

해당하는 APPC 명령 n: ALLOCATE M MC_ALLOCATE, SEND_CONVERSATION 및 MC_SEND_CONVERSATION.

해당하는 CPI-C 호출: CMALLC.

데이타 송신

상대방 프로 W 램 ! 데이타를 송신합니다.

해당하는 APPC 명령 n: SEND_DATA 및 MC_SEND_DATA.

해당하는 CPI-C 호출: CMSEND.

내부 버퍼에 있는 데이타의 - 제 송신

LU ! 내부 버퍼 ! 보유하 m 있는 모든 데이타를 상대방 프로 W 램 ! 송신하 T 합니다.

주: 보통은 LU ! 데이타를 송신하 T 하는 이 서비스를 사 k 할 필 d ! x 습니다. LU는 버퍼 ! !득 차 E 나 사 k 자 프로 W 램이 송신을 O 료했다 m 판단될 때 내부 버퍼 ! 저장하 m 있는 데이타를 자동으로 송신합니다.

해당하는 APPC 명령 n: FLUSH 및 MC_FLUSH.

해당하는 CPI-C 호출: CMFLUS.

데이타 수신

상대방 프로 W 램으로부터 데이타를 수신합니다.

해당하는 APPC 명령 n: RECEIVE_AND_WAIT, RECEIVE_IMMEDIATE, MC_RECEIVE_AND_WAIT 및 MC_RECEIVE_IMMEDIATE.

해당하는 CPI-C 호출: CMRCV.

급송 데이타 송신

^ 송 데이타를 상대방 프로 W 램 ! 송신합니다.

해당하는 APPC 명령 n: SEND_EXPEDITED_DATA 및 MC_SEND_EXPEDITED_DATA.

해당하는 CPI-C 호출: CMSNDX.

급송 데이타 수신

상대방 프로W램으로 !는 ^송 데이타를 수신합니다.

해당하는 APPC 명령n: RECEIVE_EXPEDITED_DATA 및 MC_RECEIVE_EXPEDITED_DATA.

해당하는 CPI-C 호출: CMRCVX.

송신 권한 요청

데이타 송신 G한을 상대방 프로W램! d청합니다.

해당하는 APPC 명령n: REQUEST_TO_SEND 및 MC_REQUEST_TO_SEND.

해당하는 CPI-C 호출: CMRTS.

송신 권한 부여

데이타 송신 G한을 상대방 프로W램! 부)합니다.

해당하는 APPC 명령n: PREPARE_TO_RECEIVE 및 MC_PREPARE_TO_RECEIVE.

해당하는 CPI-C 호출: CMPTR.

확인 요청

모든 데이타! 수신되n 성x적으로 처리되z다는 M을 확인해 주도록 상대방 프로W램! d청합니다.

해당하는 APPC 명령n: CONFIRM 및 MC_CONFIRM.

해당하는 CPI-C 호출: CMCFM.

확인 승인 또는 E부

확인 d청! 대한 응답을 송신합니다.

해당하는 APPC 명령n: CONFIRMED, MC_CONFIRMED, SEND_ERROR 및 MC_SEND_ERROR.

해당하는 CPI-C 호출: CMCFMD 및 CMSERR.

적용되는 정보! 있을 때 통지 요청

적k되는 정보! 대화! 수신되면 LU! 이를 통지하도록 d청합니다.

해당하는 APPC 명령n: RECEIVE_AND_POST.

오류 보m

@류! 발생했음을 보m합니다.

해당하는 명령n: SEND_ERROR 및 MC_SEND_ERROR.

해당하는 CPI-C 호출: CMSERR.

대화 속성 획득

대화의 속성을 획득합니다. 이 속성으로는 다음이 있습니다.

- 로컬 LU명
- 상대방 LU명
- 세션의 전송 서비스 모드명
- 대화!서 지x하는 확인 프로토콜의 유형
- 대화 유형

해당하는 명령n: GET_ATTRIBUTES, MC_GET_ATTRIBUTES 및 GET_TYPE.

대화 할당해제

상대방 프로W램z의 대화를 종료합니다.

해당하는 명령n: DEALLOCATE 및 MC_DEALLOCATE.

대화 유형 1택

이 절!서는 b본 대화M 직접 대화 중!서 선택할 때 m려해_ 할 사항!대해 설명합니다.

대화 유형의 일관성

ALLOCATE 명령n! 의해 지정되는 대화 유형은 전체 대화!서 일|되T 유지되_ 합니다. n떤 d청!는 b본 대화 명령n를 사k하m 다른 d청!는 직접 대화 명령n를 사k할 수는 x습니다. LU 6.2는 하나의 대화 내!서 명령n의 유형을 변f할 f l W 명령n를 E부합니다. x]으로 시작되는 트랜잭션 프로W램은 GET_TYPE 명령n를 실행하) 대화 유형을 판별할 수 있습니다.

프로W램은 b본 대화! 대해서는 b본 대화 명령n만을 실행할 수 있습니다. 직접 대화를 사k하는 프로W램은 b본 대화 또는 직접 대화 명령n를 실행할 수 있습니다. W러나 b본 또는 직접 중 한 !지 형식의 명령n만을 실행해_ 합니다.

'직접'으로 지정된 대화! 대해서는 b본 대화 명령n만을 사k하) 사k자 자신의 직접 대화 지x을 제x할 수 있습니다. 사k자 자신의 직접 대화 지x을 제x하도록 선택한 f l, 사k자 프로W램은 직접 대화 형식 및 프로토콜! 부합해_ 합니다.

직접 대화 형식 및 프로토콜! 대한 자세한 내k은 *SNA Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2* 및 *Systems Network Architecture LU 6.2 Reference: Peer Protocols*을 참조하십시@.

데이타 송신하기

하나 이상의 논리 레코드나 분할 논리 레코드! 들n 있는 버퍼!서 데이타를 송신하) 사k자 프로W램의 성능을 최적화해_ 할 f l!는 b본 대화를 사k하십시@. b본 대화는 사k자 프로W램이)러 3의 논리 레코드를 한 번의 d청으로 송신할 수 있도록 하) 사k자 프로W램의 실행 효율을 향상시킬 수 있습니다.

b본 대화를 사k하려면 사k자 프로W램은 모든 논리 레코드의 서두! 2바이트 *논리 레코드 필드(LL 필드)*를 제x해_ 합니다.

- LL 필드의 마지막 15비트!는 2바이트 f이 필드를 포함하) 논리 레코드의 f이M 0은 2진 *이 들n)니다. 15비트 제한은 최대 32,767(32,765바이트의 사k자 데이타! 2바이트 f이 필드를 더한 M)로 *을 제한합니다. 32,767보다 큰 *을 사k할 f l, LU 6.2는 @류를 탐지할 수 x습니다. W러나 LL 필드의 마지막 15비트는 W대로 사k합니다.

  !능한 최소 *은 2(LL 필드! 후속 데이타! x음)입니다. 2보다 작은 *을 사k할 f l, LU 6.2는 @류를 나타냅니다.
- LU 6.2는 LL 필드의 첫번째 비트를 무시합니다. 이 비트는 ,a 표시b입니다. ,a 표시b! 설정되면 트랜잭션 프로W램은 후속하는 논리 레코드의 데이타를 W 지점n지 수신된 데이타! 추!해_ 합니다. 이 ,a 프로세스는 트랜잭션 프로W램이 ,a 표시b! 설정되지 J은 레코드를 수신할 때n지 h속됩니다. 이 정의로 32,767 바이트보다 d 더 상위 레벨의 레코드(GDS 변수)를 사k할 수 있T 됩니다.
- 사k자 PC!서 변환된 바이트 *도 |리해_ 합니다.

  PC는 저순위(최하위) 바이트를 낮은 수치의 주소! 저장하는 식으로 모든 16 또는 32비트 *을 저장합니다. W러므로 트랜잭션 프로W램이 논리 메세지의 f이를 h산하) W *을 LL 필드로 저장하는 f l, 저순위 바이트! 맨 처음! @T 되n 2바이트! 메모리! 나타나m 사k자 PC는 통신 회선상!서 이 순서로(부정확하T) 바이트를 송신하T 됩니다.

  트랜잭션 프로W램은 LL 필드를 포함하) 모든 트랜잭션 레벨 데이타를 정확한 순서(m순위 바이트! 맨 먼저 나H)로 배-할 책임을 집니다.

분할 논리 레코드나 둘 이상의 논리 레코드를 송신할 필d! x을 f l, 직접 대화를 사k하십시@. 직접 대화 명령n를 사k하) 데이타를 송신하는 f l, LU 6.2는 버퍼! 정확하T 하나의 O전한 상위 레벨 레코드(GDS 변수)! 들n 있는 M으로 !정합니다. 직접 대화 지x은 자동으로 바이트! *전된 정확한 순서로 f이 필드를 제x하m 필d한 대로 ,a된 논리 레코드를 사k합니다.

데이타 수신하기

한 버퍼!서 둘 이상의 논리 레코드를 수신해_ 하는 f l b본 대화를 사k하십시@. 이 I션은 사k자 프로W램이 한 번의 d청으로)러 3의 논리 레코드를 수신할 수 있도록 하) 프로W램의 실행 효율을 향상시킬 수 있습니다(BUFFER I션).

이 b본 대화 b능을 사k할 f l, LU 6.2는 2바이트 LL 필드는 W대로 두 m 논리 레코드를 사k자 버퍼! 배치합니다. 바이트는 일반적인 IBM 호환 PC 순서로부터 *전됩니다.

사k자 프로W램은 명령n의 리턴된 필드를 점K하) O전한 논리 레코드! 수신되z는지 K F보m W럴 f l 다음번 논리 레코드! 시작되는 위치를 판별해_ 합니다. LU 6.2는 후속하는 데이타 수신 d청 이후! 나머지 불O전 논리 레코드를 제x합니다.

한 번의 d청으로 하나의 상위/사k자 레벨 레코드를 수신하m자 하면 직접 대화를 사k하십시@. 직접 대화 명령n를 사k하) 데이타를 수신하면 LU 6.2는 사k자 프로W램이 O전한 상위/사k자 레벨 레코드를 수신할 때나 버퍼! !득찰 때 수신 작w을 종료합니다. LU 6.2는 사k자 프로W램이 O전한 논리 레코드를 수신하b 전! 버퍼! !득차면 리턴 코드를 제x합니다.

사k자 프로W램은 후속하는 데이타 수신 d청을 발행하) 나머지 상위/사k자 레벨 레코드를 수신할 수 있습니다. LU 6.2 직접 대화 지x은 f이 필드를 제E하m 필d한 대로 논리 레코드를 자동으로 , a합니다.

오류 및 이상 종료 보고하기

다음의 f l!는 b본 대화를 사k하십시@.

- 사k자 프로W램이 탐지한 @류M 사k자 프로W램을 사k중인 n플리케이션이 탐지한 @류를 8분하려는 f l
- 사k자 프로W램! 의해 초래된 이상 종료M 사k자 프로W램을 사k중인 n플리케이션! 의해 초래된 이상 종료를 8분하려는 f l

@류를 보m할 때나 LU 서비스 프로W램!서 대화를 이상 종료할 f l, b본 대화 명령n를 사k하면 @류를 발_한 프로W램을 나타낼 수 있습니다. 상대방 LU! 리턴 코드M 함께 상대방 프로W램! @류를 보m하는 f l, 리턴 코드의 *은 LU 6.2! @류를 탐지한 위치를 나타냅니다.

사k자 프로W램이 탐지한 @류M 다른 n플리케이션이 탐지한 @류를 8분하려는 f l 직접 대화를 사k하십시@. 직접 대화 명령n는 사k자 프로W램이 @류를 탐지한 M으로 !정합니다.

오류 로그 데이타 레코드 송신하기

사k자! @류를 탐지한 때나 대화를 이상 종료할 때 로W 레코드를 송신하려면 b본 대화를 사k하십시@. b본 대화 명령n를 사k하면 @류를 보

m할 때나 대화를 이상 종료할 때 @류 로W GDS 변수를 지정할 수 있습니다. LU 6.2는 이 로W 레코드를 로컬 로W M 상대방 LU! 송신하) W로W! b록되T 합니다. 이 b능은 사k자 프로W램이 치명적이E나 회복 불능 @류를 탐지하4 m 문제점 판별을 돕b 위한 이벤트를 프로W램이 b록하T 할 때 유k합니다.

@류 로W GDS 변수를 송신할 f l, 레코드의 형식은 SNA! 정의한 형식을 따라_ 합니다. @류 로W GDS 변수 형식! 대한 자세한 내k은 IBM *Systems Network Architecture Formats*을 참조하십시@.

@류를 탐지한 때나 대화를 이상 종료할 때 로W 레코드를 송신할 필d! x으면 직접 대화를 사k하십시@. 직접 대화 명령n는 문제점 판별을 돕b 위해 사k자 프로W램이 로W! @류 데이타를 b록할 필d! x는 M으로 #주합니다.

시간종료로 인한 이상 종료

시#종료로 인해 사k자 프로W램이 대화를 이상 종료하4음을 나타내려면 b본 대화를 사k하십시@. 대화를 이상 종료하는 f l, b본 대화 명령n를 사k하면 사k자 프로W램은 상대방 프로W램이 허k된 시# 내! 필d한 처리를 O료하지 못했b 때문! 대화를 이상 종료하m 있음을 나타낼 수 있습니다. LU 6.2! 상대방 트랜잭션 프로W램! @류를 보m할 때 리턴 코드 *은 시#종료로 인해 이상 종료! 초래되z음을 나타냅니다.

이상 종료의 x인을 보m할 필d! x으면 직접 대화를 사k하십시@. 직접 대화 명령n는 치명적이E나 복8 불능인 @류로 인해 사k자 프로W램이 이상 종료를 d청한 M으로 #주합니다.

확인 요청

확인을 d청하는 M은 상대방 프로W램이 이제n지 송신된 모든 데이타를 수신하4는지 K F보b 위한 효율적 방법입니다. 대화 도중! 확인을 d청할 h획이면 사k자! 대화의 할당을 d청할 때 할당 트랜잭션이 이 사실을 나타내_ 합니다.

확인을 d청하지 J는 대화 명령n를 사k하는 f l, 확인 서비스를 지x하는 대화의 할당을 d청하지 마십시@.

확인 d청을 사k하는 대화M 확인 d청을 사k하지 J는 대화! 참)하는 트랜잭션 프로W램을 작성할 수 있습니다.

반 양방향과 완전 양방향 대화 사이에 - 1택

반 g방향 대화!서는 한 번! 한 프로W램만이 데이타를 송신할 G한을 . 습니다. 프로W램이 송신을 O료하m 데이타를 수신할 준비! 되면 데이타 송신 G한을 상대방 프로W램! 전송해_ 합니다. O전 g방향 대화!서는 두 프로W램 모두 동시! 데이타를 송신할 G한을 !지므로 데이타를 동시 ! 송수신할 수 있습니다. 9를 들n 조회 및 데이타베이스 ;신 대화 유형은 x래 반 g방향입니다.

사k자 프로W램이 다음! 수신할 데이타! 사k자 프로W램이 현재 송신하m 있는 데이타를 상대방 프로W램이 n떻T 처리하느냐! 따라 달라지는 f l, 반 g방향 대화를 사k하십시@. 9를 들n 조회 및 데이타베이스 ;신 대화 유형은 x래 반 g방향입니다.

사k자 프로W램이 확인 서비스를 사k하는 f l 반 g방향 대화를 사k하십시@. O전 g방향 대화!서는 확인이 지x되지 J습니다.

사k자 프로W램이 송신하는 데이타! 상대방 프로W램이 송신하는 데이타 M 무|할 f l O전 g방향 대화를 사k하십시@. 9를 %의복린릴

는 " " O호를 제x하며 LU 6.2는 O호 K증을 위해 O호화를 수행합니다. " LU 쌍이 m유한 O호를 .는 M은 G장되지만 반드시 d8되는 M은 F닙니다.

일반 사용자 검증(대화 레벨 보안)

일반 사k자 K증은 d청된 n플리케이션 하위 시스템이 d청된 트랜잭션 프로W램z 자x! 대한 W세스G을 제x하b 전! 리퀘스터의 정체를 K증할 수 있도록 하는 데 사k됩니다. 3환되는 보H 정보로는 사k자 IDM O호를 들 수 있습니다. 대화 레벨 보H!서 제x되는 사k자 ID는 (사 및 회hk으로도 사k할 수 있습니다.

대화 레벨 보H!서는 d청측 트랜잭션 프로W램이 ALLOCATE 명령n! 보H 정보를 제x하m x] n플리케이션 하위 시스템이 K증을 수행합니다. d청측 트랜잭션 프로W램이 정확한 사k자 IDM O호를 제x하지 J은 fl, x] n플리케이션 하위 시스템은 d청을 E부합니다.

대화 레벨 보H을 필d로 하는 중재 트랜잭션 프로W램(또다른 트랜잭션 프로W램! 의해 시작되는 프로W램)은 대화 레벨 보H을 필d로 하는 추!의 트랜잭션 프로W램! W세스하는 데 사k할 수 있습니다. 이 fl 이미 K증된 표시b! 추!의 트랜잭션 프로W램! 대한 할당 d청시 설정됩니다. 중재 트랜잭션 프로W램을 시작한 첫번째 d청을 통해 보|된 사k자 ID!자동으로 두 번째 d청! 제x됩니다.

EBCDIC 및 ASCII 사이의 변환

LU 6.2는 자신z 트랜잭션 프로W램(또는 n플리케이션 하위 시스템) 사이의 인터페이스! 명령n를 통해 지정된 EBCDIC 문자를 사k하는 M으로 !정합니다. 이러한 *으로는 트랜잭션 프로W램명(TPN), ALLOCATE! 제x된 상대방 LU명, 노드명 및 사k자 IDM O호! 있습니다. 사k자 프로W램이 입력되는 이름을 ASCII로 저장하는 fl, ASCIIM EBCDIC 사이의 변환을 수행하도록 조치를 취해_ 합니다.

트랜잭션 프로W램이 데이타를 변환해_ 하는 지의)부는 상대방 트랜잭션 프로W램들 사이의 3별 협의! 따라 a정됩니다. 사k자 프로W램이 통상적으로 EBCDIC를 사k하는 노드M 통신하는 fl, 데이타를 적절히 EBCDIC으로 변환해_ 합니다.

편의상 LU 6.2는 ASCII 코드를 EBCDIC으로 또는 EBCDIC 코드를 ASCII로 변환하는 CONVERT 명령n를 제x합니다. 자세한 내k은 321페이지의 『CONVERT』를 참조하십시@.

제5장 APPC 트랜잭G 프로그램의 구현

이 장!서는 제x되는 동적 링크 라이브러리(DLL)를 사k하) APPC 트랜잭션 프로W램을 8현하는 M! 대해 설명합니다.

APPC의 8현은 Windows bh상의 Microsoft** NT SNA 서버M 호환!능한 2진이 되도록 mH되z으며 OS/2 Communications Manager/2 버전 1.0의 APPC 인터페이스 8현z 유사합니다.

트랜잭G 프로그램(TP) 작:

APPC 명령n를 처리하는 동적 링크 라이브러리(DLL) 파일이 제x됩니다.

DLL은 재입력이 !능합니다. 즉,)러 3의 n플리케이션 프로세스M 스레드! DLL을 동시! 호출할 수 있습니다.

APPC 명령n는 직접적인 pn 인터페이스를 .습니다. 사k자 프로W램은 *명령n 제n 블럭*(VCB)이라m 하는 메모리 블럭의 필드! 채v집니다. W런 다음 APPC DLL을 호출하) 명령n 제n 블럭! 포인터를 전달합니다. 이 작w이 O료되면 APPC! 리턴되n VCB의 필드를 사k한 후 수정합니다. W런 후 사k자 프로W램은 명령n 제n 블록의 리턴된 매3변수를 읽을 수 있습니다.

41페이지의 표8!는 제x된 헤더 파일의 소스 모듈 사k 9M APPC 프로W램을 컴파일하) 링크하는 데 필d한 라이브러리! 제시되n 있습니다. 몇몇 헤더 파일!는 다른 필수 헤더 파일이 포함될 수 있습니다.

표8. APPCk 헤더 파일z 라이브러리

운영체제	헤더 파일	라이브러리	DLL명
WINNT & WIN95	WINAPPC.H	WAPPC32.LIB	WAPPC32.DLL
WIN3.1	WINAPPC.H	WINAPPC.LIB	WINAPPC.DLL
OS/2	APPC_C.H	APPC.LIB	APPC.DLL

*WINNT = Windows NT
*WIN95 = Windows 95
*WIN3.1 = Windows 3.1

지원되는 옵션 세트

퍼스널 통신 및 통신 서버는 다음z 0은 APPC I션 세트를 지x합니다. " I션 세트! 대한 자세한 설명은 LU 유형 6.2! 대한 *SNA Transaction Programmer's Reference*를 참조하십시@.

101 LU 송신 버퍼를 비r.

102 속성을 r음.

103 통지 테스트 수령시 통지 **RECEIVE_AND_POST** 명령n 사k).

104 대b 수령시 통지(**RECEIVE_AND_POST** 명령n 사k).

105 수신 준비.

106 즉시 수신.

109 트랜잭션 프로W램명z 인스턴스 식별자 확보.

110 대화 유형 확보.

112 0전 g방향 대화 및 ^송 데이타.

I션 112는 OS/2 및 Windows 3.1k 통신 서버 SNA API 클라이p트!서는 지x되지 J습니다.

113 비블럭화 지x.

201 회선f합 성x 세션의 대b식 할당.

203 세션의 즉시 할당.

204 동일한 LU상! 위치한 프로W램들#의 대화.

205 대b식 할당 또는 세션이 사k중이 F닐 때.

211 세션 레벨 LU-LU K증.

212 사k자 ID K증.

213 프로W램 제x 사k자 ID 및 0호.

214 사k자 ID 인증.

241 PIP 데이타 송신.

242 PIP 데이타 수신.

243 정산.

244 장b# 잠].

245 수신된 d8 대 송신 테스트.

247 사k자 제n 데이타.

251 변환 및 대화 상|자 발췌.

290 시스템 로W! 데이타 로k.

291 직접 대화 LU 서비스 8성d소.

401 신뢰성 있는 일방 브래킷.

501 **CHANGE_SESSION_LIMIT** 명령n.

502 **ACTIVATE_SESSION** 명령n.

504 **DEACTIVATE_SESSION** 명령n.

505 **LU-*definition*** 명령n.

601 **MIN_CONWINNERS_TARGET** 매 3 변수.

602 **RESPONSIBLE(TARGET)** 매 3 변수.

603 **DRAIN_TARGET(NO)** 매 3 변수.

604 **FORCE** 매 3 변수.

605 LU-LU 세션 한 h .

606 로컬로 확인된 LU명.

607 미해석 LU명.

610 최대 RU 크 b 범위.

612 회선 f 합 성 x 세션 자동 활성화 한 h .

613 로컬 최대(LU, 모드) 세션 한 h .

616 CPSVCMG 모드명 지 x .

완전 양방향 VCB

0 전 g 방향 대화 ! 필 d 한 포맷 1 VCB ! 대한 정의를 식별하 m ^ 송 데이타를 송수신하려면 트랜잭션 프로 W 램은 WINAPPC.H 헤더 파일을 포함시키 b 전 ! WINAPPC_FORMAT_1이라 m 하는 컴파일러 상수를 정의해 _ 합니다. 이 M 은 다음 z 0 이 C p n 를 통해 수행할 수 있습니다.

```
#define WINAPPC_FORMAT_1
#include <winappc.h>
```

이 상수 ! 정의되지 J 을 f l, n 플리케이션 ! 서는 VCB의 포맷 0 버전 ! 만 W 세스할 수 있 T 됩니다.

대기행렬 레벨 비블럭화

퍼스널 통신 및 통신 서버 APPC API는 대 b 행렬 레벨 비블럭화를 지 x 합니다. 이 지 x 은 APPC # 트리 포인트를 통해 제 x 됩니다.

비블럭화 작 w 은 명령 n 의 처리를 즉시 0 료할 수 x 는 f l n 플리케이션으로 제 n 를 리턴할 수 있으므로 W n 플리케이션은 미처리 명령 n ! 0 료되 z 다는 통보 ! 있을 때 n 지 다른 처리를 h 속할 수 있습니다. 대 b 행렬 레벨 비블럭화는 n 플리케이션이 서로 다른 대 b 행렬 ! 대해 비블럭화 명령 n 를 발행하 m 퍼스널 통신 및 통신 서버 ! 의해 명령 n 를 동시 ! 처리하 T 할 수 있음을 의미합니다. n 플리케이션은 또한 명령 n ! 0 료되 b 를 b 다리지 J m 주 n 진 대 b 행렬 ! 대해 비블럭화 명령 n 를 , 속적으로 발행할 수 있습니다.

퍼스널 통신 및 통신 서버는 비블럭 명령 n 를 위해) 섯 3 의 대 b 행렬을 유지 | 리합니다.

- 할당 대 b 행렬(" 활동 트랜잭션 프로 W 램 k)
- 송신/수신 대 b 행렬(반 g 방향 대화만 대화당 하나)

- 송신 대b 행렬(O전 g방향 대화당 하나)
- 수신 대b 행렬(O전 g방향 대화당 하나)
- 송신 ^송 대b 행렬(대화당 하나)
- 수신 ^송 대b 행렬(대화당 하나)

)섯 !지 유형의 대b 행렬 모두 수! 제한 x이 명령n를 보유할 수 있습니다. 비블럭 명령n는 또다른(블럭 또는 비블럭) 명령n ! 퍼스널 통신 또는 통신 서버 프로W램! 의해 처리되m 있을 f l 대b행렬! 배치됩니다. 할당 대b행렬의 명령n는 동시! 처리되는 반면 다른 대b행렬의 명령n는 프로W램!서 수신한 순서대로 한 번! 하나? 처리됩니다.

n플리케이션은 **opext** 필드인 **AP_NON_BLOCKING**! 플래W를 설정하) 명령n를 비블럭 모드로 처리하m자 함을 퍼스널 통신 또는 통신 서버! K립니다. n플리케이션은 비동b식 명령n O료를 통보하는 데 사k되는 비블럭 명령n ! 이벤트 핸들을 제x할 수 있습니다. 이 핸들은 **SECONDARY_RC** 필드의 퍼스널 통신 및 통신 서버! 전달됩니다. 핸들이 지정되지 J은 f l, n플리케이션!는 W 대b행렬의 다음 명령n ! 핸들이 O료되z음을 지정할 때 명령n ! O료된 M으로 통보됩니다.

핸들이 x는 모든 선행 명령n는 핸들을 지정하지 J은 동일한 대b행렬의 명령n ! O료된 후 이벤트! 신호! 보내지면 O료됩니다.

비블럭 명령n ! 플래W **AP_OPERATION_INCOMPLETE_FLAG**를 리턴하면 **opext** 필드! 설정됩니다.

할당 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)ALLOCATE

(MC_)SEND_CONVERSATION

송신/수신 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)CONFIRM

(MC_)CONFIRMED

(MC_)DEALLOCATE

(MC_)FLUSH

(MC_)PREPARE_TO_RECEIVE

(MC_)RECEIVE_AND_WAIT

(MC_)RECEIVE_IMMEDIATE

(MC_)SEND_DATA

(MC_)SEND_ERROR

(O전 g방향 대화를 위한) 송신 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)DEALLOCATE

(MC_)FLUSH

(MC_)SEND_DATA

(MC_)SEND_ERROR

(0전 g방향 대화를 위한) 수신 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)RECEIVE_AND_WAIT

(MC_)RECEIVE_IMMEDIATE

(0전 g방향 대화를 위한) 수신-^송 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)RECEIVE_EXPEDITED_DATA

송신-^송 대b행렬!서 비블럭 모드로 실행할 수 있는 APPC 명령n로는 다음이 있습니다.

(MC_)REQUEST_TO_SEND

(MC_)SEND_EXPEDITED_DATA

다음z 0은 APPC 명령n는 반드시 비동b식으로 처리되지만 대b행렬z ,|되지는 J습니다.

(MC_)RECEIVE_AND_POST

(MC_)TEST_RTS_AND_POST

비블럭 모드로 실행할 수 x는(또한 n플리케이션이 비블럭 플래W를 설정하는 f l비블 N 실

령 n 로 취 ^ 되지만 **TEST_RTS_AND_POST**는 동일한 대화 ! 다른 미처리 명령이 있는 f l M 비블럭 명령 n 대 b 행렬 ! 배치되지 J 은 f l ! 도 발행할 수 있습니다. 동일한 대 b 행렬 ! 명령 n ! x 는 f l ! 발행된 비블럭 명령 n 는 다른 대 b 행렬 ! 명령 n ! 있을지라도 정상대로 처리됩니다. TEST_RTS, GET_ATTRIBUTES, GET_STATE 및 GET_TYPE은 대 b 행렬 z | 련이 x 으며 p 제라도 실행이 ! 능하 m 절대 AP_TP_BUSY를 리턴하지 J 습니다.

디폴트 로컬 LU

퍼스널 통신 및 통신 서버는 종속 및 독립 LU 6.2 ! 대해 디폴트 로컬 LU 를 지 x 합니다. 디폴트 LU는 **lu_alias** 필드 ! x 백인 상태로 **TP_STARTED** 명령 n (90페이지의 『TP_STARTED』 참조) ! 발행된 f l 사 k 됩니다. 독립 LU 6.2의 f l, 디폴트 LU는 제 n 점(CP) LU입니다. 종속 LU 6.2의 f l, 로컬 LU 풀이 사 k 됩니다. **DEFINE_LOCAL_LU** 명령 n ! | 한 자세한 내 k 은 *System Management Programming*을 참조하십시 @. 퍼스널 통신 및 통신 서버는 디폴트 풀 ! 서 LU를 선택하 E 나 다음 z 0 이 제 n 점(CP) LU 를 사 k 합니다.

- LU ! 디폴트 로컬 LU 풀의 멤버로 8 성된 f l, 퍼스널 통신 및 통신 서버는 사 k 되지 J 는 풀 ! 서 LU를 선택합니다. 풀의 모든 LU ! 사 k 되는 f l, **TP_STARTED** 명령 n 는 실패합니다.
- 디폴트 로컬 LU 풀의 멤버로 8 성된 LU ! x 는 f l, 퍼스널 통신 및 통신 서버는 제 n 점(CP) LU를 사 k 합니다.

다음 정보는 통신 서버 Windows 95 및 Windows NT SNA API 클라이 p 트 ! 만 적 k 됩니다.

" 사 k 자 ! 대한 디폴트 로컬 LU 별명은 INI 8 성 또는 LDAP 중 적절한 8 성 유틸리티를 사 k 하) 지정할 수 있습니다.

APPC 프로 W 램은 직접 별명을 지정하 b 보다는 디폴트 로컬 LU 별명을 사 k 하도록 선택할 수 있습니다. **로컬 LU 별명** 필드 ! 2진 0으로 설정된 상태 ! 서 APPC 프로 W 램이 **TP_START** 명령 n 를 발행하면 APPC API는 8 성된 디폴트 로컬 LU 별명을 사 k 합니다.

QEL/MU 지원

이 M 은 통신 서버 SNA API 클라이 p 트 ! 만 적 k 됩니다.

통신 서버는 3270 ! 뮬레이션을 위해 Novell의 Queue Element/Message Unit(QEL/MU) 8 조를 8 현하는 ! 뮬레이터 소프트 ~ n 패키지를 실행하는

IPX- 또는 TCP/IP-접속 클라이p트! 대한 지x을 제x하m 있습니다.)b!는 전k, 풀 및 xk LU 범주(자x 유형이라m도 함)M 0은 일반적인 클라이p트 b능z 로드 U형 유지를 위한 지x이 포함됩니다.

QEL/MU !뮬레이터 소프트~n 패키지 8현! |한 내k은 *Novell Netware for SAA 3270 Client Interface Guide and Reference P/N 100-00218-001*을 참조하십시@.

제6장 CPI-C 프로그램 구현

이 장!는 CPI-C 인터페이스! 대한 퍼스널 통신 및 통신 서버 지x 내* 이 명시되n 있습니다.)b서는 다음z 0은 5*을 다루m 있습니다.

- CPI-C 프로W램을 컴파일 및 링크하는 b법
- CPI-C 프로W램을 준비 및 실행하는 방법
- 퍼스널 통신 및 통신 서버! 지x하는 CPI-C 버전의 특징

퍼스널 통신 및 통신 서버!서의 CPIC 8현은 Windows bh상의 Microsoft** NT SNA 서버M 호환!능한 2진이 되도록 mH되z으며 OS/2 Communications Manager/2 버전 1.0의 CPIC 인터페이스 8현z 유사합니다.

주: 다음 시스템!서 제x하는 CPIC API! |한 내k이 이 장! 들n 있습니다.

- Windows NT!서 실행되는 통신 서버
- 통신 서버/NTM 함께 제x되는 OS/2, Windows NT, Windows 95 및 Windows 3.1k SNA API 클라이p트
- Windows 95 및 Windows NTk 퍼스널 통신

이들 시스템!서 제x하는 지x! 상이한 점이 있으면 W 내k이 명시됩니다.

CPIC 프로그램 작:

퍼스널 통신 및 통신 서버는 CPIC 호출을 처리하는 동적 링크 라이브러리 (DLL) 파일을 제x합니다.

DLL은 재입력이 !능합니다. 즉,)러 3의 n플리케이션 프로세스M 스레드! DLL을 동시! 호출할 수 있습니다.

49페이지의 표9!는 제x된 헤더 파일의 소스 모듈 사k 9M CPIC 프로W램을 컴파일하) 링크하는데 필d한 라이브러리! 제시되n 있습니다. 몇몇 헤더 파일!는 다른 필수 헤더 파일이 포함될 수 있습니다.

표 9. CPICk 헤더 파일z 라이브러리

운영체제	헤더 파일	라이브러리	DLL명
WINNT & WIN95	WINCPIC.H	WCPIC32.LIB	WCPIC32.DLL
WIN3.1	WINCPIC.H	WINCPIC.LIB	WINCPIC.DLL
OS/2	CPIC_C.H	CPIC16.LIB 또는 CPIC32.LIB	CPIC.DLL

*WINNT = Windows NT
*WIN95 = Windows 95
*WIN3.1 = Windows 3.1

CPI-C 버전

CPI-C 인터페이스!는)러 번의 버전 변f z 확장이 있z습니다. 다음의 두 !지 이유로 이러한 버전! 대해 K m 있n _ 합니다.

- b존 시스템을 유지하E나 이식하m 있는 f l, 이식 !능한 b능 호출은 n떤 M인지M 버전을 변f할 f l n떤 M을 변f해_ 하는지 K F _ 합니다.
- 새로n 프로W램을 작성하m 있는 f l, 특정 버전! 종속하는 코드를 p제 작성할지! 대해 K F _ 합니다.

CPI-C 순응 클래스 지원

IBM 문서 *Common Programming Interface Communications CPI-C Reference* 버전 2.1(SC26-4399-08)! 정의된 M처럼 다음z 0은 CPI-C 2.1 순응 클래스! 지x됩니다.

통신 서버 클라이p트!서 지x하지 J는 클래스! 대해서는 이 장 전체!서 노트북 *F이콘*을 참조하십시@.

이 F이콘은 중d 정보를 나타냅니다.

대화 순응 클래스는 프로W램이 반 g방향 대화를 시작 및 종료할 수 있도록 합니다.

시작자 세트 호출:

CMACCP
Accept_Conversation

CMALLC
Allocate

CMDEAL
Deallocate

CMINIT
Initialize_Conversation

CMRCV
Receive

CMSEND
Send_Data

확장 b능 호출:

CMCFM
Confirm

CMCFMD
Confirmed

CMECS
Extract_Conversation_State

CMECT
Extract_Conversation_Type

CMEMBS
Extract_Maximum_Buffer_Size

CMEMN
Extract_Mode_Name

CMESL
Extract_Sync_Level

CMFLUS
Flush

CMPTR
Prepare_To_Receive

CMRTS
Request_To_Send

CMSERR
Send_Error

CMSCT
Set_Conversation_Type

CMSDT
Set_Deallocate_Type

CMSF
Set_Fill

CMSLD
Set_Log_Data

CMSMN
Set_Mode_Name

CMSPTR
Set_Prepare_To_Receive_Type

CMSRT
Set_Receive_Type

CMSRC
Set_Return_Control

CMSST
Set_Send_Type

CMSSL
Set_Sync_Level

d 8 되는 동 b 레벨 *:

CM_NONE 또는 CM_CONFIRM

CMSTPN

Set_TP_Name

CMTRTS

Test_Request_To_Send_Received

LU 6.2 순응 클래스는 프로 W 램이 LU 6.2 특정 서비스를 이 k 할 수 있 T 합니다.

CMEPLN

Extract_Partner_LU_Name

CMSED

Set_Error_Direction

CMSPLN

Set_Partner_LU_Name

대화 레벨 비블럭 순응 클래스는 호출이 즉시 O 료될 수 x 는 f l 프로 W 램이 제 n 를 재획득할 수 있도록 합니다.

CMCANC

Cancel_Conversation

CMSPM

Set_Processing_Mode

CMWAIT

Wait_For_Conversation

서버 순응 클래스는 프로 W 램이 CPI-C를 사 k 하)) 러 3 의 트랜잭션 프로 W 램명을 등록하 m 다중 입력 대화를 수 k 하며 서로 다른 클라이 p 트 ! 대한 8 문을 | 리할 수 있 T 합니다.

CMACCI

Accept_Incoming

CMECTX

Extract_Conversation_Context

통신 서버 Windows 3.1 클라이 p 트 ! 는 CMECTX Extract_Conversation_Context ! 지 x 되지 J 습니다.

CMETPN

Extract_TP_Name

CMRLTP

Release_Local_TP_Name

CMINIC

Initialize_For_Incoming

CMSLTP

Specify_Local_TP_Name

데이타 변환 순응 클래스 루틴은 프로W램이 로컬 루틴을 호출하) 로컬 인코딩!서 EBCDIC으로 또는 W 반대로 문자-의 인코딩을 변f할 수 있도록 합니다.

CMCNVI

Convert_Incoming

CMCNVO

Convert_Outgoing

보안 순응 클래스는 프로W램이 부! 정보의 W세스 보H 정보를 사k하는 대화를 설정하E나 프로W램! 의해 직접 설정할 수 있T 합니다.

CMESUI

Extract_Security_User_ID

CMSCSP

Set_Conversation_Security_Password

CMSCST

Set_Conversation_Security_Type

d8되는 대화 보H 유형 *:

CM_SECURITY_NONE

CM_SECURITY_PROGRAM

CM_SECURITY_PROGRAM_STRONG

CM_SECURITY_SAME

CMSCSU

Set_Conversation_Security_User_ID

대기행렬 레벨 비블럭: 호출이 O료되지 JR을 때 제n를 재획득하려는 fl.

CMCANC

Cancel_Conversation

CMSQPM

Set_Queue_Processing_Mode

CMWCMP

Wait_For_Completion

통신 서버 Windows 3.1 클라이p트!는 대b행렬 레벨 비블럭화! 지x되지 J습니다.

소환 함수: 호출을 O료할 수 x을 때 제n를 재획득하려는 fl.

CMCANC

Cancel_Conversation

CMSQCF

Set_Queue_Callback_Function

통신 서버 Windows 3.1 또는 OS/2 클라이 p 트 ! 는 소환 함수 ! 지 x 되지 J 습니다.

2차 정보는 2차 @ 류 리턴 정보를 발췌할 수 있도록 합니다.

CMESI

Extract_Secondary_Information

통신 서버 Windows 3.1 클라이 p 트 ! 는 2차 정보 ! 지 x 되지 J 습니다.

다음 클래스는 OS/2 및 Windows 3.1 k 통신 서버 SNA API 클라이 p 트 ! 지 x 되지 J 습니다.

완전 양방향은 0 전 g 방향 대화 ! 대한 사 k 자 W 세스를 허 k 합니다.

CMESRM

Extract_Send_Receive_Mode

CMSSRM

Set_Send_Receive_Mode

급송 데이타는 상대방 프로 W 램 z ^ 송 데이타를 3 환합니다.

CMRCVX

Receive_Expedited_Data

CMSNDX

Send_Expedited_Data

다음의 순응 클래스는 지 x 되지 J 습니다.

OSI TP 서비스

회복 ! 능 트랜잭션(자 x 복 8 인터페이스 k)

비 , 속적 트랜잭션(회복 ! 능 트랜잭션 k)

분산 보 H(분산 보 H 서버의 사 k 자 보 H 서비스)

디렉토리(분산 디렉토리 ! 저장된 사 k 자 지정 정보)

CPI-C 함수

퍼스널 통신 및 통신 서버 ! 지 x 하는 모든 CPI-C 함수 ! 55페이지의 표 10 ! 나 - 되 n 있습니다. b 존 프로 W 램을 유지 | 리하 m 있 E 나 b 존 시스템 z 호환 상태를 유지해 _ 하는 새로 n 프로 W 램을 작성하는 f l 이 표를 참조하십시 @.

주: MS Windows SNA API 클라이언트 CPI-C 애플리케이션을 작성하는 경우, Accept_Conversation(cmaccp) 호출을 통해 입력 대화를 받기 전에 Specify_Local_TP-Name(cmsltp) 호출을 통해 로컬 트랜잭션 프로그램을 지정하십시오.

표 10. 퍼스널 통신 및 통신 서버 클라이언트가 지원하는 CPI-C 함수

함수	전체 이름	Windows NT 서버 및 퍼스널 통신	Windows 95 및 Windows NT 클라이언트	OS/2 클라이언트	Windows 3.1 클라이언트
cmaccp	Accept_Conversation	x	x	x	x
cmacci	Accept_Incoming	x	x	x	x
cmallc	Allocate	x	x	x	x
cmcanc	Cancel_Conversation	x	x	x	x
cmcfm	Confirm	x	x	x	x
cmcfmd	Confirmed	x	x	x	x
cmcnvi	Convert_Incoming	x	x	x	x
cmcnvo	Convert_Outgoing	x	x	x	x
cmdeal	Deallocate	x	x	x	x
xcmdsi	Delete_CPIC_Side_Information	x	-	-	-
cmectx	Extract_Conversation_Context	x	x	x	-
xcecst	Extract_Conversation_Security_Type	x	x	x	x
cmecst	Extract_Conversation_Security_Type	x	x	x	x
cmecs	Extract_Conversation_State	x	x	x	x
cmect	Extract_Conversation_Type	x	x	x	x
xcmesi	Extract_CPIC_Side_Information	x	x	x	x
cmembs	Extract_Maximum_Buffer_Size	x	x	x	x
cmemn	Extract_Mode_Name	x	x	x	x
cmepln	Extract_Partner_LU_Name	x	x	x	x
cmesi	Extract_Secondary_Information	x	x	x	-
cmesui	Extract_Security_User_ID	x	x	x	x
cmecsu	Extract_Security_User_ID	x	x	x	x
xcecsu	Extract_Security_User_ID	x	x	x	x
cmesrm	Extract_Send_Receive_Mode	x	x	-	-
cmesl	Extract_Sync_Level	x	x	x	x
xceti	Extract_TP_ID	x	x	x	-
cmetpn	Extract_TP_Name	x	x	x	x
cmflus	Flush	x	x	x	x
cminit	Initialize_Conversation	x	x	x	x
xcinct	Initialize_Conversation_For_TP	x	x	x	-
cminic	Initialize_For_Incoming	x	x	x	x
cmptr	Prepare_To_Receive	x	x	x	x
cmrcv	Receive	x	x	x	x
cmrcvx	Receive_Expedited	x	x	-	-
cmrltp	Release_Local_TP_Name	x	x	x	x
cmrts	Request_To_Send	x	x	x	x
cmsend	Send_Data	x	x	x	x
cmsndx	Send_Expedited	x	x	-	-

표 10. 퍼스널 통신 및 통신 서버 클라이언트 지원하는 CPI-C 함수 (계속)

함수	전체 이름	Windows NT 서버 및 퍼스널 통신	Windows 95 및 Windows NT 클라이언트	OS/2 클라이언트	Windows 3.1 클라이언트
cmserr	Send_Error	x	x	x	x
cmscsp	Set_Conversation_Security_Password	x	x	x	x
xcscsp	Set_Conversation_Security_Password	x	x	x	x
cmscst	Set_Conversation_Security_Type	x	x	x	x
xcscst	Set_Conversation_Security_Type	x	x	x	x
cmscsu	Set_Conversation_Security_User_ID	x	x	x	x
xcscsu	Set_Conversation_Security_User_ID	x	x	x	x
cmsct	Set_Conversation_Type	x	x	x	x
xcmssi	Set_CPIC_Side_Information	x	-	-	-
cmsdt	Set_Deallocate_Type	x	x	x	x
cmsed	Set_Error_Direction	x	x	x	x
cmsf	Set_Fill	x	x	x	x
cmsld	Set_Log_Data	x	x	x	x
cmsmn	Set_Mode_Name	x	x	x	x
cmspln	Set_Partner_LU_Name	x	x	x	x
cmsptr	Set_Prepare_To_Receive_Type	x	x	x	x
cmspm	Set_Processing_Mode	x	x	x	x
cmsqcf	Set_Queue_Callback_Function	x	x	x	-
cmsqpm	Set-Queue_Processing_Mode	x	x	x	-
cmsrt	Set_Receive_Type	x	x	x	x
cmsrc	Set_Return_Control	x	x	x	x
cmssrm	Set_Send_Receive_Mode	x	x	-	-
cmsst	Set_Send_Type	x	x	x	x
cmssl	Set_Sync_Level	x	x	x	x
cmstpn	Set_TP_Name	x	x	x	x
cmsltp	Specify_Local_TP_Name	x	x	x	x
xchwnd*	Specify_Windows_Handle	x	x	-	x
xcstp	Start_TP	x	x	x	-
cmtrts	Test_Request_To_Send_Received	x	x	x	x
cmwcmp	Wait_For_Completion	x	x	x	-
cmwait	Wait_For_Conversation	x	x	x	x
xcendt	End_TP	x	x	x	-
WinCPICCleanup*		x	x	-	x
WinCPICIsBlocking*		-	-	-	x
WinCPICSetBlockingHook*		-	-	-	x
WinCPICStartup*		x	x	-	x
WinCPICUnhookBlockingHook*		-	-	-	x

***는:** Microsoft Windows 용 WOSA 함수를 나타냅니다.
x는: 지원되는 함수를 나타냅니다.
-는: 지원되지 않는 함수를 나타냅니다.

- 비스 TP명 지정

이 b능은 통신 서버 SNA API 클라이p트!만 지x됩니다.

CMSTPN 및 CMSLTP 함수를 사k하) 서비스 트랜잭션 프로W램명(TPN)을 지정하는 f l 특별한 T `을 따라_ 합니다. 보통 CPI-C 함수!는 표준 TP를 지정하십시@. 서비스 트랜잭션 프로W램은 다른 프로W램이나 사k자! T x통의 네트v크M 시스템 서비스를 제x하는 특화된 트랜잭션 프로W램입니다. 서비스 트랜잭션 프로W램의 9로는 프로W램 표, 디렉토리 서비스 및 스풀러! 있습니다.

CMSTPN 및 CMSL 트랜잭션 프로W램 함수를 사k하) 서비스 트랜잭션 프로W램명을 지정하b 위한 T `은 다음z 0습니다.

- 2-5바이트의 ASCII 문자를 사k하) 이름을 지정하십시@.
- 2바이트의 ASCII 문자를 사k하) 이름의 첫번째 바이트(9: 0x23)를 지정하십시@.
  - 이름의 첫번째 바이트를 두 부분으로 나누m(9: 2M 3),) b! " ASCII 바이트의 저순위 부분을 지정하십시@.
  - " ASCII 바이트의 m순위 부분을 1로 설정하십시@. 이는 사k자! 서비스 TP명을 지정하m 있음을 나타냅니다. 9!서는 지정된 처음 두 바이트! 0x12M 0x13입니다.
- 이름의 세 바이트! 남F 있는 0을 ASCII 문자로 지정하십시@(9: 007).

W러므로 서비스 트랜잭션 프로W램명 0x23 007을 0x12 0x13 007로 지정하십시@.

제7장 APPC 엔트리 포인트

다음 절 ! 서는 APPC ! 대한 프로시듀 n # 트리 포인트 ! 대해 설명합니다.

주: 다음 시스템 ! 서 제 x 하는 APPC API ! | 한 내 k 은 이 책의 제1부 ! 있는) 러 장 ! 들 n 있습니다.

- Windows NT ! 서 실행되는 통신 서버
- 통신 서버/NT M 함께 제 x 되는 OS/2, Windows NT, Windows 95 및 Windows 3.1 k SNA API 클라이 p 트
- Windows 95 및 Windows NT k 퍼스널 통신

이들 시스템 ! 서 제 x 하는 지 x ! 상이한 점이 있으면 W 내 k 이 명시됩니다.

APPC

모든 APPC 명령n ! 대한 #트리 포인트로 이M을 사k할 수 있습니다. 대H으로 이 #트리 포인트를 사k하) 2차 리턴 코드 필드 ! 이벤트 핸들을 놓m **opext** 필드(AP_NON_BLOCKING) ! 대b행렬 레벨 비블럭 플래W를 설정하) 비블럭 명령n를 발행할 수 있습니다.

구문

```
void WINAPI APPC(long)
```

입력은 명령n 제n 블럭 ! 대한 포인터입니다.

리턴 값

" @류 ! 대해 1차 리턴 코드M 2차 리턴 코드를 K사하십시@.

사용시 주의사항

| 련 참조: 64페이지의 『WinAsyncAPPCEx()』.

APPC는 SNA API OS/2 클라이p트 ! 지x되는 유일한 #트리 포인트입니다.

WinAsyncAPPC()

이 M이 모든 APPC 명령 n ! 대한 비동 b #트리 포인트입니다. n 플리케이션은 Windows 메세지를 통해 O 료되 z 음이 통보되도록 선택한 f l 이 #트리 포인트를 사 k 합니다. 퍼스널 통신 및 통신 서버는 b 존 n 플리케이션 z 의 호환을 위해 이 #트리 포인트를 제 x 합니다.

구문

```
HANDLE WINAPI WinAsyncAPPC(HWND hWnd,
                           long vcb)
```

매 3 변수

	설명
hwnd	O 료 메세지를 수신하 b 위한 창 핸들.
vcb	명령 n 제 n 블럭 ! 대한 포인터.

리턴 값

리턴 * 은 비동 b d 청이 성 x 적으로 O 료되 z 는지) 부를 지정합니다. d 청이 성 x 적인 f l, 실제 리턴 * 은 핸들입니다. 함수 ! 성 x 적인 f l, 퍼스널 통신 및 통신 서버는 0을 리턴합니다.

사용시 주의사항

블럭화할 수 있는 APPC 명령 n 는 다음 z 0 습니다.

- **[MC_]ALLOCATE**
- **[MC_]CONFIRM**
- **[MC_]CONFIRMED**
- **[MC_]DEALLOCATE**
- **[MC_]FLUSH**
- **[MC_]PREPARE_TO_RECEIVE**
- **RECEIVE_ALLOCATE**
- **[MC_]RECEIVE_AND_WAIT**
- **[MC_]RECEIVE_EXPEDITED_DATA**
- **[MC_]REQUEST_TO_SEND**
- **[MC_]SEND_CONVERSATION**
- **[MC_]SEND DATA**
- **[MC_]SEND_ERROR**
- **[MC_]SEND_EXPEDITED_DATA**
- **TP_ENDED**
- **TP_STARTED**

WinAsyncAPPC #트리 포인트는 명령n를 취소할 수 있도록 하지만 대b행렬 레벨 비블럭화는 지x하지 J습니다. APPC #트리 포인트는 대b행렬 레벨 비블럭화를 지x하지만 n플리케이션이 명령n를 취소할 수는 x습니다.

이 #트리 포인트는 대b행렬 레벨 비블럭화를 지x하지 J습니다. 대b행렬 레벨 비블럭화 플래W AP_NON_BLOCKING이 비동b 인터페이스! 지정되는 fl, 퍼스널 통신 및 통신 서버는 이를 무시합니다. 비동b #트리 포인트를 사k하는 fl, n플리케이션은 한 번! 하나의 대화!서만 미처리 함수를 처리할 수 있습니다. 두 번째 함수를 시작하려 하면 @류 코드 AP_CONV_BUSY! 초래됩니다. 이벤트 핸들을 통해 비동b 0료! n플리케이션! 통보되T 해_ 하는 fl, n플리케이션은 **WinAsyncAPPCEx** 또는 **APPC** #트리 포인트를 사k할 수 있습니다. 9\로 **RECEIVE_AND_POST**M **RECEIVE_AND_WAIT**! 있습니다. 비동b 지x을 0전히 사k할 수 있T 하b 위해 퍼스널 통신 및 통신 서버는 비동b로 발행된 **RECEIVE_AND_WAIT** 명령n! **RECEIVE_AND_POST** 명령n처럼 작k하도록 f로를 보냅니다. 특히 비동b인 **RECEIVE_AND_POST** 또는 **RECEIVE_AND_WAIT**! 미처리 상태인 fl, n플리케이션은 동일한 대화!서 다음 명령n를 발행할 수 있습니다.

- **REQUEST_TO_SEND**
- **GET_TYPE**
- **GET_ATTRIBUTES**
- **TEST_RTS**
- **DEALLOCATE**(AP_ABEND_PROG, AP_ABEND_SVC 또는 AP_ABEND_TIMER)
- **SEND_ERROR**
- **TP_ENDED**

서버M 0은 n플리케이션이 비동b **RECEIVE_AND_WAIT**를 사k하) 데이타를 수신할 수 있T 합니다. **RECEIVE_AND_POST** 또는 **RECEIVE_AND_WAIT**! 미처리 상태일 때!도 n플리케이션은 **SEND_ERROR** 및 **REQUEST_TO_SEND**를 사k할 수 있습니다.

비동b 작w이 0료되면 n플리케이션의 창 *hWnd*!는 『**WinAsyncAPPC**』를 입력 문자-로 하) **RegisterWindowMessage**! 리턴한 메세지! 수신됩니다. *wParam* 인수!는 x래의 함수 호출! 의해 리턴된 비동b 타스크 핸들이 포함됩니다. *lParam* 인수!는 x래의 VCB 포인터! 포함되며 최종 리턴 코드를 판별하는 데 사k할 이 인수를 사k할 수 있습니다.

WinAPPCCancelAsyncRequest는 n플리케이션이 비동b APPC 조치를 취소할 수 있T 하지만 |련 대화나 트랜잭션 프로W램을 적절한 M으로 종료합니다. 미처리 작w은 AP_CANCELED를 리턴 코드로 하) 리턴됩니다.

함수! 성x적으로 리턴되면 퍼스널 통신 및 통신 서버는 작w이 0료되E나 대화! 취소될 때 **WinAsyncAPPC()** 메세지를 n플리케이션! 통지합니다.

|련 참조:

64페이지의 『WinAsyncAPPCEx()』.

66페이지의 『WinAPPCCancelAsyncRequest()』.

WinAsyncAPPCEx()

이M이 모든 APPC 명령n ! 대한 비동b #트리 포인트입니다.)러 세션이 동일한 스레드 !서 처리되T 하려면 이 호출을 사k하십시@.

이벤트를 통해 O료 ! n플리케이션 ! 통보되m 사k자 n플리케이션 ! 미처리 명령n를 취소할 G한이 필d한 f l 이 #트리 포인트를 사k하십시@. W렇지 J으면 APPC 대b행렬 레벨 비블럭 #트리 포인트를 사k하십시@.

구문

```
HANDLE WINAPI WinAsyncAPPCEx(HANDLE handle,
                             long vcb);
```

매3변수

설명

handle

n플리케이션이 처리하는 이벤트의 핸들.

vcb 명령n 제n 블럭 ! 대한 포인터.

리턴 값

리턴 *은 비동b d청이 성x적이z는지)부를 지정합니다. 함수 ! 성x적인 f l 실제 리턴 *이 핸들입니다. 함수 ! 성x적인 f l 퍼스널 통신 및 통신 서버는 0을 리턴합니다.

사용시 주의사항

이 명령n는 Win32** API !서 **WaitForMultipleObjects**M 함께 사k하b 위한 M입니다.

블럭화할 수 있는 APPC 명령n는 다음z 0습니다.

- **[MC_]ALLOCATE**
- **[MC_]CONFIRM**
- **[MC_]CONFIRMED**
- **[MC_]DEALLOCATE**
- **[MC_]FLUSH**
- **[MC_]PREPARE_TO_RECEIVE**
- **RECEIVE_ALLOCATE**
- **[MC_]RECEIVE_AND_WAIT**
- **[MC_]REQUEST_TO_SEND**
- **[MC_]SEND_CONVERSATION**
- **[MC_]SEND_DATA**
- **[MC_]SEND_ERROR**

- **TP_ENDED**
- **TP_STARTED**

이 #트리 포인트는 대b행렬 레벨 비블럭화를 지x하지 J습니다. 대b행렬 레벨 비블럭화 플래W AP_NON_BLOCKING이 비동b 인터페이스! 지정되는 f l, 퍼스널 통신 및 통신 서버는 이를 무시합니다. 비동b #트리 포인트를 사k하는 f l, n플리케이션은 한 번! 하나의 대화!서만 미처리 함수를 처리할 수 있습니다. 두 번째 함수를 시작하려 하면 @류 코드 AP_CONV_BUSY! 초래됩니다.

WinAsyncAPPCEx #트리 포인트는 명령n를 취소할 수 있도록 하지만 대b행렬 레벨 비블럭화는 지x하지 J습니다. **APPC** #트리 포인트는 대b행렬 레벨 비블럭화를 지x하지만 n플리케이션이 명령n를 취소할 수는 x습니다. 이의 9\로 **RECEIVE_AND_POST**M **RECEIVE_AND_WAIT**! 있습니다. 비동b 지x을 O전히 사k할 수 있T 하b 위해 퍼스널 통신 및 통신 서버는 비동b로 발행된 **RECEIVE_AND_WAIT** 명령n! **RECEIVE_AND_POST** 명령n처럼 작k하도록 f로를 보냅니다. 특히 비동b인 **RECEIVE_AND_POST** 또는 **RECEIVE_AND_WAIT**! 미처리 상태인 f l, n플리케이션은 동일한 대화!서 다음 명령n를 발행할 수 있습니다.

- **REQUEST_TO_SEND**
- **GET_TYPE**
- **GET_ATTRIBUTES**
- **TEST_RTS**
- **DEALLOCATE**(AP_ABEND_PROG, AP_ABEND_SVC 또는 AP_ABEND_TIMER)
- **SEND_ERROR**
- **TP_ENDED**

이는 n플리케이션 특히 서버 n플리케이션이 비동b **RECEIVE_AND_WAIT**를 사k하) 데이타를 수신할 수 있T 합니다. **RECEIVE_AND_POST** 또는 **RECEIVE_AND_WAIT**! 미처리 상태일 때!도 n플리케이션은 **SEND_ERROR** 및 **REQUEST_TO_SEND**를 사k할 수 있습니다.

비동b 작w이 O료되면 퍼스널 통신 및 통신 서버는 이벤트를 통한 신호로 n플리케이션! 이를 통보합니다. n플리케이션은 신호를 수신하면 @류 조G! 대한 1차 리턴 코드 및 2차 리턴 코드를 K사합니다.

|련 참조:

61페이지의 『WinAsyncAPPC()』.

66페이지의 『WinAPPCCancelAsyncRequest()』.

60페이지의 『APPC』.

WinAPPCCancelAsyncRequest()

이 함수는 미처리의 **WinAsyncAPPC** 방식 d청을 취소합니다.

구문

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

매3변수

설명

handle

제x되는 매3변수. 취소할 d청의 핸들을 지정합니다.

리턴 값

리턴 *은 비동b d청이 취소되z는지)부를 지정합니다. *이 0이면 퍼스널 통신 및 통신 서버는 d청을 취소하4습니다. W렇지 J으면 *은 다음 @류 코드 중 하나입니다.

WAPPCINVALID

지정된 비동b 타스크 ID! 유효하지 J습니다.

WAPPCALREADY

취소할 비동b 루틴이 이미 0료되z습니다.

사용시 주의사항

n플리케이션 프로W램은 **WinAPPCCancelAsyncRequest()** 호출을 발행한 후 비동b 이벤트를 핸들의 최초 함수! 의해 리턴된 M으로 지정하) 0료되b 전! **WinAsyncAPPC** 함수들 중 하나를 사k하) 발행된 비동b 타스크를 취소할 수 있습니다.

미처리 명령n! 대화M |련된 fl(9: **SEND_DATA** 또는 **RECEIVE_AND_WAIT**), 퍼스널 통신 및 통신 서버는 명령n를 제E하m 세션을 비활성화합니다. 명령n! 트랜잭션 프로W램z |련된 fl(9: **RECEIVE_ALLOCATE** 또는 **TP_STARTED**), 퍼스널 통신 및 통신 서버는 트랜잭션 프로W램을 종료합니다. 두 fl 모두 퍼스널 통신 및 통신 서버! !능한 한 z끗하T 대화M 세션을 비활성화하더라도 송신 버퍼나 확인 대b 또는 다른 대b 조치를 비l지는 J습니다. 이 호출은 동b식입니다. 이전! b술된 처리! 0료되면 퍼스널 통신 및 통신 서버는 취소된 명령n! 대해 0료 메세지를 통지합니다.

b존의 비동b **WinAsyncAPPC** 루틴 취소 시도! WAPPCALREADY의 @류 코드M 함께 실패하더라도 x래 루틴은 이미 0료되z습니다. n플리케이션은 az 통보를 처리하E나 0료 통보를 처리하지 J습니다. APPC 대b행렬 레벨 비블럭화 #트리 포인트를 통해 발행된 비동b 명령n를 취소할 수는 x습니다.

| 련 참조: 61페이지의 『WinAsyncAPPC()』.

WinAPPCCancelBlockingCall()

이 함수는 스레드! 대한 미처리 블럭 작w을 취소합니다. 퍼스널 통신 및 통신 서버! 미처리 블럭화 호출을 취소하는 f l, AP_CANCELLED의 @ 류 코드! 생성됩니다. 블럭화 후크 함수 내!서만 이 호출을 사k하십시 @. 퍼스널 통신 및 통신 서버는 b존 n플리케이션z의 *호환을 위해 이 함수를 제x합니다.

구문

```
Int WINAPI WINAPPCCancelBlockingCall(void);
```

리턴 값

리턴 *은 취소 d청이 성x적이z는지)부를 지정합니다. *이 0이면 퍼스널 통신 및 통신 서버는 d청을 취소합니다. W렇지 J으면 다음 @류 코드! *입니다.

WAPPCINVALID
미처리 블럭화 호출이 x습니다.

사용시 주의사항

미처리 명령n! 대화M |련된 f l(9: **SEND_DATA** 또는 **RECEIVE_AND_WAIT**), 퍼스널 통신 및 통신 서버는 명령n를 제E하m 세션을 비활성화합니다. 명령n! 트랜잭션 프로W램z |련된 f l(9: **RECEIVE_ALLOCATE** 또는 **TP_STARTED**), 퍼스널 통신 및 통신 서버는 트랜잭션 프로W램을 종료합니다. 두 f l 모두 퍼스널 통신 및 통신 서버! !능한 한 z끗하T 대화M 세션을 비활성화하더라도 송신 버퍼나 확인 대b 또는 다른 대b 조치를 비l지는 J습니다. 이 호출은 동b식입니다. 이전! b술된 처리! O료되면 함수! 종료됩니다.

다중 스레드 n플리케이션은 미처리 상태의 블럭 작w을)러 3 !질 수 있지만 스레드당 하나만 허k됩니다. 다중 미처리 호출들을 8분하b 위해 **WinAPPCCancelBlockingCall()**은 현재의 미처리 작w이나 호출측 n플리케이션 스레드(있을 f l)를 취소합니다. W렇지 J으면 실패합니다. APPC는 작w이 미처리 상태일 때!는 호출측 n플리케이션 스레드를 일시중단합니다. a z적으로 비블럭 작w이 시작된 스레드는 n플리케이션이 **WinAPPCSetBlockingHook**를 사k하) 스레드! 대한 블럭화 후크를 이전! 등록해둔 f l! F니면 제n를 재획득할 수 x습니다. (W러므로 **WinAPPCCancelBlockingCall()**! 대한 호출을 발행할 수 x습니다.)

Windows, Windows 95 및 Windows NT SNA API 클라이p트!는 지x되지 J습니다.

WinAPPCCleanup()

이 함수는 APPC API로부터 n 플리케이션을 종료하 m 등록해제합니다.

구문

```
BOOL WINAPI WinAPPCCleanup(void);
```

리턴 값

리턴 * 은 등록해제 ! 성 x 적이 z 는지) 부를 지정합니다. * 이 0이 F 니면 퍼스널 통신 및 통신 서버는 n 플리케이션을 성 x 적으로 등록해제하 4 습니다. 퍼스널 통신 및 통신 서버 ! n 플리케이션을 등록해제하지 J 은 f l 0 * 을 리턴합니다.

사용시 주의사항

APPC API로부터 퍼스널 통신 및 통신 서버 n 플리케이션을 등록해제하려면 **WinAPPCCleanup()**을 사 k 하십시 @ .

퍼스널 통신 및 통신 서버는 F 직 활동중인 대화를 종 a 하 m 트랜잭션 프로 W 램을 종료합니다. 이 함수는 n 플리케이션이 속한 모든 트랜잭션 프로 W 램 ! 서 **TP_ENDED(HARD)**를 발행하는 M z 0 습니다.

| 련 참조: 71페이지의 『WinAPPCStartup()』.

WinAPPCIsBlocking()

이 함수는 스레드! 이전의 블럭 호출이 O료되b를 b다리면서 실행되m 있는지)부를 판별합니다. 퍼스널 통신 및 통신 서버는 b존 n플리케이션z의 *호환을 위해 이 함수를 제x합니다.

구문

```
BOOL WlNAPI WinAPPCIsBlocking(void);
```

리턴 *은 함수의 실행 az를 나타냅니다. *이 0이 F니면 미처리 블럭화 호출이 O료를 b다리m 있습니다. 0의 *은 미처리 블럭화 호출이 x음을 의미합니다.

사용시 주의사항

퍼스널 통신 및 통신 서버 DLL은 스레드당 둘 이상의 블럭화 호출을]지합니다. 따라서 둘 이상의 블럭화 호출이 있을 fl!는 AP_THREAD_BLOCKING를 리턴합니다. 블럭화 호출을 실행하는 스레드는 블럭화 후크 함수! 설정되지 J으면 재입력되지 J습니다. 이 fl **WinAPPClsBlocking**은 블럭화 후크 함수 내!서만 참의 *을 리턴합니다.

|련 참조:

68페이지의 『WinAPPCCancelBlockingCall()』.

72페이지의 『WinAPPCSetBlockingHook()』.

74페이지의 『WinAPPCUnhookBlockingHook()』.

Windows 95 및 Windows NT SNA API 클라이p트!는 지x되지 J습니다.

WinAPPCStartup()

이 함수는 n플리케이션이 d 8되는 퍼스널 통신 및 통신 서버의 버전을 지정하m 퍼스널 통신 및 통신 서버 ! 서 버전 정보를 K색할 수 있도록 합니다. 이 호출은 필수 항목이 F닙니다.

구문

```
int WINAPI WinAPPCStartup(WORD wVersionRequired,
                          LPWAPPCDATA wappcdata);
```

매 3 변수

설명

wVersionRequired

d 8되는 퍼스널 통신 및 통신 서버 지x의 버전을 지정합니다. m 순위 바이트는 별로 중d하지 J은 버전(3정) 번호를 지정합니다. 저순위 바이트는 주d 버전 번호를 지정합니다.

wappcdata

APPC API의 버전z API 8현의 설명을 리턴합니다.

리턴 값

리턴 *은 퍼스널 통신 및 통신 서버 ! n플리케이션을 등록하4는지)부 M 지정된 버전 번호를 지x할 수 있는지)부를 나타냅니다. *이 0이면 퍼스널 통신 및 통신 서버는 지정된 버전을 지x하m n플리케이션을 등록합니다. W렇지 J으면 다음 * 중 하나 ! 리턴됩니다.

WAPPCSYSNOTREADY

b초 ! 되는 네트v크 하위 시스템이 네트v크 통신k으로 준비되지 J R습니다.

WAPPCVERNOTSUPPORTED

특정 퍼스널 통신 또는 통신 서버 8현이 d청된 퍼스널 통신 또는 통신 서버 지x의 버전을 지x하지 J습니다.

WAPPCINVALID

퍼스널 통신 및 통신 서버 ! 지정된 버전을 판별할 수 x습니다.

사용시 주의사항

WinAPPCStartup()은 향후 발표될 APIM의 호환을 돕b 위한 M입니다. 이 DLL은 버전 1.0을 지x합니다.

| 련 참조: 69페이지의 『WinAPPCCleanup()』.

WinAPPCSetBlockingHook()

이 함수는 APPC API의 APPC 8현!서 APPC 함수 호출을 블럭할 수 있도록 합니다.

퍼스널 통신 및 통신 서버는 b존 n플리케이션z의 호환을 위해 이 함수를 제x합니다.

구문

```
FARPROC WINAPI WinAPPCSetBlockingHook(FARPROC IpBlockFunc);
```

매3변수
설명

IpBlockFunc
설치할 블럭화 함수의 프로시듀n 인스턴스 주소를 지정합니다.

리턴 값

리턴 *은 이전! 설치된 블럭화 함수의 프로시듀n 인스턴스를 !리킵니다. **SetBlockingHook** 함수를 호출하는 n플리케이션이나 라이브러리는 필d시 복x할 수 있도록 이 리턴 *을 보|해_ 합니다. (내포! 중d하지 J을 fl, n플리케이션은 **WinAPPCSetBlockingHook()**! 의해 리턴된 *을 폐b하m az적으로 **WinAPPCUnhookBlockingHook**를 사k하) 디폴트 메카니즘을 복x합니다.)

사용시 주의사항

블럭화 함수는 WM_QUIT 메세지! 수신되면 FALSE를 리턴하) 메세지를 처리한 후 제대로 종료하도록 퍼스널 통신 및 통신 서버! n플리케이션! 제n를 리턴할 수 있T 합니다. W렇지 J으면 함수는 TRUE를 리턴합니다.

디폴트 블럭화 후크! 8현되지 J습니다. n플리케이션이 블럭화 후크를 설정하지 J은 fl, n플리케이션 스레드는 블럭화 호출이 리턴되b를 무b한 b다립니다.

블럭화 후크 함수! TRUE를 리턴하지 J은 fl, 1차 리턴 코드를 AP_CANCELLED로 하) 블럭화 명령n를 n플리케이션! 리턴합니다.

이 함수는 스레드! 의해 8현됩니다. 이M은 다른 스레드!는 5향을 주지 J으면서 블럭화 메카니즘을 대체하도록 특정 스레드! 제x됩니다.

|련 참조:

68페이지의 『WinAPPCCancelBlockingCall()』.

70페이지의 『WinAPPCIsBlocking()』.

74페이지의 『WinAPPCUnhookBlockingHook()』.

Windows 95 및 Windows NT SNA API 클라이 p 트 ! 는 지 x 되지 J 습니다.

WinAPPCUnhookBlockingHook()

이 함수는 설치된 이전의 블럭화 후크를 제E합니다.

퍼스널 통신 및 통신 서버는 b존 n플리케이션z의 *호환을 위해 이 함수를 제x합니다.

구문

```
BOOL WINAPI WinAPPCUnhookBlockingHook (void);
```

리턴 값

리턴 *은 함수의 실행 a z를 나타냅니다. 퍼스널 통신 및 통신 서버! 디폴트 메카니즘을 재설치한 f l, 이M은 0이 F닙니다. 퍼스널 통신 및 통신 서버! 디폴트 메카니즘을 재설치하지 J은 f l *은 0입니다.

사용시 주의사항

함수! 호출된 후 이 n플리케이션 스레드는 향후의 모든 블럭화 호출이 0료되b를 무한정 b다립니다.

|련 참조: 72페이지의 『WinAPPCSetBlockingHook()』.

Windows 95 및 Windows NT SNA API 클라이p트!는 지x되지 J습니다.

GetAppcConfig()

이 함수는 8현되지 J습니다. W러나 *호환을 위한 #트리 포인트는 제x됩니다. 유효한 매3변수 세트! 지정되면 퍼스널 통신 및 통신 서버는 APPC_CFG_SUCESS_NO_DEFAULT_REMOTE를 리턴하m NULL 종료자를 RemLu 버퍼의 첫번째 바이트! 배치합니다.

많은 fl 퍼스널 통신 및 통신 서버는 APPN !능 노드이므로 이 호출은 필d하지 J습니다. ALLOCATE! 상대방 LU명을 지정할 수 있으며 LU! 대한 K색이 시작됩니다. W러나 n플리케이션은 노드 ,산자 b능(NOF) 인터페이스를 사k하) 이 정보를 K색할 수 있습니다. NOF 인터페이스 사k! 대한 내k은 *System Management Programming*을 참조하십시@.

GetAppcReturnCode()

이 함수는 VCB의 1차 및 2차 리턴 코드를 인쇄 ! 능 문자 - 로 변환합니다. APPC n 플리케이션 ! 서 사 k 하도록 표준 @ 류 문자 - 세트를 제 x 합니다.

구문

```
int WINAPI GetAppcReturnCode (struct appc_hdr *vcb,
                              UINT buffer_length,
                              unsigned char *buffer_addr);
```

매 3 변수

설명

vcb 제 x 되는 매 3 변수. 명령 n 제 n 블럭의 주소를 지정합니다.

buffer_length

제 x 되는 매 3 변수. **buffer_addr** ! 지시하는 버퍼의 f 이를 지정합니다. G 장되는 f 이는 256입니다.

buffer_addr

제 x 또는 리턴되는 매 3 변수. 포맷된 널(null) 종료 문자 - 을 보 | 하는 버퍼의 주소를 지정합니다. 지정된 버퍼의 문자 - f 이.

리턴 값

0x20000001

매 3 변수 ! 유효하지 J 습니다. 지정된 명령 n 제 n 블럭 ! 서 함수 ! 읽 b 를 수행할 수 x E 나 지정된 버퍼 ! 5 수 x 습니다.

0x20000002

지정된 버퍼 ! 너무 작습니다.

사용시 주의사항

buffer_addr ! 리턴되는 설명식 @ 류 문자 - 은 종료시 새로 n 행 문자(**\n**)를 사 k 하지 J 습니다.

제8장 APPC 명령어

이 장!는 APPC API!서 통k되는 " 명령n의 8문을 나-하m " 명령n! 전달 및 리턴되는 매3변수! 대해 설명합니다.

또한 APPC g방향 및 반 g방향 대화! 제x되는 b본 및 직접 대화 명령n! 대한 참조 정보도 제x합니다. 이 장을 읽다 보면 b본 대화 명령nM 직접 대화 명령n! F주 유사하며 이M 때문! 이 둘이 한 장! 통합되n 있음을 KT 될 M입니다. W러나 `#의 차이는 있습니다. 차이점은F래M 0이 명시됩니다.

이 b호는 정보! b본 명령n!만 적k될 때 나타납니다.

이 b호는 정보! 직접 명령n!만 적k될 때 나타납니다.

정보에 분류! 하나 더 있음! 대화 명령n! b본 또는 직접이 될 수 있는fl, 다음z 0이 명시됩니다.

[MC_]VERBNAME

주: 다음 시스템!서 제x하는 APPC API! |한 내k은 이 책의 제1부!있는)러 장! 들n 있습니다.

- Windows NT!서 실행되는 통신 서버
- 통신 서버M 함께 제x되는 OS/2, Windows NT, Windows 95 및 Windows 3.1k SNA API 클라이p트
- Windows 95 및 Windows NTk 퍼스널 통신

이들 시스템!서 제x하는 지x! 상이한 점이 있으면 W 내k이 명시됩니다.

명령어 제어 블럭

이 절!는 필드! 대한 일반 정보M " 명령n! 대한 9! 나M 있습니다.

공통 필드

" VCB는 다음z 0은 다수의 x통 필드를 .습니다.

opcode
명령n n5 코드. 명령n의 이름이 들n 있는 식별 필드.

format
VCB의 포맷을 식별합니다. VCB의 현재 버전을 지정하려면 이 필드! 설정되n_ 하는 *이 " 명령n F래! 3별적으로 나-되n 있습니다.

primary_rc
1차 리턴 코드. " 명령n ! !능한 *은 다음 절! 나-되n 있습니다.

secondary_rc
2차 리턴 코드. 이는 1차 리턴 코드! 제x한 정보를 보충합니다. " 명령n ! !능한 *은 다음 절! 나-되n 있습니다. 몇몇 VCB !는 다음z 0은 필드도 포함되n 있습니다.

opext 명령n 확장 코드. 이는 명령n n5 코드! 제x하는 정보를 보충합니다. 명령n 신호! 비블럭 모드로 처리될 fl, 플래WAP_NON_BLOCKING이 설정되n_ 합니다. F래 설명되는 신호! 이들 x통 필드! 포함은 되지만 3별적으로 설명되지는 J습니다.

TP Identifiers
""의 활동 트랜잭션 프로W램! 8바이트의 트랜잭션 프로W램 식별자(ID)! 할당됩니다. 이 식별자는 퍼스널 통신 및 통신 서버! 의해 할당됩니다.

트랜잭션 프로W램 식별자는 **TP_ENDED** 명령n를 f로지정하는 데사k되m 대화 명령n !서 상|자로 사k됩니다.

" 신호! 대한 명령n 제n 블럭은 다음 절!서 설명됩니다.

APPC API 지원

지원되는 명령어

퍼스널 통신 및 통신 서버는 APPC API!서 다음 명령n를 지x합니다.

유형z 무|한 명령어

GET_TP_PROPERTIES
GET_TYPE
RECEIVE_ALLOCATE
TP_ENDED
TP_STARTED

대화 명령어

[MC_]ALLOCATE
[MC_]CONFIRM
[MC_]CONFIRMED
[MC_]DEALLOCATE
[MC_]FLUSH
[MC_]GET_ATTRIBUTES
[MC_]PREPARE_TO_RECEIVE
[MC_]RECEIVE_AND_POST
[MC_]RECEIVE_AND_WAIT
[MC_]RECEIVE_EXPEDITED_DATA

[MC_]RECEIVE_IMMEDIATE
[MC_]REQUEST_TO_SEND
[MC_]SEND_CONVERSATION
[MC_]SEND_DATA
[MC_]SEND_ERROR
[MC_]SEND_EXPEDITED_DATA
[MC_]TEST_RTS
[MC_]TEST_RTS_AND_POST

GET_TP_PROPERTIES

GET_TP_PROPERTIES는 트랜잭션 프로W램z , |된 속성을 리턴합니다.

VCB 구조

```
typedef struct get_tp_properties
{
 unsigned short   opcode;              /* verb operation code            */
 unsigned char    opext;               /* verb extension code            */
 unsigned char    format;              /* format                         */
 unsigned char    reserv2[2]           /* verb format                    */
 unsigned short   primary_rc;          /* primary return code            */
 unsigned long    secondary_rc;        /* secondary return code          */
 unsigned char    tp_id[8];            /* TP identifier                  */
 unsigned char    tp_name[64];         /* TP name                        */
 unsigned char    lu_alias[8];         /* LU alias                       */
 luw_id_overlay   luw_id;              /* LUW identifier       */
 unsigned char    fqlu_name[17];       /* fully qualified LU name        */
 unsigned char    reserv3[10];         /* reserved                       */
 unsigned char    user_id[10];         /* user id                        */
} GET_TP_PROPERTIES;
typedef struct luw_id_overlay
{
 unsigned char    fqlu_name_len;       /* fully qualified LU name length */
 unsigned char    fqlu_name[17];       /* fully qualified LU name        */
 unsigned char    instance[6];         /* instance number                */
 unsigned char    sequence[2];         /* sequence number                */
} LUW_ID_OVERLAY;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_GET_TP_PROPERTIES

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

opext AP_BASIC_CONVERSATION

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하는 *으로 이 필드를 설정하십시@.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

tp_name

로컬 트랜잭션 프로W램의 이름 즉, 이 명령n를 발행하는 트랜잭션 프로W램. 퍼스널 통신 및 통신 서버는 이 필드의 문자 세트를 점K하지 J습니다.

lu_alias

트랜잭션 프로W램z ,|된 로컬 LU의 별명. 이M은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자-입니다. 8바이트! 모두 중d하며 반드시 설정해_ 합니다.

luw_id 필드는 비보호 대화(**sync_level**이 AP_NONE 또는 AP_CONFIRM_SYNC_LEVEL인 대화)M ,|된 논리 작w 단위 식별자입니다. **luw_id_overlay**!는 다음 매3변수! 포함됩니다.

luw_id_overlay.fq_lu_name_len

논리 작w 단위M ,|된 전체 정식 LU명의 f이.

luw_id_overlay.fq_lu_name

논리 작w 단위M ,|된 전체 정식 LU명. 이 이름은 최대 17바이트이며 @른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 ,a된 두 3의 유형 A EBCDIC 문자-로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 !질 수 있습니다. 네트v크 ID! x으면 점을 생략하십시@.) 이름의 f이! 17바이트 미만이면 **인스턴스**M **순서**! 바로 뒤! 이n집니다. (이는 LUW_ID_OVERLAY 8조의 필드를 사k하) **인스턴스** 또는 **순서**! W세스할 수 x음을 의미합니다.)

luw_id_overlay.instance

논리 작w 단위 인스턴스 번호. 이M은 6바이트 f이의 2진 문자-입니다.

luw_id_overlay.sequence

논리 작w 단위 순서 번호. 이M은 2바이트 f이의 2진 문자-입니다.

luw_id_overlay.fq_lu_name_len이 17 미만일 f l, **luw_id_overlay**의 @른쪽은 EBDIC x백으로 채v집니다(**인스턴스** 및 **순서** 다음).

luw_id_overlay.fq_lu_name

트랜잭션 프로W램z ,|된 로컬 LU의 전체 정식 이름. 이 이름은 17바이트 f이이며 @른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 ,a된 두 3의 유형 A EBCDIC 문자-로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 !질 수 있습니다. 네트v크 ID! x으면 점을 생략하십시@.)

user_id

트랜잭션 3시자의 사k자 ID. 이M은 10바이트의 유형 AE EBCDIC 문자-이며 @른쪽은 EBCDIC x백으로 채v집니다.

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_TP_ID

다음 z 0 은 1차 리턴 코드(**primary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

GET_TYPE

GET_TYPE 명령ｎ는 특정 대화의 대화 유형(ｂ본 또는 직접)을 리턴합니다.

VCB 구조

```
typedef struct get_type
{
 unsigned short    opcode;             /* verb operation code      */
 unsigned char     opext;              /* verb extension code      */
 unsigned char     format;             /* format                   */
 unsigned short    primary_rc;         /* primary return code      */
 unsigned long     secondary_rc;       /* secondary return code    */
 unsigned char     tp_id[8];           /* TP identifier            */
 unsigned long     conv_id;            /* conversation identifier  */
 unsigned char     conv_type;          /* conversation type        */
 unsigned char     conv_style;         /* conversation style       */
} GET_TYPE;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

opcode

AP_GET_TYPE

opext AP_BASIC_CONVERSATION

format

VCB의 포맷을 식별합니다. 위！ 나-된 VCB 버전을 지정하려면 이 필드를 1로 설정하십시＠.

tp_id 로컬 트랜잭션 프로Ｗ램！ 대한 식별자. 이 매３변수의 *은 ｂ동 트랜잭션 프로Ｗ램의 **TP_STARTED** 명령ｎ나 피ｂ동 트랜잭션 프로Ｗ램의 **RECEIVE_ALLOCATE**！ 의해 리턴되[illegible]

conv_id

대화 식별자(ID). 이 매３변수유[illegible]

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

conv_style

conv_id로 식별되는 대화의 대화 g식. 이 필드!는 VCB의 포맷 1 버전이 필d합니다. 포맷 1 VCB! W세스하는 데 대한 자세한 내k은 43페이지의 『O전 g방향 VCB』를 참조하십시@.

AP_HALF_DUPLEX
AP_FULL_DUPLEX

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_TP_ID

AP_BAD_CONV_ID

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

RECEIVE_ALLOCATE

RECEIVE_ALLOCATE 명령 n 는 **ALLOCATE** 또는 **MC_ALLOCATE** 명령 n 를 발행한 상대방 트랜잭션 프로W 램 z 의 대화를 위한 새로 n 트랜잭션 프로W 램을 설정하는 데 필 d 한 정보를 d 청합니다.

VCB 구조

```
typedef struct receive_allocate
{
 unsigned shor    opcode;             /* verb operation code             */
 unsigned char    opext;              /* verb extension code             */
 unsigned char    format;             /* format                          */
 unsigned shor    primary_rc;         /* primary return code             */
 unsigned long    secondary_rc;       /* secondary return code           */
 unsigned char    tp_name[64];        /* TP name                         */
 unsigned char    tp_id[8];           /* TP identifier                   */
 unsigned long    conv_id;            /* conversation identifier         */
 unsigned char    sync_level;         /* sync Level                      */
 unsigned char    conv_type;          /* conversation type               */
 unsigned char    user_id[10];        /* user ID                         */
 unsigned char    lu_alias[8];        /* LU alias                        */
 unsigned char    plu_alias[8];       /* partner LU alias                */
 unsigned char    mode_name[8];       /* mode name                       */
 unsigned char    reserv3[2];         /* reserved                        */
 unsigned long    conv_group_id;      /* conversation group ID           */
 unsigned char    fqplu_name[17];     /* fully qualified partner LU name */
 unsigned char    pip_incoming;       /* received PIP data               */
 unsigned char    conversation_style; /* conversation style              */
 unsigned char    reserv4[3];         /* reserved                        */
 unsigned char    password[10];       /* security password               */
 unsigned char    reserv5[2];         /* reserved                        */
 unsigned char    dload_id[8];        /* user ID                         */
} RECEIVE_ALLOCATE;
```

제공되는 매개변수

트랜잭션 프로W 램은 다음 매 3 변수를 퍼스널 통신 및 통신 서버 ! 제 x 합니다.

opcode

AP_RECEIVE_ALLOCATE

opext AP_BASIC_CONVERSATION

format

VCB의 포맷을 식별합니다. 위 ! 나 - 된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@ .

tp_name

트랜잭션 프로W 램의 이름. 퍼스널 통신 및 통신 서버는 이 필드의 문자 세트를 점 K 하지 J 습니다.

리턴되는 매개변수

명령 n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_OK

tp_id 로컬 트랜잭션 프로 W 램 ! 대한 식별자. 이 * 은 퍼스널 통신 및 통신 서버 ! 의해 트랜잭션 프로 W 램 ! 할당됩니다. 트랜잭션 프로 W 램은 이 식별자(ID)를 후속하는 모든 APPC 명령 n 의 퍼스널 통신 및 통신 서버 ! 전달합니다.

conv_id
대화 식별자(ID). 이 * 은 두 트랜잭션 프로 W 램 사이 ! 설정된 대화를 식별합니다.

sync_level
대화의 동 b 화 레벨.

AP_CONFIRM_SYNC_LEVEL

AP_NONE

conv_type
conv_id로 식별되는 대화의 대화 유형.

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

user_id
상대방 트랜잭션 프로 W 램이 제 x 하는 사 k 자 ID. 이 M 은 10바이트의 유형 AE EBCDIC 문자 - 이며 @ 른쪽은 EBCDIC x 백으로 채 v 집니다.

lu_alias
로컬 LU를 식별하는 별명. 이 M 은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자 - 입니다. 8바이트 ! 모두 중 d 하며 반드시 설정해 _ 합니다.

plu_alias
로컬 트랜잭션 프로 W 램 ! 상대방 LU를 식별해주는 별명. 이 M 은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자 - 입니다. 8바이트 ! 모두 중 d 하며 반드시 설정해 _ 합니다.

mode_name
8 성시 정의된 네트 v 크 특성 세트의 이름. 이 M 은 8바이트의 5 숫자로 된 유형 A EBCDIC 문자 - (문자로 시작됨)이며 @ 른쪽은 EBCDIC x 백으로 채 v 집니다.

conv_group_id
이 대화 ! 사 k 되 m 있는 세션의 대화 W 룹 식별자(CGID).

fqplu_name

상대방 LU! 대한 전체 정식 LU명. 이 이름은 17바이트 f이이며 @ 른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 , a 된 두 3의 유형 A EBCDIC 문자 - 로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 !질 수 있습니다. 네트v크 ID! x으면 점을 생략하십시@.)

pip_incoming

상대방 트랜잭션 프로W램이 제x한 프로W램 초b화 매3변수(PIP)! [MC_]ALLOCATE d청! 있는지)부를 지정합니다. AP_YES 또는 AP_NO로 설정하십시@. AP_YES일 f l 이 대화!서 발행된 첫번째 **[MC_]RECEIVE_*** 명령n! PIP 데이타! 수신됩니다.

conversation_style

conv_id로 식별되는 대화의 대화 g식.

AP_HALF_DUPLEX
AP_FULL_DUPLEX

password

user_idM , |된 0호. 이M은 10바이트의 유형 AE EBCDIC 문자 - 이며 @른쪽은 EBCDIC x백으로 채v집니다. 보H=프로W램 (AP_PGM 또는 AP_PGM_STRONG)일 f l 이M이 필d하며 W렇지 J으면 생략!능합니다.

dload_id

이 필드는 포맷 필드! 1로 설정된 f l !만 설정할 수 있습니다. DYNAMIC_LOAD_INDICATION! 대한 응답으로 RECEIVE_ALLOCATE! 발행된 f l, 이 필드는 다음 방식으로 두 3의 신호를 , |시키는 데 사k할 수 있습니다.

RECEIVE_ALLOCATE는 **dload_id**! 다음 중 하나로 설정된 f l !만 DYNAMIC_LOAD_INDICATIONz , |됩니다.

- 모두 0.
- DYNAMIC_LOAD_INDICATION의 **dload_id** 필드.

주: 이 매3변수는 SNA API 클라이p트!는 지x되지 J습니다.

매3변수 @류로 인해 명령n! 실행되지 J은 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_UNDEFINED_TP_NAME

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_UNEXPECTED_SYSTEM_ERROR

TP_ENDED

TP_ENDED 명령 n 는 지정된 트랜잭션 프로 W 램이 종료되 z 음을 퍼스널 통신 및 통신 서버 ! 통보합니다.

VCB 구조

```
typedef struct tp_ended
{
 unsigned short    opcode;             /* verb operation code   */
 unsigned char     opext;              /* verb extension code   */
 unsigned char     format;             /* format                */
 unsigned short    primary_rc;         /* primary return code   */
 unsigned long     secondary_rc;       /* secondary return code */
 unsigned char     tp_id[8];           /* TP identifier         */
 unsigned char     type;               /* type of TP ended      */
} TP_ENDED;
```

제공되는 매개변수

트랜잭션 프로 W 램은 다음 매 3 변수를 퍼스널 통신 및 통신 서버 ! 제 x 합니다.

opcode
AP_TP_ENDED

opext AP_BASIC_CONVERSATION

format
VCB의 포맷을 식별합니다. 위 ! 나 - 된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시 @ .

tp_id 로컬 트랜잭션 프로 W 램 ! 대한 식별자. 이 매 3 변수의 * 은 b 동 트랜잭션 프로 W 램의 **TP_STARTED** 명령 n 나 피 b 동 트랜잭션 프로 W 램의 **RECEIVE_ALLOCATE** 명령 n ! 의해 리턴되 z 습니다.

type **TP_ENDED**의 유형.

AP_HARD
AP_SOFT
AP_ABEND
AP_CANCEL

유형이 AP_ABEND인 f l 퍼스널 통신 및 통신 서버는 **TP_ENDED** 신호 ! 응답하지 J 습니다.

리턴되는 매개변수

명령 n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_OK

리턴되는 매개변수

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_TP_ID

AP_BAD_TYPE

다음 z 0 은 1차 리턴 코드(**primary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

TP_STARTED

TP_STARTED 명령 n는 입력 할당 d청이 F닌, 로컬 명령의 a z 시작된 트랜잭션 프로W램! 대한 자x을 프로W램이 d청했음을 퍼스널 통신 및 통신 서버! 통보합니다.

VCB 구조

```
typedef struct tp_started
{
 unsigned short   opcode;              /* verb operation              */
 unsigned char    opext;               /* verb extension              */
 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    tp_id[8];            /* TP identifier               */
 unsigned char    tp_name[64];         /* TP name                     */
} TP_STARTED;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_TP_STARTED

opext AP_BASIC_CONVERSATION

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

lu_alias

로컬 LU를 식별하는 별명. 이M이 0으로 설정되면 퍼스널 통신 및 통신 서버는 제n점(CP) LU를 사k합니다. 이M은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자-입니다. 8바이트! 모두 중d하며 반드시 설정해_ 합니다. x백 **lu_alias** 필드도 !능합니다. 이 f l 제n점(CP) LU! 사k됩니다.

다음 정보는 통신 서버 Windows 95 및 Windows NT SNA API 클라이p트!만 적k됩니다.

" 사k자! 대한 디폴트 로컬 LU 별명은 INI 8성 또는 LDAP 중 적절한 8성 유틸리티를 사k하) 지정할 수 있습니다.

APPC 프로W램은 직접 별명을 지정하b보다는 디폴트 로컬 LU 별명을 사k하도록 선택할 수 있습니다. **local_LU_alias** 필드! 2진 0으로 설정된 상태!서 APPC 프로W램이 **TP_START** 명령n를 발행하면 APPC API는 8성된 디폴트 로컬 LU 별명을 사k합니다.

tp_name
트랜잭션 프로W램의 이름. 퍼스널 통신 및 통신 서버는 이 필드의 문자 세트를 점K하지 J습니다.

리턴되는 매개변수

명령n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

tp_id 로컬 트랜잭션 프로W램 ! 대한 식별자. 이 * 은 퍼스널 통신 및 통신 서버 ! 의해 트랜잭션 프로W램 ! 할당됩니다. 트랜잭션 프로W램은 이 식별자(ID)를 후속하는 모든 APPC 명령n의 퍼스널 통신 및 통신 서버 ! 전달합니다.

매3변수 @류로 인해 명령n ! 실행되지 J은 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_INVALID_LU_NAME

AP_INVALID_ENABLE_POOL

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드 ! 설명되n 있습니다.

AP_UNEXPECTED_SYSTEM_ERROR

[MC_]ALLOCATE

[MC_]ALLOCATE 명령n는 b동 트랜잭션 프로W램! 의해 발행됩니다. 이 명령n는 로컬 LUM 상대방 LU 사이! 세션을 할당한 후 (**RECEIVE_ALLOCATE** 명령nM 함께) b동 트랜잭션 프로W램z 피b동 트랜잭션 프로W램 사이! 대화를 설정합니다.

ALLOCATE 명령n는 b본 또는 직접 대화를 설정할 수 있습니다. **ALLOCATE** 명령n를 사k하) 직접 대화를 설정하면 트랜잭션 프로W램은 b본 대화 명령n를 사k하) 직접 대화 상대방 트랜잭션 프로W램z 대화할 수 있습니다.

퍼스널 통신 및 통신 서버는 이 명령n! 실행되면 대화 식별자(**conv_id**)를 생성합니다. 이 식별자는 다른 모든 APPC 대화 명령n! d8되는 매3변수입니다.

VCB 구조

```
typedef struct allocate
{
 unsigned short    opcode;              /* verb operation code           */
 unsigned char     opext;               /* verb extension code           */
 unsigned char     format;              /* format                        */
 unsigned short    primary_rc;          /* primary return code           */
 unsigned long     secondary_rc;        /* secondary return code         */
 unsigned char     tp_id[8];            /* TP identifier                 */
 unsigned long     conv_id;             /* conversation identifier       */
 unsigned char     conv_type;           /* conversation type             */
 unsigned char     sync_level;          /* sync level                    */
 unsigned char     reserv3[2];          /* reserved                      */
 unsigned char     rtn_ctl;             /* return control                */
 unsigned char     conversation_style;  /* conversation style            */
 unsigned long     conv_group_id;       /* conversation group identifier */
 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];         /* partner TP name               */
 unsigned char     security;            /* security level                */
 unsigned char     reserv5[11];         /* reserved                      */
 unsigned char     pwd[10];             /* security password             */
 unsigned char     user_id[10];         /* security user_id              */
 unsigned short    pip_dlen;            /* PIP data length               */
 unsigned char    *pip_dptr;            /* pointer to PIP data           */
 unsigned char     reserv5a;            /* reserved                      */
 unsigned char     fqplu_name[17];      /* fully qualified partner LU    */
                                        /* name                          */
 unsigned char     reserv6[8];          /* reserved                      */
} ALLOCATE;

typedef struct mc_allocate
{
 unsigned short     opcode;             /* verb operation code           */
 unsigned char      opext;              /* verb extension code           */
 unsigned char      format;             /* format                        */
 unsigned short     primary_rc;         /* primary return code           */
 unsigned long      secondary_rc;       /* secondary return code         */
 unsigned char      tp_id[8];           /* TP identifier                 */
```

```
 unsigned long      conv_id;            /* conversation identifier       */
 unsigned char      reserv3;            /* reserved                      */
 unsigned char      sync_level;         /* sync level                    */
 unsigned char      reserv4[2];         /* reserved                      */
 unsigned char      rtn_ctl;            /* return control                */
 unsigned char      conversation_style; /* conversation style            */
 unsigned long      conv_group_id;      /* conversation group identifier */
 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];        /* partner TP name               */
 unsigned char      security;           /* security level                */
 unsigned char      reserv6[11];        /* reserved                      */
 unsigned char      pwd[10];            /* security password             */
 unsigned char      user_id[10];        /* security user_id              */
 unsigned short     pip_dlen;           /* PIP data length               */
 unsigned char     *pip_dptr;           /* pointer to PIP data           */
 unsigned char      reserv6a;           /* reserved                      */
 unsigned char      fqplu_name[17];     /* fully qualified partner LU    */
                                        /* name                          */
 unsigned char      reserv7[8];         /* reserved                      */
} MC_ALLOCATE;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_ALLOCATE 

AP_M_ALLOCATE 

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE** 명령n! 의해 리턴되z습니다.

conv_type 

할당할 대화 유형.

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

ALLOCATE 명령ｎ！ 직접 대화를 설정하면 로컬 트랜잭션 프로Ｗ램은 ｂ본 대화 명령ｎ를 발행하ｍ 자신의 맵핑 층을 제ｘ하) 데이타 레코드를 논리 레코드로, 논리 레코드를 데이타 레코드로 변환해＿ 합니다. 상대방 트랜잭션 프로Ｗ램은 ｂ본 대화 명령ｎ를 발행, 직접 대화 층을 제ｘ하Ｅ나 (상대방 트랜잭션 프로Ｗ램이 사ｋ하ｍ 있는 APPC ８현이 직접 대화 명령ｎ를 지ｘ하는 ｆｌ) 직접 대화 명령ｎ를 사ｋ할 수 있습니다. 자세한 내ｋ은 *IBM Systems Network Architecture: LU 6.2 Reference: Peer Protocols*를 참조하십시＠.

sync_level
대화의 동ｂ화 레벨.

AP_CONFIRM_SYNC_LEVEL
AP_NONE

rtn_ctl
로컬 트랜잭션 프로Ｗ램의 세션 ｄ청！ 따라 작ｋ하는 로컬 LU！ 로컬 트랜잭션 프로Ｗ램！ 제ｎ를 리턴하는 시ｂ를 지정합니다.

AP_IMMEDIATE
AP_WHEN_SESSION_ALLOCATED
AP_WHEN_SESSION_FREE
AP_WHEN_CONV_GROUP_ALLOC
AP_WHEN_CONWINNER_ALLOC
AP_WHEN_CONLOSER_ALLOC

conversation_style
conv_id로 식별되는 대화의 대화 ｇ식.

AP_HALF_DUPLEX
AP_FULL_DUPLEX

이Ｍ은 OS/2 및 Windows 3.1ｋ 통신 서버 SNA API 클라이ｐ트！는 지ｘ되지 Ｊ습니다.

conv_group_id
할당할 세션！ 대한 대화 Ｗ룹 식별자(CGID). 이 매３변수는 **rtn_ctl**이 AP_WHEN_CONV_GROUP_ALLOC로 설정된 ｆｌ！만 제ｘ됩니다.

plu_alias
로컬 트랜잭션 프로Ｗ램！ 상대방 LU를 식별해주는 별명. 이Ｍ은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자 - 입니다. 8바이트！ 모두 중ｄ하며 반드시 설정해＿ 합니다. 이 이름은 ８성시 설정된 상대방 LU의 이름ｚ 일치해＿ 합니다. 이 필드！ 모두 0으로 설정되면 퍼스널 통신 및 통신 서버는 **fqplu_name** 필드를 사ｋ하) ｄ８되는 상대방 LU를 지정합니다.

다음 정보는 통신 서버 Windows 95 및 Windows NT SNA API 클라이p트!만 적k됩니다.

" 사k자! 대한 디폴트 상대방 LU 별명은 INI 8성 또는 LDAP 중 적절한 8성 유틸리티를 사k하) 지정할 수 있습니다.

APPC 프로W램은 직접 별명을 지정하b보다는 디폴트 상대방 LU 별명을 사k하도록 선택할 수 있습니다. **partner_LU_alias** 필드M **fully_qualified_partner_LU** 필드! 2진 0으로 설정된 상태!서 APPC 프로W램이 **ALLOCATE** 명령n를 발행하면 APPC API는 8성된 디폴트 상대방 LU 별명을 사k합니다.

mode_name

일반적으로 8성시 정의된 네트v크 특성 세트의 이름. 이M은 8바이트의 5숫자로 된 유형 A EBCDIC 문자 -(문자로 시작됨)이며 @ 른쪽은 EBCDIC x백으로 채v집니다.

tp_name

피b동 트랜잭션 프로W램의 이름. 퍼스널 통신 및 통신 서버는 이 필드의 문자 세트를 점K하지 J습니다. b동 트랜잭션 프로W램의 **ALLOCATE** 명령n! 의해 지정된 **tp_name**의 *은 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 지정된 **tp_name**의 * z 일치해_ 합니다.

security

상대방 LU! 피b동 트랜잭션 프로W램! 대한 W세스를 K증하려면 필d로 하는 정보를 지정합니다.

AP_NONE

피b동 트랜잭션 프로W램은 대화 보H을 사k하지 J습니다.

AP_PGM

피b동 트랜잭션 프로W램은 대화 보H을 사k하며)b!는 사k자 IDM 0호! d8됩니다.

AP_SAME

피b동 트랜잭션 프로W램은 대화 보H을 사k하며 이미 K증된 표시b를 수락하도록 8성됩니다. 사k자 ID! 이미 K증된 표시bM 함께 송신되n 0호! 필d하지 J음을 피b동 트랜잭션 프로W램! K려줍니다.

AP_PGM_STRONG

AP_PGMz 동일하지만 ALLOCATE는 상대방 LU! 대한 세션이 0호 대체를 지x하는 fl!만 성x합니다.

주: [MC_]ALLOCATE! 보H 유형 AP_SAME을 지정했지만 사k자 IDM 0호를 지정하지 J은 fl, 이전 SET_TP_PROPERTIES 명령n(있을 fl)! 지정된 사k자 IDM 0호! 사k됩니다. [MC_]ALLOCATE! 사k자 IDM 0호를 지정하는 fl, 이들 * 이 SET_TP_PROPERTIES 명령n! 지정된 * 대신 사k됩니다.

pwd **user_id**M ,|된 0호. 이M은 10바이트의 유형 AE EBCDIC 문자-이며 @른쪽은 EBCDIC x백으로 채v집니다. 보H=프로W램(AP_PGM 또는 AP_PGM_STRONG)일 fl 이M이 필d하며 W렇지 J으면 생략!능합니다.

user_id
상대방 트랜잭션 프로W램! W세스하는 데 필d한 사k자 ID. 이M은 10바이트의 유형 AE EBCDIC 문자-이며 @른쪽은 EBCDIC x백으로 채v집니다. 보H=프로W램(AP_PGM 또는 AP_PGM_STRONG)일 fl 이M이 필d하며 W렇지 J으면 생략!능합니다.

pip_dlen
상대방 트랜잭션 프로W램! 전달될 프로W램 초b화 매3변수(PIP)의 f이. 범위: 0 – 32767

pip_dptr
PIP 데이타! 들n 있는 버퍼의 주소. **pip_dlen**이 0보다 큰 fl!만 이 매3변수를 사k하십시@.

fqplu_name
상대방 LU! 대한 전체 정식 LU명. 이 이름은 17바이트 f이이며 @른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 ,a된 두 3의 유형 A EBCDIC 문자-로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 !질 수 있습니다. 네트v크 ID! x으면 점을 생략하십시@.) 이 필드는 **plu_alias** 필드! 모두 0으로 설정된 fl!만 의미! 있습니다.

리턴되는 매개변수

명령n! 성x적으로 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

conv_id
대화 식별자(ID). 이 *은 두 트랜잭션 프로W램 사이! 설정된 대화를 식별합니다.

conv_group_id
대화! 할당된 세션의 대화 W룹 식별자(CGID).

명령n! 비블럭 명령n이m F직 0료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

rtn_ctl 매 3 변수 ! AP_IMMEDIATE로 설정됐 m 즉시 사 k 할 수 있는 세션이 x 는 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_UNSUCCESSFUL

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_TYPE 

AP_BAD_DUPLEX_TYPE
AP_BAD_RETURN_CONTROL
AP_BAD_SECURITY
AP_BAD_SYNC_LEVEL
AP_CONFIRM_INVALID_FOR_FDX
AP_NO_USE_OF_SNASVCMG_CPSVCMG 

AP_BAD_TP_ID
AP_PIP_LEN_INCORRECT
AP_UNKNOWN_PARTNER_MODE

sense_data
[MC_]ALLOCATE의 실패 x 인 ! 대한 추 ! 정보를 제 x 합니다.

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_ALLOCATION_ERROR
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_FDX_NOT_SUPPORTED_BY_LU
AP_SEC_REQUESTED_NOT_SUPPORTED

AP_TP_BUSY
AP_UNSUCCESSFUL
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

primary_rcｉ AP_ALLOCATION_ERROR로 설정된 ｆｌ, **sense_data** 필드ｉ
는 실패ｉ |한 좀더 많은 정보ｉ 포함됩니다.

[MC_]CONFIRM

CONFIRM 명령ｎ는 로컬 LU 송신 버퍼의 내ｋｚ 확인 ｄ청을 상대방 트랜잭션 프로Ｗ램！ 송신합니다. **CONFIRM** 명령ｎ！ 대한 응답！서 상대방 트랜잭션 프로Ｗ램은 보통 **CONFIRMED** 명령ｎ를 발행하） @류 ｘ이 데이타를 수신했음을 확인합니다. (상대방 트랜잭션 프로Ｗ램이 @류를 만나면 **SEND_ERROR** 명령ｎ를 발행하Ｅ나 대화를 비정상적으로 할당해제합니다.)

트랜잭션 프로Ｗ램은 **ALLOCATE** 명령ｎ！ 의해 설정된 대화의 동ｂ화 레벨이 AP_CONFIRM_SYNC_LEVEL인 ｆｌ！만 **CONFIRM** 명령ｎ를 발행할 수 있습니다.

VCB 구조

```
typedef struct confirm
{
 unsigned short    opcode;             /* verb operation code      */
 unsigned char     opext;              /* verb extension code      */
 unsigned char     format;             /* format                   */
 unsigned short    primary_rc;         /* primary return code      */
 unsigned long     secondary_rc;       /* secondary return code    */
 unsigned char     tp_id[8];           /* TP identifier            */
 unsigned long     conv_id;            /* conversation identifier  */
 unsigned char     rts_rcvd;           /* request to send received */
#ifdef WINAPPC_FORMAT_1
 unsigned char     expd_data_rcvd;     /* expedited data received  */
#endif
} CONFIRM;
```

```
typedef struct mc_confirm
{
 unsigned short      opcode;             /* verb operation code      */
 unsigned char       opext;              /* verb extension code      */
 unsigned char       format;             /* format                   */
 unsigned short      primary_rc;         /* primary return code      */
 unsigned long       secondary_rc;       /* secondary return code    */
 unsigned char       tp_id[8];           /* TP identifier            */
 unsigned long       conv_id;            /* conversation identifier  */
 unsigned char       rts_rcvd;           /* request to send received */
#ifdef WINAPPC_FORMAT_1
 unsigned char       expd_data_rcvd;     /* expedited data received  */
#endif
} MC_CONFIRM;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

opcode

AP_B_CONFIRM 

AP_M_CONFIRM 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

format
VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 1로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id
대화 식별자(ID). 이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

rts_rcvd
수신된 송신 d청 표시b.

AP_YES
AP_NO

expd_data_rcvd
수신된 ^송 데이타 표시b. 이 표시는 RECEIVE_EXPEDITED_DATA! 발행될 때n지 AP_YES로 설정되n 있습니다.

AP_YES
AP_NO

이 필드!는 VCB의 포맷 1 버전이 필d합니다. 포맷 1 VCB! W세스하는 데 대한 자세한 내k은 43페이지의 『0전 g방향 VCB』를 참조하십시@.

명령n! 비블럭 명령n이m F직 0료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext 명령n! 비블럭 명령n이m F직 0료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

AP_OPERATION_INCOMPLETE_FLAG

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_CONFIRM_INVALID_FOR_FDX
AP_CONFIRM_ON_SYNC_LEVEL_NONE

트랜잭션 프로 W 램이 이 명령 n 를 발행할 때 대화 상태 ! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_CONFIRM_BAD_STATE

AP_CONFIRM_NOT_LL_BDY 

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_ALLOCATION_ERROR

AP_SECURITY_NOT_VALID
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TP_NAME_NOT_RECOGNIZED
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_CONVERSATION_TYPE_MISMATCH
AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY

AP_DEALLOC_ABEND 

AP_DEALLOC_ABEND_PROG 

AP_DEALLOC_ABEND_TIMER 

MC_CONFIRM

AP_PROG_ERROR_PURGING

AP_SVC_ERROR_PURGING 

AP_CONVERSATION_TYPE_MIXED 

AP_UNEXPECTED_SYSTEM_ERROR 

AP_TP_BUSY
AP_CANCELLED

주: 성능상의 이유로 SNA API 클라이ｐ트는 리턴 코드를 서버！ 이송하지 Ｊｍ 성ｘ적인 리턴 코드를 **[MC_]SEND_DATA** 명령ｎ！ 리턴할 수 있습니다. 후속 **[MC_]CONFIRM** 명령ｎ！ 발행되면 **[MC_]SEND_DATA** ！ 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** @류 리턴 코드！ 있는 ｆｌ, **[MC_]CONFIRM** 명령ｎ！ 리턴됩니다. @류 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령ｎ를 참조하십시@.

[MC_]CONFIRMED

CONFIRMED 명령n는 상대방 트랜잭션 프로W램의 확인 d청! 응답합니다. 로컬 트랜잭션 프로W램이 수신된 데이타!서 @류를 탐지하지 못했음을 상대방 트랜잭션 프로W램! 통지합니다.

확인 d청을 실행한 트랜잭션 프로W램이 확인을 b다리m 있b 때문! **CONFIRMED** 명령n는 두 트랜잭션 프로W램의 처리를 동b화합니다.

VCB 구조

```
typedef struct confirmed
{
 unsigned short    opcode;             /* verb operation code     */
 unsigned char     opext;              /* verb extension code     */
 unsigned char     format;             /* format                  */
 unsigned short    primary_rc;         /* primary return code     */
 unsigned long     secondary_rc;       /* secondary return code   */
 unsigned char     tp_id[8];           /* TP identifier           */
 unsigned long     conv_id;            /* conversation identifier */
} CONFIRMED;
```

```
typedef struct mc_confirmed
{
 unsigned short     opcode;              /* verb operation code     */
 unsigned char      opext;               /* verb extension code     */
 unsigned char      format;              /* format                  */
 unsigned short     primary_rc;          /* primary return code     */
 unsigned long      secondary_rc;        /* secondary return code   */
 unsigned char      tp_id[8];            /* TP identifier           */
 unsigned long      conv_id;             /* conversation identifier */
} MC_CONFIRMED;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_CONFIRMED 

AP_M_CONFIRMED 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트

MC_CONFIRMED

랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id
대화 식별자(ID). 이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

명령n! 비블럭 명령n이m F직 O료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_CONFIRMED_INVALID_FOR_FDX

트랜잭션 프로세서! 이 명령n를 발행할 때 대화 상태! 나쁜 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_CONFIRMED_BAD_STATE

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

AP_CONVERSATION_TYPE_MIXED

[MC_]DEALLOCATE

DEALLOCATE 명령n는 두 트랜잭션 프로W램 사이의 대화를 할당해제합니다. 대화를 할당해제하b 전! 이 명령n는 다음 명령n 중 하나M 일치하는 명령n를 수행합니다.

- 로컬 LU의 송신 버퍼 내k을 상대방 LU(및 트랜잭션 프로세서)! 송신하는 **FLUSH** 명령n.
- 로컬 LU의 송신 버퍼 내k z 확인 d청을 상대방 트랜잭션 프로W램! 송신하는 **CONFIRM** 명령n.

이 명령n ! 실행되면 대화 ID! 더 이상 유효하지 J습니다.

반 g방향 대화의 f l.

- 지정된 대화를 트랜잭션 프로W램!서 할당해제하며 **FLUSH** 또는 **CONFIRM** 명령n의 b능을 포함할 수 있습니다.

O전 g방향 대화의 f l.

- **TYPE(FLUSH)**을 사k한 **DEALLOCATE**는 로컬 프로W램의 송신 대b행렬을 닫습니다. 따라서 로컬 및 x] 프로W램은 자신들의 송신 대b행렬을 독립적으로 닫F _ 하므로 대화를 종료하는 데!는 두 3의 **DEALLOCATE TYPE(FLUSH)** 명령n ! 필d합니다. 상대방이 송신 대b행렬을 닫R다는 통지는 **DEALLOCATE_NORMAL** 리턴 코드의 g식으로 수신 대b행렬! 제x됩니다.
- **TYPE(ABEND)**를 사k한 **DEALLOCATE**는 g쪽의 대화를 닫는 이상종료입니다. 이 통지는 x] 프로W램의 송신 대b행렬! **ERROR_INDICATION** 리턴 코드로 리턴되m x] 프로W램의 수신 대b행렬!는 **DEALLOCATE_ABEND** 리턴 코드로 리턴됩니다.

VCB 구조

```
typedef struct deallocate
{
 unsigned short    opcode;              /* verb operation code     */
 unsigned char     opext;               /* verb extension code     */
 unsigned char     format;              /* format                  */
 unsigned short    primary_rc;          /* primary return code     */
 unsigned long     secondary_rc;        /* secondary return code   */
 unsigned char     tp_id[8];            /* TP identifier           */
 unsigned long     conv_id;             /* conversation identifier */
#ifdef WINAPPC_FORMAT_1
 unsigned char     expd_data_rcvd;      /* expedited data received */
 unsigned char     reserv3;             /* reserved                */
#endif
 unsigned char     dealloc_type;        /* deallocate type         */
 unsigned short    log_dlen;            /* log data length         */
 unsigned char    *log_dptr;            /* pointer to log data     */
} DEALLOCATE;

typedef struct mc_deallocate
{
 unsigned short    opcode;              /* verb operation code     */
 unsigned char     opext;               /* verb extension code     */
 unsigned char     format;              /* format                  */
```

```
 unsigned short    primary_rc;          /* primary return code     */
 unsigned long     secondary_rc;        /* secondary return code   */
 unsigned char     tp_id[8];            /* TP identifier           */
 unsigned long     conv_id;             /* conversation identifier */
#ifdef WINAPPC_FORMAT_1
 unsigned char     expd_data_rcvd;      /* expedited data received */
 unsigned char     reserv3;             /* reserved                */
#endif

 unsigned char     dealloc_type;        /* deallocate type         */
 unsigned char     reserv4[2];          /* reserved                */
 unsigned char     reserv5[4];          /* reserved                */
} MC_DEALLOCATE;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_DEALLOCATE 

AP_M_DEALLOCATE 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATIONz 논리합(OR)이 되n_ 합니다. 

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 1로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자 (ID). 이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

dealloc_type

할당해제를 수행하는 방법을 지정합니다.

AP_ABEND 

AP_ABEND_PROG

AP_ABEND_SVC
AP_ABEND_TIMER
AP_FLUSH
AP_SYNC_LEVEL

다음 *은 본 명령에만 적용됩니다. 

AP_TP_NOT_AVAIL_NO_RETRY
AP_TP_NOT_AVAIL_RETRY
AP_TPN_NOT_RECOGNIZED
AP_PIP_DATA_NOT_ALLOWED
AP_PIP_DATA_INCORRECT
AP_RESOURCE_FAILURE_NO_RETRY
AP_CONV_TYPE_MISMATCH
AP_SYNC_LVL_NOT_SUPPORTED
AP_SECURITY_PARAMS_INVALID

log_dlen 

@송 로W 파일! 송신될 데이터의 바이트 수.

범위: 0 – 32767

n플리케이션은 데이터를 VCB의 끝! 추!할 수 있으며 이 fl 이 필드는 0보다 커지며 **log_dptr**은 NULL로 설정되n_ 합니다. (f이 ! 0이면 @송 로W 데이타! x음을 나타냅니다.)

log_dptr 

@송 정보! 들n 있는 데이타의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 fl **log_dptr**는 NULL로 설정되n_ 합니다.

이 데이타는 로컬 @송 로WM 상대방 LU로 송신됩니다. 트랜잭션 프로세서는 @송 데이타를 일반 데이타 스트림(GDS) @송 로W 변수로 포맷해_ 합니다. 자세한 내k은 *IBM System Network Architecture: LU 6.2 Reference: Peer Protocols*를 참조하십시@.

리턴되는 매개변수

명령 n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_OK

expd_data_rcvd
수신된 ^ 송 데이타 표시 b . 이 표시는 RECEIVE_EXPEDITED_DATA ! 발행될 때 n 지 AP_YES로 설정되 n 있습니다.

이 필드 ! 는 VCB의 포맷 1 버전이 필 d 합니다. 포맷 1 VCB ! W 세스하는데 대한 자세한 내 k 은 43페이지의 『 0 전 g 방향 VCB』를 참조하십시 @ .

AP_YES
AP_NO

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID
AP_BAD_TP_ID
AP_DEALLOC_BAD_TYPE
AP_DEALLOC_LOG_LL_WRONG 

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다(직접 명령 n ! 한함). 

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

트랜잭션 프로세서 ! 이 명령 n 를 발행할 때 대화 상태 ! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc

AP_DEALLOC_CONFIRM_BAD_STATE
AP_DEALLOC_FLUSH_BAD_STATE
AP_DEALLOC_NOT_LL_BDY

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_ALLOCATION_ERROR
- AP_SECURITY_NOT_VALID
- AP_TRANS_PGM_NOT_AVAIL_RETRY
- AP_TRANS_PGM_NOT_AVAIL_NO_RTRY
- AP_TP_NAME_NOT_RECOGNIZED
- AP_PIP_NOT_ALLOWED
- AP_PIP_NOT_SPECIFIED_CORRECTLY
- AP_CONVERSATION_TYPE_MISMATCH
- AP_SYNC_LEVEL_NOT_SUPPORTED
- AP_CONV_FAILURE_NO_RETRY
- AP_CONV_FAILURE_RETRY
- AP_DEALLOC_ABEND 
- AP_DEALLOC_ABEND_PROG 
- AP_DEALLOC_ABEND_SVC 
- AP_DEALLOC_ABEND_TIMER 
- AP_PROG_ERROR_PURGING
- AP_SVC_ERROR_PURGING 

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED
AP_ERROR_INDICATION
- AP_ALLOCATION_ERROR_PENDING
- AP_DEALLOC_ABEND_PROG_PENDING
- AP_DEALLOC_ABEND_SVC_PENDING
- AP_DEALLOC_ABEND_TIMER_PENDING
- AP_UNKNOWN_ERROR_TYPE_PENDING

주: 성능상의 이유로 SNA API 클라이언트는 리턴 코드를 서버로 이송하지 않을 수도 있습니다. 후속 **[MC_]DEALLOCATE** 명령은 발행되면 **[MC_]SEND_DATA** 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** 오류 리턴 코드는 있는 경우, **[MC_]DEALLOCATE** 명령으로 리턴됩니다. 오류 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령을 참조하십시오.

[MC_]FLUSH

FLUSH 명령n는 로컬 LU의 송신 버퍼 내k을 상대방 LU(및 트랜잭션 프로W램)! 송신합니다. 송신 버퍼! 비n 있는 fl, 조치! 취해지지 J습니다.

VCB 구조

```
typedef struct flush
{
 unsigned short    opcode;             /* verb operation code     */
 unsigned char     opext;              /* verb extension code     */
 unsigned char     format;             /* format                  */
 unsigned short    primary_rc;         /* primary return code     */
 unsigned long     secondary_rc;       /* secondary return code   */
 unsigned char     tp_id[8];           /* TP identifier           */
 unsigned long     conv_id;            /* conversation identifier */
} FLUSH;

typedef struct mc_flush
{
 unsigned short      opcode;             /* verb operation code     */
 unsigned char       opext;              /* verb extension code     */
 unsigned char       format;             /* format                  */
 unsigned short      primary_rc;         /* primary return code     */
 unsigned long       secondary_rc;       /* secondary return code   */
 unsigned char       tp_id[8];           /* TP identifier           */
 unsigned long       conv_id;            /* conversation identifier */
} MC_FLUSH;
```

제공되는 매개변수

트랜잭션 프로세서는 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_FLUSH 

AP_M_FLUSH 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATIONz 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트

랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id
대화 식별자(ID). 이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

명령n! 비블럭 명령n이m F직 0료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID

트랜잭션 프로W램이 이 명령n를 발행할 때 대화 상태! 나쁜 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_FLUSH_NOT_SEND_STATE

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED

AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_ERROR_INDICATION

- AP_ALLOCATION_ERROR_PENDING
- AP_DEALLOC_ABEND_PROG_PENDING
- AP_DEALLOC_ABEND_SVC_PENDING
- AP_DEALLOC_ABEND_TIMER_PENDING
- AP_UNKNOWN_ERROR_TYPE_PENDING

주: 성능상의 이유로 SNA API 클라이언트는 리턴 코드를 서버로 즉시 송신하지 않을 수 있습니다. 후속 **[MC_]FLUSH** 명령이 발행되면 **[MC_]SEND_DATA** 로 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** 에 대한 리턴 코드로 있는 것은 **[MC_]FLUSH** 명령에 대한 리턴됩니다. 이에 대한 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령을 참조하십시오.

[MC_]GET_ATTRIBUTES

GET_ATTRIBUTES 명령 n는 대화의 속성을 리턴합니다.

VCB 구조

```
typedef struct get_attributes
{
 unsigned short    opcode;             /* verb operation code           */
 unsigned char     opext;              /* verb extension code           */
 unsigned char     format;             /* verb format                   */
 unsigned short    primary_rc;         /* primary return code           */
 unsigned long     secondary_rc;       /* secondary return code         */
 unsigned char     tp_id[8];           /* TP identifier                 */
 unsigned long     conv_id;            /* conversation identifier       */
 unsigned char     reserv3;            /* reserved                      */
 unsigned char     sync_level;         /* sync_level                    */
 unsigned char     mode_name[8];       /* mode name                     */
 unsigned char     net_name[8];        /* network name of local LU      */
 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     plu_un_name[8];
                                       /* partner LU uninterpreted name */
 unsigned char     reserv4[2];         /* reserved                      */
 unsigned char     fqplu_name[17];
                                       /* fully qualified partner LU    */
                                       /* name                          */
 unsigned char     reserv5;            /* reserved                      */
 unsigned char     user_id[10];        /* user identifier               */
 unsigned long     conv_group_id;      /* conversation group identifier */
 unsigned char     conv_corr_len;      /* conversation correlator       */
                                       /* length                        */
 unsigned char     conv_corr[8];       /* conversation correlator       */
 unsigned char     reserv6[13];        /* reserved                      */
} GET_ATTRIBUTES;

typedef struct mc_get_attributes
{
 unsigned short    opcode;             /* verb operation code           */
 unsigned char     opext;              /* verb extension code           */
 unsigned char     format;             /* verb format                   */
 unsigned short    primary_rc;         /* primary return code           */
 unsigned long     secondary_rc;       /* secondary return code         */
 unsigned char     tp_id[8];           /* TP identifier                 */
 unsigned long     conv_id;            /* conversation identifier       */
 unsigned char     reserv3;            /* reserved                      */
 unsigned char     sync_level;         /* sync_level                    */
 unsigned char     mode_name[8];       /* mode name                     */
 unsigned char     net_name[8];        /* network name of local LU      */
 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     plu_un_name[8];     /* partner LU uninterpreted name */
 unsigned char     reserv4[2];         /* reserved                      */
 unsigned char     fqplu_name[17];     /* fully qualified partner LU    */
                                       /* name                          */
 unsigned char     reserv5;            /* reserved                      */
 unsigned char     user_id[10];        /* user identifier               */
 unsigned long     conv_group_id;      /* conversation group identifier */
```

```
 unsigned char     conv_corr_len;      /* conversation correlator       */
                                       /* length                        */
 unsigned char     conv_corr[8];       /* conversation correlator       */
 unsigned char     reserv6[13];        /* reserved                      */
} MC_GET_ ATTRIBUTES;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_GET_ATTRIBUTES 

AP_M_GET_ATTRIBUTES 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATIONz 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

sync_level

대화의 동b화 레벨.

AP_CONFIRM_SYNC_LEVEL
AP_NONE

mode_name

대화! 할당된 세션z , |된 네트v크 특성 세트의 이름. 이M은 8바이트의 5숫자로 된 유형 A EBCDIC 문자 - (문자로 시작됨)이며 @른쪽은 EBCDIC x백으로 채v집니다.

net_name

로컬 LU! 들n 있는 네트v크의 이름. 이M은 8바이트의 5숫자로 된 유형 A EBCDIC 문자 - (문자로 시작됨)이며 @른쪽은 EBCDIC x백으로 채v집니다.

lu_name

로컬 LU의 이름. 이M은 8바이트의 5숫자로 된 유형 A EBCDIC 문자 - (문자로 시작됨)이며 @른쪽은 EBCDIC x백으로 채v집니다.

lu_alias

로컬 트랜잭션 프로W램! 로컬 LU를 식별해주는 별명. 이M은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자 - 입니다. 8바이트! 모두 중d하며 반드시 설정해_ 합니다.

plu_alias

로컬 트랜잭션 프로W램! 상대방 LU를 식별해주는 별명. 이M은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자 - 입니다. 8바이트! 모두 중d하며 반드시 설정해_ 합니다.

plu_un_name

상대방 LU의 미해석 이름. 즉, 시스템 서비스 제n점(SSCP)! 정의된 상대방 LU의 이름. 이M은 8바이트의 유형 A EBCDIC 문자 - 입니다.

fqplu_name

상대방 LU의 전체 정식 이름. 이 이름은 17바이트 f이이며 @른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 , a된 두 3의 유형 A EBCDIC 문자 - 로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 !질 수 있습니다. 네트v크 ID! x으면 점을 생략하십시@.)

user_id

피b동 트랜잭션 프로W램(적k !능한 f l)! W세스하b 위해 **ALLOCATE** 명령n를 통해 b동 트랜잭션 프로W램! 송신되는 사k자 ID. 이M은 10바이트의 유형 AE EBCDIC 문자 - 이며 @른쪽은 EBCDIC x백으로 채v집니다.

conv_group_id

대화! 할당된 세션의 대화 W룹 식별자(CGID).

conv_corr_len

항상 0으로 설정됩니다.

범위: 0 – 8

conv_corr

항상 0으로 설정됩니다.

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED

AP_DUPLEX_TYPE_MIXED

AP_UNEXPECTED_SYSTEM_ERROR

[MC_]PREPARE_TO_RECEIVE

PREPARE_TO_RECEIVE 명령n는 로컬 트랜잭션 프로W램! 대한 대화 상태를 SEND 또는 SEND_PENDING!서 RECEIVE로 변f합니다.

대화의 상태를 변f하b 전! 이 명령n는 다음 명령n 중 하나M 일치하는 명령n를 수행합니다.

- 로컬 LU의 송신 버퍼 내k을 상대방 LU(및 트랜잭션 프로W램)! 송신하는 **FLUSH** 명령n.
- 로컬 LU의 송신 버퍼 내kz 확인 d청을 상대방 트랜잭션 프로W램! 송신하는 **CONFIRM** 명령n.

이 명령n! 실행되면 로컬 트랜잭션 프로W램은 데이타를 수신할 수 있습니다.

VCB 구조

```
typedef struct prepare_to_receive
{
 unsigned short    opcode;              /* verb operation code      */
 unsigned char     opext;               /* verb extension code      */
 unsigned char     format;              /* format                   */
 unsigned short    primary_rc;          /* primary return code      */
 unsigned long     secondary_rc;        /* secondary return code    */
 unsigned char     tp_id[8];            /* TP identifier            */
 unsigned long     conv_id;             /* conversation identifier  */
 unsigned char     ptr_type;            /* prepare to receive type  */
 unsigned char     locks;               /* prepare to receive locks */
} PREPARE_TO_RECEIVE;

typedef struct mc_prepare_to_receive
{
 unsigned short     opcode;             /* verb operation code      */
 unsigned char      opext;              /* verb extension code      */
 unsigned char      format;             /* format                   */
 unsigned short     primary_rc;         /* primary return code      */
 unsigned long      secondary_rc;       /* secondary return code    */
 unsigned char      tp_id[8];           /* TP identifier            */
 unsigned long      conv_id;            /* conversation identifier  */
 unsigned char      ptr_type;           /* prepare to receive type  */
 unsigned char      locks;              /* prepare to receive locks */
} MC_PREPARE_TO_RECEIVE;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_PREPARE_TO_RECEIVE 

AP_M_PREPARE_TO_RECEIVE 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

format

VCB의 포맷을 식별합니다. 위! 나 - 된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

ptr_type

상태 변f을 수행하는 방법을 지정합니다.

AP_FLUSH
AP_SYNC_LEVEL
AP_P_TO_R_CONFIRM

locks 퍼스널 통신 및 통신 서버! 로컬 트랜잭션 프로세서! 제n를 리턴하는 시b를 지정합니다.

AP_LONG

AP_SHORT

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

명령n! 비블럭 명령n이m F직 0료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_P_TO_R_INVALID_FOR_FDX
AP_P_TO_R_INVALID_TYPE

트랜잭션 프로세서! 이 명령n를 발행할 때 대화 상태! 나쁜 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_STATE_CHECK

secondary_rc

AP_TO_R_NOT_LL_BDY 

AP_P_TO_R_NOT_SEND_STATE

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_ALLOCATION_ERROR

AP_SECURITY_NOT_VALID

AP_TRANS_PGM_NOT_AVAIL_RETRY

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

AP_TP_NAME_NOT_RECOGNIZED

AP_PIP_NOT_ALLOWED

AP_PIP_NOT_SPECIFIED_CORRECTLY

AP_CONVERSATION_TYPE_MISMATCH

AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY

AP_CONV_FAILURE_RETRY

AP_DEALLOC_ABEND 

AP_DEALLOC_ABEND_PROG 

AP_DEALLOC_ABEND_SVC 

AP_DEALLOC_ABEND_TIMER 

AP_PROG_ERROR_PURGING 

AP_SVC_ERROR_PURGING 

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED

AP_UNEXPECTED_SYSTEM_ERROR

AP_CANCELLED

주: 성능상의 이유로 SNA API 클라이언트는 리턴 코드를 서버로 이송하지 않을 수도 있습니다. 후속 **[MC_]PREPARE_TO_RECEIVE** 명령이 발행되면 **[MC_]SEND_DATA** 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** 리턴 코드에 있는 오류, **[MC_]PREPARE_TO_RECEIVE** 명령이 리턴됩니다. 오류 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령을 참조하십시오.

[MC_]RECEIVE_AND_POST

RECEIVE_AND_POST 명령 n는 n플리케이션 데이타 M 상태 정보를 비동 b로 수신합니다. 이 M 0이 하면 트랜잭션 프로 W램은 데이타 ! 로컬 LU ! 도착하는 중 ! 도 처리를 h속할 수 있습니다. 이 명령 n는 APPC #트리 포인트를 통해서만 발행할 수 있습니다.

Win 3.1 SNA API 클라이 p트 ! 서는 사 k할 수 x음.

VCB 구조

```
typedef struct receive_and_post
{
 unsigned short    opcode;             /* verb operation code        */
 unsigned char     opext;              /* verb extension code        */
 unsigned char     format;             /* format                     */
 unsigned short    primary_rc;         /* primary return code        */
 unsigned long     secondary_rc;       /* secondary return code      */
 unsigned char     tp_id[8];           /* TP identifier              */
 unsigned long     conv_id;            /* conversation identifier    */
 unsigned short    what_rcvd;          /* what received              */
 unsigned char     rtn_status;         /* return status with data    */
 unsigned char     fill;               /* data fill                  */
 unsigned char     rts_rcvd;           /* request to send received   */
 unsigned char     expd_data_rcvd;     /* expedited data received    */
 unsigned short    max_len;            /* maximum length of received */
                                       /* data                       */
 unsigned short    dlen;               /* actual length of received  */
                                       /* data                       */
 unsigned char    *dptr;               /* pointer to data buffer     */
 unsigned long    *sema;               /* post handle for verb       */
 unsigned char     reserv5;            /* reserved                   */
} RECEIVE_AND_POST;

typedef struct mc_receive_and_post
{
 unsigned short   opcode;            /* verb operation code        */
 unsigned char    opext;             /* verb extension code        */
 unsigned char    format;            /* format                     */
 unsigned short   primary_rc;        /* primary return code        */
 unsigned long    secondary_rc;      /* secondary return code      */
 unsigned char    tp_id[8];          /* TP identifier              */
 unsigned long    conv_id;           /* conversation identifier    */
 unsigned short   what_rcvd;         /* what received              */
 unsigned char    rtn_status;        /* return status with data    */
 unsigned char    reserv4;           /* reserved                   */
 unsigned char    rts_rcvd;          /* request to send received   */
 unsigned char    expd_data_rcvd;    /* expedited data received    */
 unsigned short   max_len;           /* maximum length of received */
                                     /* data                       */
 unsigned short   dlen;              /* actual length of received  */
                                     /* data                       */
 unsigned char   *dptr;              /* pointer to data buffer     */
 unsigned long   *sema;              /* post handle for verb       */
 unsigned char    reserv6;           /* reserved                   */
} MC_RECEIVE_AND_POST;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_RECEIVE_AND_POST 

AP_M_RECEIVE_AND_POST 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자. 이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

rtn_status

상태 정보M 데이타! 동일한 명령n! 리턴되는지)부를 나타냅니다.

AP_YES

AP_NO

fill 

로컬 트랜잭션 프로W램이 데이타를 수신하는 방식을 나타냅니다.

AP_BUFFER

AP_LL

max_len

로컬 트랜잭션 프로W램이 수신할 수 있는 데이타 바이트의 최대 수.

범위: 0 – 65535

이 *은 수신 데이타를 담을 수 있는 버퍼의 f이를 초z하면 H 됩니다.

dptr 로컬 LU! 의해 수신된 데이타를 담b 위한 버퍼의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 fl **dptr**는 NULL로 설정되n _ 합니다.

sema n플리케이션이 처리하는 이벤트의 핸들. 이 명령n는 Win32 API! 서 WaitForMultipleObjectsM 함께 또는 OS/2k DosWaitEventSem! 사k하b 위한 M입니다.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

AP_DEALLOC_NORMAL

what_rcvd

입력 데이타M 함께 수신된 상태 정보. **rtn_status**! AP_NO로 설정된 fl, 이 필드!는 반드시 다음 리스트의 첫번째 부분! 있는 * 이 들n)니다. **rtn_status**! AP_YES로 설정된 fl, 이 필드!는 리스트의 F무 *이나 들n% 수 있습니다.

AP_NONE

AP_CONFIRM_DEALLOCATE

AP_CONFIRM_SEND

AP_CONFIRM_WHAT_RECEIVED

AP_DATA 

AP_DATA_COMPLETE

AP_DATA_INCOMPLETE

AP_SEND

AP_USER_CONTROL_DATA_COMPLETE 

AP_USER_CONTROL_DATA_INCMP 

AP_PS_HEADER_COMPLETE 

AP_PS_HEADER_INCOMPLETE 

AP_DATA_CONFIRM 

AP_DATA_COMPLETE_CONFIRM

AP_DATA_CONFIRM_DEALLOCATE

AP_DATA_COMPLETE_CONFIRM_DEALL
AP_DATA_CONFIRM_SEND
AP_DATA_COMPLETE_CONFIRM_SEND
AP_DATA_SEND
AP_DATA_COMPLETE_SEND

다음 매 3 변수는 직접 명령 n 전 k 입니다. 

AP_UC_DATA_COMPLETE_CONFIRM
AP_UC_DATA_COMPLETE_CNFM_DEALL
AP_UC_DATA_COMPLETE_CNFM_SEND
AP_UC_DATA_COMPLETE_SEND
AP_PS_HDR_COMPLETE_CONFIRM
AP_PS_HDR_COMPLETE_CNFM_DEALL
AP_PS_HDR_COMPLETE_CNFM_SEND
AP_PS_HDR_COMPLETE_SEND

rts_rcvd

수신된 송신 d 청 표시 b .

AP_YES
AP_NO

expd_data_rcvd

수신된 ^ 송 데이타 표시 b . 이 표시는 RECEIVE_EXPEDITED_DATA ! 발행될 때 n 지 AP_YES로 설정되 n 있습니다.

AP_YES
AP_NO

이 포맷 필드 ! 는 VCB의 포맷 1 버전이 필 d 합니다. 포맷 1 VCB ! W 세스하는 데 대한 자세한 내 k 은 43페이지의 『 0 전 g 방향 VCB』를 참조하십시 @ .

dlen 수신된 데이타의 바이트 수(데이타는 **dptr** 매 3 변수 ! 의해 지정된 버퍼 ! 저장됩니다.) f 이 ! 0이면 데이타 ! 수신되지 J R 음을 나타냅니다. 이 매 3 변수는 **what_rcvd** 매 3 변수 ! 데이타의 수신을 나타내는 f l ! 만 사 k 됩니다.

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l , 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID
AP_BAD_RETURN_STATUS_WITH_DATA

AP_BAD_TP_ID

AP_RCV_AND_POST_BAD_FILL 

트랜잭션 프로W램이 이 명령n를 발행할 때 대화 상태! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_STATE_CHECK

secondary_rc

AP_RCV_AND_POST_BAD_STATE

AP_RCV_AND_POST_NOT_LL_BDY 

트랜잭션 프로W램이 발행한 또다른 명령n ! 의해 명령n ! 취소되n 실행되지 J은 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_CANCELLED

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_ALLOCATION_ERROR

AP_SECURITY_NOT_VALID

AP_TRANS_PGM_NOT_AVAIL_RETRY

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

AP_TP_NAME_NOT_RECOGNIZED

AP_PIP_NOT_ALLOWED

AP_PIP_NOT_SPECIFIED_CORRECTLY

AP_CONVERSATION_TYPE_MISMATCH

AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY

AP_CONV_FAILURE_RETRY

AP_DEALLOC_ABEND 

AP_DEALLOC_ABEND_PROG 

AP_DEALLOC_ABEND_SVC 

AP_DEALLOC_ABEND_TIMER 

AP_DEALLOC_NORMAL

AP_PROG_ERROR_NO_TRUNC

AP_PROG_ERROR_PURGING 

AP_PROG_ERROR_TRUNC 

AP_SVC_ERROR_NO_TRUNC 

AP_SVC_ERROR_PURGING 

AP_SVC_ERROR_TRUNC

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED

AP_UNEXPECTED_SYSTEM_ERROR

AP_CANCELLED

주: 성능상의 이유로 SNA API 클라이언트는 리턴 코드를 서버로 이송하지 못할 수도 있습니다. 후속 **[MC_]RECEIVE_AND_POST** 명령에 대한 응답으로 예상되는 리턴 코드를 **[MC_]SEND_DATA** 명령에 대한 응답으로 리턴할 수 있습니다. 후속 **[MC_]RECEIVE_AND_POST** 명령에 대한 응답으로 발행되면 **[MC_]SEND_DATA** 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** 에 대한 리턴 코드에 있는 값에 따라, **[MC_]RECEIVE_AND_POST** 명령에 대한 리턴됩니다. 에 대한 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령에 대한 참조하십시오.

[MC]RECEIVE_AND_WAIT

RECEIVE_AND_WAIT 명령n는 현재 상대방 트랜잭션 프로W램!서 제x하는 모든 데이타를 수신합니다. 제x되는 데이타! x는 f l, 로컬 트랜잭션 프로W램은 데이타! 도착하b를 b다립니다.

반 g방향 대화의 f l.

- 프로W램은 대화! 송신 상태! 있을 때 이 명령n를 발행할 수 있습니다. 이 f l LU는 송신 버퍼를 비v 버퍼된 모든 정보M SEND 표시를 x] 프로W램! 송신합니다. 이는 대화를 수신 상태로 변f합니다. W런 다음 LU는 정보! 도착하b를 b다립니다. x] 프로W램은 SEND 표시를 수신한 후 로컬 프로W램으로 데이타를 송신할 수 있습니다.

O전 g방향 대화의 f l.

- 송신 버퍼! 대화 할당 d청이 들n 있는 f l, 이는 비v집니다. W렇지 J으면 이 명령n로 인해 LU! 자신의 송신 버퍼를 비l지는 J습니다. 데이타를 수신하b 전! 송신 버퍼! 남F 있는 데이타를 전송해_ 하는 f l, 로컬 프로W램은 이 명령n를 발행하b 전! FLUSH를 발행해_ 합니다.

VCB 구조

```
typedef struct receive_and_wait
{
 unsigned short    opcode;             /* verb operation code        */
 unsigned char     opext;              /* verb extension code        */
 unsigned char     format;             /* format                     */
 unsigned short    primary_rc;         /* primary return code        */
 unsigned long     secondary_rc;       /* secondary return code      */
 unsigned char     tp_id[8];           /* TP identifier              */
 unsigned long     conv_id;            /* conversation identifier    */
 unsigned short    what_rcvd;          /* what received              */
 unsigned char     rtn_status;         /* return status with data    */
 unsigned char     fill;               /* data fill                  */
 unsigned char     rts_rcvd;           /* request to send received   */
 unsigned char     expd_data_rcvd;     /* expedited data received    */
 unsigned short    max_len;            /* maximum length of received */
                                       /* data                       */
 unsigned short    dlen;               /* actual length of received  */
                                       /* data                       */
 unsigned char    *dptr;               /* pointer to data buffer     */
 unsigned char     reserv5[5];         /* reserved                   */
} RECEIVE_AND_WAIT;

typedef struct mc_receive_and_wait
{
 unsigned short      opcode;             /* verb operation code        */
 unsigned char       opext;              /* verb extension code        */
 unsigned char       format;             /* format                     */
 unsigned short      primary_rc;         /* primary return code        */
 unsigned long       secondary_rc;       /* secondary return code      */
 unsigned char       tp_id[8];           /* TP identifier              */
 unsigned long       conv_id;            /* conversation identifier    */
 unsigned short      what_rcvd;          /* what received              */
 unsigned char       rtn_status;         /* return status with data    */
```

```
 unsigned char       reserv4;            /* reserved                   */
 unsigned char       rts_rcvd;           /* request to send received   */
 unsigned char       expd_data_rcvd;     /* expedited data received    */
 unsigned short      max_len;            /* maximum length of received */
                                         /* data                       */
 unsigned short      dlen;               /* actual length of received  */
                                         /* data                       */
 unsigned char      *dptr;               /* pointer to data buffer     */
 unsigned char       reserv6[5];         /* reserved                   */
} MC_RECEIVE_AND_WAIT;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_RECEIVE_AND_WAIT 

AP_M_RECEIVE_AND_WAIT 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATIONz 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

rtn_status

상태 정보M 데이타! 동일한 명령n! 리턴되는지)부를 나타냅니다.

AP_YES

AP_NO

fill

로컬 트랜잭션 프로W램이 데이타를 수신하는 방식을 나타냅니다.

AP_BUFFER

AP_LL

max_len

로컬 트랜잭션 프로W램이 수신할 수 있는 데이타 바이트의 최대 수.

범위: 0 – 65535

이 *은 수신 데이타를 담을 수 있는 버퍼의 f이를 초z하면 H 됩니다.

dptr 로컬 LU! 의해 수신된 데이타를 담b 위한 버퍼의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 f l **dptr**은 NULL로 설정되n _ 합니다.

리턴되는 매개변수

명령n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

AP_DEALLOC_NORMAL

what_rcvd

입력 데이타M 함께 수신된 상태 정보. **rtn_status**! AP_NO로 설정된 f l, 이 필드!는 반드시 다음 리스트의 첫번째 부분! 있는 * 이 들n)니다. **rtn_status**! AP_YES로 설정된 f l, 이 필드!는 리스트의 F무 *이나 들n % 수 있습니다.

AP_NONE

AP_CONFIRM_DEALLOCATE

AP_CONFIRM_SEND

AP_CONFIRM_WHAT_RECEIVED

AP_DATA 

AP_DATA_COMPLETE

AP_DATA_INCOMPLETE

AP_SEND

AP_USER_CONTROL_DATA_COMPLETE 

AP_USER_CONTROL_DATA_INCMP 

AP_PS_HEADER_COMPLETE 

AP_PS_HEADER_INCOMPLETE 

AP_DATA_CONFIRM 

AP_DATA_COMPLETE_CONFIRM

AP_DATA_CONFIRM_DEALLOCATE 

AP_DATA_COMPLETE_CONFIRM_DEALL

AP_DATA_CONFIRM_SEND 

AP_DATA_COMPLETE_CONFIRM_SEND

AP_DATA_SEND 

AP_DATA_COMPLETE_SEND

다음 매 3 변수는 직접 명령 n ! 만 적 k 됩니다. 

AP_UC_DATA_COMPLETE_CONFIRM

AP_UC_DATA_COMPLETE_CNFM_DEALL

AP_UC_DATA_COMPLETE_CNFM_SEND

AP_UC_DATA_COMPLETE_SEND

AP_PS_HDR_COMPLETE_CONFIRM

AP_PS_HDR_COMPLETE_CNFM_DEALL

AP_PS_HDR_COMPLETE_CNFM_SEND

AP_PS_HDR_COMPLETE_SEND

rts_rcvd

수신된 송신 d 청 표시 b .

AP_YES

AP_NO

다음 명령 n 의 이 포맷은 VCB의 포맷 1 버전이 n _ 합니다. 포맷 1 VCB ! W 세스하는 데 대한 자세한 내 k 은 43페이지의 『 0 전 g 방향 VCB』를 참조하십시 @ .

expd_data_rcvd

수신된 ^ 송 데이타 표시 b . 이 표시는 RECEIVE_EXPEDITED_DATA ! 발행될 때 n 지 AP_YES로 설정되 n 있습니다.

AP_YES

AP_NO

dlen 이 매개변수는 **what_rcvd** 매개변수가 데이타의 수신을 나타내는 경우에만 사용됩니다. 수신된 데이타의 바이트 수(데이타는 **dptr** 매개변수에 의해 지정된 버퍼에 저장됩니다). 이 0이면 데이타가 수신되지 않음을 나타냅니다.

명령이 비블럭 명령이며 아직 완료되지 않은 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

매개변수 오류로 인해 명령이 실행되지 않은 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID
AP_BAD_RETURN_STATUS_WITH_DATA
AP_BAD_TP_ID
AP_RCV_AND_WAIT_BAD_FILL 

트랜잭션 프로그램이 이 명령을 발행할 때 대화 상태가 나쁜 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_RCV_AND_WAIT_BAD_STATE
AP_RCV_AND_WAIT_NOT_LL_BDY 

다음 표는 1차 리턴 코드(**primary_rc**)에 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조건을 부록A. APPC 공통 리턴 코드에 설명되어 있습니다.

AP_ALLOCATION_ERROR
AP_SECURITY_NOT_VALID
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_TRANS_PGM_NOT_AVAIL_NO_RTRY
AP_TP_NAME_NOT_RECOGNIZED
AP_PIP_NOT_ALLOWED

AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_CONVERSATION_TYPE_MISMATCH
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_DEALLOC_ABEND 

다음 세 3의 매3변수는 b본 명령n!만 적k됩니다. 

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER

AP_DEALLOC_NORMAL
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING

다음 네 3의 매3변수는 b본 명령n!만 적k됩니다. 

AP_PROG_ERROR_TRUNCL
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

주: 성능상의 이유로 SNA API 클라이p트는 리턴 코드를 서버! 이송하지 Jm 성x적인 리턴 코드를 **[MC_]SEND_DATA** 명령n! 리턴할 수 있습니다. 후속 **[MC_]RECEIVE_AND_WAIT** 명령n! 발행되면 **[MC_]SEND_DATA**! 처리를 위해 서버! 이송됩니다. **[MC_]SEND_DATA** @류 리턴 코드! 있는 fl, **[MC_]RECEIVE_AND_WAIT** 명령n! 리턴됩니다. @류 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령n를 참조하십시@.

[MC_]RECEIVE_EXPEDITED_DATA

OS/2 및 Windows 3.1 k 통신 서버 SNA API 클라이 p 트 ! 는 지 x 되지 J 습니다.

[MC_]RECEIVE_EXPEDITED_DATA 명령 n 는 현재 상대방 TP ! 서 사 k 할 수 있는 ^ 송 데이타를 수신합니다. 현재 ^ 송 데이타를 사 k 할 수 있는 f l, 로컬 트랜잭션 프로 W 램은 b 다리지 J m 데이타를 수신합니다. W 렇지 J 으면 **rtn_ctl** 필드 ! 의해 처리 ! 통제됩니다.

VCB 구조

```
typedef struct receive_expedited_data
{
 unsigned short    opcode;             /* verb operation code        */
 unsigned char     opext;              /* verb extension code        */
 unsigned char     format;             /* format                     */
 unsigned short    primary_rc;         /* primary return code        */
 unsigned long     secondary_rc;       /* secondary return code      */
 unsigned char     tp_id[8];           /* TP identifier              */
 unsigned long     conv_id;            /* conversation identifier    */
 unsigned char     return_control;     /* when to return control     */
 unsigned char     reserv1[3];         /* reserved                   */
 unsigned char     rts_rcvd;           /* request to send received   */
 unsigned char     expd_data_rcvd;     /* expedited data received    */
 unsigned short    max_len;            /* maximum length of received */
                                       /* data                       */
 unsigned short    dlen;               /* actual length of received  */
                                       /* data                       */
 unsigned char    *dptr;               /* pointer to data buffer     */
} RECEIVE_EXPEDITED_DATA

typedef struct mc_receive_expedited_data
{
 unsigned short    opcode;             /* verb operation code        */
 unsigned char     opext;              /* verb extension code        */
 unsigned char     format;             /* format                     */
 unsigned short    primary_rc;         /* primary return code        */
 unsigned long     secondary_rc;       /* secondary return code      */
 unsigned char     tp_id[8];           /* TP identifier              */
 unsigned long     conv_id;            /* conversation identifier    */
 unsigned char     return_control;     /* when to return control     */
 unsigned char     reserv1[3];         /* reserved                   */
 unsigned char     rts_rcvd;           /* request to send received   */
 unsigned char     expd_data_rcvd;     /* expedited data received    */
 unsigned short    max_len;            /* maximum length of received */
                                       /* data                       */
 unsigned short    dlen;               /* actual length of received  */
                                       /* data                       */
 unsigned char    *dptr;               /* pointer to data buffer     */
} MC_RECEIVE_EXPEDITED_DATA
```

제공되는 매개변수

트랜잭션 프로 W 램은 다음 매 3 변수를 퍼스널 통신 및 통신 서버 ! 제 x 합니다.

opcode

AP_B_RECEIVE_EXPEDITED_DATA 

AP_M_RECEIVE_EXPEDITED_DATA 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATION z 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

return_control

트랜잭션 프로W램으로 제n를 리턴하는 시b를 지정합니다.

AP_WHEN_EXPD_RECEIVED

AP_IMMEDIATE

max_len

로컬 트랜잭션 프로W램이 수신할 수 있는 데이타 바이트의 최대 수.

범위: 0 – 86

이 *은 수신 데이타를 담을 수 있는 버퍼 f이를 초z하면 H 됩니다.

dptr 로컬 LU! 의해 수신된 데이타를 담b 위한 버퍼의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 fl **dptr**은 NULL로 설정되n_ 합니다.

리턴되는 매개변수

명령ｎ！ 실행되면 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_OK

rts_rcvd
수신된 송신 ｄ청 표시ｂ.
AP_YES
AP_NO

expd_data_rcvd
수신된 ＾송 데이타 표시ｂ. 이 표시는 RECEIVE_EXPEDITED_DATA ！ 발행될 때ｎ지 AP_YES로 설정되ｎ 있습니다.

AP_YES
AP_NO

dlen 수신된 데이타의 바이트 수(데이타는 **dptr** 매３변수！ 의해 지정된 버퍼！ 저장됩니다). ｆ이！ 0이면 데이타！ 수신되지 ＪＲ음을 나타냅니다. 수신되는 데이타는 포맷되지 ＪＲ습니다. 2바이트 ｆ이의 필드(LL)는 존재하지 Ｊ습니다.

명령ｎ！ 비블럭 명령ｎ이ｍ Ｆ직 Ｏ료되지 Ｊ은 ｆｌ, 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE 

opext AP_OPERATION_INCOMPLETE_FLAG

ｘ］LU！ ＾송 데이타를 지ｘ하지 Ｊｂ 때문！ 명령ｎ！ 실행되지 Ｊ은 ｆｌ, 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_EXPD_NOT_SUPPORTED_BY_LU

상대방 트랜잭션 프로Ｗ램！ 즉시 사ｋ할 수 있는 데이타！ ｘｍ **rtn_ctl** 플래Ｗ！ AP_IMMEDIATE인 ｆｌ, 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_UNSUCCESSFUL

트랜잭션 프로Ｗ램이 제ｘ하는 데이타 버퍼！ LU！서 사ｋ할 수 있는 ＾송 데이타를 모두 담을 만큼 크지 Ｊ은 ｆｌ, 데이타！ 리턴되지 Ｊ으며 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_BUFFER_TOO_SMALL

dlen LU ! 수신할 수 있는 ^ 송 데이타의 바이트 수.

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_EXPD_BAD_RETURN_CONTROL
AP_RCV_EXPD_INVALID_LENGTH

트랜잭션 프로 W 램이 이 명령 n 를 발행할 때 대화 상태 ! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_EXPD_DATA_BAD_CONV_STATE

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_ALLOCATION_ERROR
- AP_SECURITY_NOT_VALID
- AP_TRANS_PGM_NOT_AVAIL_RETRY
- AP_TRANS_PGM_NOT_AVAIL_NO_RTRY
- AP_TP_NAME_NOT_RECOGNIZED
- AP_PIP_NOT_ALLOWED
- AP_PIP_NOT_SPECIFIED_CORRECTLY
- AP_CONVERSATION_TYPE_MISMATCH
- AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_DEALLOC_NORMAL
AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED

MC_RECEIVE_EXPEDITED_DATA

AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

AP_ERROR_INDICATION 

[MC_]RECEIVE_IMMEDIATE

[MC_]RECEIVE_IMMEDIATE 명령ｎ는 현재 상대방 트랜잭션 프로Ｗ램！서 제ｘ하는 상태 정보나 데이타를 수신합니다. 제ｘ되는 데이타！ ｘ는 ｆｌ, 로컬 트랜잭션 프로Ｗ램은 즉시 리턴되ｍ 데이타를 ｂ다리지 Ｊ습니다.

VCB 구조

```
typedef struct receive_immediate
{
 unsigned short      opcode;           /* verb operation code        */
 unsigned char       opext;            /* verb extension code        */
 unsigned char       format;           /* format                     */
 unsigned short      primary_rc;       /* primary return code        */
 unsigned long       secondary_rc;     /* secondary return code      */
 unsigned char       tp_id[8];         /* TP identifier              */
 unsigned long       conv_id;          /* conversation identifier    */
 unsigned short      what_rcvd;        /* what received              */
 unsigned char       rtn_status;       /* return status with data    */
 unsigned char       fill;             /* data fill                  */
 unsigned char       rts_rcvd;         /* request to send received   */
 unsigned char       expd_data_rcvd;   /* expedited data received    */
 unsigned short      max_len;          /* maximum length of received */
                                       /* data                       */
 unsigned short      dlen;             /* actual length of received  */
                                       /* data                       */
 unsigned char      *dptr;             /* pointer to data buffer     */
 unsigned char       reserv5[5];       /* reserved                   */
} RECEIVE_IMMEDIATE;

typedef struct mc_receive_immediate
{
 unsigned short    opcode;             /* verb operation code        */
 unsigned char     opext;              /* verb extension code        */
 unsigned char     format;             /* format                     */
 unsigned short    primary_rc;         /* primary return code        */
 unsigned long     secondary_rc;       /* secondary return code      */
 unsigned char     tp_id[8];           /* TP identifier              */
 unsigned long     conv_id;            /* conversation identifier    */
 unsigned short    what_rcvd;          /* what received              */
 unsigned char     rtn_status;         /* return status with data    */
 unsigned char     reserv4;            /* reserved                   */
 unsigned char     rts_rcvd;           /* request to send received   */
 unsigned char     expd_data_rcvd;     /* expedited data received    */
 unsigned short    max_len;            /* maximum length of received */
                                       /* data                       */
 unsigned short    dlen;               /* actual length of received  */
                                       /* data                       */
 unsigned char    *dptr;               /* pointer to data buffer     */
 unsigned char     reserv6[5];         /* reserved                   */
} MC_RECEIVE_IMMEDIATE;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

MC_RECEIVE_IMMEDIATE

opcode

AP_B_RECEIVE_IMMEDIATE 

AP_M_RECEIVE_IMMEDIATE 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATIONz 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **[MC_]ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

rtn_status

상태 정보M 데이타! 동일한 명령n! 리턴되는지)부를 나타냅니다.

AP_YES

AP_NO

fill 

로컬 트랜잭션 프로W램이 데이타를 수신하는 방식을 나타냅니다.

AP_BUFFER

AP_LL

max_len

로컬 트랜잭션 프로W램이 수신할 수 있는 데이타 바이트의 최대수.

범위: 0 – 65535

이 *은 수신 데이타를 담을 수 있는 버퍼 f이를 초z하면 H 됩니다.

dptr 로컬 LU! 의해 수신된 데이타를 담b 위한 버퍼의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 f l **dptr**은 NULL로 설정되n _ 합니다.

리턴되는 매개변수

명령n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

AP_DEALLOC_NORMAL

what_rcvd

입력 데이타M 함께 수신된 상태 정보. **rtn_status**! AP_NO로 설정된 f l, 이 필드!는 반드시 다음 리스트의 첫번째 부분! 있는 *이 들n)니다. **rtn_status**! AP_YES로 설정된 f l, 이 필드!는 리스트의 F무 *이나 들n% 수 있습니다.

AP_NONE

AP_CONFIRM_DEALLOCATE

AP_CONFIRM_SEND

AP_CONFIRM_WHAT_RECEIVED

AP_DATA

AP_DATA_COMPLETE

AP_DATA_INCOMPLETE

AP_SEND 

AP_USER_CONTROL_DATA_COMPLETE 

AP_USER_CONTROL_DATA_INCMP 

AP_PS_HEADER_COMPLETE 

AP_PS_HEADER_INCOMPLETE 

AP_DATA_CONFIRM 

AP_DATA_COMPLETE_CONFIRM

AP_DATA_CONFIRM_DEALLOCATE

AP_DATA_COMPLETE_CONFIRM_DEALL

AP_DATA_CONFIRM_SEND

AP_DATA_COMPLETE_CONFIRM_SEND

AP_DATA_SEND 

다음 매 3 변수는 직접 명령 n ! 만 적 k 됩니다. 

AP_DATA_COMPLETE_SEND

AP_UC_DATA_COMPLETE_CONFIRM

AP_UC_DATA_COMPLETE_CNFM_DEALL

AP_UC_DATA_COMPLETE_CNFM_SEND

AP_UC_DATA_COMPLETE_SEND

AP_PS_HDR_COMPLETE_CONFIRM

AP_PS_HDR_COMPLETE_CNFM_DEALL

AP_PS_HDR_COMPLETE_CNFM_SEND

AP_PS_HDR_COMPLETE_SEND

expd_data_rcvd

수신된 ^ 송 데이타 표시 b .

AP_YES

AP_NO

rts_rcvd

수신된 송신 d 청 표시 b .

AP_YES

AP_NO

dlen 이 매 3 변수는 **what_rcvd** 매 3 변수 ! 데이타의 수신을 나타내는 f l ! 만 사 k 됩니다. 수신된 데이타의 바이트 수(데이타는 **dptr** 매 3 변수 ! 의해 지정된 버퍼 ! 저장됩니다.) f 이 ! 0이면 데이타 ! 수신되지 J R 음을 나타냅니다.

명령 n ! 비블럭 명령 n 이 m F 직 O 료되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_BASIC_CONVERSION 또는 AP_MAPPED_CONVERSATION

AP_NON_BLOCKING

AP_OPERATION_INCOMPLETE_FLAG

상대방 트랜잭션 프로 W 램 ! 즉시 사 k 할 수 있는 데이타 ! x 는 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_UNSUCCESSFUL

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID
AP_BAD_RETURN_STATUS_WITH_DATA
AP_BAD_TP_ID
AP_RCV_IMMD_BAD_FILL 

트랜잭션 프로 W 램이 이 명령 n 를 발행할 때 대화 상태 ! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_RCV_IMMD_BAD_STATE

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_ALLOCATION_ERROR
AP_SECURITY_NOT_VALID
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_TRANS_PGM_NOT_AVAIL_NO_RTRY
AP_TP_NAME_NOT_RECOGNIZED
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_CONVERSATION_TYPE_MISMATCH
AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY

AP_DEALLOC_ABEND 

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_SVC

MC_RECEIVE_IMMEDIATE

AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_DEALLOC_NORMAL
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING

AP_PROG_ERROR_TRUNC 

AP_SVC_ERROR_NO_TRUNC 

AP_SVC_ERROR_PURGING 

AP_SVC_ERROR_TRUNC 

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_DUPLEX_TYPE_MIXED
AP_CANCELLED

주: 성능상의 이유로 SNA API 클라이언트는 리턴 코드를 서버로 이송하지 않을 수도 있습니다. 후속 **[MC_]RECEIVE_IMMEDIATE** 명령은 리턴할 수 있는 유효한 리턴 코드를 **[MC_]SEND_DATA** 명령으로 발행되면 **[MC_]SEND_DATA** 처리를 위해 서버로 이송됩니다. **[MC_]SEND_DATA** 에 대한 리턴 코드로 있는 경우, **[MC_]RECEIVE_IMMEDIATE** 명령은 리턴됩니다. 유효한 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령을 참조하십시오.

[MC_]REQUEST_TO_SEND

[MC_]REQUEST_TO_SEND 명령ｎ는 로컬 트랜잭션 프로Ｗ램이 데이타를 송신하ｍ자 함을 상대방 트랜잭션 프로Ｗ램！ 통지합니다.

VCB 구조

```
typedef struct request_to_send
{
 unsigned short    opcode;              /* verb operation code     */
 unsigned char     opext;               /* verb extension code     */
 unsigned char     format;;             /* format                  */
 unsigned short    primary_rc;          /* primary return code     */
 unsigned long     secondary_rc;        /* secondary return code   */
 unsigned char     tp_id[8];            /* TP identifier           */
 unsigned long     conv_id;             /* conversation identifier */
} REQUEST_TO_SEND;

typedef struct mc_request_to_send
{
 unsigned short      opcode;            /* verb operation code     */
 unsigned char       opext;             /* verb extension code     */
 unsigned char       format;            /* format                  */
 unsigned short      primary_rc;        /* primary return code     */
 unsigned long       secondary_rc;      /* secondary return code   */
 unsigned char       tp_id[8];          /* TP identifier           */
 unsigned long       conv_id;           /* conversation identifier */
} MC_REQUEST_TO_SEND;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

opcode

AP_B_REQUEST_TO_SEND 

AP_M_REQUEST_TO_SEND 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 ｎ５의 ｆｌ, 이 플래Ｗ는 AP_NON_BLOCKINGｚ 논리합(OR)이 될 수 있습니다.

format

VCB의 포맷을 식별합니다. 위！ 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시＠.

tp_id 로컬 트랜잭션 프로Ｗ램！ 대한 식별자.

이 매３변수의 ＊은 ｂ동 트랜잭션 프로Ｗ램의 **TP_STARTED** 명령ｎ나 피ｂ동 트랜잭션 프로Ｗ램의 **RECEIVE_ALLOCATE**！ 의해 리턴되ｚ습니다.

conv_id
대화 식별자(ID).

이 매개변수의 값은 동기 트랜잭션 프로그램의 **[MC_]ALLOCATE** 명령이나 비동기 트랜잭션 프로그램의 **RECEIVE_ALLOCATE** 에 의해 리턴됩니다.

리턴되는 매개변수

명령이 실행되면 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_OK

명령이 비블럭 명령이며 아직 완료되지 않은 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

[MC_]REQUEST_TO_SEND 는 비블럭 모드로 발행되며(43페이지의 『대화 레벨 비블럭화』 참조) 송신/수신 대화의 명령을 처리하는 동안 대화가 종료되면 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_CONVERSATION_ENDED

응용프로그램은 이 대화에 대해 더 이상의 명령을 발행하면 안 됩니다.

매개변수 오류로 인해 명령이 실행되지 않은 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_R_T_S_INVALID_FOR_FDX

트랜잭션 프로그램이 이 명령을 발행할 때 대화 상태가 나쁜 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_STATE_CHECK

secondary_rc
AP_R_T_S_BAD_STATE

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x 통 리턴 코드! 설명되n 있습니다.

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

[MC_]SEND_CONVERSATION

[MC_]SEND_CONVERSATION 명령ｎ는 로컬 LUＭ 상대방 LU 사이의 세션！ 대화를 할당하ｍ(상대방 LU의 트랜잭션 프로Ｗ램이 시작되Ｔ 함) 이 대화의 단일 대화 레코드를 송신하며 확인을 ｂ다리지 Ｊｍ 대화를 할당해제합니다. 이Ｍ은 **[MC_]ALLOCATE**, **[MC_]SEND_DATA**, **[MC_]DEALLOCATE(FLUSH)**의 명령ｎ 순서Ｍ 동일합니다(보통 『단일 일방향 브래킷』이라ｍ도 함).

VCB 구조

```
typedef struct send_conversation
{
 unsigned short   opcode;                /* verb operation code           */
 unsigned char    opext;                 /* verb extension code           */
 unsigned char    format;                /* format                        */
 unsigned short   primary_rc             /* primary return code           */
 unsigned long    secondary_rc;          /* secondary return code         */
 unsigned char    tp_id[8];              /* TP identifier                 */
 unsigned char    reserv3[8];            /* reserved                      */
 unsigned char    rtn_ctl;               /* return control                */
 unsigned char    reserv4;               /* reserved                      */
 unsigned long    conv_group_id;         /* conversation group identifier */
 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];           /* TP name                       */
 unsigned char    security;              /* security                      */
 unsigned char    reserv5[11];           /* reserved                      */
 unsigned char    pwd[10];               /* security password             */
 unsigned char    user_id[10];           /* security user_id              */
 unsigned short   pip_dlen;              /* PIP data length               */
 unsigned char   *pip_dptr;              /* pointer to PIP data           */
 unsigned char    reserv5a;              /* reserved                      */
 unsigned char    fqplu_name[17];        /* fully qualified partner LU    */
                                         /* name                          */
 unsigned char    reserv6[8];            /* reserved                      */
 unsigned short   dlen;                  /* data length                   */
 unsigned char   *dptr;                  /* pointer to data buffer        */
} SEND_CONVERSATION;

typedef struct mc_send_conversation
{
 unsigned short    opcode;               /* verb operation code           */
 unsigned char     opext;                /* verb extension code           */
 unsigned char     format;               /* format                        */
 unsigned short    primary_rc;           /* primary return code           */
 unsigned long     secondary_rc;         /* secondary return code         */
 unsigned char     tp_id[8];             /* TP identifier                 */
 unsigned char     reserv3[8];           /* reserved                      */
 unsigned char     rtn_ctl;              /* return control                */
 unsigned char     reserv4;              /* reserved                      */
 unsigned long     conv_group_id;        /* conversation group identifier */
 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];          /* TP name                       */
 unsigned char     security;             /* security                      */
```

```
 unsigned char     reserv6[11];          /* reserved                      */
 unsigned char     pwd[10];              /* security password             */
 unsigned char     user_id[10];          /* security user_id              */
 unsigned short    pip_dlen;             /* PIP data length               */
 unsigned char    *pip_dptr;             /* pointer to PIP data           */
 unsigned char     reserv6a;             /* reserved                      */
 unsigned char     fqplu_name[17];       /* fully qualified partner LU    */
                                         /* name                          */
 unsigned char     reserv7[8];           /* reserved                      */
 unsigned short    dlen;                 /* data length                   */
 unsigned char    *dptr;                 /* pointer to data buffer        */
} MC_SEND_CONVERSATION;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_SEND_CONVERSATION 

AP_M_SEND_CONVERSATION 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKINGz 논리합(OR)이 될 수 있습니다.

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE** 명령n! 의해 리턴되z습니다.

rtn_ctl

로컬 트랜잭션 프로세서의 세션 d청! 따라 작k하는 로컬 LU! 로컬 트랜잭션 프로W램! 제n를 리턴하는 시b를 지정합니다.

AP_IMMEDIATE
AP_WHEN_SESSION_ALLOCATED
AP_WHEN_SESSION_FREE
AP_WHEN_CONV_GROUP_ALLOC
AP_WHEN_CONWINNER_ALLOC
AP_WHEN_CONLOSER_ALLOC

conv_group_id

할당할 세션! 대한 대화 W룹 식별자. 이 매3변수는 **rtn_ctl**이 AP_WHEN_CONV_GROUP_ALLOC로 설정된 fl!만 제x됩니다.

plu_alias

로컬 트랜잭션 프로W램! 상대방 LU를 식별해주는 별명. 이M은 로컬로 표시할 수 있는 문자 세트의 8바이트 문자- 입니다. 8바이트! 모두 중d하며 반드시 설정해_ 합니다. 이 이름은 8성시 설정된 상대방 LU의 이름z 일치해_ 합니다.

이 필드! 모두 0으로 설정되면 퍼스널 통신 및 통신 서버는 **fqplu_name** 필드를 사k하) d8되는 상대방 LU를 지정합니다.

mode_name

8성시 정의된 네트v크 특성 세트의 이름. 이M은 8바이트의 5숫자로 된 유형 A EBCDIC 문자-(문자로 시작됨)이며 @른쪽은 EBCDIC x백으로 채v집니다.

tp_name

피b동 트랜잭션 프로W램의 이름. 퍼스널 통신 및 통신 서버는 이 필드의 문자 세트를 점K하지 J습니다. b동 트랜잭션 프로W램의 **ALLOCATE** 명령n! 의해 지정된 **tp_name**의 *은 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 지정된 **tp_name**의 * z 일치해_ 합니다.

security

상대방 LU! 피b동 트랜잭션 프로W램! 대한 W세스를 K증하려면 필d로 하는 정보를 지정합니다.

AP_NONE

AP_PGM

AP_SAME

AP_PGM_STRONG

pwd **user_id**M , |된 0호. 10바이트의 유형 AE EBCDIC 문자-이며 @른쪽은 EBCDIC x백으로 채v집니다. 보H=프로W램(AP_PGM 또는 AP_PGM_STRONG)일 f l 이M이 필d하며 W렇지 J으면 생략!능합니다.

user_id

상대방 트랜잭션 프로W램! W세스하는 데 필d한 사k자 ID. 10바이트의 유형 AE EBCDIC 문자-이며 @른쪽은 EBCDIC x백으로 채v집니다. 보H=프로W램(AP_PGM 또는 AP_PGM_STRONG)일 f l 이M이 필d하며 W렇지 J으면 생략!능합니다.

pip_dlen

상대방 트랜잭션 프로W램! 전달될 프로W램 초b화 매3변수(PIP)의 f이.

범위: 0 – 32767

pip_dptr

PIP 데이타! 들n 있는 버퍼의 주소. **pip_dlen**이 0보다 큰 f l !만 이 매3변수를 사k하십시@.

fqplu_name

상대방 LU ! 대한 전체 정식 LU명. 이 이름은 17바이트 f이이며 @ 른쪽이 EBCDIC x백으로 채v집니다. 이M은 EBCDIC 점으로 , a 된 두 3의 유형 A EBCDIC 문자 - 로 8성됩니다. (" 이름은 x백 x이 8바이트의 최대 f이를 ! 질 수 있습니다. 네트v크 ID ! x으면 점을 생략하십시@.) 이 필드는 **plu_alias** 필드 ! 모두 0으로 설정된 f l ! 만 의미 ! 있습니다.

dlen 송신할 데이타의 바이트 수.

범위: 0 – 65535

dptr 송신할 데이타 ! 들n 있는 버퍼의 주소. n플리케이션은 VCB의 끝 ! 데이타를 추 ! 할 수 있으며 이 f l **dptr**은 NULL로 설정되n _ 합니다.

리턴되는 매개변수

명령n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

conv_group_id

대화 ! 할당된 세션의 대화 W룹 식별자.

명령n ! 비블럭 명령n이m F직 O료되지 J은 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

rtn_ctl 매3변수 ! AP_IMMEDIATE로 설정되z m 즉시 사k할 수 있는 세션이 x는 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_UNSUCCESSFUL

매3변수 @류로 인해 명령n ! 실행되지 J은 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_TP_ID

AP_BAD_LL 

AP_BAD_RETURN_CONTROL
AP_BAD_SECURITY
AP_PIP_LEN_INCORRECT

AP_NO_USE_OF_SNASVCMG 

AP_UNKNOWN_PARTNER_MODE

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_UNSUCCESSFUL 

AP_ALLOCATION_ERROR
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_SEC_REQUESTED_NOT_SUPPORTED 

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED 

AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

primary_rc ! AP_ALLOCATION_ERROR로 설정된 f l, **sense_data** 필드 ! 는 실패 ! | 한 좀더 많은 정보 ! 포함됩니다.

[MC_]SEND_DATA

[MC_]SEND_DATA 명령ｎ는 상대방 트랜잭션 프로Ｗ램으로 전송되도록 로컬 LU의 송신 버퍼！ 데이타를 저장합니다.

VCB 구조

```
typedef struct send_data
{
 unsigned short    opcode;               /* verb operation code      */
 unsigned char     opext;                /* verb extension code      */
 unsigned char     format;               /* format                   */
 unsigned short    primary_rc;           /* primary return code      */
 unsigned long     secondary_rc;         /* secondary return code    */
 unsigned char     tp_id[8];             /* TP identifier            */
 unsigned long     conv_id;              /* conversation identifier  */
 unsigned char     rts_rcvd;             /* request to send received */
 unsigned char     expd_data_rcvd;       /* expedited data received  */
 unsigned short    dlen;                 /* data length              */
 unsigned char    *dptr;                 /* pointer to data          */
 unsigned char     type;                 /* send data type           */
 unsigned char     reserv4;              /* reserved                 */
} SEND_DATA;

typedef struct mc_send_data
{
 unsigned short    opcode;             /* verb operation code      */
 unsigned char     opext;              /* verb extension code      */
 unsigned char     format;             /* format                   */
 unsigned short    primary_rc;         /* primary return code      */
 unsigned long     secondary_rc;       /* secondary return code    */
 unsigned char     tp_id[8];           /* TP identifier            */
 unsigned long     conv_id;            /* conversation identifier  */
 unsigned char     rts_rcvd;           /* request to send received */
#ifdef WINAPPC_FORMAT_1
 unsigned char     expd_data_rcvd;     /* expedited data received  */
#else
 unsigned char     data_type;          /* data type received       */
#endif
 unsigned short    dlen;               /* data length              */
 unsigned char    *dptr;               /* pointer to data          */
 unsigned char     type;               /* send data type           */
#ifdef WINAPPC_FORMAT_1
 unsigned char     data_type;          /* data type received       */
#else
 unsigned char     reserv4;            /* reserved                 */
#endif
} MC_SEND_DATA;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

opcode

AP_B_SEND_DATA 

AP_M_SEND_DATA 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

0전 g방향 대화!서 이 플래W는 AP_FULL_DUPLEX_CONVERSATION z 논리합(OR)이 되n_ 합니다.

format

VCB의 포맷. 위! 나-된 포맷을 r으려면 이M을 1로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **[MC_]ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

dlen 로컬 LU의 송신 버퍼! 놓일 데이타의 바이트 수.

범위: 0 – 65535

dptr 로컬 LU의 송신 버퍼! 놓일 데이타! 들n 있는 버퍼 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 fl **dptr**은 NULL로 설정되n_ 합니다.

type **SEND_DATA** \! 또다른 명령n의 b능을 수행할지)부를 지정합니다.

AP_NONE
AP_SEND_DATA_CONFIRM
AP_SEND_DATA_FLUSH
AP_SEND_DATA_P_TO_R_FLUSH
AP_SEND_DATA_P_TO_R_SYNC_LEVEL
AP_SEND_DATA_P_TO_R_CONFIRM
AP_SEND_DATA_DEALLOC_FLUSH
AP_SEND_DATA_DEALLOC_SYNC_LEVE
AP_SEND_DATA_DEALLOC_CONFIRM
AP_SEND_DATA_DEALLOC_ABEND

리턴되는 매개변수

명령ｎ！ 실행되면 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

주: 성능상의 이유로 SNA API 클라이ｐ트는 리턴 코드를 서버！ 이송하지 Ｊｍ 성ｘ적인 리턴 코드를 **[MC_]SEND_DATA** 명령ｎ！ 리턴할 수 있습니다. 후속 **[MC_]SEND_DATA** 명령ｎ！ 발행되면 **[MC_]SEND_DATA** ！ 처리를 위해 서버로 이송됩니다.

SEND_DATA @류 리턴 코드！ 있으면 후속되는 명령ｎ！ 리턴됩니다.

primary_rc
AP_OK

rts_rcvd
수신된 송신 ｄ청 표시ｂ.
AP_YES
AP_NO

expd_data_rcvd
수신된 ^송 데이타 표시ｂ. 이 표시는 RECEIVE_EXPEDITED_DATA ！ 발행될 때ｎ지 AP_YES로 설정되ｎ 있습니다.
AP_YES
AP_NO

매３변수 @류로 인해 명령ｎ！ 실행되지 Ｊ은 ｆｌ, 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

opext AP_OPERATION_INCOMPLETE_FLAG

매３변수 @류로 인해 명령ｎ！ 실행되지 Ｊ은 ｆｌ, 퍼스널 통신 및 통신 서버는 다음 매３변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID
AP_BAD_TP_ID

AP_BAD_LL 

AP_SEND_DATA_INVALID_TYPE
AP_SEND_DATA_CONFIRM_SYNC_NONE
AP_SEND_TYPE_INVALID_FOR_FDX

MC_SEND_DATA

트랜잭션 프로W램이 이 명령n를 발행할 때 대화 상태! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_STATE_CHECK

secondary_rc

AP_SEND_DATA_NOT_SEND_STATE

AP_SEND_DATA_NOT_LL_BDY 

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_ALLOCATION_ERROR

- AP_SECURITY_NOT_VALID
- AP_TRANS_PGM_NOT_AVAIL_RETRY
- AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
- AP_TP_NAME_NOT_RECOGNIZED
- AP_PIP_NOT_ALLOWED
- AP_PIP_NOT_SPECIFIED_CORRECTLY
- AP_CONVERSATION_TYPE_MISMATCH
- AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY

AP_DEALLOC_ABEND 

AP_DEALLOC_ABEND_PROG 

AP_DEALLOC_ABEND_SVC 

AP_DEALLOC_ABEND_TIMER 

AP_PROG_ERROR_PURGING

AP_SVC_ERROR_PURGING 

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED

AP_DUPLEX_TYPE_MIXED 

AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED
AP_ERROR_INDICATION

- AP_ALLOCATION_ERROR_PENDING
- AP_DEALLOC_ABEND_PROG_PENDING
- AP_DEALLOC_ABEND_SVC_PENDING
- AP_DEALLOC_ABEND_TIMER_PENDING
- AP_UNKNOWN_ERROR_TYPE_PENDING

[MC_]SEND_ERROR

[MC_]SEND_ERROR 명령 n는 로컬 트랜잭션 프로W램！ n플리케이션 레벨의 @류！ 발생했음을 상대방 트랜잭션 프로W램！ 통보합니다.

VCB 구조

```
typedef struct send_error
{
 unsigned short     opcode;              /* verb operation code      */
 unsigned char      opext;               /* verb extension code      */
 unsigned char      format;              /* format                   */
 unsigned short     primary_rc;          /* primary return code      */
 unsigned long      secondary_rc;        /* secondary return code    */
 unsigned char      tp_id[8];            /* TP identifier            */
 unsigned long      conv_id;             /* conversation identifier  */
 unsigned char      rts_rcvd;            /* request to send received */
 unsigned char      err_type;            /* error type               */
 unsigned char      err_dir;             /* error direction          */
 unsigned char      expd_data_rcvd;      /* expedited data received  */
 unsigned short     log_dlen;            /* log data length          */
 unsigned char     *log_dptr;            /* pointer to log data      */
} SEND_ERROR;
```

```
typedef struct mc_send_error
{
 unsigned short   opcode;              /* verb operation code      */
 unsigned char    opext;               /* verb extension code      */
 unsigned char    format;              /* format                   */
 unsigned short   primary_rc;          /* primary return code      */
 unsigned long    secondary_rc;        /* secondary return code    */
 unsigned char    tp_id[8];            /* TP identifier            */
 unsigned long    conv_id;             /* conversation identifier  */
 unsigned char    rts_rcvd;            /* request to send received */
 unsigned char    err_type;            /* error type               */
 unsigned char    err_dir;             /* error direction          */
 unsigned char    expd_data_rcvd;      /* expedited data received  */
 unsigned char    reserv5[2];          /* reserved                 */
 unsigned char    reserv6[4];          /* reserved                 */
} MC_SEND_ERROR;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매 3변수를 퍼스널 통신 및 통신 서버！ 제 x합니다.

opcode

AP_B_SEND_ERROR 

AP_M_SEND_ERROR 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n5의 fl, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

0 전 g 방향 대화 ! 서 이 플래 W 는 AP_FULL_DUPLEX_CONVERSATION z 논리합(OR)이 되 n _ 합니다.

format

VCB의 포맷을 식별합니다. 위 ! 나 - 된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시 @ .

tp_id 로컬 트랜잭션 프로 W 램 ! 대한 식별자.

이 매 3 변수의 * 은 b 동 트랜잭션 프로 W 램의 **TP_STARTED** 명령 n 나 피 b 동 트랜잭션 프로 W 램의 **RECEIVE_ALLOCATE** ! 의해 리턴되 z 습니다.

conv_id

대화 식별자(ID).

이 매 3 변수의 * 은 b 동 트랜잭션 프로 W 램의 **[MC_]ALLOCATE** 명령 n 나 피 b 동 트랜잭션 프로 W 램의 **RECEIVE_ALLOCATE** ! 의해 리턴되 z 습니다.

err_type

보 m 되 m 있는 @ 류의 유형 즉, n 플리케이션 프로 W 램 또는 서비스 프로 W 램을 나타냅니다.

AP_PROG
AP_SVC

err_dir

보 m 되 m 있는 @ 류 ! 상대방 트랜잭션 프로 W 램 ! 서 수신된 데이타 ! 서 발생했는지 또는 로컬 트랜잭션 프로 W 램이 송신하려던 데이타 ! 서 발생했는지를 나타냅니다.

이 매 3 변수는 **SEND_ERROR** 명령 n ! SEND_PENDING 상태 ! 서 발행되는 f l ! 만 사 k 됩니다.

AP_RCV_DIR_ERROR

AP_SEND_DIR_ERROR

log_dlen

@ 류 로 W 파일 ! 송신될 데이타의 바이트 수.

범위: 0 – 32767

n 플리케이션은 데이타를 VCB의 끝 ! 추 ! 할 수 있으며 이 f l 이 필드는 0보다 커지며 **log_dptr**은 NULL로 설정되 n _ 합니다. (f 이 ! 0이면 @ 류 로 W 데이타 ! x 음을 나타냅니다.)

log_dptr 

@류 정보! 들n 있는 데이타의 주소. n플리케이션은 VCB의 끝! 데이타를 추!할 수 있으며 이 fl **log_dptr**는 NULL로 설정되n _ 합니다.

이 데이타는 로컬 @류 로WM 상대방 LU로 송신됩니다. 이 매3변수는 **log_dlen**이 0보다 큰 fl **SEND_ERROR** 명령n!서 사k됩니다.

트랜잭션 프로W램은 @류 데이타를 일반 데이타 스트림(GDS) @류로W 변수로 포맷해_ 합니다. 자세한 내k은 *IBM System Network Architecture: LU 6.2 Reference: Peer Protocols*를 참조하십시@.

리턴되는 매개변수

명령n! 실행되면 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OK

rts_rcvd
수신된 송신 d청 표시b.
AP_YES
AP_NO

expd_data_rcvd
수신된 ^송 데이타 표시b. 이 표시는 RECEIVE_EXPEDITED_DATA ! 발행될 때n지 AP_YES로 설정되n 있습니다.
AP_YES
AP_NO

명령n! 비블럭 명령n이m F직 O료되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_ERROR_DIRECTION

AP_BAD_TP_ID

AP_SEND_ERROR_BAD_TYPE 

AP_SEND_ERROR_LOG_LL_WRONG 

트랜잭션 프로W램이 이 명령n를 발행할 때 대화 상태! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_STATE_CHECK

secondary_rc

AP_SEND_ERROR_BAD_STATE

다음z 0은 1차 리턴 코드(**primary_rc**)M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

허용된 상태에서 발행되는 명령어

다음 리턴 코드는 **[MC_]SEND_ERROR** 명령n! 허k된 상태!서 발행된 f l! 생성됩니다.

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED
AP_ERROR_INDICATION

AP_ALLOCATION_ERROR_PENDING

AP_DEALLOC_ABEND_PROG_PENDING

AP_DEALLOC_ABEND_SVC_PENDING

AP_DEALLOC_ABEND_TIMER_PENDING

AP_UNKNOWN_ERROR_TYPE_PENDING

송신 상태에서 발행된 명령어: 다음 리턴 코드는 **[MC_]SEND_ERROR** 명령n! 송신 상태!서 발행된 f l!만 생성됩니다.

AP_ALLOCATION_ERROR

AP_SECURITY_NOT_VALID

AP_TRANS_PGM_NOT_AVAIL_RETRY

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

AP_TP_NAME_NOT_RECOGNIZED
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_CONVERSATION_TYPE_MISMATCH
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_DEALLOC_ABEND 
AP_DEALLOC_ABEND_PROG 
AP_DEALLOC_ABEND_SVC 
AP_DEALLOC_ABEND_TIMER 
AP_PROG_ERROR_PURGING
AP_SVC_ERROR_PURGING 

수신 상태에서 발행된 명령어: 다음 리턴 코드는 수신 상태!서 명령n! 발행된 fl!만 생성됩니다.

AP_DEALLOC_NORMAL

주: 성능상의 이유로 SNA API 클라이p트는 리턴 코드를 서버! 이송하지 Jm 성x적인 리턴 코드를 **[MC_]SEND_DATA** 명령n! 리턴할 수 있습니다. 다음의 **[MC_]SEND_ERROR** 명령n! 발행되면 **[MC_]SEND_DATA**! 처리를 위해 서버로 이송됩니다.

[MC_]SEND_DATA @류 리턴 코드! 있는 fl, **[MC_]SEND_ERROR** 명령n! 리턴됩니다. @류 리턴 코드의 리스트를 보려면 **[MC_]SEND_DATA** 명령n를 참조하십시@.

[MC_]SEND_EXPEDITED_DATA

OS/2 및 Windows 3.1 k 통신 서버 SNA API 클라이 p 트 ! 는 지 x 되지 J 습니다.

[MC_]SEND_EXPEDITED_DATA 명령 n 는 상대방 트랜잭션 프로 W 램으로 전송되도록 로컬 LU의 ^ 송 송신 버퍼 ! 데이타를 저장합니다. 이 데이타는 비 ^ 송 데이타 ! 전송되 b 이전 ! 상대방 트랜잭션 프로 W 램 ! 도착합니다.

VCB 구조

```
typedef struct send_expedited_data
{
 unsigned short     opcode;                /* verb operation code      */
 unsigned char      opext;                 /* verb extension code      */
 unsigned char      format;                /* format                   */
 unsigned short     primary_rc;            /* primary return code      */
 unsigned long      secondary_rc;          /* secondary return code    */
 unsigned char      tp_id[8];              /* TP identifier            */
 unsigned long      conv_id;               /* conversation identifier  */
 unsigned char      rts_rcvd;              /* request to send received */
 unsigned char      expd_data_rcvd;        /* expedited data received  */
 unsigned short     dlen;                  /* data length              */
 unsigned char     *dptr;                  /* pointer to data          */
 unsigned char      reserve4[2];           /* TP identifier            */
} SEND_EXPEDITED_DATA;

typedef struct mc_send_expedited_data
{
 unsigned short    opcode;              /* verb operation code        */
 unsigned char     opext;               /* verb extension code        */
 unsigned char     format;              /* format                     */
 unsigned short    primary_rc;          /* primary return code        */
 unsigned long     secondary_rc;        /* secondary return code      */
 unsigned char     tp_id[8];            /* TP identifier              */
 unsigned long     conv_id;             /* conversation identifier    */
 unsigned char     rts_rcvd;            /* request to send received   */
 unsigned char     expd_data_rcvd;      /* expedited data received    */
                                        /* transaction plan           */
                                        /* data                       */
 unsigned short    dlen;                /* actual length of received  */
                                        /* data                       */
 unsigned char    *dptr;                /* pointer to data buffer     */
 unsigned char     reserv4[2];          /* reserved                   */
} MC_SEND_EXPEDITED_DATA
```

제공되는 매개변수

트랜잭션 프로 W 램은 다음 매 3 변수를 퍼스널 통신 및 통신 서버 ! 제 x 합니다.

opcode

AP_B_SEND_EXPEDITED_DATA 

AP_M_SEND_EXPEDITED_DATA

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION 비블럭 n 5의 f l, 이 플래W는 AP_NON_BLOCKING z 논리합(OR)이 될 수 있습니다.

0 전 g 방향 대화 ! 서 이 플래 W 는 AP_FULL_DUPLEX_CONVERSATION z 논리합(OR)이 되 n _ 합니다.

format

VCB의 포맷을 식별합니다. 위 ! 나 - 된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램 ! 대한 식별자.

이 매 3 변수의 * 은 b 동 트랜잭션 프로W램의 **TP_STARTED** 명령 n 나 피 b 동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE** ! 의해 리턴되 z 습니다.

conv_id

대화 식별자(ID).

이 매 3 변수의 * 은 b 동 트랜잭션 프로W램의 **[MC_]ALLOCATE** 명령 n 나 피 b 동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE** ! 의해 리턴되 z 습니다.

dlen 로컬 LU의 송신 버퍼 ! 놓일 데이타의 바이트 수.

범위: 1 – 86

dptr @ 류 정보 ! 들 n 있는 데이타의 주소. n 플리케이션은 VCB의 끝 ! 데이타를 추 ! 할 수 있으며 이 f l **dptr**은 NULL로 설정되 n _ 합니다.

포맷되지 J 은 2바이트 f 이의 필드(LL)는 존재하지 J 습니다.

리턴되는 매개변수

명령 n ! 실행되면 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_OK

rts_rcvd

수신된 송신 d 청 표시 b .

AP_YES

AP_NO

expd_data_rcvd

수신된 ^ 송 데이타 표시 b . 이 표시는 RECEIVE_EXPEDITED_DATA ! 발행될 때 n 지 AP_YES로 설정되 n 있습니다.

AP_YES

AP_NO

명령 n ! 비블럭 명령 n 이 m F 직 O 료되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

x] LU ! ^ 송 데이타를 지 x 하지 J b 때문 ! 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

매 3 변수 @ 류로 인해 명령 n ! 실행되지 J 은 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID

AP_SEND_EXPD_INVALID_LENGTH

AP_RCV_EXPD_INVALID_LENGTH 

트랜잭션 프로 W 램이 이 명령 n 를 발행할 때 대화 상태 ! 나쁜 f l, 퍼스널 통신 및 통신 서버는 다음 매 3 변수를 리턴합니다.

primary_rc

AP_STATE_CHECK

secondary_rc

AP_EXPD_DATA_BAD_CONV_STATE

다음 z 0 은 1차 리턴 코드(**primary_rc**) M 인덴트된 2차 리턴 코드(**secondary_rc**)를 생성하는 조 G 은 부록A. APPC x 통 리턴 코드 ! 설명되 n 있습니다.

AP_ALLOCATION_ERROR

AP_SECURITY_NOT_VALID

AP_TRANS_PGM_NOT_AVAIL_RETRY

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

AP_TP_NAME_NOT_RECOGNIZED

AP_PIP_NOT_ALLOWED

MC_SEND_EXPEDITED_DATA

AP_PIP_NOT_SPECIFIED_CORRECTLY

AP_CONVERSATION_TYPE_MISMATCH

AP_SYNC_LEVEL_NOT_SUPPORTED

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

[MC_]TEST_RTS

[MC_]TEST_RTS 명령ｎ는 송신 ｄ청 통지！ 상대방 트랜잭션 프로Ｗ램으로부터 수신되ｚ는지)부를 판별합니다.

VCB 구조

```
typedef struct test_rts
{
 unsigned short     opcode;              /* verb operation code     */
 unsigned char      opext;               /* verb extension code     */
 unsigned char      format;              /* format                  */
 unsigned short     primary_rc;          /* primary return code     */
 unsigned long      secondary_rc;        /* secondary return code   */
 unsigned char      tp_id[8];            /* TP identifier           */
 unsigned long      conv_id;             /* conversation identifier */
 unsigned char      reserv3;             /* reserved                */
} TEST_RTS;

typedef struct mc_test_rts
{
 unsigned short    opcode;             /* verb operation code     */
 unsigned char     opext;              /* verb extension code     */
 unsigned char     format;             /* format                  */
 unsigned short    primary_rc;         /* primary return code     */
 unsigned long     secondary_rc;       /* secondary return code   */
 unsigned char     tp_id[8];           /* TP identifier           */
 unsigned long     conv_id;            /* conversation identifier */
 unsigned char     reserv3;            /* reserved                */
} MC_TEST_RTS;
```

제공되는 매개변수

트랜잭션 프로Ｗ램은 다음 매３변수를 퍼스널 통신 및 통신 서버！ 제ｘ합니다.

opcode

AP_B_TEST_RTS 

AP_M_TEST_RTS 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION

format

VCB의 포맷을 식별합니다. 위！ 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시＠.

tp_id 로컬 트랜잭션 프로Ｗ램！ 대한 식별자. 이 매３변수의 *은 ｂ동 트랜잭션 프로Ｗ램의 **TP_STARTED** 명령ｎ나 피ｂ동 트랜잭션 프로Ｗ램의 **RECEIVE_ALLOCATE**！ 의해 리턴되ｚ습니다.

conv_id

대화 식별자(ID).

이 매개변수의 값은 동기 트랜잭션 프로그램의 **[MC_]ALLOCATE** 명령나 비동기 트랜잭션 프로그램의 **RECEIVE_ALLOCATE** 의해 리턴됩니다.

리턴되는 매개변수

명령나 실행되면 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
송신 대기 통지 상대방 트랜잭션 프로그램으로 수신되는지 부를 나타냅니다.

AP_OK

AP_UNSUCCESSFUL

매개변수 오류로 인해 명령나 실행되지 않은 경우, 퍼스널 통신 및 통신 서버는 다음 매개변수를 리턴합니다.

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_TEST_INVALID_FOR_FDX

다음 표는 1차 리턴 코드(**primary_rc**)를 생성하는 조건을 부록A. APPC 통신 리턴 코드 설명되어 있습니다.

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR

[MC_]TEST_RTS_AND_POST

[MC_]TEST_RTS_AND_POST 명령n는 송신 d청 통지! 상대방 트랜잭션 프로W램으로부터 수신되z는지)부를 비동b로 판별합니다. 트랜잭션 프로W램은 대화! 또다른 명령n! 미해a 상태로 있는 fl라도 **[MC_]TEST_RTS_AND_POST**를 발행할 수 있습니다. **[MC_]TEST_RTS_AND_POST**는 송신 d청 통지! 수신될 때 대화! 종료될 때 또는 대화 장V! 탐지될 때 리턴됩니다.

이 명령n는 APPC #트리 포인트를 통해서만 발행할 수 있습니다.

VCB 구조

```
typedef struct test_rts_and_post
{
 unsigned short      opcode;              /* verb operation code     */
 unsigned char       opext;               /* verb extension code     */
 unsigned char       format;              /* format                  */
 unsigned short      primary_rc;          /* primary return code     */
 unsigned long       secondary_rc;        /* secondary return code   */
 unsigned char       tp_id[8];            /* TP identifier           */
 unsigned long       conv_id;             /* conversation identifier */
 unsigned char       reserv3;             /* reserved                */
 unsigned long       sema;                /* post handle for verb    */
} TEST_RTS_AND_POST;

typedef struct mc_test_rts_and_post
{
 unsigned short    opcode;             /* verb operation code     */
 unsigned char     opext;              /* verb extension code     */
 unsigned char     format;             /* format                  */
 unsigned short    primary_rc;         /* primary return code     */
 unsigned long     secondary_rc;       /* secondary return code   */
 unsigned char     tp_id[8];           /* TP identifier           */
 unsigned long     conv_id;            /* conversation identifier */
 unsigned char     reserv3;            /* reserved                */
 unsigned long     sema;               /* post handle for verb    */
} MC_TEST_RTS_AND_POST;
```

제공되는 매개변수

트랜잭션 프로W램은 다음 매3변수를 퍼스널 통신 및 통신 서버! 제x합니다.

opcode

AP_B_TEST_RTS_AND_POST 

AP_M_TEST_RTS_AND_POST 

opext AP_BASIC_CONVERSATION 또는 AP_MAPPED_CONVERSATION

format

VCB의 포맷을 식별합니다. 위! 나-된 VCB 버전을 지정하려면 이 필드를 0으로 설정하십시@.

tp_id 로컬 트랜잭션 프로W램! 대한 식별자.

이 매3변수의 *은 b동 트랜잭션 프로W램의 **TP_STARTED** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

conv_id

대화 식별자(ID).

이 매3변수의 *은 b동 트랜잭션 프로W램의 **[MC_]ALLOCATE** 명령n나 피b동 트랜잭션 프로W램의 **RECEIVE_ALLOCATE**! 의해 리턴되z습니다.

sema n플리케이션이 처리하는 이벤트의 핸들. 이 명령n는 Win32 API! 서 WaitForMultipleObjectsM 함께 사k하b 위한 M입니다. 이 함수! 대한 자세한 내k은 Win32 API! 대한 프로W래밍 책을 참조하십시@.

리턴되는 매개변수

명령n! 실행되면(즉, 송신 d청 통지! 수신되면) 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_OK

대화! 종료되E나 대화 장V! 탐지되n 명령n! 리턴되는 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_UNSUCCESSFUL

매3변수 @류로 인해 명령n! 실행되지 J은 fl, 퍼스널 통신 및 통신 서버는 다음 매3변수를 리턴합니다.

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_TEST_INVALID_FOR_FDX

다음z 0은 1차 리턴 코드(**primary_rc**)를 생성하는 조G은 부록A. APPC x통 리턴 코드! 설명되n 있습니다.

AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

제2부 LUA API

제9장 IBM 관용 LU 어플리케이션의 기본 개념

이 장에서는 IBM 에서 일반 LU 응용 프로그램 인터페이스(LUA) 액세스 방법을 설명하며 시스템 네트워크 체계(SNA)에 대한 개요를 설명합니다.

주: 이 책의 제2부에 포함된 정보는 다음 시스템에서 제공하는 LUA API에 대한 정보입니다.

- Windows NT에서 수행하는 통신 서버
- 통신 서버/NT 제품과 함께 제공되는 OS/2, Windows NT, Windows 95 및 Windows 3.1용 SNA API 클라이언트
- Windows 95 및 Windows NT용 커널 통신

이들 시스템이 제공하는 지원에 대한 차이점 있는 경우, 명시됩니다.

LUA 및 SNA 이해

IBM LUA 액세스 방법은 2차 종속 LU에 대한 응용 프로그래밍 인터페이스(API)를 제공합니다. LUA는 시스템 소프트웨어에 대한 입출력 서비스 루틴을 제공하는 인터페이스로 구성되어 LU 유형 0, 1 및 3 SNA 프로토콜을 사용하여 통신을 지원합니다. 통신 서버는 LUA의 RUI 및 SLI 인터페이스를 지원합니다.

통신 서버는 Microsoft** NT SNA 서버에 호환성이 있는 2진이 되도록 디자인되었으며 OS/2 Communications Manager/2 버전 1.0 LUA의 구현과 유사합니다.

LUA에 대한 응용 프로그램에 대한 제공하는 서비스는 데이타 통신을 지원하는 서비스만을 포함하며 LUA는 장치 에뮬레이션 능력을 제공하지 않습니다. 그러나 LUA는 강력한 프리젠테이션 서비스 계층 능력의 서브세트를 제공합니다.

LUA 응용 프로그램이 워크스테이션에서 수행하기 전에 통신 서버에 설치되고 구성되어야 합니다. 통신 서버를 설치 및 구성하는 정보는 통신 *서버: 빠른 시작*을 참조하십시오.

연결 기능

통신 시스템의 주요 목적은 다른 시스템과 통신하는 데입니다. SNA의 목적은 일반 통신성을 제공하는 것입니다. LUA 통신 및 통신성 구현 사항은 System/370*(S/370*) 통신을 포함합니다.

LUA 어플리케이G 프로그램

이 책!서 *LUA n플리케이션 프로W램*은 LUA 통신 b능을 사k하는 n플리케이션 프로W램이나 n플리케이션 프로W램의 일부를 의미합니다. n플리케이션 프로W램은 이들 b능을 사k하) LU 유형 0, 1, 2 또는 3을 지x하는 다른 시스템!서의 n플리케이션 프로W램z 통신합니다.

로컬 LUA n플리케이션 프로W램이 수행함! 따라 x] 호스트 n플리케이션 프로W램z 데이타를 3환합니다. 로컬 및 x] n플리케이션 프로W램을 *상대방* n플리케이션 프로W램이라m 합니다.

LUA 명령어

명령n는 LUA! 프로세스하는 형식화된 d청입니다. n플리케이션 프로W램은 LUA! 일부 조치를 취하는 d청! 명령n를 발행합니다. LUA 명령n는 제n 블럭으로서 코드화됩니다. " 명령n 제n 블럭은 정확하T 정의된 형식을 .습니다. LUA b능을 사k하b 위해 n플리케이션 프로W램은 명령n 제n 블럭을 LUA API! 제x합니다.

LUA 명령n는 즉시 호출자로 리턴합니다. 리턴 코드! IN_PROGRESS이면 n플리케이션은 명령n d청! 지정된 통지 방법을 사k하) 명령n의 O료를 대b할 필d! 있습니다. LUA 명령n 통지의 설명은 제12장 RUI LUA #트리 포인트를 참조하십시@.

명령n 제n 블럭 레이F t은 **INCLUDE** 디렉토리!서 사k!능합니다. 명령n 제n 블럭 레이F t z 샘플 프로W램을 사k하) LUA n플리케이션 프로W램 작성을 도o 수 있습니다.

LU, 로컬 LU 및 상대방 LU

논리 장치(LU)는 n플리케이션 프로W램#의 데이타 3환을 |리합니다. 모든 LUA n플리케이션 프로W램은 LUA n플리케이션 프로W램z SNA 네트v크#의 중3자로서 *할하는 LU를 통해 SNA 네트v크! 대한 W세스G한을 r습니다.

LUA!는 LUA n플리케이션 프로W램 프로세스 및 LU#의 일 대 다 |h! 있습니다. 하나의 LUA n플리케이션 프로W램 프로세스는 동시! 다중 LU를 소유할 수 있지만 주n진 LU는 @직 하나의 LUA n플리케이션 프로W램 프로세스! 의해 소유될 수 있습니다. 두 번째 n플리케이션 프로W램 프로세스! LU를 사k하b 전! 첫번째 n플리케이션 프로W램이 LU를 해제해_ 합니다.

LUA n플리케이션 프로W램은 로컬 LU! LUA 명령n를 발행합니다. 이들 명령n는 명령z 데이타! 네트v크를 통해 상대방 LU! 흐르도록 합니다.

주: *빠른 시작*!서 설명한 대로 " b h ! 대해 한번만 로컬 LU를 정의할 필d ! 있습니다.

시스템 - 비스 제어점(SSCP)

호스트 시스템!서 시스템 서비스 제n점(SSCP) 8성d소는 호스트 n플리케이션 시작, 종속 LU로 호스트 n플리케이션 a합, LU#의 , a 작성 및 종료! 책임이 있습니다.

SNA 계층

SNA는 73의 잘 정의된 h층으로 8성된 h층적 8조입니다. 8조의 " h층은 m유한 b능을 수행합니다. SNA h층 8조의 이해는 LUA! 제x하는 다g한 b능의 이해를 돕습니다. 53의 상위 레벨 SNA h층! 대한 다음 설명은 LUA 및 SNA#의 | h를 보)줍니다.

데이타 링크 제어 계층

데이타 링크 제n(DLC) h층은 하드~n ! 인터페이스를 제x하는 d소로 8성됩니다. DLC d소는 동b 데이타 링크 제n(SDLC) 및 IBM 토큰링 네트v크M 0은 다g한 DLC 프로토콜! 대한 지x을 제x합니다. DLC h층은 f로 제n(PC) h층!서 d소! 대한 x k 링크 \ |을 제x합니다. DLC h층은 LUA를 포함하) 모든 퍼스널 통신 및 통신 서버 LU 8현! x통입니다.

경로 제어 계층

주변 노드의 f로 제n(PC) h층은 노드#의 복수 반 세션으로의 f로지정 z 0은 b본 b능을 제x합니다. SNA는 한번! @직 하나의 데이타 링크로의 f로! PC h층을 허k합니다. PC h층은 LUA를 포함하) 모든 퍼스널 통신 및 통신 서버 LU 8현! x통입니다.

전송 제어 계층

SNA의 전송 제n(TC) h층은 9지적으로 지x되는 " 반 세션! 대한 세션 제n b능z , a점 |리 프로W램 b능을 지x합니다. , a점 |리 프로W램 b능은 순서 번호 점K, 페이싱, 반 세션 데이타 흐름! |련된 다른 지x b능을 제n합니다. 세션 제n b능은 시작, 페이싱, 0호화, 0호 해독 및 세션 |련 데이타 흐름! |련된 다른 지x b능을 지x합니다. LUA!는 퍼스널 통신 및 통신 서버 내의 LU 유형 0, 1, 2! 대한 TC h층 8현이 들n 있습니다.

데이타 흐름 제어 계층

SNA의 데이타 흐름 제n(DFC) h층은 세션! 있는 FMD 쌍#z 세션#의 b능 |리 데이타(FMD) 및 FMD 응답의 흐름을 제n합니다. 데이타 흐름

제n h층은 d청/응답 형식화, 데이타 체인 프로토콜, d청/응답 상|, 송신 및 수신 모드 프로토콜, 브래킷 프로토콜, @류 복8 프로토콜, 정지 브래킷 시작 프로토콜 및 대b행렬화된 응답 프로토콜z 0은 다g한 b능을 지x합니다. LUA!는 퍼스널 통신 및 통신 서버 내의 LU 유형 0, 1, 2! 대한 데이타 흐름 h층 8현이 들n 있습니다.

프리젠테이션 서비스 계층

SNA의 프리젠테이션 서비스(PS) h층!는 사k자! 대한 통신 데이타 인터페이스를 나타내는 b능이 들n 있습니다. 프리젠테이션 서비스 h층은 LU 0를 제\한 모든 LU 유형! 대한 8조! 정의됩니다. LUA!는 퍼스널 통신 및 통신 서버 내의 프리젠테이션 서비스 h층의 m유한 서브세트! 들n 있습니다. 프리젠테이션 서비스 h층! 대한 자세한 내k은 *Systems Network Architecture Concepts and Products*을 참조하십시@.

LU 서비스 b능은 SNA 세션 메세지 흐름 h층의 일부입니다. 이들 b능은 세션을 확립하b 전! 세션 8조 만들b를 지x하며 또한 세션 8조를 해체합니다. xk 퍼스널 통신z 통신 서버로 LUA b능 인터페이스는 LU를 정의하m SNA 세션을 시작z 종료를 지x합니다.

SNA <G 사용법

LUA n플리케이션 프로W램이 상대방 호스트 n플리케이션 프로W램z 통신하b 전! " LU는 세션이라m 하는 상호 |h!서 ,a되n_ 합니다. SNA 세션은 두 3의 네트v크 지정 장치(NAU)! 서로 통신하T 하는 논리 ,a이며 LU는 NAU의 한 종류입니다. 세션은 두 3의 LU를 ,a하므로 *LU-LU 세션*이라m 합니다. LU-LU 세션은 일반 사k자! 데이타를 3환할 수 있T 합니다.

세션은 SNA 네트v크!서 LU 쌍#의 데이타 이동 방식을 |리합니다. W러므로 세션은 전송되는 데이타 수량, 데이타 보H, 네트v크 f로지정, 데이타 유실 및 트래픽 z잉 등! |련됩니다. 세션 특성은 2차 LU! **BIND** 명령을 수k할 때 1차 LU!서 bx하는 SNA **BIND** 명령! 의해 a정됩니다.

SNA 세션에 대한 선수사항

LU-LU 세션은 1차 논리 장치(PLU) 및 2차 논리 장치(SLU)#의 통신으로 8성됩니다. SLU는 LUA n플리케이션 프로W램! 의해 8현됩니다. 데이타! LU-LU 세션!서 PLU 및 SLU#! 전송되b 전! 다음 이벤트! 발생해_ 합니다.

1. 퍼스널 통신 및 통신 서버는 데이타 링크를 활성화시킵니다.
2. 데이타 링크! 준비되면 시스템 서비스 제n점(SSCP)은 물리 장치(PU) 활성화(**ACTPU**) 명령을 송신하m 퍼스널 통신이나 통신 서버 프로W램 중 하나!서 `정 응답을 읽n 자신z 물리 장치(SSCP-PU 세션)#의 세

션을 만듭니다. W런 다음 **ACTPU** 명령으로부터의 PU 주소! 8성 정보! 대응하는 f l n느 한쪽의 프로W램이 `정 응답을 송신합니다.

3. SSCP는 논리 장치(LU) 활성화(**ACTLU**) 명령을 송신하m 퍼스널 통신이나 통신 서버 프로W램 중 하나!서 `정 응답을 읽n 자신z 논리 장치(SSCP-PU 세션)#의 세션을 만듭니다. W런 다음 **ACTLU** 명령으로부터의 LU 주소! 8성 정보! 대응하는 f l n느 한쪽의 프로W램이 `정 응답을 송신합니다.

세션 시작하기

SLU 또는 PLU 중 하나! LU-LU 세션을 시작할 수 있습니다.

SLU에서 LU-LU 세션 시작하기

SSCP-LU 세션이 성립된 후! SLU 프로W램은 자동 초b화(**INITSELF**) 명령을 SSCP! 송신하) LU-LU 세션을 d청할 수 있습니다. SSCP는 **INITSELF** 명령을 수신하며 명명된 호스트 n플리케이션 프로W램이 유효한지 점K합니다. 호스트 n플리케이션 프로W램이 K려지m 활성화된 f l 유효합니다. 호스트 n플리케이션 프로W램이 유효한 f l, SSCP는 `정 응답을 SLU! 보내m PLU는 세션을 시작합니다. 호스트 n플리케이션 프로W램이 유효하지 J은 f l, SSCP는 부정 응답을 SLU! 보내m PLU는 세션을 시작하지 J습니다.

SSCP! `정 응답을 **INITSELF** 명령! 보내지만 세션이 성립될 수 x는 f l, SSCP는 네트v크 서비스 프로시듀n @류(**NSPE**) 명령을 SLU로 보내 세션 성립 시도를 중단합니다. SLU는 **NSPE** 명령 다음! **INITSELF** 명령을 재발행할 수 있습니다.

PLU에서 LU-LU 세션 시작하기

PLU 프로W램은 d청되지 J은 LU-LU 세션을 시작할 수 있습니다. PLU는 **BIND** 명령을 생성하) 세션을 시작합니다. 다음의 `정 응답은 통신! 동의합니다. **BIND** 명령z , |된 데이타 필드!는 PLU n플리케이션 프로W램의 이름 및 세션 **BIND** 매3변수! 들n 있습니다. 이 데이타 필드의 형식! 대한 자세한 내k은 *Systems Network Architecture: Formats*을 참조하십시@.

조정할 수 x는 **BIND**의 f l, SLU는 매3변수! 수k할 수 있을 때 `정 응답을 리턴합니다. 매3변수! 수k할 수 x는 f l, SLU는 (지 데이타 ! 있는 부정 응답을 PLU로 리턴합니다.

조정할 수 있는 **BIND** 명령은 PLU 매3변수M 호환성을 나타내는 ;신된 세션 매3변수의 최소 26바이트로 `정 응답을 SLU! 리턴하도록 허k합니다. PLU! 수k할 수 있는 리턴 매3변수를 찾으면 시작 데이타 트래픽(**SDT**) 명령을 송신합니다. 리턴된 매3변수! 수k할 수 x으면 PLU는 SLU로부터 수k할 수 x는 조정!능 **BIND** 명령 매3변수를 나타내는 **UNBIND** 명령을 송신합니다.

LU-LU 세션에서 데이타 전송하기

LU-LU 세션이 성립되m SLU 프로W램이 **SDT** 명령! 응답한 후! 데이타 전송을 시작할 수 있습니다. 데이타 전송 작w의 f l, 메세지! 전송될 때 n지 일반 사k자 b o 5 * !서 퍼스널 통신이나 통신 서버 b o 5 * 으로 이동합니다. 데이타 수신 작w의 f l, n느 한쪽 프로W램이 메세지를 자신의 b o 5 * 으로 위치시킨 다음 메세지를 일반 사k자 b o 5 * 으로 이동시킵니다.

Quiesce 프로토콜은 LU-LU 세션!서 데이타의 전송을 중단합니다. PLU 또는 SLU는 다음 Quiesce 프로토콜 명령을 전송할 수 있습니다.

- **QEC**(체인 끝!서 Quiesce). 이 명령은 수신자! 데이타 체인!서 최종 부분을 전송한 후! 데이타 전송을 중단하도록 d청합니다. 데이타 체인은 일련의 |련 메세지입니다. 데이타 체인! 대한 자세한 내k은 183페이지의 『데이타 체인 프로토콜 사k법』을 참조하십시@.
- **QC**(Quiesce 0료). 이 명령은 데이타 전송이 중단되z음을 **QEC** 명령! 통지합니다. SLU! **QC** 명령을 송신하면 퍼스널 통신이나 통신 서버 중 하나! **RELQ**(Quiesce 해제) 명령을 수신할 때n지 SLU! 정상 흐름 메세지를 송신하지 못하T 합니다.
- **RELQ**(Quiesce 해제). 이 명령은 데이타! 다시 전송될 수 있음을 수신자 !T 통지합니다.

세션 종료하기

모든 데이타! 전송되m K증되면 세션이 종료할 수 있습니다. 동일 또는 다른 PLU 중 하나로 다른 세션을 시작하b 전! SLU는 하나의 세션을 종료해_ 합니다.

SLU로 LU-LU 세션 종료하기

SLU는 다음 두 !지 방법 중 하나로 LU-LU 세션을 종료할 수 있습니다.

- 자체 종료(**TERMSELF**) 명령 또는 **UNBIND** 명령 송신하b. 둘 중 하나의 명령은 즉"적인 종료 a z를 !져I니다.
- **RSHUTD**(시스템 종료 d청) 명령 송신하b. 이 명령은 PLU!서 **UNBIND**를 d청합니다.

즉시 세션을 종료하b 위해 SLU는 **TERMSELF** 명령을 SSCP로 송신하) 명명된 LUA n플리케이션 프로W램이 이 세션!서 참)하m 있는지)부를 점K합니다. 참)하m 있는 f l, SSCP는 `정적인 '자료 x음' 응답을 송신합니다. 사k중인 호스트 SNA 버전! 따라 SSCP는 LU-LU 세션!서 모든 메세지를 제E하는 **CLEAR** 명령을 송신한 다음 **UNBIND** 명령을 송신하) 세션을 종료할 수 있습니다. 대신 SLU는 **UNBIND** 명령을 PLU로 송신할 수 있습니다.

PLU로 LU-LU 세션 종료하기

PLU는 다음 두 !지 방법 중 하나로 LU-LU 세션을 종료할 수 있습니다.

- **CLEAR** 명령을 송신한 후 **UNBIND** 명령을 송신하E나 **UNBIND** 명령만을 송신하b. 둘 중 하나의 방법은 즉"적인 종료 az를 !져I니다.
- **SHUTD**(시스템 종료) 명령 송신하b. 이 명령은 순서대로 세션 종료 az를 !져I니다. SLU 및 PLU는 데이타 송신을 중단하도록 서로 K리m 송신된 데이타! 수신되z음을 확인하는 대화를 .습니다.

LU-LU 세션 종료는 SSCP-LU 세션! 5향을 미치지 J습니다.

SSCP-LU 세션 및 SSCP-PU 세션 종료하기

호스트! **DACTLU**(논리 장치(LU) 비활성화) 명령을 SLU! 송신할 때 SSCP-LU 세션이 종료합니다. 퍼스널 통신 및 통신 서버k 최종 SSCP-LU 세션이 종료하면 SSCP는 **DACTPU**(물리 장치(PU) 비활성화) 명령을 송신하) SSCP-PU 세션을 종료할 수 있습니다.

호스트 링크 연결단절

호스트! **DACTPU** 명령! 대한 응답을 수신할 때 SDLC 프로토콜을 사k할 때 **SDRM**(,a단절 응답 모드 설정) 명령z 0이 퍼스널 통신 및 통신 서버! 명령을 리턴합니다. 또한 SSCP는 퍼스널 통신 및 통신 서버! 동일한 명령을 송신하) p제라도 즉시 ,a단절하m 모든 세션을 종료할 수 있습니다. 세션이 이런 방식으로 종료되면 이전! 활성화된 모든 SLU는 ,a단절 표시를 수신합니다.

메<지 번호

LU-LU 세션 동H SLU 및 PLU#! 전송되는 모든 정상적인 흐름 메세지는 순서대로 번호를 매i니다. SLU는 SLU!서 PLU로의 정상적인 흐름 메세지! 대한 순서 번호M PLU!서 SLU로의 정상적인 흐름 메세지! 대한 다른 순서 번호를 유지합니다. " 정상 흐름 메세지는 선행하는 정상 흐름 메세지의 순서 번호보다 더 큰 순서 번호를 r습니다. SLUM PLU#! 성립된 " 세션! 대한 순서 번호 쌍이 있습니다.

LU-LU d^ 흐름 메세지M 모든 SSCP-LU 및 SSCP-PU 메세지의 fl, 순서화되지 J은 식별자(ID)! 순서 번호 대신 사k됩니다.

세션이 재성립되E나 **CLEAR** 명령이 송신되면 PLU 및 SLU는 W들의 순서 번호를 0으로 설정합니다. PLU는 순서 번호를 **STSN**(순서 번호 설정 및 테스트) 명령으로 변f할 수 있습니다. 이M은 세션이 복8되E나 재시작될 때 순서 번호를 정정할 수 있T 합니다.

SLU ! 순서 번호 @ 류를 만나면 응답이 d 청된 f l 부정 응답을 PLU로 보냅니다. SLU ! 응답을 읽을 때 SLU는 응답 순서 번호를 사 k 하) 응답을 x 래의 d 청 z | 련시킵니다. SLU ! 응답을 작성할 때 SLU는 x 래 d 청의 순서 번호를 제 x 해 _ 합니다.

< G 재시작 및 재동기화하기

PLU 또는 SLU ! 회선 장 V M 0 은 복 8 할 수 x 는 @ 류를 만나면 LU-LU 세션을 재시작한 후 LU-LU 세션을 재동 b 화할 필 d ! 있습니다. LU-LU 세션 재동 b 화는 회복할 수 있는 메세지 재처리 및(선택적으로) 메세지 순서 번호 재설정을 포함합니다. n 플리케이션 프로 W 램은 유실된 메세지를 재전송하 b 위해 루틴을 포함할 수 있습니다.

세션이 재시작되 m 재동 b 화되면 PLU는 **BIND**, **STSN** 및 **SDT** 명령을 송신합니다. **STSN** 명령이 송신되면 PLU 및 SLU g 쪽 ! 서 수 k 할 수 있는 순서 번호를 설정하 b 위해 대화 ! 발생할 수 있습니다. 이 대화는 **STSN** 메세지 및 ` 정 응답으로 8 성됩니다.

SLU ! 재동 b 화의 필 d 성을 a 정하면 SLU는 **RQR**(복 8 d 청) 명령, 부정 응답 또는 **LUSTAT**(LU 상태) 명령 z 사 k 자 (지 바이트로 실패 ! 대한 설명을 송신할 수 있습니다. PLU ! 장 V 를 발 _ 하 E 나 SLU ! 서 **RQR** 명령을 수신하면 PLU는 **CLEAR** 명령을 송신하) 네트 v 크 ! 서 모든 LU-LU 메세지를 제 E 하며 **STSN** 명령을 송신하) 새 순서 번호를 설정한 후 **SDT** 명령을 송신합니다.

요청 및 응답을 제어하기 위한 프로토콜 사용법

다 g 한 프로토콜은 d 청 및 응답 ! 대한 순서 T 칙을 제 n 할 수 있습니다. 이 절 ! 서는 SNA 네트 v 크 | 리, 데이타 전송 및 네트 v 크 8 성 d 소 상태 동 b 화 ! 사 k 되는 일부 프로토콜을 설명합니다.

페이싱 프로토콜 사용법

메세지 흐름 비율이 퍼스널 통신 및 통신 서버나 호스트 ! 대해 너무 빠르지 J T 하 b 위해 **BIND** 명령 ! 페이싱을 지정할 수 있습니다. 페이싱은 LU-LU 정상 흐름 ! 만 적 k 합니다. 페이싱하는 동 H 퍼스널 통신 및 통신 서버는 지정된 수의 메세지 ! 흐르도록 하며 추 ! 메세지 ! 송신되도록 허 k 되 b 전 ! 응답을 b 다립니다. 퍼스널 통신 및 통신 서버 대 호스트 흐름, 호스트 대 퍼스널 통신 및 통신 서버 흐름 또는 g 쪽 다 ! 서 페이싱을 지정할 수 있습니다. LU-LU 세션이 시작되면 LUA는 n 플리케이션 프로 W 램의 참) x 이 모든 페이싱을 처리합니다.

수신 페이싱 프로토콜

수신 페이싱 프로토콜은 LU-LU 세션의 SLU ! 서 송신한 메세지의 번호M 주 b ! 서 PLU 제 n 를 제 x 합니다. SLU ! **BIND** 명령 ! 서 페이싱 * 을 수신하면 퍼스널 통신 및 통신 서버는 자동으로 호스트M 통신하는 " SLU ! 대한 페이싱을 - 화합니다.

조정할 수 있는 **BIND** 명령 ! 대한 ` 정 응답 동H ! 페이싱 * 을 0 이 \ 의 숫자로 변 f 할 수 있습니다. SLU ! 순서의 첫번째 메세지를 송신하면 퍼스널 통신 및 통신 서버는 페이싱 응답이 리턴됨을 나타내는 d 청/응답 헤더(RH) ! 서 비트를 설정합니다. 둘 중 하나의 프로W 램이 PLU로부터 페이싱 응답을 수신하b 전 ! 페이싱 카 n 트를 소비하는 f l, n 느 쪽 프로W 램도 추 ! 적인 데이타 메세지를 송신할 수 x 습니다. n 플리케이션 프로W 램이 2 b 작 w 을 수행하m 페이싱 응답이 수신되지 J 는 f l, 퍼스널 통신 및 통신 서버는 2 b 작 w 을 지 , 합니다.

송신 페이싱 프로토콜

SLU는 자동으로 송신 페이싱 프로토콜을 제 n 합니다. 페이싱 표시 b ! PLU ! 서 SLU로의 메세지 ! 설정되 n 있는 f l, SLU는 n 플리케이션 프로W 램이 메세지를 읽을 때 페이싱 응답을 발행합니다. 메세지 응답은 페이싱 표시 b 를 포함할 수 있E 나 응답이 수신 메세지 ! 필 d 하지 J 은 f l 페이싱 응답은 분리 페이싱 응답(IPR)일 수 있습니다. W 러면 PLU는 다른 메세지 페이싱 창을 보낼 수 있습니다.

반 양방향(전송) 회선경합/플립플롭 프로토콜 사용법

방향 변 f (CD) 표시 b 는 다음 프로토콜 둘다 ! 서 사 k 됩니다.

- 세션의 시작이나 체인의 최종 d 청을 송수신한 후 ! 둘 중 하나의 반 세션이 정상 흐름 d 청을 송신하는 정상 흐름 송수신 모드인 반 g 방향(전송) 회선 f 합 프로토콜.
- 하나의 반 세션이 체인 끝의 응답 헤더(RH) ! CD 표시 b 를 설정하) 다른 반 세션이 송신을 시작하T 하는 정상 흐름 송수신 모드인 반 g 방향(전송) 플립플롭 프로토콜.

CD 표시 b 는 전송을 시작할 수 있음을 수신자 ! T K 립니다.

9 를 들n SLU ! 트랜잭션을 시작하는 f l, SLU는 트랜잭션을 O 전하T b 술하는 메세지를 송신하) 시작합니다. 최종 메세지 ! 서, SLU는 CD 표시 b 를 설정하) 전송 응답을 시작할 수 있음을 PLU ! K 립니다. PLU ! 트랜잭션을 O 료하는데 추 ! 적인 정보를 필 d 로 하면 조회를 전송하) CD 표시 b 를 설정합니다. 트랜잭션이 O 료될 때 n 지 대화는 이 반 g 방향(전송) 모드 ! 서 진행합니다. 반 g 방향(전송) 대화 동H SLU는 **SIG** 명령을 사 k 하) 데이타 송신을 중단하m 데이타 흐름 방향을 변 f 하도록 PLU ! K 립니다.

브래킷 프로토콜 사용법

브래킷 프로토콜은 세션이 단일 트랜잭션! |h함을 나타내며 데이타 전송의 SLU 및 PLU 문맥 제n를 제x합니다. 브래킷 프로토콜은 동시 트랜잭션! 의한 !로채b로부터 현재 세션을 보호합니다. 브래킷은 트랜잭션의 지속 b#을 포함합니다.

브래킷의 첫번째 메세지!는 시작 브래킷(BB) 표시b! 들n 있으며 브래킷의 마지막 메세지!는 종료 브래킷(EB) 표시b! 들n 있습니다. 단일 메세지! g쪽 표시b! 들n 있는 fl 단일 메세지! 브래킷이 될 수 있습니다.

브래킷 세션의 fl, **BIND** 명령은 첫번째 스피커로서 하나의 LU를 지정하며 다른 LU를 명령자로서 지정합니다. 첫번째 스피커는 다른 LU의 허k x이 브래킷을 시작할 수 있습니다. W러나 명령자는 첫번째 스피커로부터 허k을 d청하m 수신하) 브래킷을 시작해_ 합니다.

BID 명령은 명령자! 발행한 정상 흐름 d청으로 브래킷을 시작하도록 사kG한을 d청합니다. **BID** 명령! 대한 `정 응답은 첫번째 스피커! 브래킷을 시작하지 J으나 명령자! 브래킷 시작을 b다립니다. **BID** 명령! 대한 부정 응답은 명령자! 브래킷을 시작하T 하는 사kG한을 첫번째 스피커! E부함을 나타냅니다. 브래킷을 시작하도록 사kG한이 부)될 때 첫번째 스피커는 **RTR**(수신 준비) 명령을 송신할 수 있습니다.

첫번째 스피커는 다음 두 3의 코드 중 하나를 !진 **BID** 명령! 대한 부정 응답을 나타냅니다.

Bracket-Bid-Reject-RTR-Forthcoming
: 해당 **BID** 명령! 대한 **RTR** 명령이 나중! 송신됨을 나타냅니다(브래킷을 시작할 사kG한 부)). 명령자는 **RTR** 명령을 대b하E나 **BID** 명령을 다시 전송할 수 있습니다.

Bracket-Bid-Reject-No-RTR-Forthcoming
: 해당 **BID** 명령! 대한 **RTR** 명령이 나중! 송신되지 J음을 나타냅니다. 명령자! 브래킷을 시작하도록)전히 x하는 fl 명령자는 **BID** 명령을 다시 송신해_ 합니다.

BB 표시b! 있는 체인의 첫번째 FMD! 뒤따르는 **BID** 명령을 송신하는 대신 명령자는 BB 표시b! 있는 체인의 첫번째 FMD를 송신하) 브랫킷을 시작할 수 있습니다. 첫번째 스피커는 해당 시도! `정 응답을 부)하E나 부정 응답 코드 중 하나를 나타내는 부정 응답으로 해당 시도를 E부할 수 있습니다. W러나 명령자! **CANCEL** 명령을 송신하) BB 표시b! 있는 체인을 종료하는 fl, 브래킷은 응답! |hx이 시작되지 J습니다. **RTR** 명령은 명령자! 브래킷을 시작하E나 명령자! 브래킷을 시작하b를 x하는지)부를 KF내b 위한 사kG한을 명령자!T 부)하도록 첫번째 스피커! 발행할 수 있습니다.

명령자! 다음 브래킷을 시작할 M임을 나타내는 **RTR** 명령! 대한 `정 응답. 명령자! 브래킷을 시작하려m 하지 J는 f l, 명령자는 RTR-Not-Required (지 코드로 부정 응답을 발행합니다.

데이타 체인 프로토콜 사용법

데이타 체인은 |련된 메세지의 W룹을 전송하는 선택적 프로토콜입니다. SLU로부터 체인 메세지를 전송하b 위해 SLU는 체인!서 첫번째 메세지를 나타내b 위해 메세지! 대한 시작 체인(BC) 표시b를 1로 설정합니다. 체인!서 첫번째M 마지막 사이의 모든 메세지! 대해 SLU는 BC 및 종료 체인(EC) 표시b를 0으로 설정합니다. 체인의 마지막 메세지의 f l, EC는 다시 1로 설정됩니다. SLU! 메세지를 수신하면 메세지! 체인되z는지)부를 판별하b 위해 체인 표시b를 테스트합니다.

데이타 체인 프로토콜은 다음z 0이 3!지 유형의 체인으로 이루n집니다.

- 응답이 x는 체인. 체인!서의 " d청은 *응답 x음*으로 표시됩니다.
- 9 \ 응답 체인. 체인!서의 " d청은 *9 \ 응답*으로 표시됩니다.
- 절대 응답 체인. 체인!서의 마지막 d청은 *절대 응답*으로 표시되며 체인!서의 다른 모든 d청은 *9 \ 응답*으로 표시됩니다.

PLU! 메세지 체인을 송신할 때 SLU나 PLU! 메세지 @류를 발_하는 f l SLU는 **CANCEL** 명령을 송신할 수 있습니다. SLU! **CANCEL** 명령을 PLU! 송신하는 f l, PLU는 수신한 모든 메세지를 체인!서 삭제합니다. PLU! 부정 응답을 체인의 임의 d소로 송신하는 f l, SLU는 정상적으로 체인을 종료하E나 **CANCEL** 명령을 송신합니다.

데이타 교환 제어 방법

SNA 세션은 순차적인 데이타 3환을 위한 T칙으로 처리됩니다.

흐름 프로토콜

전송 레벨!서 데이타는 반 g방향(전송)(HDX) 프로토콜이나 O전 g방향(전송) 프로토콜 중 하나를 통해 3환됩니다.

반 g방향(전송) 프로토콜이 사k되면, 한번! @직 한 방향으로 데이타! 흐르며 하나의 LU는 송신 전k이며 다른 LU는 수신 전k입니다. 반 g방향(전송) 플립플룹 프로토콜!서 g쪽 LU는 n느쪽 LU! 송신 G한을 !지며 n느쪽 LU! 수신 G한을 .는지 인식합니다. 지정 시#! 상대방 LU는 방향을 변f하는 데 동의하) 수신자는 송신할 수 있으며 송신자는 수신할 수 있습니다.

O전 g방향(전송) 프로토콜이 사k되면 데이타는 p제라도 n느 한쪽 방향으로 흐를 수 있습니다. g쪽 LU는 제한 x이 송수신할 수 있습니다.

응답 모드

" SNA 메세지는 d청 또는 응답 중 하나입니다. 하나의 LU로부터의 모든 d청은 상대방 LU로부터의 일치 응답을 유도합니다. 응답은 d청z 동일한 전송 순서 번호를 제x하므로 응답 및 d청은 순서 번호! 의해 일치될 수 있습니다.

RH! 주d 응답을 지정하는 d청을 n플리케이션이 수신하면 n플리케이션은 응답 메세지를 생성하) 송신해_ 합니다. 응답 모드 T칙은 응답이 송신되n _ 하는 시b를 a정합니다.

즉시 응답 모드!서 n플리케이션은 자신의 d청을 송신하b 전! d청! 대한 응답을 송신해_ 합니다. W러나 지,된 응답 모드!서 d청이 수신된 후 p제라도 응답이 송신될 수 있습니다.

LUA 상관 테이블

LUA는 응답을 수신할 때n지(n플리케이션이 입력 d청! 대한 응답을 발행하E나 PLU! 전송 d청! 응답할 때n지) 입력 및 전송 d청의 순서 번호의 트랙을 유지합니다. 이들 번호는 *상/ 테이블*이라m 하는 퍼스널 통신 및 통신 서버 5*! b록됩니다.

즉시 응답 모드!서 @직 몇 3의 미a d청만이 b껏해_ 보통 하나의 세션!서 생성될 수 있습니다. 지,된 응답 노드!서 번호는 더 커질 수 있습니다.

LUA 상| 테이블은 동적으로 |리됩니다. LUA는 몇 3의 응답을 b록할 수 있습니다. 매l 많은 수의 응답이 누적되면(프로W램 논리 @류 때문!) 서버는 메모리!서 느리T 수행하며 퍼스널 통신 및 통신 서버는 시스템 종료될 수 있습니다.

예외 응답 요청(RQE)

대부분의 f l LUA는 프로W램의 도r x이 d청 및 응답을 자동으로 서로 |련시킬 수 있습니다. LUA는 세션!서 흐르는 d청 및 응답 RU를 |찰합니다. LUA는 d청이 응답을 필d로 하는 시b M 응답이 송신된 시b를 K릴 수 있습니다. W러나 응답이 송신될지)부를 LUA! K릴 수 x는 한!지 f l! 있으며 사k자 프로W램이 W M을 K려_ 합니다.

d청의 RH! 있는 비트 필드는 응답이 필수적인지, x하지 J는 M인지 F니멂 OOQO040 1 Tf 1.0001 0Tf 0.9867 0 (Q40 1 Tf 1.0001 0 T127 1 Tf 1T139 1 Tf 1.00

상 자동으로 상 | 테이블을 | 리할 수 x습니다. 표 11은 LUA ! 상 | 테이블 ! 서 수신된 RQE를 자동으로 소 E 할 수 있는 인스턴스 M 수신된 RQE를 소 E 하 b 전 ! LUA ! n 플리케이션 ! 서 신호를 b 다려 _ 하는 인스턴스를 d ` 합니다.

표 11. RQE의 소 E

즉시 응답 모드	지연 응답 모드			
명령 n	HDX	FDX	HDX	FDX
RUI_READ	자동	자동	n 플리케이션 응답	n 플리케이션 응답
RUI_WRITE	자동	n 플리케이션 응답	n 플리케이션 응답	n 플리케이션 응답

HDX 또는 FDX 세션의 즉시 응답 모드 ! 서 LUA는 n 플리케이션이 입력을 d 청하자마자(RUI_READ 사 k) RQE의 번호를 삭제할 수 있는데 이는 즉시 응답 모드 ! 서는 응답이 다른 d 청이 일 n 나 b 전 ! 송신되 n _ 하 b 때문입니다. 또한, HDX , a 의 즉시 응답 모드 ! 서 LUA는 n 플리케이션이 출력을 d 청하자마자(RUI_WRITE 사 k) RQE의 번호를 삭제할 수 있는데 이는 출력이 RQE 응답이 E 나 n 떠한 응답도 송신되지 J b 때문입니다.

다른 모든 인스턴스 ! 서 LUA는 RQE ! 대한 응답이 생성될지) 부를 확인할 수 x습니다. n 플리케이션은 PLU(부정 응답만을 x 함)의 이점을 위해서 ! F 니라 RQE ! 수 k 되 n 부정 응답을 생성하지 J 을 M 임을 LUA ! K 리 b 위해 ` 정 응답을 형식화하) RQE ! 송신해 _ 합니다.

W 런 다음 LUA는 표 ! 서 RQE를 소 E 할 수 있습니다. 응답이 ` 정 응답이 m PLU 는 부정 응답만을 x 하므로 LUA는 n 플리케이션의 응답을 네트 v 크
크 p 픳현 있습

표 12. TS 프로파일 특성

프로파일	페이싱 사용	CLEAR	CRV	RQR	SDT	STSN
2	항상	사k	사k되지 J음	사k되지 J음	사k되지 J음	사k되지 J음
3	항상	사k	선택적	사k되지 J음	사k	사k되지 J음
4	항상	사k	선택적	사k	사k	사k
7	선택적	사k되지 J음	선택적	사k되지 J음	사k되지 J음	사k되지 J음

FM 프로파일

0, 2, 3, 4, 6, 7, 18 및 19의 83 FM 프로파일은 SNA! 의해 정의됩니다. W러나 프로파일 0 및 6은 SSCP! 의해서만 사k되m 프로파일 19는 LU 유형 6.2로만 사k되므로 53의 프로파일은 LUA n플리케이션! 적k될 수 있습니다. " 프로파일은 제한될 수 있는 SNA b능!서 다릅니다.

FM 프로파일의 대략적인 d`은 186페이지의 표13! 표시됩니다. 표!서 x백 셀은 SNA b능이 이 프로파일!서 제한되지 JR음을 의미하며 BIND 매3변수! 지정되도록 사k할 수 있습니다.

LUA RUI는 FM 프로파일 2, 3, 4, 7 및 18을 지x합니다.

표 13. FM 프로파일 특성

SNA 기능	FMP 2	FMP 3	FMP 4	FMP 7	FMP 18
d청 모드	SLU! 지,을 사k합니다				
응답 모드	SLU는 즉시를 사k합니다	즉시	즉시	즉시	즉시
RU 체인	단일 RU 체인만				
f이 점K P축				LU 0만	
FMH-1 세션 제n 블럭(SCB) P축	허k되지 J음				

표 13. FM 프로파일 특성 (h 속)

SNA 기능	FMP 2	FMP 3	FMP 4	FMP 7	FMP 18
데이타 흐름 제 n RU ! 허 k 됩니다	x 음	• CANCEL • SIGNAL • L U S T A T (SLU만) • CHASE • SHUTD • SHUTC • RSHUTD • BID, RTR	• CANCEL • SIGNAL • LUSTAT • QEC • QC • RELQ • CHASE • SHUTD • SHUTC • RSHUTD • BID, RTR	• CANCEL • SIGNAL • LUSTAT • RSHUTD	• CANCEL • SIGNAL • LUSTAT • CHASE • BIS, SBI • BID, RTR
FM 헤더	허 k 되지 J 음				
브래킷	제한 사 k				
흐름 프로토콜	FDX				
복 8	PLU만 ! 의해				

RUI LUA 명령어 사용법

n 플리케이션은 LUA 명령 n 를 통해 LUA ! W 세스합니다. " 명령 n 는 LUA ! 대한 매 3 변수를 제 x 하며 x 하는 b 능을 수행하 m 매 3 변수를 n 플리케이션 ! 리턴합니다.

명령어 요약

다음은 n 플리케이션이 사 k 할 수 있는 7 3 의 LUA 명령 n 의 # 단한 d ` 입니다. (" 명령 n ! 대한 자세한 설명은 제13장 RUI 명령 n 를 참조하십시 @ .)

RUI_BID

호스트로부터 정보를 읽을 수 있는 시 b 를 n 플리케이션이 판별할 수 있 T 합니다.

RUI_INIT

LUA n 플리케이션 ! 대한 LU-SSCP 세션을 설정합니다.

RUI_PURGE

미 a **RUI_READ** 명령 n 를 취소합니다.

RUI_READ

LU-SSCP 세션이나 LU-LU 세션 중 하나 ! 서 호스트 ! 서 LUA n 플리케이션의 LU로 전송하는 데이타나 상태 정보를 수신합니다.

RUI_TERM

LUA n플리케이션! 대한 LU-SSCP 세션을 종료합니다. 또한 LU-LU 세션이 활동중인 f l 종료합니다.

RUI_WRITE

LU-SSCP 세션이나 LU-LU 세션 중 하나!서 호스트! 데이타를 전송합니다.

W리m 퍼스널 통신 및 통신 서버는 RUI_INIT_STATUS 표시를 LUA n플리케이션! 리턴할 수 있습니다. 미처리 **RUI_INIT** 명령n 처리시 도달하는 상태! 대한 정보를 제x합니다.

RUI < G

RUI 세션은 n플리케이션이 a정하는 시# 동H LU의 소유G으로 8성되 n SSCP 및 LU(SSCP-LU 세션이라m 함)#의 세션 만들b를 포함할 수 있습니다. 또한 RUI 세션은 하나 이상의 중복되지 J은 LU-LU 세션 설정을 포함할 수 있습니다. 접속 단절이나 다른 재설정 조G 때문! SSCP-LU 세션이 실패하면 RUI 세션이 종료합니다. RUI 세션은 **RUI_INIT** 명령n로 시작하m **RUI_TERM** 명령n로 종료합니다.

RUI 명령어 발행

189페이지의 표14는 RUI n플리케이션 프로W램이 주n진 LU! 대한 RUI API! 명령n를 발행할 수 있는 유효한 조G을 나타냅니다. !장 ^쪽 -! 있는 항목은 입력 명령n를 나타냅니다. 첫번째 행! 있는 항목은 진행중인 명령n를 나타냅니다. 표! 있는 항목이 『9』이면 명령n 조합은 유효한 조G을 나타냅니다. 표! 있는 항목이 『@류』이면 명령n 조합은 틀린 조G을 나타내며 @류 코드! LUA n플리케이션 프로W램으로 리턴됩니다.

표 14. RUI 명령n 조G

	진행중인 명령						
입력 명령	**현재 세션 없음**	**RUI_INIT**	**RUI_TERM**	**RUI_WRITE**	**RUI_READ**	**RUI_PURGE**	**RUI_BID**
RUI_INIT	9	@류	@류	@류	@류	@류	@류
RUI_TERM	@류	9	@류	9	9	9	9
RUI_WRITE	@류	@류	@류	9 (주의사항 1 참조)	9	9	9
RUI_READ	@류	@류	@류	9	9 (주의사항 2 참조)	9	9
RUI_PURGE	@류	@류	@류	9	9	@류	9
RUI_BID	@류	@류	@류	9	9	9	@류

주의사항:

1. RUI는 동시! 세션당 최대 두 3의 활동중인 **RUI_WRITE** 명령n를 허k합니다. 활동중인 **RUI_WRITE** 명령n는 다른 세션 흐름k이n_ 합니다. 43의 세션 흐름이 !능합니다.
   - SSCP-LU d^
   - SSCP-LU 정상
   - LU-LU d^
   - LU-LU 정상
2. RUI는 동시! 세션 당 최대 네 3의 활동중인 **RUI_READ** 명령n를 허k합니다. 활동중인 **RUI_READ** 명령n는 다른 세션 흐름k이n_ 합니다.

비동기 명령어 완료

일부 LUA 명령n는 일부 로컬 프로세스 이후!(9를 들n **RUI_PURGE** 명령n) 신속하T O료되지만 대부분의 명령n는 호스트 n플리케이션으로부터 송수신하는 메세지를 필d로 하므로 O료하는 데 시#이 I립니다. 이 때문! LUA는 비동b 인터페이스로서 8현되며 명령n! 진핸중인 동H n 플리케이션으로 제n! 리턴될 수 있으므로 n플리케이션이 프로세스(다른 LUA 명령n 발행 포함)를 h속하도록 해제됩니다. LUA! n플리케이션! 제n를 리턴하는 방식은 명령n!서의 이벤트 처리 방식! 의합니다.

퍼스널 통신 및 통신 서버의 명령n 응답 신호! 지,되면(9를 들n x] 노드!서 정보를 대b하는 M이 필d하므로) 절취 부분이 비동b적으로 명령n를 리턴해_ 합니다. API는 1차 리턴 코드를 LUA_IN_PROGRESS로, **lua_flag2**를 LUA_ASYNC로 설정하) 수행합니다. n플리케이션은 이제 다른 프로세스를 수행하E나 명령n! O료됐음을 K리는 통지를 API!서 b

다릴 수 있습니다. 명령 n ! 0 료되면 VCB ! 서 1차 리턴 코드의 설정을 최종 * 으로 설정하 m **lua_flag2**를 LUA_ASYNC로 설정하) n 플리케이션 ! 통지됩니다.

샘플 LUA 통신 순서

다음은 LUA 통신 순서의 9 제입니다.) b ! 서는 세션을 시작하 m 데이타를 3 환하며 세션을 종료하는데 사 k 되는 LUA 명령 n M 송수신되는 SNA 메세지를 보) 줍니다. 화살표는 SNA 메세지 ! 흐르는 방향을 나타냅니다.

다음 z 0 은 ` n ! 사 k 됩니다.

SSCP norm
LU-SSCP 세션, 정상 흐름

LU norm
LU-LU 세션, 정상 흐름

LU exp
LU-LU 세션, d ^ 흐름

+rsp 표시된 메세지 ! 대한 ` 정 응답

BC 시작 체인

MC 체인 중 #

EC 종료 체인

CD 변 f 방향 표시 b 세트

RQD 절대 응답 d 8

LUA 어플리케이션으로 발행하는 명령어	SNA 메세지	흐름 방향 어플리케이션	호스트
RUI_INIT	(ACTLU)	<-----	
	(ACTLU +rsp)	----->	
RUI_WRITE(SSCP norm)	INITSELF	----->	
RUI_READ(SSCP norm)	INITSELF +rsp	<-----	
RUI_READ(LU exp)	BIND	<-----	
RUI_WRITE(LU exp)	BIND +rsp	----->	
RUI_READ(LU exp)	SDT	<-----	
RUI_WRITE(LU exp)	SDT +rsp	----->	
RUI_WRITE(LU norm)	data, BC	----->	
RUI_WRITE(LU norm)	data, MC	----->	
RUI_WRITE(LU norm)	data, EC, CD, RQD	----->	
RUI_READ(LU norm)	data +rsp	<-----	
RUI_READ(LU norm)	data, BC	<-----	
RUI_READ(LU norm)	data, MC	<-----	
RUI_READ(LU norm)	data, EC, RQD	<-----	
RUI_WRITE(LU norm)	data +rsp	----->	
RUI_READ(LU exp)	UNBIND	<-----	
RUI_WRITE(LU exp)	UNBIND +rsp	----->	
RUI_TERM	(NOTIFY)	----->	
	(NOTIFY +rsp)	<-----	

LUA 어플리케이션으로 발행하는 명령어	SNA 메세지	흐름 방향 어플리케이션	호스트

이 9 ! 서 n 플리케이션은 다음 단 h 를 수행합니다.

1. **RUI_INIT** 명령 n 를 발행하) LU-SSCP 세션을 만듭니다. (**RUI_INIT** 명령 n 는 퍼스널 통신 및 통신 서버 프로 W 램이 호스트로부터 ACTLU 메세지를 수신하 m ` 정 응답을 송신할 때 n 지 O 료되지 J 지만, 이들 메세지는 " 프로 W 램 ! 의해 처리되며 LUA n 플리케이션 ! 노출되지 J 습니다.)
2. **INITSELF** 메세지를 SSCP ! 보내 **BIND** d 청을 하 m 응답을 읽습니다.
3. 호스트 ! 서 **BIND** 메세지를 읽 m 응답을 작성합니다. 이 M 은 LU-LU 세션을 만듭니다.
4. 호스트 ! 서 **SDT** 메세지를 읽 n 초 b 화 ! O 료됐으며 데이타 전송을 시작할 수 있음을 나타냅니다.
5. 세 3 의 RU(마지막 RU는 명확한 응답이 필 d 함을 나타냅니다)로 8 성된 체인을 송신하며 응답을 읽습니다.
6. 세 3 의 RU로 8 성된 데이타의 체인을 읽으며 응답을 작성합니다.
7. 호스트 ! 서 **UNBIND** 메세지를 읽 m 응답을 작성합니다. 이 M 은 LU-LU 세션을 종료합니다.
8. **RUI_TERM** 명령 n 를 발행하) LU-SSCP 세션을 종료합니다. (퍼스널 통신 및 통신 서버 프로 W 램이 NOTIFY 메세지를 호스트 ! 송신하 m ` 정 응답을 b 다리지만 이들 메세지는 " 프로 W 램 ! 의해 처리되며 LUA n 플리케이션 ! 노출되지 J 습니다.)

BIND 점검

LU-LU 세션 초 b 화 동 H ! 호스트는 **BIND** 메세지를 LU-LU 세션 ! 서 사 k 할 RU 크 b 의 정보 ! 들 n 있는 퍼스널 통신 및 통신 서버 LUA n 플리케이션으로 송신합니다. 퍼스널 통신 및 통신 서버는 이 메세지를 **RUI_READ** 명령 n ! 있는 LUA n 플리케이션으로 리턴합니다. **BIND** ! 지정한 매 3 변수 ! 적합한지 점 K 하는 M 은 LUA n 플리케이션의 책임입니다. n 플리케이션은 다음 z 0 은 I 션을 ! 집니다.

- **BIND** ! 대한 9 응답이 들 n 있는 **RUI_WRITE** 명령 n 를 발행하) 있는 W 대로 **BIND**를 수 k 하십시 @ . 응답 ! 서는 데이타를 송신할 필 d ! x 습니다.
- 하나 이상의 **BIND** 매 3 변수를 조정하십시 @ . (이 M 은 **BIND** ! 조정 ! 능한 f l ! 만 허 k 됩니다.) 이를 수행하 b 위해 n 플리케이션은 9 응답이 들 n 있지만 수정된 **BIND**를 데이타로서 포함하는 **RUI_WRITE** 명령 n 를 발행합니다.
- 해당 SNA (지 코드를 데이타로 사 k 하) 부정 응답이 들 n 있는 **RUI_WRITE** 명령 n 를 발행하) **BIND**를 E 부(reject)하십시 @ .

RUI_WRITE 명령n ! 대한 자세한 내k은 제13장 RUI 명령n를 참조하십시@.

주: **BIND** 매3변수의 유효성 K증 및 모든 메세지 ! 매3변수M 일치되는지 확인하는 M은 LUA n플리케이션의 책임입니다. W러나 다음 두 ! 지 제한사항이 적k됩니다.

- 퍼스널 통신 및 통신 서버는 **BIND** ! 서 지정한 f이보다 더 d RU f이를 지정하는 **RUI_WRITE** 명령n를 E부합니다.
- 퍼스널 통신 및 통신 서버는 **BIND** ! 2차 LU ! 회선f합 성x 세션이며 @류 복8는 회선f합 실패 세션의 책임임을 지정하도록 d8합니다.

부정 응답 및 SNA 감지 코드

SNA (지 코드는 다음 f l ! LUA n플리케이션 ! 리턴될 수 있습니다.

- 호스트 ! LUA n플리케이션으로부터의 d청 ! 부정 응답을 송신할 때 부정 응답 ! 대한 이유를 나타내는 SNA (지 코드를 포함합니다. 다음z 0이 차후의 **RUI_READ** 명령n ! 서 n플리케이션 ! b록됩니다.
  - 1차 리턴 코드는 LUA_OK입니다.
  - d청/응답 표시b, 응답 유형 표시b 및 (지 데이타 포함 표시b는 (지 데이타를 포함하는 부정 응답을 표시하며 모두 1로 설정됩니다.
  - **RUI_READ** 명령n ! 리턴하는 데이타는 SNA (지 코드입니다.
- 퍼스널 통신 및 통신 서버 ! 호스트로부터 틀린 데이타를 수신할 때 보통 부정 응답을 호스트로 송신하며 틀린 데이타를 LUA n플리케이션으로 전달하지 J습니다. 이M은 다음z 0이 다음의 **RUI_READ** 또는 **RUI_BID** 명령n ! 서 n플리케이션 ! b록됩니다.
  - 1차 리턴 코드는 LUA_NEGATIVE_RSP입니다.
  - 2차 리턴 코드는 호스트 ! 송신되는 SNA (지 코드입니다.
- n떤 f l 퍼스널 통신 및 통신 서버는 호스트 ! 제x한 데이타 ! 유효하지 J음을 K출하지만 송신할 C바른 (지 코드를 판별할 수 x습니다. 이 f l 다음z 0은 방식으로 **RUI_READ** 명령n ! 서 9 \ d청(EXR) ! 있는 틀린 데이타를 LUA n플리케이션으로 전달합니다.
  - d청/응답 표시b는 d청을 나타내며 0으로 설정됩니다.
  - (지 데이타 포함 표시b는 (지 데이타 ! 포함되n 있음을 나타내며 1로 설정됩니다. (이 표시b는 보통 응답k으로만 사k됩니다.)
  - 메세지 데이타는 제시된 SNA (지 코드 ! 의해 대체됩니다.

  W러면 n플리케이션은 부정 응답을 메세지로 보내_ 하며 퍼스널 통신 및 통신 서버 ! 제시한 (지 코드를 사k하E나 변f할 수 있습니다.
- 퍼스널 통신 및 통신 서버는 n플리케이션이 제x한 데이타 ! 유효하지 J음을 나타내b 위해 n플리케이션 ! (지 코드를 송신할 수 있습니다. 이M은 다음z 0이 데이타를 제x한 **RUI_WRITE** 명령n ! 서 n플리케이션으로 b록됩니다.

– 1차 리턴 코드는 LUA_UNSUCCESSFUL입니다.
– 2차 리턴 코드는 SNA (지 코드입니다.

다른 2차 리턴 코드에서 SNA 감지 코드 구별하기

(지 코드 ! F 닌 2차 리턴 코드의 f l, 이 * 의 첫번째 두 바이트는 항상 0입니다. SNA (지 코드의 f l, 첫번째 두 바이트는 0이 F 니며 첫번째 바이트는 (지 코드 범주를 제 x 하 m 두 번째는 해당 범주 내의 특정 (지 코드를 식별합니다. (세 번째 및 네 번째 바이트 ! 는 추 ! 적인 정보 ! 들 n 있 E 나 0일 수 있습니다.)

SNA 감지 코드에 있는 정보

리턴된 (지 코드 ! 서 정보 ! 필 d 하면 *IBM 시스템 네트 v 크 체 h (SNA): 형식*을 참조하십시 @. (지 코드는 범주 ! 의해 숫자순으로 나 - 됩니다.

페이싱

페이싱은 LUA ! 의해 처리되며 LUA n 플리케이션은 페이싱을 제 n 할 필 d ! x 으며 페이싱 표시 b 플래 W 를 설정하지 말 F _ 합니다.

페이싱이 LUA n 플리케이션으로부터 호스트로 전송된 데이타 ! 서 사 k 되는 f l(이 M 은 **BIND** ! 의해 판별됩니다), **RUI_WRITE** 명령 n 를 0 료하려면 시 # 이 I 릴 수 있습니다. V 냐하면 퍼스널 통신 및 통신 서버 ! 더 많은 데이타를 송신하 b 전 ! 페이싱 응답을 b 다려 _ 하 b 때문입니다.

LUA n 플리케이션이 호스트 ! 또는 호스트로부터의 한쪽 방향으로 많은 데이타를 전송하 b 위해 사 k 되는 f l(9 를 들 n 파일 전송 n 플리케이션), 호스트 8 성은 페이싱이 해당 방향 ! 서 사 k 됨을 지정해 _ 하며 이 M 은 데이타를 수신하는 노드 ! 데이타로 넘치지 J 으며 데이타 b o 5 * 을 다 소모하지 J 도록 보장합니다.

세그멘테이션

RU의 세 W 멘테이션은 LUA ! 의해 처리됩니다. LUA는 항상 0 료 RU를 n 플리케이션으로 전달하며 n 플리케이션은 0 료 RU를 LUA로 전달해 _ 합니다.

Courtesy 수신응답

퍼스널 통신 및 통신 서버는 해당 d 청으로 n 플리케이션 ! 의해 송신된 응답 ! | 련시키 b 위해 호스트로부터 수신한 d 청 레코드를 유지합니다. n 플리케이션이 응답을 송신할 때 퍼스널 통신 및 통신 서버 프로 W 램은 x 래 d 청으로부터 데이타 M 이 응답을 | 련시키며 이 M , | 된 b o 5 * 을 해제시킬 수 있습니다.

호스트 ! 9 \ 응답만을 지정하는 f l(부정 응답은 송신될 수 있으나 ` 정 응답은 송신되지 J 습니다), 퍼스널 통신 및 통신 서버는 n 플리케이션이 다

음! 부정 응답을 송신하는 f l d청 레코드를 유지해_ 합니다. n플리케이션이 응답을 송신하지 J는 f l, 이 d청z , |된 b o 5 * 은 해제될 수 x습니다.

이 때문! 퍼스널 통신 및 통신 서버는 LUA n플리케이션이 호스트로부터 9 \ 응답의 d청!만 `정 응답을 발행할 수 있T 합니다 (이M을 courtesy 수신응답이라m 합니다). 응답은 호스트! 송신되지 J지만 d청z , |된 b o 5 * 을 소E하b 위해 퍼스널 통신 및 통신 서버! 사k합니다.

체인의 끝까지 데이타 제거

호스트! d청 단위(RU)의 체인을 LUA n플리케이션! 보내면 n플리케이션은 응답을 송신하b 전! 체인의 마지막 RU! 수신될 때n지 b다릴 수 있E나 체인!서 마지막이 F닌 RU! 부정 응답을 송신할 수 있습니다. 부정 응답이 체인 중#! 송신되면 퍼스널 통신 및 통신 서버는 이 체인!서 다음의 모든 RU를 제E하며 n플리케이션! 송신하지 J습니다.

퍼스널 통신 및 통신 서버! 체인!서 마지막 RU를 수신하면 **RUI_READ** 또는 **RUI_BID** 명령n의 1차 리턴 코드를 0의 2차 리턴 코드! 있는 LUA_NEGATIVE_RSP! 설정하) n플리케이션! 이M을 나타냅니다.

주: 호스트는 체인 중#! CANCEL z 0은 메세지를 송신하) 체인을 종료할 수 있습니다. 이 f l CANCEL 메세지는 **RUI_READ** 명령n!서 n플리케이션! 리턴되며 LUA_NEGATIVE_RSP 리턴 코드는 사k되지 J습니다.

구:

LUA n플리케이션이 사k하는 " LU는 퍼스널 통신 및 통신 서버 NOF 명령n를 사k하E나 SNA 노드 8성 프로W램을 통해 8성되n_ 합니다. (자세한 내k은 *시스템 |리 프로W래밍*을 참조하십시@.) W리m 8성!는 LUA LU 풀이 포함될 수 있습니다. 풀은 n플리케이션이 W룹!서 해제 LU를 사k할 수 있는 M z 0은 유사한 특성을 !진 LU W룹입니다. 사k!능한 LU보다 더 많은 n플리케이션이 있는 f l 이M은 처음 제x되는 b본!서 LU를 할당하E나 다른 링크!서 LU 선택사항을 제x하b 위해 사k될 수 있습니다.

LUA LU 풀(선택적)

필d하다면 n플리케이션이 사k할 둘 이상의 LUA LU를 8성하m LU들을 풀로 W룹화할 수 있습니다. 이M은 n플리케이션이 세션을 시작하려m 시도할 때 특정 LU보다는 풀을 지정할 수 있으며 풀로부터 첫번째 사k!능한 LU! 지정될 M임을 의미합니다.

LUA n플리케이션은 LU명으로 **RUI_INIT** 명령n를 발행하) 세션을 시작하려 함을 퍼스널 통신 및 통신 서버! K립니다. 이 이름은 이전! *System*

Management Programming! 정의한 LUA LU 또는 LU 풀의 이름z 일치해 _ 합니다. 퍼스널 통신 및 통신 서버는 다음z 0이 이 이름을 사k합니다.

- 제x된 이름이 풀! x는 LU의 이름인 f l, 세션은 해당 LU! 사k! 능한 f l W LU를 사k하) 지정됩니다(즉, LUA n플리케이션! 의해 이미 사k중이지 J은 f l).
- 제x된 이름이 LU 풀의 이름이E나 이미 사k중인 풀 내의 특정 LU 이름인 f l, 세션은 풀!서 사k!능한 첫번째 LU를 사k하) 지정됩니다(LU! 사k!능한 f l).

  주: 이M은 **RUI_INIT** 명령n! 지정한 LU 이름이 F닐 수 있습니다.

SNA API 클라이언트 고려사항

LUA n플리케이션이 클라이p트 v크스테이션! 있는 f l, LUA 세션도 로컬 v크스테이션!서 정의되n _ 합니다. 이 LUA 세션 이름은 다중 통신 서버 및 LUA 정의를 포함할 수 있으므로 , a이 사k불능이 되면 SNA 클라이p트 코드! 새 서버로 % 수 있도록 허k합니다.

제10장 RUI LUA 명령어의 특징

이 장!서는 LUA 명령n! 대해 다음z 0은 특수 f l M 사k법! 대해 다룹니다.

- 9 \ d청 처리— 사k자 프로W램이 부정 응답을 발행하T 하는 LUA로부터의 d청
- 프로W램 디자인을 통해 LAN 트래픽 최소화
- LUA 명령n의 무한 중단 처리
- 세션 장V로부터의 복8

예외 요청 처리

RUI 및 SLI 모니터는 둘다)러 프로토콜의 상태를 모니터하며 RU 형식의 유효성을 K증합니다. 인터페이스! 1차 논리 장치(PLU)로부터의 부적절한 RU를 K출하면 부정 응답을 발행해_ 합니다. LUA는 9 \ d청(EXR)으로 입력된 RU를 형식화하) K출된 이 @류를 n플리케이션! 통지합니다. EXR은 송수신 d 8 명령n(**RUI_BID** 또는 **SLI_BID**)나 입력 명령n(**RUI_READ** 또는 **SLI_RECEIVE**) 중 하나!서 사k자 프로W램으로 인도됩니다. EXR은 d청 헤더(RH)!서 다음 조G! 의해 표시됩니다.

- *lua_rh.rri*를 0으로 설정(RU는 d청 단위(RU)입니다).
- *lua_rh.sdi*를 1로 설정((지 데이타 포함).

이M은 비정상적인 RH 비트 조합입니다. (지 데이타는 보통 응답 RU의 내k이며 d청 RU! F닙니다. LUA는 이 비정상적인 조합을 사k하) PLU! @류를 발생한 비정상적인 사실을 사k자 프로W램! f m합니다. 4바이트 (지 코드는 EXR의 부분이며 K출된 @류를 지정합니다. (지 데이타이\! LUA는 x래 RU의 3바이트n지 리턴합니다.

명령어 레코드 변경

사k자 n플리케이션은 EXR을 부정 응답으로 형식화하) 사k중인 API! 따라 RUI_WRITE 중 하나를 사k하) PLU! 송신해_ 합니다. 응답 출력! 대한 EXR 입력을 변환하려면 명령n 레코드!서 다음 변f사항을 작성하십시@.

- *lua_rh.rri*를 1로 설정하) 이M이 응답임을 보)줍니다.
- *lua_rh.ri*를 1로 설정하) 부정 응답을 나타냅니다.
- *lua_flag2*! 있는 * ! Y E하) *lua_flag1*!서 해당 데이타 흐름 플래W를 설정하십시@.
- *lua_message_type*을 LUA_MESSAGE_TYPE_0으로 설정하십시@.
- 사k중인 API! 따라 *lua_opcode*를 LUA_OPCODE _RUI_WRITE로 설정하십시@.

- *lua_data_length*를 (지 데이타의 f 이인 4로 설정하십시@.
- *lua_data_ptr*을 (지 데이타의 주소로 설정하십시@. 이 위치는 EXR을 K 출한 명령n ! 좌l됩니다. 명령n ! RUI_BID인 f l (지 데이타는 명령n 레코드 ! 있는 ″peek 버퍼″ ! 있으며 명령n ! RUI_READ인 f l, (지 데이타는 입력 버퍼 ! 있습니다.
- *lua_max_length*를 0으로 설정하십시@.

사k자 프로W램은 이제 EXR ! 대해 명령n 레코드 및 버퍼를 사k하) RUI_WRITE를 시작하) 부정 응답을 송신할 수 있습니다.

브래킷 송수신 요구 거부 처리

한 ! 지 f l를 제\하m EXR ! 서 LUA ! 제x하는 (지 코드는 PLU ! 리턴하려는 유일한 M입니다. W러나 브래킷이 사k중이m PLU ! 스피커 ! 되도록 d청하는 f l, n플리케이션은 (지 코드의 선택사항을 ! 집니다.

- LUA는 PLU로부터의 BID 명령을 E부(reject)할 수 있습니다. BID를 E부하b 위해 LUA는 (지 코드 LUA_BB_REJECT_NO_RTR이 들n 있는 EXR을 형식화하며 브래킷 송수신 d 8 ! E부되z으며 RTR 명령은 나중 ! 발행되지 J음을 p ^ 합니다. 이 (지 코드의 숫자 * 은 0x00001308L입니다(C 프로W램 ! 서 코드화하는 M처럼 Intel** 또는 바 방 문물물물 물

- v크스테이션! 데이타를 리턴하는 메세지

W러나 RUI_READ는 한 단h로 동일한 작w을 수행할 수 있습니다. 단순히 RUI_READ 명령n를 시작하) O료되b를 b다리는 f l, 두 3의 LAN 메세지! 줄n 듭니다.

『송수신 로직』의 유일한 이점은 메세지를 수신하b 전! 메세지의 크b를 K 수 있다는 M입니다. 이M은 s마나 큰 버퍼! 필d한지 K 때n지 데이타 버퍼 할당을 지,하T 합니다. 유일한 입력 명령n를 사k하는 f l, 송수신 d8! O료된 후! 버퍼를 할당하b보다는 미리 최대 버퍼 크b를 KF_ 합니다.

일시중단 처리하기

RUI 명령n의 O료는 PLU n플리케이션, 호스트 시스템, 네트v크 및 퍼스널 통신 및 통신 서버의 조치! 좌l됩니다. 이들 중 하나! 느리T 응답하E나 응답! 실패하는 f l, 명령n는 무한정 일시중단될 수 있습니다. 프로W램을 디자인할 때 사k자나 프로W램! 일시중단된 명령n를 종료하는 방법을 제x하) 일시중단을 9상할 수 있습니다.

RUI_INIT 취소하기

RUI_INIT 명령n는 호스트! 지정된 LU를 활성화시킬 때n지 일시중단합니다. 보통 호스트는 n플리케이션이 시작하b 전! ACTLU 명령을 송신하지만 W렇T 하는 M이 d8되지는 J습니다. n플리케이션이 시작할 때 메인프레임은 종료되E나)전히 시작되m 있을 수 있습니다.

프로W램이 일시중단된 RUI_INIT를 취소할 필d! 있는 f l, RUI_TERM 명령n를 발행할 수 있습니다.

RUI_WRITE 취소하기

페이싱이 사k중이면 출력이 일시중단될 수 있습니다. 호스트! 일시적으로 데이타 읽b를 중단하E나 페이싱 응답 전송! 실패하면 RUI_WRITE는 페이싱 창이 -리b를 b다리는 동H 일시중단될 수 있습니다.

프로W램이 일시중단된 RUI_WRITE를 취소할 필d! 있다면 RUI_TERM으로 세션을 닫F_ 합니다.

RUI_READ 취소하기

명령n! 지정된 흐름! 입력이 도달할 때n지 입력 명령n는 보통 일시중단됩니다. 사k자 프로W램은 RUI_PURGE를 사k하) 보류중인 RUI_READ를 취소할 수 있습니다. 세션을 닫으면 보류중인 입력 명령n도 취소됩니다.

명령어 완료 확인하기

명령n O료를 잘못 처리하면 프로W램!서 무한 대b를 생성할 수 있습니다. 프로W램이 명령n를 시작하m 명령n! 동b적으로 O료하도록 주의시키는데 실패하m 비동b O료를 대b하는 f l, 프로W램은 5x히 대b합니다.

RUI #트리 포인트는 실행된 명령n의 1차 리턴 코드를 명시적 az로서 리턴합니다. 명령n! 비동b적으로 O료될지)부를 K리는 !장 #단한 방법은 W림8!서M 0이 LUA_IN_PROGRESS! 대한 이 명시적 리턴을 테스트하는 M입니다.

```
unsigned short rc;
rc = RUI(ptrToTheVerb);
if (LUA_IN_PROGRESS == rc)
   // verb will complete later; the callback function will be entered
else
   // verb is finished now; the callback function will never be entered
```

W림8. 명령n O료 테스트하b

데이타 압축하기

데이타 P축은 RUI 및 SLI API 인터페이스 g쪽!서 지x됩니다. 데이타 P축의 사k은 BIND 및 BIND 응답! 의해 세션마다 조정됩니다. P축이 세션!서 사k되도록 조정된 f l, LZ9나 수행 f이 인코딩(RLE) P축 Km리즘은 1차 LU (PLU)!서 인바n드 수k되며 RLE는 PLU로 데이타를 송신하b 위해 사k됩니다.

RUI 및 SLI API g쪽 f l!는 데이타 P축이 다음 중 하나! 의해 처리될 수 있습니다.

- n플리케이션이 데이타를 P축하m P축해제합니다
- 통신 서버는 호스트로 데이타를 P축하m P축해제하지만 P축해제된 데이타를 n플리케이션으로부터 W리m n플리케이션으로 수k합니다.

세션마다 데이타 압축을 조정하는 규칙

다음은 세션당 RUI 및 SLI API g쪽! 대한 데이타 P축을 조정하는 T칙입니다.

RUI 규칙

1. RUI n플리케이션이 데이타의 P축 및 P축해제를 처리하도록 허k하려면,
   - RUI n플리케이션은 P축이 제x되E나 d청됐음을 나타내b 위해 바이트 25의 비트 6 및 7을 .는 BIND d청을 수신합니다.

- RUI n플리케이션은 ″제x되E나 d8된 P축이 수k됨″을 나타내b 위해 바이트 25의 비트 6 및 7로 `정 BIND 응답을 리턴해_ 합니다.

2. 통신 서버! RUI n플리케이션의 측면!서 P축을 처리하도록 허k하려면,
   - 다음을 수행함으로써 노드의 P축 지x을 나타내려면 통신 서버 SNA 노드 8성 유틸리티를 사k하십시@.
     - 8성 노드 선택
     - 확장 선택
     - 노드! 지x하는 최대 P축 레벨을 RLE로 설정
   - RUI n플리케이션은 P축이 제x되E나 d청됐음을 나타내b 위해 바이트 25의 비트 6 및 7로 BIND 응답을 수신합니다.
   - RUI n플리케이션은 ″P축 x음″을 나타내b 위해 바이트 25의 비트 6 및 7로 `정 BIND 응답을 리턴합니다. 통신 서버는 BIND 응답을 해석하m 수정한 다음, 호스트로 데이타를 P축하m P축해제합니다.

SLI 규칙

1. SLI n플리케이션이 데이타의 P축 및 P축해제를 처리하도록 허k하려면,
   - SLI n플리케이션이 **SLI_OPEN** 명령을 발행할 때 BIND Callback 루틴을 제x해_ 합니다.
   - SLI n플리케이션의 BIND callback 루틴이 시작되면 SLI는 P축이 제x되E나 d청됐음을 나타내b 위해 바이트 25의 비트 6 및 7을 .는 BIND d청을 수신합니다.
   - SLI n플리케이션은 ″제x되E나 d8된 P축이 수k됨″을 나타내b 위해 바이트 25의 비트 6 및 7로 BIND 응답을 리턴해_ 합니다.
2. 통신 서버! SLI의 측면!서 P축을 처리하도록 허k하려면,
   - 다음을 수행함으로써 노드의 P축 지x을 나타내려면 통신 서버 SNA 노드 8성 유틸리티를 사k하십시@.
     - 8성 노드 선택
     - 확장 선택
     - 노드! 지x하는 최대 P축 레벨을 RLE로 설정
   - n플리케이션이 **SLI_OPEN** 명령n!서 BIND callback 루틴을 제x하지 J은 fl, SLI는 통신 서버! SLI! 대한 데이타를 P축하m P축해제할 M임을 나타내b 위해 BIND 응답을 디폴트로 설정합니다.
   - n플리케이션이 BIND callback 루틴을 제x하지 J은 fl,
     - BIND callback 루틴이 시작되면 P축이 제x되E나 d청됐음을 나타내b 위해 바이트 25의 비트 6 및 7을 .는 BIND d청을 수신합니다.
     - SLI n플리케이션은 ″P축 x음″을 나타내b 위해 바이트 25의 비트 6 및 7로 BIND 응답을 리턴합니다. 통신 서버는 BIND 응답을 해석, 수정하m 호스트로 데이타를 P축하m P축해제합니다.

< G 장애로부터 복구하기

LUA 세션이 @류 때문! 닫힌 두 3의 인스턴스! 있습니다.

- LUA 명령n ! 1차 리턴 코드 LUA_SESSION_FAILURE로 0료된 f l 또는
- RUI_INIT! 0료된 후! LUA 명령n ! 1차 리턴 코드 LUA_STATE_CHECK 및 2차 리턴 코드 LUA_NO_RUI_SESSION으로 0료된 f l.

세션은 자주 재8성될 수 있습니다. 프로W램이 d청하는 f l LUA는 복8를 시도합니다.

프로W램이 @류 때문! 닫힌 LUA 세션을 수신하면 복8하려는 f l 다음을 수행해_ 합니다.

- 세션을 닫지 마십시@. 세션은 이미 닫혀 있습니다.
- 세션을 - b 위해 x래 사k한 명령n를 사k하) 세션을 다시)십시@(RUI_INIT). 이 명령n ! 0이 F닌 1차 리턴 코드로 0료하는 f l, 세션은 이번! 재시작될 수 x습니다.
- 복8 ! 시#이 I리므로 복8 ! 진행중이면 대화방식의 사k자 ! T 통지하십시@. 사k자의 작w 상태는 PLU n플리케이션의 디자인! 좌l됩니다.

제11장 LUA 프로그램 구현

이 장!서는 LUA 프로W램 8현 및 작성! 대해 설명합니다. 다음 주제를 포함합니다.

- LUA 서비스 호출 및 순서화
- LUA 프로W램 작성
- 비동b 0료 및 callback 함수 사k
- 다른 플랫폼!서 컴파일 및 링크

LUA의 통신 서버 8현은 Microsoft** NT SNA 서버M 호환!능한 2진이 되도록 디자인되z으며 OS/2 Communications Manager/2 버전 1.0 LUA의 RUI 및 SUI 인터페이스 8현z 유사합니다.

LUA 프로그램 작:하기

LUA!는 RUI 명령n 및 SLI 명령n! 대한 하나의 b본 DLL이 들n 있습니다. LUA n플리케이션 프로W램은 이 DLL을 호출하) 명령n를 수행합니다.

LUA n플리케이션 프로W램은 명령n 제n 블럭!서 선택된 필드를 설정하m RUI나 SLI를 호출하) 명령n 제n 블럭! 포인터를 전달합니다. 명령n 제n 블럭! 있는 필드는 LUA! d청된 조치를 정의합니다. LUA는 명령n 제n 블럭!서 필드를 수정하) LUA! n플리케이션 프로W램으로 제n를 리턴하b 전! 조치 az를 표시합니다. W러면 n플리케이션 프로W램은 다음 프로세싱!서 명령n 제n 블럭으로부터 리턴된 매3변수를 사k할 수 있습니다.

F래 표는 RUI 프로W램을 컴파일하m 링크하는 데 필d한 제x 헤더 파일 및 라이브러리의 소스 모듈 사k을 보)줍니다.

표 15. n5체제의 RUI API! 대한 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL명
WINNT & WIN95	WINLUA.H	WINRUI32.LIB	WINRUI32.DLL
WIN3.1	WINRUI.H	WINRUI.LIB	WINRUI.DLL
OS/2	LUA_C.H	ACSRUI.LIB	ACSRUI.DLL

표 16. SLI API! 대한 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL명
WINNT & WIN95	WINLUA.H	WINSLI32.LIB	WINSLI32.DLL

표 16. SLI API ! 대한 헤더 파일 및 라이브러리 (h 속)

운영체제	헤더 파일	라이브러리	**DLL명**
WIN3.1	WINSLI.H	WINSLI.LIB	WINSLI.DLL
OS/2	LUA_C	ACSSLI.LIB	ACSSLI.DLL

주: SLI API는 서버 ! 서 지 x 되며 통신 서버 클라이 p 트 ! 서 지 x 되지 J 습니다.

*WINNT = Windows NT
*WIN95 = Windows 95
*WIN3.1 = Windows 3.1

LUA - 비스 호출

사 k 자 프로 W 램은 지정된 # 트리 포인트를 호출하 m 단일 매 3 변수(*명령 n 레코드*라 m 하는 데이타 8 조의 주소)를 전달하) LUA 서비스를 b 동합니다. 레코드 ! 는 특정 함수 ! 대한 입력 매 3 변수 ! 들 n 있습니다. LUA는 작 w 의 a z 인 출력 매 3 변수로 레코드를 ; 신합니다.

명령어 레코드 내용 이해하기

명령 n 레코드의 3 ! 지 유형은 모두 다르 T 8 성되 z 지만 다음 z 0 은 매 3 변수 ! 대한 필드를 제 x 합니다.

작업 수행되는 특정 작 w 을 지정하는 번호. 작 w 의 b 호식 이름은 『_cons.h』 포함 파일 ! 선 p 됩니다.

명령어 레코드 길이
작 w ! 서 작 w 으로 변화할 수 있 m LUA ! 레코드를 프로세스하 b 위해 필 d 로 하는 명령 n 레코드의 크 b .

세션 식별자(ID)
통신 및 서비스 명령 n ! 서 세션이나 세션명을 식별하 b 위한 번호.

1차 리턴 코드
일반 성 x 이나 장 V 를 나타내 b 위해 LUA ! 리턴하는 번호.

2차 리턴 코드
특정 문제점을 나타내 b 위해 실패시 LUA ! 리턴하는 번호.

상 | 자
명령 n 레코드를 다른 데이타 ! | 련시키 E 나 비동 b O 료 동 H 명령 n 레코드 식별을 위해 n 플리케이션이 사 k 할 수 있는 d 정수.

통지 핸들
명령 n ! 비동 b 적으로 O 료할 때 통지되는 이벤트의 핸들.

Windows NT 및 Windows 95의 f l, 이 M 은 이벤트 핸들이 n _ 합니다. Windows 3.1 ! 서 이 M 은 창 핸들이 n _ 합니다. OS/2의 f l, 이 M 은 신호 b 핸들이 n _ 합니다.

이 대부분의 필드는 동일한 데이타 유형을 ! 지며 이들 필드 ! 나타나는 " 명령 n 레코드 ! 서 동일한 @ 프셋 ! 있습니다. W 러나 작 w 코드 및 명령 n f 이 필드는 다른 특성을 ! 집니다.

다중 프로세스

LUA n 플리케이션 프로 W 램은 단일 프로세스로 제한됩니다. 동일한 LUA n 플리케이션 프로 W 램의 다른 인스턴스 ! 다른 프로세스 ! 서 시작할 수 있지만 " n 플리케이션 프로 W 램은 다른 LUA LU를 사 k 해 _ 합니다.

W 리 m 단일 프로세스는 " b m 유한 LUA LU를 . 는 다중 LUA n 플리케이션 프로 W 램으로 8 성될 수 있습니다.

다중 스레드

단일 LUA n 플리케이션 프로 W 램은 다중 스레드를 사 k 하) 명령 n 를 발행할 수 있습니다. 이 M 은 단일 LUA n 플리케이션 프로 W 램으로부터 동시 ! 다중 명령 n 를 발행할 수 있 T 합니다. 동일한 LUA n 플리케이션 프로 W 램의 다른 인스턴스는 다른 스레드 ! 서 시작할 수 있으나 " n 플리케이션 프로 W 램은 다른 LUA LU를 사 k 할 수 있습니다.

주: LUA n 플리케이션 프로 W 램이 명령 n 를 발행한 후 ! 명령 n ! O 료될 때 n 지 명령 n 제 n 블럭의 일부를 변 f 하지 말 F _ 합니다. RUI는 명령 n 제 n 블럭의 n 플리케이션 사본만을 사 k 합니다. 자세한 정보는 205페이지의 『LUA 명령 n 통지하 b 』를 참조하십시 @ .

LUA 명령어 통지하기

LUA 명령 n 는 동 b 적으로 또는 비동 b 적으로 O 료합니다. 동 b 명령 n O 료는 RUI ! LUA n 플리케이션 프로 W 램 ! 리턴할 때 LUA ! 대한 호출 이후 ! 해당 명령 n ! 대한 모든 프로세싱이 O 료되며 비동 b 통지 방법은 사 k 되지 J 습니다. 타이밍 조 G 때문 ! 명령 n 는 비동 b 적으로 O 료할 수 있으나 모든 프로세싱은 LUA ! LUA n 플리케이션 프로 W 램으로 리턴하는 시 # ! O 료됩니다. 비동 b 명령 n O 료는 프로세스 ! 성 x 하 E 나 실패하) O 료된 시 b 를 n 플리케이션 프로 W 램 ! 통지하 b 위해 LUA ! 통지 방법을 사 k 함을 의미합니다.

LUA n 플리케이션 프로 W 램은 명령 n ! 비동 b 적으로 O 료할 때 다음 z 0 은 방법 중 하나로 통지될 수 있습니다.

- LUA n 플리케이션 프로 W 램은 **lua_flag2_async** 및 **lua_prim_rc** 매 3 변수를 사 k 하) 명령 n 프로세싱 상태를 판별합니다.
- n 플리케이션은 **lua_post_handle** 매 3 변수 ! 서 이벤트를 지정합니다. 이 M 은 명령 n ! O 료될 때 설정됩니다.

ASCII에서 EBCDIC으로 변환하기

보통 호스트로 송신되는 모든 메세지는 EBCDIC이며 PLU는 메세지! EBCDIC임을 !정합니다. 9를 들n **BIND**! 표시되는 PLU명은 EBCDIC 문자 - 이n _ 합니다. ASCII로 문자 - 을 저장하는 LUA n플리케이션 프로W램은 문자 - 이 SNA 메세지로 송신되b 전! EBCDIC으로 변환해_ 합니다.

LUA n플리케이션 프로W램이 n플리케이션 데이타를 변환할 필d! 있는지)부는 상대방 n플리케이션 프로W램#의 3별 동의! 좌l됩니다. LUA n플리케이션 프로W램이 보통 EBCDIC을 사k하는 노드M 통신하는 f l, 적당한 w!서 ASCII 데이타를 EBCDIC으로 변환시켜_ 합니다.

ASCII를 EBCDIC(또는 b타)으로 변환하는 M은 315페이지의 『제17장 x통 서비스 명령n (CSV)』!서 설명한 변환 명령n로 수행할 수 있습니다.

제12장 RUI LUA 엔트리 포인트

이 장 ! 서는 LUA ! 대한 프로시듀 n #트리 포인트를 설명합니다.

RUI DLL은 다음 프로시듀 n #트리 포인트를 정의합니다.

주: 이 책의 제2부 ! 포함된 M 은 다음 시스템 ! 서 제 x 하는 LUA API ! 있는 정보입니다.

- Windows NT ! 서 수행하는 통신 서버
- 통신 서버/NT 제품으로 제 x 되는 OS/2, Windows NT, Windows 95 및 Windows 3.1 k SNA API 클라이 p 트.
- Windows 95 및 Windows NT k 퍼스널 통신.

이들 시스템이 제 x 하는 지 x # 의 차이점이 있는 f l, 명시됩니다.

RUI()

모든 **RUI** 명령 n ! 대한 이벤트 통지를 제 x 합니다.

구문

```
void WINAPI RUI (LUA_VERB_RECORD* vcb);
```

매 3 변수

설명

vcb 제 x 된 매 3 변수로 명령 n 제 n 블럭의 주소를 지정합니다.

리턴 값

lua_prim_rc ! 리턴된 * 은 비동 b 통지 ! 발생할지) 부를 나타냅니다. 필드 ! LUA_IN_PROGRESS로 설정된 f l, 비동 b 통지 ! 이벤트 신호를 통해 발생합니다. 플래 W ! LUA_IN_PROGRESS ! F 닌 f l, d 청이 비동 b 적으로 0 료됐습니다. 모든 @ 류 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시 @ .

사용시 주의사항

n 플리케이션은 명령 n 제 n 블럭의 *lua_post_handle* 매 3 변수 ! 서 이벤트 ! 대한 핸들을 제 x 해 _ 합니다. 이벤트는 신호 x 음 상태 ! 있 n _ 합니다.

비동 b 작 w 이 0 료되면 n 플리케이션은 이벤트의 신호로 통지됩니다. 이벤트의 신호 ! 따라 @ 류 상태 ! 대한 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시 @ . **| 련 참조:** 209페이지의 『WinRUI』.

OS/2 클라이 p 트 ! 지 x 되는 유일한 RUI # 트리 포인트입니다.

WinRUI

모든 RUI 명령n ! 대한 비동b 메세지 통지를 제x합니다.

구문

```
int WINAPI WinRUI (HWND hWnd,
                   LUA_VERB_RECORD* vcb);
```

매3변수

설명

hwnd O료 메세지를 수신하b 위한 창 핸들.

vcb 명령n 제n 블럭 ! 대한 포인터.

리턴 값

프로세스를 위해 RUI ! d청을 수k했는지)부를 나타내는 *을 함수 ! 리턴합니다. 리턴 * 0은 d청이 수k됐으며 처리될 M임을 나타냅니다. 0이 F닌 *은 @류를 나타냅니다. !능한 @류 코드는 다음z 0습니다.

WLUAINVALIDHANDLE

제x되는 창 핸들은 유효하지 J습니다.

lua_flag2.async ! 리턴된 *은 비동b 통지 ! 발생할지)부를 나타냅니다. 플래W ! 설정된 f l(0이 F님), 비동b 통지는 n플리케이션의 메세지 대b행렬 ! 통지된 메세지를 통해 발생합니다. 플래W ! 설정되지 J은 f l, d청이 비동b적으로 O료됩니다. 모든 @류 상태 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시@.

사용시 주의사항

명령n를 O료하면 n플리케이션의 창 *hWind*는 입력 문자-로 **WinRUI**를 .는 **RegisterWindowMessage** ! 의해 리턴되는 메세지를 수신합니다. **lParam** 인수 !는 O료된 M으로 통지되는 VCB의 주소 ! 들n 있습니다. *wParam* 인수는 정의되n 있지 J습니다. 처리를 위해 d청을 수k하지만(함수 호출 ! 서 0을 리턴합니다) VCB !서 설정된 1차 및 2차 리턴 코드로 나중 ! E부하는 M이 !능합니다. 모든 @류 상태 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시@.

n플리케이션이 **WinRUIStartup**을 사k하) 세션을 l선 초b화하지 J m **WinRUI**을 호출하는 f l, @류 ! 리턴됩니다. **|련 참조:** 208페이지의 『RUI()』.

이 #트리 포인트는 통신 서버 Windows 3.1 클라이p트 !서 지x되지 J습니다.

WinRUICleanup()

RUI API ! 서 n 플리케이션을 종료하 m 등록취소합니다.

구문

```
BOOL WINAPI WinRUICleanup (void);
```

리턴 값

리턴 * 은 등록취소의 성 x 또는 실패를 나타냅니다. * 이 0이 F 니면 n 플리케이션이 등록취소되 z 습니다. * 이 0이면 n 플리케이션이 등록취소되지 J R 습니다.

사용시 주의사항

RUI API를 등록취소하려면, 9 를 들 n 특정 n 플리케이션 ! 할당된 자 x 을 해제하려면 **WinRUICleanup**을 사 k 하십시@.

LU ! 세션 ! 있는 동 H **WinRUICleanup**이 호출되면(**RUI_TERM** ! 발행되지 J 음) 종 a 처리 코드는 - 린 모든 세션의 n 플리케이션 ! 대해 **RUI_TERM** 닫 b 유형 ABEND를 발행합니다. **| 련 참조:** 215페이지의 『WinRUIStartup()』.

WinRUIGetLastInitStatus()

이 b능은 **RUI_INIT**의 상태를 판별하는 방법을 n플리케이션! 제x하므로 n플리케이션은 **RUI_INIT**! 시#종료되n _ 하는지)부를 a정할 수 있습니다. 이 호출을 사k하) 상태의 b록을 시작하E나 상태 b록을 종료하E나 현재 상태를 KF내십시@. 자세한 내k은 사k시 주의사항절을 참조하십시@.

구문

```
int WINAPI WinRUIGetLastInitStatus (DWORD dwSid,
                                    HANDLE hStatusHandle,
                                    DWORD dwNotifyType,
                                    BOOL bClearPrevious);
```

매3변수

설명

dwSid 상태! 판별되는 세션의 세션 식별자(ID). *이 0이면 *hStatusHandle*이 모든 세션의 상태를 b록하b 위해 사k됩니다. **RUI_INIT** VCB! 있는 *lua_sid* 매3변수는 **RUI_INIT**! 대한 **RUI**() 또는 **WinRUI**()의 호출이 리턴되자마자 유효합니다.

hStatusHandle

세션! 대한 상태! 변f되z음을 n플리케이션! 신호하는 데 사k되는 핸들. 창 핸들, 이벤트 핸들 또는 널(null)일 수 있습니다. *dwNotifyType*이 이! 따라 설정되n _ 합니다.

- *hStatusHandle*이 창 핸들이면 상태는 창 메세지를 통해 n플리케이션! 송신됩니다. 프로W램은 **WinRUI** 문자-을 사k하) **RegisterWindowMessage**!서 메세지를 r습니다. 매3변수 wParam!는 세션 상태! 들n 있습니다(리턴 * 참조). *dwNotifyType*의 *! 따라 **lParam**!는 세션의 RUI 세션 ID나 **RUI_INIT** 명령n로부터의 **lua_correlator** *이 들n 있습니다.
- *hStatusHandle*이 이벤트 핸들이면 *dwSid*! 의해 지정한 세션! 대한 상태! 변하면 이벤트! 신호 상태로 입력됩니다. W러면 n플리케이션은 새 상태를 KF내b 위해 **WinRUIGetLastInitStatus()**! 대한 호출을 해_ 합니다. 이벤트는 **RUI** 명령n의 0료를 신호하b 위해 사k되는 Mz 0지 JF_ 합니다.
- *hStatusHandle*이 널(null)이면 *dwSid*! 지정하는 세션 상태는 리턴 코드로 리턴됩니다. 이 fl *dwSid*는 *bClearPrevious*! TRUE! F니라면 0이 F니n _ 합니다. *hStatusHandle*이 널이면 *dwNotifyType*이 무시됩니다.

dwNotifyType

표시 유형이 필d합니다. 이M은 창 메세지의 **lParam** 내kz **WinRUIGetLastInitStatus()**! *hStatusHandle*을 해석하는 방식을 판별합니다. 허k *은 다음z 0습니다.

WLUA_NTFY_EVENT
hStatusHandle 매개변수는 이벤트 핸들이 들어 있습니다.

WLUA_NTFY_MSG_CORRELATOR
hStatusHandle 매개변수는 창 핸들이 들어 있으며 리턴된 창 메세지의 **lParam**은 LUA 상관자 및 RUI를 포함해야 합니다.

WLUA_NTFY_MSG_SID
hStatusHandle 매개변수는 창 핸들이 들어 있으며 리턴된 창 메세지의 **lParam**은 LUA 세션 식별자(ID)를 포함해야 합니다.

bClearPrevious
TRUE면 *dwSid*로 식별되는 세션에 대해 더 이상 상태 메세지가 송신되지 않습니다. *dwSid*가 0이면 상태 메세지는 세션에 더 이상 송신되지 않습니다. *bClearPrevious*가 TRUE면, *hStatusHandle* 및 *dwNotifyType*이 무시됩니다.

리턴 값

WLUASYSNOTREADY
퍼스널 통신 또는 통신 서버가 수행중입니다.

WLUANTFYINVALID
dwNotifyType 매개변수가 유효하지 않습니다.

WLUAINVALIDHANDLE
hStatusHandle 매개변수는 유효한 핸들이 들어 있지 않습니다.

WLUALINKINACTIVE
호스트에 대한 링크가 활동중이지 않습니다.

WLUAPUINACTIVE
호스트에 대한 링크가 활동중이지 않지만 **ACTPU**가 수신되지 않았습니다.

WLUAPUACTIVE
ACTPU가 수신되었습니다.

WLUAPUREACTIVATED
PU가 재활성화되었습니다.

WLUAUINACTIVE
호스트에 대한 링크가 활동중이며 **ACTPU**가 수신되었지만 **ACTLU**가 수신되지 않았습니다.

WLUALUACTIVE
LU가 활동중입니다.

WLUAUNKNOWN
세션이 알 수 없는 상태에 있습니다. (내부 오류입니다.)

WLUASIDINVALID
지정된 SID가 **RUI**에 의해 알려진 사항과 일치하지 않습니다.

WLUASIDZERO

hStatusHandle 매 3 *Previous*는 FALSE이지만
*dwSid*는 0입니다.

WLUAGLOBALHANDL

dwSid 매 3 변수는 메세지는 통지됩니다.
(이 M 은 정상적인

사용시 주의사항

이 함수는 상태 변 f 의
트 핸들 중 하나를 사 k

트 핸들 중 하나를 사용해야 합니다. 노

SG_CORRELATOR,FALSE)28))

세션의 현재 상태를 조회하b 위해 이 함수를 사k하려면 이벤트나 창 핸들 사k이 필d하지 J습니다. 대신 다음 명령을 사k하십시@.

```
Status = WinRUIGetLastInitStatus(Sid,NULL,0,0,FALSE);
```

주: **WinRUIGetLastInitStatus**는 통신 서버 SNA API 클라이p트!서 지x되지 J습니다.

WinRUIStartup()

n 플리케이션을 사k ! 능하T 하) RUI API의 필d 한 버전을 지정하m API의 세부사항을 K색하T 합니다.

구문

```
int WINAPI WinRUIStartup (WORD wVersionRequired,
                          LPWLUADATA* luadata);
```

매 3 변수

설명

wVersionRequired

필d 한 RUI API 지x 의 버전을 지정합니다. 상위 바이트는 소 버전(3 정) 번호를 지정하며 하위 바이트는 주 버전 번호를 지정합니다.

luadata

RUI 8 현 버전을 리턴합니다.

리턴 값

리턴 * 은 n 플리케이션이 등록되z 는지M RUI API ! 지정 버전 번호를 지x 할 수 있는지) 부를 지정합니다. * 이 0이면 등록됐으며 지정 버전이 지x 될 수 있습니다. W렇지 J으면 리턴 * 이 다음 * 중 하나입니다.

WLUAVERNOTSUPPORTED

등록된 RUI API 지x 버전이 이 특정 RUI API ! 의해 제x 되지 J습니다.

WLUAINVALID

d 청된 버전을 판별할 수 x 습니다.

사용시 주의사항

이 호출은 API의 장래 버전z 의 호환성을 돕b 위해 시도됐습니다. 현재 버전은 1.0입니다. **| 련 참조:** 210페이지의 『WinRUICleanup()』.

이 # 트리 포인트는 통신 서버 Windows 3.1 클라이p 트 ! 서 지x 되지 J습니다.

GetLuaReturnCode()

VCB의 1차 및 2차 리턴 코드를 인쇄 ! 능한 문자 - 로 변환합니다. 이 함수는 LUA n플리케이션이 사 k할 @류 문자 - 의 표준 세트를 제 x합니다.

구문

```
int WINAPI GetLuaReturnCode (lua_common* vcb,
                             UINT buffer_length,
                             unsigned char* buffer_addr);
```

매 3 변수

설명

vcb 제 x 된 매 3 변수로 명령 n 제 n 블럭의 주소를 지정합니다.

buffer_length

제 x 된 매 3 변수로 buffer_addr ! 지정된 버퍼 f 이(바이트)를 지정합니다. G 장 f 이는 256입니다.

buffer_addr

제 x /리턴된 매 3 변수로, 형식화된 널(null) 종료 문자 - 을 보유하는 버퍼의 주소 M 지정 버퍼 ! 있는 문자 - f 이를 지정합니다.

리턴 값

0x20000001

매 3 변수 ! 유효하지 J으며 지정된 명령 n 제 n 블럭 ! 서 함수를 읽을 수 x E나 지정 버퍼 ! 5 수 x습니다.

0x20000002

지정된 버퍼 ! 너무 작습니다.

사용시 주의사항

buffer_addr ! 리턴된 설명 @류 문자 - 은 새 행 문자(**/n**)로 종료하지 J습니다.

호출 예제

다음 9 제는 **WINRUI32.DLL**을 호출하는 방식을 보) 줍니다. 이 DLL ! 대한 헤더 파일은 **WINLUA.H**입니다. 이 9 제는 **RUI DLL**을 호출하) 프로 W 램 ! 서 **RUI** 명령 n 를 발행합니다.

```
#include "WINLUA.H"                    /* LUA C include file for
                                          the LUA Application. */
. . .


. . .

example()
{
LUA_VERB_RECORD    VerbRecord;         /* Declare VerbRecord as a verb
                                          control block using the
```

```
                                                    TYPEDEF in WINLUA.H */
. . .

WINRUI((LUA_VERB_RECORD *) &VerbRecord); /* Call the RUI API */
. . .
}
```

제13장 RUI 명령어

이 장!는 다음 정보! 들n 있습니다.

- LUA xk 제n 블럭 8조의 세부사항
- 모든 LUA 명령nM LUA 명령n 레코드! 대한 " LUA 명령n의 설명

다음 정보는 " LUA 명령nk으로 제x됩니다.

- 명령n의 목적.
- LUA! 제x되m LUA! 의해 리턴되는 매3변수. " 매3변수의 설명!는 해당 매3변수! 유효한 *! 대한 정보M 필d한 추! 정보! 들n 있습니다.
- 다른 명령nM의 상호작k.
- 명령n의 사k을 설명하는 추! 정보.

주: *9비*로 표시되는 매3변수는 항상 0으로 설정되n_ 합니다.

LUA 명령어 제어 블럭 형식

명령n 제n 블럭은 다음z 0이 8성됩니다.

- **lua_common**, 모든 명령n! 사k되며 219페이지의 『xk 명령n 헤더』!서 설명됩니다.
- **specific**, **RUI_BID** 명령n!만 사k됩니다. (224페이지의 『RUI_BID 데이타 8조』!서 설명합니다.)

8조는 다음z 0이 정의됩니다.

```
typedef struct lua_verb_record
{
 LUA_COMMON        common;                 /* The common verb header     */
 union
 {
   unsigned char   lua_peek_data[12];      /* field specific to RUI_BID  */
 }
} LUA_VERB_RECORD;
```

공용 명령어 헤더

퍼스널 통신 LUA는 일반 *xk 명령n 헤더*를 사k하) 모든 입력 및 전송 데이타를 전송합니다. 명령n 제n 블럭! 있는 필드는 다음z 0이 정의됩니다.

```
typedef struct lua_common
{
 unsigned short    lua_verb;           /* LUA_VERB_RUI               */
 unsigned short    lua_verb_length;    /* VCB length                 */
 unsigned short    lua_prim_rc;        /* primary return code        */
 unsigned long     lua_sec_rc;         /* secondary return code      */
 unsigned short    lua_opcode;         /* verb opcode                */
```

```
 unsigned long     lua_correlator;      /* verb correlator            */
 unsigned char     lua_luname[8];       /* local LU name              */
 unsigned short    lua_extension_list_offset;
                                        /* reserved                   */
 unsigned short    lua_cobol_offset;    /* reserved                   */
 unsigned long     lua_sid;             /* session ID                 */
 unsigned short    lua_max_length;      /* max buffer length          */
 unsigned short    lua_data_length;     /* actual data length         */
 unsigned char    *lua_data_ptr;        /* data pointer               */
 unsigned long     lua_post_handle;     /* post handle                */
 LUA_TH            lua_th;              /* TH structure               */
 unsigned char     lua_rh;              /* message RH                 */
 unsigned char     lua_flag1;           /* application message flag   */
 unsigned char     lua_message_type;    /* SNA message type           */
 unsigned char     lua_flag2;           /* LUA message flag           */
 unsigned char     lua_resv56[7];       /* reserved                   */
 unsigned char     lua_encr_decr_option;/* cryptography               */
} LUA_COMMON;

typedef struct lua_th
{
 unsigned char     reserv1;             /* reserved                   */
 unsigned char     daf;                 /* DAF                        */
 unsigned char     oaf;                 /* OAF                        */
 unsigned char     snf[2];              /* SNF                        */
} LUA_TH;
```

lua_verb

LUA 명령n로 식별합니다. 항상 **LUA_VERB_RUI**로 설정합니다.

lua_verb_length

명령n 제n 블럭의 f이.

lua_prim_rc

LUA! 설정한 1차 리턴 코드.

lua_sec_rc

LUA! 설정한 2차 리턴 코드.

lua_opcode

수행되는 **LUA** 명령n를 식별하는 명령n 작w 코드.

lua_correlator

n플리케이션!서 다른 프로세스M 이 명령n를 ,|시키b 위해 사k할 수 있는 4바이트 상|자. LUA는 이 매3변수를 사k하지 J습니다.

lua_luname

LU 세션!서 사k하는 로컬 LU명(ASCII). 이M은 LU명이E나 LU 풀명입니다. 자세한 내k은 231페이지의 『RUI_INIT』를 참조하십시@.

lua_sid

이 명령n! 수행되는 LUA 세션의 세션 ID.

lua_max_length

데이타를 수신하b 위해 사k되는 버퍼의 f이.

lua_data_length
송신되는 데이타의 f이 또는 수신하는 데이타의 실제 f이.

lua_data_ptr
송신되는 데이타! 대한 포인터 또는 데이타를 수신하는 데이타 버퍼.

lua_th.flags
전송 헤더!서 설정되는 플래W를 지정합니다. (자세한 내k은 *Systems Network Architecture: Formats*을 참조하십시@.) 하나 이상의 다음 *들이 함께 논리합(OR)이 될 수 있습니다.

LUA_FID
형식 식별 유형 2

LUA_MPF
세W멘트 매핑 필드

LUA_BBIU
BIU 시작

LUA_EBIU
BIU 종료

LUA_ODAI
OAF-DAF 지정자 표시b

LUA_EFI
d^ 흐름 표시b

lua_th.daf
DAF(목적지 주소 필드).

lua_th.oaf
OAF(Yx지 주소 필드).

lua_th.snf
순서 번호 필드.

lua_rh
송신되E나 수신되는 d청/응답 헤더(RH)를 지정합니다. (자세한 내k은 *System Network Architecture: Formats*를 참조하십시@.) 이M은 하나 이상의 다음 *들이 함께 논리합(OR)이 될 수 있습니다.

LUA_RRI
d청 응답 표시b

LUA_RH_FMD
RU 범주. FMI 데이타 세W멘트

LUA_RH_NC
RU 범주. 네트v크 제n

LUA_RH_DFC
RU 범주. 데이타 흐름 제n

LUA_RH_SC
RU 범주. 세션 제n

LUA_FI
형식 표시b

LUA_SDI
표시b ! 포함된 (지 데이타

LUA_BCI
시작 체인 표시b

LUA_ECI
종료 체인 표시b

LUA_DR1I
절대 응답 1 표시b

LUA_DR2I
절대 응답 2 표시b

LUA_RI
9\ 응답 표시b(d청k) 또는 응답 유형 표시b(응답k)

LUA_QRI
대b행렬 응답 표시b

LUA_PI
페이싱 표시b

LUA_BBI
시작 브래킷 표시b

LUA_EBI
종료 브래킷 표시b

LUA_CDI
변f 방향 표시b

LUA_CSI
코드 선택 표시b

LUA_EDI
0호화 데이타 표시b

LUA_PDI
채r 데이타 표시b

lua_flag1
n플리케이션이 제x하는 메세지! 대한 플래W를 지정합니다. (자세한 내k은 *System Network Architecture: Formats*를 참조하십시@.) 플래W는 하나 이상의 다음 *들이 함께 논리합(OR)이 될 수 있습니다.

LUA_BID_ENABLE
송수신 사k!능 표시b

LUA_NOWAIT
데이타 플래W를 b다리지 J음

LUA_SSCP_EXP
SSCP d^ 흐름

LUA_SSCP_NORM
SSCP 정상 흐름

LUA_LU_EXP
LU d^ 흐름

LUA_LU_NORM
LU 정상 흐름

LUA_CLOSE_ABEND

LUA_RESERVE1

lua_message_type
RUI_READ 명령n! 수신하는 (또는 **RUI_BID** 명령n! 표시된) SNA 메세지의 유형. 다음 * 중 하나일 수 있습니다.

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIND
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_CLEAR
LUA_MESSAGE_TYPE_CRV
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RQR
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SHUTD
LUA_MESSAGE_TYPE_SIGNAL
LUA_MESSAGE_TYPE_SDT
LUA_MESSAGE_TYPE_STSN
LUA_MESSAGE_TYPE_UNBIND

lua_flag2

LUA ! 리턴하는 메세지 ! 대한 플래 W를 지정합니다. (자세한 내 k은 *System Network Architecture: Formats*를 참조하십시 @ .) 플래 W는 하나 이상의 다음 * 들이 함께 논리합(OR)이 될 수 있습니다.

LUA_BID_ENABLE

송수신 사 k ! 능 표시 b

LUA_ASYNC

비동 b 명령 n O 료 플래 W

LUA_SSCP_EXP

SSCP d ^ 흐름

LUA_SSCP_NORM

SSCP 정상 흐름

LUA_LU_EXP

LU d ^ 흐름

LUA_LU_NORM

LU 정상 흐름

lua_encr_decr_option

O 호화 I 션.

RUI_BID 데이타 구조

다음 매 3 변수는 **RUI_BID** 명령 n ! 만 제 x 되며 m 유한 M 입니다.

lua_peek_data

읽히 b 를 b 다리는 최대 12바이트의 데이타.

RUI_BID

RUI_BID 명령n는 수신된 메세지! 읽히b를 b다리는 시b를 나타내b 위해 n플리케이션!서 사k합니다. **RUI_READ** 명령n를 수행하b 전! n플리케이션이 사k할 수 있는 자료! 있는지 판별할 수 있T 합니다. 메세지! 사k!능하면 **RUI_BID** 명령n는 수신한 메세지 흐름, 메세지 유형, 메세지의 THM RH W리m 최대 12바이트의 메세지 데이타! 대한 세부사항을 리턴합니다. **RUI_BID** 및 **RUI_READ**#의 중d한 차이점은 **RUI_BID**! n플리케이션이 입력 메세지 대b행렬로부터 데이타를 제E하지 Jm 점K할 수 있도록 남\두n 나중 단h! W세스할 수 있습니다. **RUI_READ**는 대b행렬로부터 메세지를 제E하므로 n플리케이션이 데이타를 읽m나면 반드시 프로세스해_ 합니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령n 레코드의 바이트 f이.

```
sizeof(struct LUA_COMMON) + 12로 설정하십시@.
```

lua_opcode
LUA_OPCODE_RUI_BID

lua_correlator
선택적. n플리케이션 내!서 다른 프로세스M 이 명령n를 상|시키b 위해 사k할 수 있는 4바이트 *. LUA는 이 정보를 사k하E나 변f하지 J습니다.

lua_luname
세션이 사k하는 로컬 LU의 ASCII 이름. 활동중 LUA 세션의 LU명z 일치해_ 합니다.

이 매3변수는 **lua_sid** 매3변수! 0인 fl!만 필d합니다. 세션ID! **lua_sid**!서 제x되면 LUA는 이 매3변수를 사k하지 J습니다.

이 매3변수는 8바이트 f이)_ 하며 이름이 8자보다 짧은 fl 0x20 x백으로 @른쪽을 채s니다.

lua_sid
세션의 세션 ID. 이M은 이전 **RUI_INIT** 명령n!서 리턴되는 세션IDM 일치해_ 합니다. 이 매3변수는 선택적입니다. 세션 ID를 지정하지 J은 fl, **lua_luname** 매3변수! 세션! 대한 LU명을 지정해_ 합니다.

RUI_BID

lua_post_handle
비동b 명령n의 0료를 통지하b 위해 사k되는 4바이트 핸들입니다.

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2
명령n! 비동b적으로 0료된 fl!만 LUA_ASYNC로 설정됩니다.

리턴된 다른 매3변수는 명령n! 0료됐는지)부! 좌l되며 다음 절을 참조하십시@.

명령n! 0료되면 다음 매3변수! 리턴됩니다.

lua_prim_rc
LUA_OK

lua_sid
이 명령n를 수행할 때 n플리케이션!서 세션 ID를 지정하지 Jm **lua_luname** 매3변수를 지정한 fl, LUA는 세션 ID를 제x합니다.

lua_max_length
수신된 메세지!서의 데이타 바이트 수.

lua_data_length
lua_peek_data 매3변수!서 리턴되는 데이타의 0 - 12바이트 수.

lua_th 수신된 메세지의 전송 헤더(TH)로부터의 정보.

lua_rh
수신된 메세지의 d청/응답 헤더(RH)로부터의 정보.

lua_message_type
다음 * 중 하나인 수신된 메세지의 메세지 유형.

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIND
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_CLEAR
LUA_MESSAGE_TYPE_CRV
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP

LUA_MESSAGE_TYPE_QC

LUA_MESSAGE_TYPE_QEC

LUA_MESSAGE_TYPE_RELQ

LUA_MESSAGE_TYPE_RTR

LUA_MESSAGE_TYPE_SBI

LUA_MESSAGE_TYPE_SHUTD

LUA_MESSAGE_TYPE_SIGNAL

LUA_MESSAGE_TYPE_SDT

LUA_MESSAGE_TYPE_STSN

LUA_MESSAGE_TYPE_UNBIND

lua_flag2

데이타! 수신되는 메세지 흐름을 나타내b 위해 다음 플래W 중 하나! 설정됩니다.

LUA_SSCP_EXP
SSCP d^ 흐름

LUA_LU_EXP
LU d^ 흐름

LUA_SSCP_NORM
SSCP 정상 흐름

LUA_LU_NORM
LU 정상 흐름

lua_peek_data

메세지 데이타의 첫번째 12바이트(또는 메세지 데이타! 12바이트보다 짧은 fl 모든 메세지 데이타).

다음 리턴 코드는 명령n! 다른 명령n! 의해 취소되n 0료되지 JR음을 나타냅니다.

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

이 명령n! 보류중인 동H **RUI_TERM** 명령n! 수행되z습니다.

다음 리턴 코드는 제x된 매3변수! @류! 있n 명령n! 0료되지 JR음을 나타냅니다.

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

!능한 *:

LUA_BID_ALREADY_ENABLED
이전 **RUI_BID** 명령ｎ！ 미처리되ｎ **RUI_BID** 명령ｎ！ Ｅ부됐습니다. 한번！ @직 하나의 **RUI_BID**만이 미처리될 수 있습니다.

LUA_RESERVED_FIELD_NOT_ZERO
명령ｎ 레코드의 ９｀ 필드나 이 명령ｎ！ 사ｋ하지 Ｊ은 매 ３변수！ 0이 Ｆ닌 *으로 설정되ｚ습니다.

LUA_VERB_LENGTH_INVALID
lua_verb_length 매３변수의 *은 이 명령ｎ！ 필ｄ한 명령ｎ 레코드의 ｆ이보다 적습니다.

다음 리턴 코드는 유효하지 Ｊ은 세션 상태！서 명령ｎ！ 수행됐음을 나타냅니다.

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

RUI_INIT 명령ｎ！ 이 세션！서 Ｏ료되지ＪＲＥ나 세션 중단이 발생했습니다.

다음 리턴 코드는 제ｘ된 명령ｎ 레코드！ 유효하지만 명령ｎ！ Ｏ료되지ＪＲ음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
LUA_INVALID_PROCESS

이 명령ｎ를 수행한 ｎ플리케이션 인스턴스！ 이 세션！ 대한 **RUI_INIT** 명령ｎ를 수행한 인스턴스Ｍ 동일하지 Ｊ습니다.

다음 리턴 코드는 퍼스널 통신 및 통신 서버！ 호스트！서 수신한 데이타！서 @류를 Ｋ출했음을 나타냅니다. 수신 메세지를 **RUI_READ** 명령ｎ로 ｎ플리케이션！ 전달하는 대신, 퍼스널 통신 및 통신 서버는 메세지를 삭제하며(메세지！ 체인！ 있는 ｆｌ 나머지 체인을 삭제하며) 부정 응답을 호스트로 송신합니다. LUA는 차후의 **RUI_READ**나 부정 응답이 송신된 **RUI_BID** 명령ｎ로 ｎ플리케이션！ Ｋ립니다.

lua_prim_rc
LUA_NEGATIVE_RSP

lua_sec_rc
2차 리턴 코드！는 부정 응답시 호스트！ 송신되는 (지 코드！ 들ｎ 있습니다. 리턴될 수 있는 (지 코드를 해석하는 정보는 175페이지의 『SNA ｈ층』을 참조하십시@.

0의 2차 리턴 코드는 체인의 중# ! 서 메세지 ! 대한 부정 응답의 이전 **RUI_WRITE** ! 따라 퍼스널 통신이 이 체인으로부터 모든 메세지를 수신하) 삭제했음을 나타냅니다.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령 n ! O 료되지 J R 음을 나타냅니다.

lua_prim_rc

LUA_SESSION_FAILURE

세션이 종료됐습니다.

lua_sec_rc

! 능한 *:

LUA_LU_COMPONENT_DISCONNECTED

통신 링크나 호스트 LU M의 문제 때문 ! LUA 세션이 실패했습니다.

LUA_RUI_LOGIC_ERROR

리턴 코드는 다음 중 하나를 나타냅니다.

- 호스트 시스템이 SNA 프로토콜을 위반했습니다.
- 내부 @ 류 ! LUA 내 ! 서 K 출됐습니다.

추적 활성화로 문제점을 재생성하도록 시도하 m 호스트 ! C 바른 데이타를 송신하는지 점 K 하십시 @.

lua_prim_rc

LUA_INVALID_VERB

lua_verb 매 3 변수나 **lua_opcode** 매 3 변수 중 하나 ! 유효하지 J 습니다. 명령 n ! 실행되지 J R 습니다.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

자 x 부족 z 0 은 n 5 체제 @ 류 ! 발생했습니다.

lua_sec_rc

이 * 은 n 5 체제 리턴 코드입니다. 이 리턴 코드의 의미 ! 대해서는 n 5 체제 문서를 점 K 하십시 @.

주석

이 명령 n 를 수행하 b 전 ! **RUI_INIT** 명령 n ! O 료되 n _ 합니다.

한번 ! @ 직 하나의 **RUI_BID**만이 미처리될 수 있습니다. **RUI_BID** 명령 n ! O 료된 후 ! 차후의 **RUI_READ** 명령 n ! 서 **lua_flag1**을 LUA_BID_ENABLE로 설정하) 이 명령 n 를 재수행할 수 있습니다. 명령 n ! 이런 방식으로 재수행되는 f l, **RUI_BID** 명령 n 레코드 M | 련된 b o 5 * 을 해제하 E 나 수정하지 말 F _ 합니다.

RUI_BID

RUI_READ 및 **RUI_BID** 둘다 미처리시 호스트!서 메세지! 도달하면 **RUI_READ**! 0료되며 **RUI_BID**! 진행 상태로 남습니다.

사용시 주의사항

도달하는 " 메세지는 @직 한번만 송수신이 d8됩니다. 데이타! 특정 세션 흐름!서 대b중임을 **RUI_BID** 명령n! 나타내m 나면 n플리케이션은 **RUI_READ** 명령n를 수행하) 데이타를 수신해_ 합니다. 차후의 모든 **RUI_BID**는 송수신을 d8한 메세지! **RUI_READ** 명령n를 발행하) 수k될 때n지 해당 세션 흐름!서 도착한 데이타를 b록하지 J습니다.

일반적으로 이 명령n!서 리턴한 **lua_data_length** 매3변수는 대b 메세지의 총 데이타 f이! F닌 **lua_peek_data**! 있는 데이타의 f이만을 나타냅니다(12보다 적은 *이 리턴되는 f l 제\). **lua_max_length** 매3변수는 수신 메세지!서 바이트 수를 리턴합니다. n플리케이션은 데이타를 수k하는 **RUI_READ** 명령n의 데이타 f이! 메세지를 포함하b! 충분한지 확인해_ 합니다.

RUI_INIT

RUI_INIT 명령n는 주n진 LUA LU! 대한 SSCP-LU 세션을 설정합니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령n 레코드의 바이트 f이. sizeof(struct LU_COMMON)로 설정하십시@.

lua_opcode
LUA_OPCODE_RUI_INIT

lua_correlator
선택적. n플리케이션 내!서 다른 프로세스M 이 명령n를 ,|시키b 위해 사k할 수 있는 4바이트 *. LUA는 이 정보를 사k하E나 변f하지 J습니다.

lua_luname
세션을 시작하려는 로컬 LU나 LU 풀의 ASCII 이름. 이M은 8성된 LUA LU명이나 LU 풀명z 일치해_ 합니다. 퍼스널 통신 및 통신 서버!서의 n플리케이션의 fl, 이름은 다음z 0이 사k됩니다.

이름이 풀! x는 LU명인 fl, 퍼스널 통신 및 통신 서버는 이 LU를 사k하) 세션을 시작하려m 시도합니다.

이름이 LU 풀명이E나 풀 내의 LU명인 fl, 퍼스널 통신 및 통신 서버는 풀!서 사k!능한 첫번째 LU를 사k하) 세션을 시작하려m 시도합니다. 이 필드는 필d한 fl 후미 x백(0x20)으로 채v지는 8바이트 ASCC 문자-입니다.

SNA API 클라이p트! 있는 n플리케이션의 fl, 8성된 LUA 세션 이름z 일치해_ 합니다.

다음 정보는 통신 서버 Windows 95 및 Windows NT SNA API 클라이p트!만 적k됩니다.

" 사k자! 대한 디폴트 LUA 세션명은 INI 8성이나 LDAP 중 하나의 해당 8성 유틸리티를 사k하) 지정될 수 있습니다.

3270 !뮬레이터M 0은 LUA 프로W램은 직접 하나의 세션 이름을 지정하b보다는 디폴트 LUA 세션명을 사k하도록 선택할 수 있습니다. LUA 프로W램 **lua_name** 필드를 2진 0이나 ASCII x백으로 설정한 **RUI_INIT** 명령n를 수행할 때 RUI API는 8성된 디폴트 LUA 세션명을 사k합니다.

lua_post_handle
비동b 명령n의 0료를 통지하b 위해 사k되는 4바이트 핸들입니다.

lua_flag1
n플리케이션은 이M을 LUA_ASYNC_STATUS로 설정하) **RUI_INIT** 명령n를 프로세스할 때 퍼스널 통신 및 통신 서버!서 RUI_INIT_STATUS 표시를 수신해_ 합니다. (RUI_INIT_STATUS 메세지는 240페이지의 『RUI_INIT_STATUS』!서 설명됩니다.)

lua_encr_decr_option
세션 레벨 0호화 I션. 퍼스널 통신 및 통신 서버는 다음z 0은 두!지 *을 수k합니다.

0 세션 레벨 0호화! 사k되지 J습니다.

128 0호화 및 0호해독이
n플리케이션 프로W램! 의해 수행됩니다.

다른 모든 *은 리턴 코드 LUA_ENCR_DECR_LOAD_ERROR의 az를 !져I니다. (사k자 정의 0호화 및 0호해독 루틴을 나타내는 1-127의 *은 퍼스널 통신 및 통신 서버! F닌 OS/2 Communications Manager/2의 LUA 8현으로 지x됩니다.)

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2
명령n! 비동b적으로 0료된 fl!만 이M이 LUA_ASYNC로 설정됩니다.

주: RUI_INIT는 LUA_PARAMETER_CHECKM 0은 @류를 리턴하지 J는 한 항상 비동b적으로 0료합니다.

리턴된 다른 매3변수는 명령n! 0료되z는지의)부! 좌l되며 다음 절을 참조하십시@.

명령n! 실행되면 LUA는 다음 매3변수를 리턴합니다.

lua_prim_rc
LUA_OK

lua_sid
새 세션! 대한 세션 ID. 이M은 이 세션을 식별하b 위한 다음의 명령n로 사k될 수 있습니다.

lua_luname
세션이 사k하는 로컬 LU의 이름. 이M은 n플리케이션이 LU 풀을 지정한 fl 필d하며 풀 내의 LU! 사k됐음을 F는 M이 필d합니다.

다음 리턴 코드는 명령n ! 다른 명령n ! 의해 취소되n 0료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

RUI_TERM 명령n !
RUI_INIT의 0료 전 ! 수행됐습니다.

다음 리턴 코드는 제x된 매3변수 ! @류 ! 있n 명령n ! 0료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
!능한 *:

LUA_INVALID_LUNAME
lua_luname 매3변수를 찾을 수 x습니다. LU명이나 LU 풀명이 *퍼스널 통신 및 통신 서버 System Management Programming* API !서 정의되z는지 점K하십시@.

LUA_RESERVED_FIELD_NOT_ZERO
명령n 레코드의 9 ` 필드나 이 명령n ! 사k하지 J은 매3변수 ! 0이 F닌 *으로 설정됐습니다.

LUA_VERB_LENGTH_INVALID
lua_verb_length 매3변수의 *은 이 명령n ! 필d한 명령n 레코드의 f이보다 적습니다.

다음 리턴 코드는 유효하지 J은 세션 상태 !서 명령n ! 수행됐음을 나타냅니다.

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT

이 n플리케이션이 사k하m 있는 LU명이나 LU 풀명을 지정한 **lua_luname** 매3변수(또는 이 n플리케이션이 진행중인 **RUI_INIT** 명령n를 !지m 있는 f l).

다음 리턴 코드는 제x된 명령n 레코드 ! 유효하지만 명령n ! 0료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

! 능한 *:

LUA_COMMAND_COUNT_ERROR

풀 내의 모든 LU ! 사 k 중이지만 LU 풀명이나 풀 내의 LU 명을 지정하는 명령 n .

LUA_ENCR_DECR_LOAD_ERROR

0 또는 128이 F 닌 **lua_encr_decr_option** ! 대한 * 을 지정하는 명령 n .

LUA_INVALID_PROCESS

다른 매 3 변수 ! 사 k 중인 **lua_luname** 매 3 변수 ! 지정하는 LU.

LUA_LINK_NOT_STARTED

호스트 ! 대한 링크 ! 시작되지 J R 습니다.

lua_sec_rc ! 대한 다음 * 은 퍼스널 통신 및 통신 서버 (지 코드이며 **lua_prim_rc** ! LUA_UNSUCCESSFUL인 f l 리턴될 수 있습니다. (이들 * 은 LU의 상태를 반 5 합니다.)

X10020000

ACTPU ! 수신되지 J R 습니다. **RUI_INIT**는 PU를 활성화시키지 J 습니다.

X10100000

ACTPU ! 수신되지 J R 습니다. **RUI_INIT**는 PU를 활성화시킵니다.

X10110000

ACTPU ! 수신되 z 습니다. **ACTLU** ! 수신되지 J R 습니다. SSCP는 자체 정의 종속 LU(SSDLU)를 지 x 하지 J 습니다. **RUI_INIT**는 LU를 활성화시킵니다.

X10120000

ACTPU ! 수신됐습니다. **ACTLU** ! 수신되지 J R 습니다. SSCP는 SSDLU를 지 x 합니다. **RUI_INIT**는 LU를 활성화시킵니다.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령 n ! O 료되지 J R 음을 나타냅니다.

lua_prim_rc

LUA_SESSION_FAILURE

세션이 종료됐습니다.

lua_sec_rc

LUA_LU_COMPONENT_DISCONNECTED

통신 링크나 호스트 LU M 의 문제 때문 ! LUA 세션이 실패했습니다.

lua_prim_rc

LUA_INVALID_VERB

lua_verb 매 3 변수나 **lua_opcode** 매 3 변수 중 하나 ! 유효하지 J 습니다. 명령 n ! 실행되지 J R 습니다.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

자 x 부족 z 0 은 n 5 체제 @ 류 ! 발생했습니다.

lua_sec_rc

이 * 은 n 5 체제 리턴 코드입니다. 이 리턴 코드의 의미 ! 대해서는 n 5 체제 문서를 점 K 하십시 @.

주석

이 명령 n 는 세션 ! 대해 수행된 첫번째 LUA 명령 n) _ 합니다. 이 명령 n ! 0 료될 때 n 지 이 세션 ! 대해 수행될 수 있는 다른 유일한 LUA 명령 n 는 **RUI_TERM**(보류 중인 **RUI_INIT**를 종료합니다)입니다. 이 세션 ! 서 수행하는 다른 모든 명령 n 는 이 명령 n 로부터 다음 매 3 변수 중 하나를 사 k 하) 세션을 식별해 _ 합니다.

- 세션 ID는 **lua_sid** 매 3 변수 ! 서 n 플리케이션으로 리턴됩니다.
- LU명은 **lua_luname** 매 3 변수 ! 있는 n 플리케이션 ! 의해 제 x 됩니다.

사용시 주의사항

RUI_INIT 명령 n 는 **ACTLU** ! 호스트 ! 서 수신된 후 ! 0 료됩니다. 필 d 하다면 명령 n ! 무 b 한 b 다립니다. **ACTLU** ! **RUI_INIT** 명령 n 이전 ! 수신된 f l, LUA는 LU ! 사 k 준비되 z 음을 K 리 b 위해 호스트 ! **NOTIFY** 를 송신합니다.

주: **ACTLU** 또는 **NOTIFY** 중 n 느 M 도 LUA n 플리케이션 ! 보이지 J 습니다.

RUI_INIT 명령 n 중 하나 ! 0 료되면 이 세션은 세션이 시작된 LU를 사 k 합니다. 다른 LUA 세션은 **RUI_TERM** 명령 n ! 수행될 때 n 지 LU를 사 k 할 수 x 습니다.

RUI_PURGE

RUI_PURGE 명령n는 이전 **RUI_READ**를 취소합니다. **lua_flag1**을 LUA_NO WAIT(즉시 리턴 I션)로 설정하지 Jm **RUI_READ**! 송신된 fl 무한히 b다릴 수 있으며 n떠한 데이타도 지정 흐름!서 사k할 수 xm **RUI_PURGE**는 대b 명령n를 리턴하도록 -제합니다(1차 리턴 코드 CANCELLED를 사k하)).

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령n 레코드의 바이트 f이. sizeof(struct LUA_COMMON)로 설정하십시@.

lua_opcode
LUA_OPCODE_RUI_PURGE

lua_correlator
선택적. n플리케이션 내!서 다른 프로세스M 이 명령n를 ,|시키b 위해 사k할 수 있는 4바이트 *. LUA는 이 정보를 사k하E나 변f하지 J습니다.

lua_luname
세션이 사k하는 로컬 LU의 ASCII 이름. 이M은 활동중 LUA 세션의 LU명z 일치해_ 합니다.

이 매3변수는 **lua_sid** 매3변수! 0인 fl!만 필d합니다. 세션 ID! **lua_sid**!서 제x되면 LUA는 이 매3변수를 사k하지 J습니다.

이 매3변수는 8바이트 f이)_ 하며 이름이 8자보다 짧은 fl 0x20 x백으로 @른쪽을 채s니다.

lua_sid
세션의 세션 ID. 이M은 이전 **RUI_INIT** 명령n!서 리턴되는 세션 IDM 일치해_ 합니다.

이 매3변수는 선택적입니다. 세션 ID를 지정하지 J은 fl, **lua_luname** 매3변수! 세션! 대한 LU명을 지정해_ 합니다.

lua_data_ptr
제E되는 **RUI_READ** LUA_VERB_RECORD! 대한 포인터.

lua_post_handle
이M은 비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다.

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2
명령n ! 비동b적으로 0료된 f l !만 LUA_ASYNC로 설정됩니다.

리턴된 다른 매3변수는 명령n ! 0료됐는지의)부 ! 좌l되며 다음 절을 참조하십시@.

명령n ! 0료되면 다음 매3변수 ! 리턴됩니다.

lua_prim_rc
LUA_OK

lua_sid
이 명령n를 수행할 때 n플리케이션 !서 세션 ID를 지정하지 J m **lua_luname** 매3변수를 지정한 f l, LUA는 세션 ID를 제x합니다.

다음 리턴 코드는 명령n ! 다른 명령n ! 의해 취소되n 0료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

이 명령n ! 보류중인 동H **RUI_TERM** 명령n ! 수행됐습니다.

다음 리턴 코드는 제x된 매3변수 ! @류 ! 있n 명령n ! 0료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
!능한 *:

LUA_BAD_DATA_PTR
lua_data_ptr 매3변수 ! 0으로 설정됐습니다.

LUA_RESERVED_FIELD_NOT_ZERO
명령n 레코드의 9 ` 필드나 이 명령n ! 사k하지 J은 매3변수 ! 0이 F닌 *으로 설정되z습니다.

LUA_VERB_LENGTH_INVALID
lua_verb_length 매3변수의 *은 이 명령n ! 필d한 명령n 레코드의 f이보다 적습니다.

다음 리턴 코드는 유효하지 J은 세션 상태 !서 명령n ! 수행됐음을 나타냅니다.

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
!능한 *:

LUA_SEC_RC_OK
이전 **RUI_PURGE** 명령n ! 이 세션!서)전히 진행중입니다.

LUA_NO_RUI_SESSION
RUI_INIT 명령n ! 이 세션!서 O료되지 JRE나 세션 중단이 발생했습니다.

다음 리턴 코드는 제x된 명령n 레코드! 유효하지만 명령n ! O료되지 JR음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
!능한 *:

LUA_INVALID_PROCESS
이 명령n를 수행한 n플리케이션 인스턴스! 이 세션! 대한 **RUI_INIT** 명령n를 수행한 인스턴스M 동일하지 J습니다.

LUA_NO_READ_TO_PURGE
lua_data_ptr 매3변수! **RUI_READ** LUA_VERB_RECORD! 대한 포인터를 포함하지 JRE나 **RUI_READ** 명령n ! **RUI_PURGE** 명령n의 수행 이전! O료됐습니다.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령n ! O료되지 JR음을 나타냅니다.

lua_prim_rc
LUA_SESSION_FAILURE

세션이 종료되z습니다.

lua_sec_rc
!능한 *:

LUA_LU_COMPONENT_DISCONNECTED
통신 링크나 호스트 LUM의 문제 때문! LUA 세션이 실패했습니다.

LUA_RUI_LOGIC_ERROR
리턴 코드는 다음 중 하나를 나타냅니다.

- 호스트 시스템이 SNA 프로토콜을 위반했습니다.
- 내부 @류! LUA 내!서 K출됐습니다.

추적 활성화로 문제점을 재생성하도록 시도하m 호스트! C 바른 데이타를 송신하는지 점K하십시@.

lua_prim_rc

LUA_INVALID_VERB

lua_verb 매3변수나 **lua_opcode** 매3변수 중 하나! 유효하지 J습니다. 명령n! 실행되지 JR습니다.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

자x 부족z 0은 n5체제 @류! 발생했습니다.

lua_sec_rc

이 *은 n5체제 리턴 코드입니다. 이 리턴 코드의 의미! 대해서는 n5체제 문서를 점K하십시@.

주석

이 명령n는 **RUI_READ**! 수행되zm O료! 보류중인 fl(즉, 1차 리턴 코드! IN_PROGRESS인 fl)!만 사k될 수 있습니다. 이 명령n는 다른 **RUI_PURGE**! 이 세션!서 진행중인 동H 수행될 수 x습니다.

RUI_INIT_STATUS

n플리케이션은 **RUI_INIT_STATUS** 표시를 수행할 수 x습니다. 퍼스널 통신은 **RUI_INIT** 프로세스 동H ! n플리케이션 ! 이 표시를 전송하) LU-SSCP 세션의 상태 ! 대한 정보를 제x합니다. **RUI_INIT_STATUS** 표시는 n플리케이션이 **RUI_INIT**를 수행할 때 상태 정보를 d청하는 f l ! 만 송신됩니다(231페이지의 『RUI_INIT』 참조).

제공되는 매개변수

다음 매3변수는 **RUI_INIT_STATUS** 표시 ! 서 설정됩니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령n 레코드의 바이트 f이(퍼스널 통신 및 통신 서버 ! sizeof(LUA_COMMON)로 설정).

lua_opcode
LUA_OPCODE_RUI_INIT_STATUS

lua_primary_rc
LU-SSCP 세션의 상태 ! 대한 정보 ! 들n 있습니다. ! 능한 *은 다음z 0습니다.

LUA_LINK_INACTIVE
호스트 ! 대한 링크 ! 활동중이지 J습니다.

LUA_PU_INACTIVE
ACTPU ! 수신되지 J R E나 **DACTPU** ! 수신되z습니다.

LUA_PU_ACTIVE
ACTPU ! SSCP ! 서 수신됐습니다.

LUA_PU_REACTIVATED
PU ! 활동중인 동H **ACTPU(COLD)** ! 수신됐습니다.

LUA_LU_INACTIVE
ACTLU ! E부되z E나 **DACTLU** ! 수신됐습니다.

LUA_UNKNOWN
데이타 링크 제n 링크 @류 때문 ! LU-SSCP 세션이 활동중이 F닙니다.

lua_correlator
퍼스널 통신 및 통신 서버는 이 매3변수k으로 **RUI_INIT** 명령n ! 지정된 *을 사k합니다.

lua_post_handle
이M은 비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다. 퍼스널 통신 및 통신 서버는 이 매3변수k으로 **RUI_INIT** 명령n ! 지정한 *을 사k합니다.

RUI_READ

RUI_READ 명령 n 는 호스트 ! 서 n 플리케이션의 LU로 송신한 자료나 상태 정보를 수신합니다. 자료를 읽을 특정 메세지 흐름(LU 정상, LU d ^, SSCP 정상 또는 SSCP d ^)을 지정하 E 나 둘 이상의 메세지 흐름을 지정할 수 있습니다. 다중 **RUI_READ** 명령 n 를 미처리하) 이들 중 n 떤 두 3 도 동일한 흐름을 지정하지 J 도록 준비할 수 있습니다.

제공되는 매개변수

n 플리케이션은 다음 매 3 변수를 제 x 합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령 n 레코드의 바이트 f 이. sizeof(struct LUA_COMMON)로 설정하십시 @.

lua_opcode
LUA_OPCODE_RUI_READ

lua_correlator
선택적. n 플리케이션 내 ! 서 다른 프로세스 M 이 명령 n 를 , | 시키 b 위해 사 k 할 수 있는 4바이트 *. LUA는 이 정보를 사 k 하 E 나 변 f 하지 J 습니다.

lua_luname
세션이 사 k 하는 로컬 LU의 ASCII 이름. 이 M 은 활동중인 LUA 세션의 LU명 z 일치해 _ 합니다.

이 매 3 변수는 **lua_sid** 매 3 변수 ! 0인 f l ! 만 필 d 합니다. 세션 ID ! **lua_sid** ! 서 제 x 되면 LUA는 이 매 3 변수를 사 k 하지 J 습니다.

이 매 3 변수는 8바이트 f 이) _ 하며 이름이 8자보다 짧은 f l 0x20 x 백으로 @ 른쪽을 채 s 니다.

lua_sid
세션의 세션 ID. 이 M 은 이전 **RUI_INIT** 명령 n ! 서 리턴되는 세션 ID M 일치해 _ 합니다.

이 매 3 변수는 선택적입니다. 세션 ID를 지정하지 J 은 f l, **lua_luname** 매 3 변수 ! 세션 ! 대한 LU명을 지정해 _ 합니다.

lua_max_length
데이타를 수신하 b 위해 제 x 되는 버퍼의 f 이(**lua_data_ptr** 참조).

lua_data_ptr
데이타를 수신하도록 제 x 되는 버퍼 ! 대한 포인터.

RUI_READ

lua_post_handle

비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다.

lua_flag1

플래W는 하나 이상의 다음 *들이 함께 논리합(OR)이 될 수 있습니다.

- 데이타! 읽혀질 수 있는지)부! 상|x이 **RUI_READ** 명령n! 즉시 리턴되T 하려면 LUA_NOWAIT를 설정하E나 명령n! 리턴하b 전! 데이타를 b다리T 하려면 이M을 설정하지 마십시@.
- !장 최Y의 **RUI_BID** 명령n를 재사k하려면 LUA_BID_ENABLE를 설정하E나(이전z 동일한 매3변수를 사k하) 다시 **RUI_BID**를 수행하는 Mz 동일) **RUI_BID**를 재사k하지 J으려면 이 명령n를 설정하지 마십시@.

  주: 이전 **RUI_BID** 재사k은 x래 할당된 LUA_VERB_RECORD를 재사k하며 LUA_VERB_RECORD! 해제 또는 수정되도록 허k하지 J습니다.
- n떤 메세지 흐름이 데이타를 읽는지 나타내려면 다음 플래W 중 둘 이상을 설정하십시@.

  LUA_SSCP_EXP
  SSCP d^ 흐름

  LUA_LU_EXP
  LU d^ 흐름

  LUA_SSCP_NORM
  SSCP 정상 흐름

  LUA_LU_NORM
  LU 정상 흐름

  둘 이상의 플래W! 설정된 fl, 사k!능한 !장 높은 l선순위 데이타! 리턴됩니다. l선순위 순서(!장 높은 l선순위!서 !장 낮은 l선순위로)는 다음z 0습니다.

  1. SSCP d^
  2. LU d^
  3. SSCP 정상
  4. LU 정상

  동일한 플래W! **lua_flag2**! 설정되n 데이타! 읽을 흐름을 나타냅니다(242페이지의 『리턴되는 매3변수』 참조).

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2

명령n ! 비동b적으로 O료되는 f l LUA_ASYNC ! 설정됩니다. (명령n ! 동b적으로 O료되는 f l 설정되지 J습니다.)

RUI_BID ! 재사k할 수 있는 f l LUA_BID_ENABLE이 설정됩니다. (W리m 재사k할 수 x는 f l 설정되지 J습니다.)

리턴된 다른 매3변수는 명령n ! O료되z는지)부 ! 좌l되며 다음 절을 참조하십시@.

명령n ! 실행되면 LUA는 다음 매3변수를 리턴합니다.

lua_prim_rc

LUA_OK

명령n ! O료되는 f l 다음 매3변수 ! 리턴됩니다. 제x되는 **lua_data_length** 매3변수 ! 너무 작F 명령n ! 잘린 데이타로 리턴하는 f l !도 다음 매3변수 ! 리턴됩니다.

lua_sid

이 명령n를 수행할 때 n플리케이션 !서 세션 ID를 지정하지 J m **lua_luname** 매3변수를 지정한 f l, LUA는 세션 ID를 제x합니다.

lua_data_length

수신된 데이타의 f이. LUA는 **lua_data_ptr** ! 지정한 버퍼 ! 데이타를 위치시킵니다.

lua_th 수신된 메세지의 전송 헤더(TH)로부터의 정보.

lua_rh

수신된 메세지의 d청/응답 헤더(RH)로부터의 정보.

lua_message_type

다음 * 중 하나인 수신된 메세지의 메세지 유형.

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIND
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_CLEAR
LUA_MESSAGE_TYPE_CRV
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC

LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SHUTD
LUA_MESSAGE_TYPE_SIGNAL
LUA_MESSAGE_TYPE_SDT
LUA_MESSAGE_TYPE_STSN
LUA_MESSAGE_TYPE_UNBIND

lua_flag2 매 3 변수
데이타 ! 수신되는 메세지 흐름을 나타내 b 위해 다음 * 중 하나로 설정됩니다.

LUA_SSCP_EXP
SSCP d ^ 흐름

LUA_LU_EXP
LU d ^ 흐름

LUA_SSCP_NORM
SSCP 정상 흐름

LUA_LU_NORM
LU 정상 흐름

다음 리턴 코드는 다른 명령 n 나 내부 @ 류로 명령 n ! 취소되 n O 료되지 J R 음을 나타냅니다.

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
! 능한 *:

LUA_PURGED
이 **RUI_READ** 명령 n 는 **RUI_PURGE** 명령 n ! 의해 취소됐습니다.

LUA_TERMINATED
이 명령 n ! 보류중인 동 H **RUI_TERM** 명령 n ! 수행됐습니다.

다음 리턴 코드는 제 x 된 매 3 변수 ! @ 류 ! 있 n 명령 n ! O 료되지 J R 음을 나타냅니다.

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
! 능한 *:

LUA_BAD_DATA_PTR
lua_data_ptr 매 3 변수 ! 틀린 * 이 들 n 있습니다.

LUA_BID_ALREADY_ENABLED
lua_flag1을 LUA_BID_ENABLE로 설정하) **RUI_BID** 명령 n 를 재사 k 할 수 있 T 했으나 이전 **RUI_BID** 명령 n !) 전히 진행중이 z 습니다.

LUA_DUPLICATE_READ_FLOW
lua_flag1의 흐름 플래 W ! **RUI_READ** 명령 n ! 미처리된 하나 이상의 세션 흐름을 지정했습니다. 한번 ! @ 직 하나의 **RUI_READ** ! " 세션 흐름 ! 서 대 b 할 수 있습니다.

LUA_INVALID_FLOW
lua_flag1 흐름 플래 W 중 n 느 M 도 설정되지 J R 습니다. 최소한 이들 플래 W 중 하나 ! 읽을 흐름을 나타내도록 설정되 n _ 합니다.

LUA_NO_PREVIOUS_BID_ENABLED
lua_flag1을 LUA_BID_ENABLE로 설정하) **RUI_BID** 명령 n 을 재사 k 할 수 있 T 했으나 사 k ! 능한 이전 **RUI_BID** 명령 n ! x 습니다. (자세한 내 k 은 247페이지의 『주석』을 참조하십시 @ .)

LUA_RESERVED_FIELD_NOT_ZERO
명령 n 레코드의 9 ` 필드나 이 명령 n ! 사 k 하지 J 은 매 3 변수 ! 0이 F 닌 * 으로 설정되 z 습니다.

LUA_VERB_LENGTH_INVALID
lua_verb_length 매 3 변수의 * 은 이 명령 n ! 필 d 한 명령 n 레코드 f 이보다 적습니다.

다음 리턴 코드는 유효하지 J 은 세션 상태 ! 서 명령 n ! 수행됐음을 나타냅니다.

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

RUI_INIT 명령 n ! 이 세션 ! 서 O 료되지 J R E 나
세션 중단이 발생했습니다.

다음의 1차 리턴 코드는 2차 리턴 코드 ! 의해 8 별할 수 있는 다음 두 ! 지 f l 중 하나를 나타냅니다.

- 퍼스널 통신 및 통신 서버 ! 호스트 ! 서 수신한 데이타 ! 서 @ 류를 K 출했습니다. 수신 메세지를 **RUI_READ** 명령 n 로 n 플리케이션 ! 전달하는 대신, 퍼스널 통신은 메세지를 삭제하며(메세지 ! 체인 ! 있는 f l 나머

지 체인을 삭제하며) 부정 응답을 호스트로 송신합니다. LUA는 차후의 **RUI_READ**나 부정 응답이 송신된 **RUI_BID** 명령ｎ로 ｎ플리케이션！ Ｋ립니다.

- LUA ｎ플리케이션은 이전！ 부정 응답을 체인 중#의 메세지로 송신했습니다. 퍼스널 통신은 이 체인！서 차후 메세지를 제Ｅ했으며 이제 체인으로부터의 모든 메세지！ 수신되ｍ 제Ｅ됐음을 ｎ플리케이션！ ｂ록하ｍ 있습니다.

lua_prim_rc
LUA_NEGATIVE_RSP

lua_sec_rc
0이 Ｆ닌 2차 리턴 코드！는 부정 응답시 호스트！ 송신되는 (지 코드！ 들ｎ 있습니다. 이Ｍ은 퍼스널 통신이 호스트 데이타！서 @류를 Ｋ출하) 부정 응답을 호스트！ 송신했음을 나타냅니다. 리턴될 수 있는 (지 코드를 해석하는 정보는 175페이지의 『SNA ｈ층』을 참조하십시@.

0의 2차 리턴 코드는 체인의 중#！서 메세지！ 대한 부정 응답의 이전 **RUI_WRITE**！ 따라 퍼스널 통신이 이 체인으로부터 모든 메세지를 수신하) 삭제했음을 나타냅니다.

다음 리턴 코드는 제ｘ된 명령ｎ 레코드！ 유효하지만 명령ｎ！ Ｏ료되지 ＪＲ음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
！능한 *:

LUA_DATA_TRUNCATED
lua_data_length 매３변수！ 메세지！서 수신한 데이타의 실제ｆ이보다 더 작습니다. 데이타의 **lua_data_length** 바이트만이 명령ｎ로 리턴되ｚ으며 나머지 데이타는 삭제되ｚ습니다. 이 2차 리턴 코드를 ｒ은 ｆｌ 추！ 매３변수！ 리턴됩니다.

LUA_NO_DATA
lua_flag1은 LUA_NOWAIT로 설정되ｎ 데이타를 ｂ다리지 Ｊｍ 즉시 리턴을 나타내며 지정된 세션 흐름！서 현재 사ｋ！능한 데이타！ ｘ습니다.

LUA_INVALID_PROCESS
이 명령ｎ를 수행한 ｎ플리케이션 인스턴스！ 이 세션！ 대한 **RUI_INIT** 명령ｎ를 수행한 인스턴스Ｍ 동일하지 Ｊ습니다.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령ｎ！ Ｏ료되지 ＪＲ음을 나타냅니다.

lua_prim_rc

LUA_SESSION_FAILURE

세션이 종료됐습니다.

lua_sec_rc

!능한 *:

LUA_LU_COMPONENT_DISCONNECTED

통신 링크나 호스트 LUM의 문제 때문! LUA 세션이 실패했습니다.

LUA_RUI_LOGIC_ERROR

리턴 코드는 다음 중 하나를 나타냅니다.

- 호스트 시스템이 SNA 프로토콜을 위반했습니다.
- 내부 @류! LUA 내!서 K출됐습니다.

추적 활성화로 문제점을 재생성하도록 시도하m 호스트! C바른 데이타를 송신하는지 점K하십시@.

lua_prim_rc

LUA_INVALID_VERB

lua_verb 매3변수나 **lua_opcode** 매3변수 중 하나! 유효하지 J습니다. 명령n! 실행되지 JR습니다.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

자x 부족z 0은 n5체제 @류! 발생했습니다.

lua_sec_rc

이 *은 n5체제 리턴 코드입니다. 이 리턴 코드의 의미! 대해서는 n5체제 문서를 점K하십시@.

주석

이 명령n를 수행하b 전! **RUI_INIT** 명령n! 0료되n_ 합니다. b존 **RUI_READ**! 보류중인 동H 보류중인 **RUI_READ**!서 다른 세션 흐름을 지정하는 fl!만 다른 **RUI_READ**를 수행할 수 있습니다. 즉, 동일한 세션 흐름! 대해 둘 이상의 미처리 **RUI_READ**를 !질 수 x습니다.

다음이 사실인 fl **lua_flag1**은 LUA_BID_ENABLE로만 설정될 수 있습니다.

- **RUI_BID**! 수행됐으며 0료됐습니다.
- **RUI_BID** 명령n! 할당된 bo5*이 해제되E나 수정되지 JR습니다.
- 보류중인 다른 **RUI_BID**는 x습니다.

사용시 주의사항

수신한 데이타 ! **lua_max_length** 매 3 변수보다 더 f 다면 잘릴 M 이며 데이타의 **lua_max_length** 바이트만이 리턴됩니다. 1차 및 2차 리턴 코드 LUA_UNSUCCESSFUL 및 LUA_DATA_TRUNCATED ! 리턴됩니다.

RUI_READ 명령 n 를 사 k 하) 메세지를 읽은 다음 ! 는 입력 명령 n 대 b 행렬 ! 서 메세지 ! 제 E 되며 다시 W 세스할 수 x 습니다.

주: **RUI_BID** 명령 n 는 비파 + 적인 읽 b 로 사 k 될 수 있으므로 n 플리케이션은 이를 사 k 하) 사 k ! 능한 데이타의 유형을 점 K 할 수 있지만 데이타는 입력 대 b 행렬 ! 남 F 있으며 즉시 사 k 될 필 d ! x 습니다.

페이싱은 1차 대 2차 반 세션(이 M 은 호스트 8 성 ! 서 지정됩니다) ! 서 사 k 되 n 퍼스널 통신 및 통신 서버 노드 ! 메세지 ! 밀려드는 M 을 막습니다. LUA n 플리케이션이 느리 T 메세지를 읽는 f l , 퍼스널 통신 및 통신 서버 ! 호스트 ! 대한 페이싱 응답 송신을 지 , 시켜 속도를 늦춥니다.

RUI_TERM

RUI_TERM 명령n는 주n진 LUA LU! 대한 LU-LU 세션 및 LU-SSCP 세션 둘다를 종료합니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령n 레코드의 바이트 f이. sizeof(struct LUA_COMMON)로 설정하십시@.

lua_opcode
LUA_OPCODE_RUI_TERM

lua_correlator
선택적. n플리케이션 내!서 다른 프로세스M 이 명령n를 ,|시키b 위해 사k할 수 있는 4바이트 *. LUA는 이 정보를 사k하E나 변f하지 J습니다.

lua_luname
세션이 사k하는 로컬 LU의 ASCII 이름. 이M은 활동중 LUA 세션의 LU명(또는 **RUI_INIT** 미처리 명령! 지정한 LU명)z 일치해_ 합니다.

이 매3변수는 **lua_sid** 매3변수! 0인 fl!만 필d합니다. 세션ID! **lua_sid**!서 제x되면 LUA는 이 매3변수를 사k하지 J습니다.

이 매3변수는 8바이트 f이)_ 하며 이름이 8자보다 짧은 fl 0x20 x백으로 @른쪽을 채s니다.

lua_sid
세션의 세션 ID. 이M은 이전 **RUI_INIT** 명령n!서 리턴되는 세션IDM 일치해_ 합니다.

이 매3변수는 선택적입니다. 세션 ID를 지정하지 J은 fl, **lua_luname** 매3변수! 세션! 대한 LU명을 지정해_ 합니다.

lua_post_handle
이M은 비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다.

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2
명령n！ 비동b적으로 O료된 f l！만 이M이 LUA_ASYNC로 설정됩니다.

리턴된 다른 매3변수는 명령n！ O료됐는지의)부！ 좌l되며 다음 절을 참조하십시@.

명령n！ 실행되면 LUA는 다음 매3변수를 리턴합니다.

lua_prim_rc
LUA_OK

다음 리턴 코드는 제x된 매3변수！ @류！ 있n 명령n！ O료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
！능한 *:

LUA_RESERVED_FIELD_NOT_ZERO
명령n 레코드의 9` 필드나 이 명령n！ 사k하지 J은 매3변수！ 0이 F닌 *으로 설정되z습니다.

LUA_VERB_LENGTH_INVALID
lua_verb_length 매3변수의 *은 이 명령n！ 필d한 명령n 레코드의 f이보다 적습니다.

다음 리턴 코드는 유효하지 J은 세션 상태！서 명령n！ 수행되z음을 나타냅니다.

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

R_IT 명령n！ 이 세션！ 대해 O료되지 J R E나
세션 중단이 발생했습니다.

다음 리턴 코드는 제x된 명령n 레코드！ 유효하지만 명령n！ O료되지 J R음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
！능한 *:

LUA_COMMAND_COUNT_ERROR
명령n！ 발행되z을 때 **RUI_TERM**이 보류중이z습니다.

LUA_INVALID_PROCESS
이 명령n를 수행한 n플리케이션 인스턴스! 이 세션! 대한 **RUI_INIT** 명령n를 수행한 인스턴스M 동일하지 J습니다.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령n! O료되지 JR음을 나타냅니다.

lua_prim_rc
LUA_SESSION_FAILURE

세션이 종료되z습니다.

lua_sec_rc
!능한 *:

LUA_LU_COMPONENT_DISCONNECTED
통신 링크나 호스트 LUM의 문제 때문! LUA 세션이 실패했습니다.

LUA_RUI_LOGIC_ERROR
리턴 코드는 다음 중 하나를 나타냅니다.

- 호스트 시스템이 SNA 프로토콜을 위반했습니다.
- 내부 @류! LUA 내!서 K출되z습니다.

추적 활성화로 문제점을 재생성하도록 시도하m 호스트! C바른 데이타를 송신하는지 점K하십시@.

lua_prim_rc
LUA_INVALID_VERB

lua_verb 매3변수나 **lua_opcode** 매3변수 중 하나! 유효하지 J습니다. 명령n! 실행되지 JR습니다.

lua_prim_rc
LUA_UNEXPECTED_DOS_ERROR

자x 부족z 0은 n5체제 @류! 발생했습니다.

lua_sec_rc
이 *은 n5체제 리턴 코드입니다. 이 리턴 코드의 의미! 대해서는 n5체제 문서를 점K하십시@.

주석

RUI_INIT 명령n! 수행된 후!(명령n의 O료)부! |hx이) p제라도 이 명령n를 수행할 수 있습니다. **RUI_TERM**이 수행될 때 다른 LUA 명령n! 보류중인 fl, 보류중인 명령n! 더 이상의 처리는 일n나지 J으며 LUA_CANCELLED의 1차 리턴 코드로 리턴합니다.

이 명령n! O료된 후! 다른 LUA 명령n는 이 세션! 대해 수행될 수 x습니다.

RUI_WRITE

RUI_WRITE 명령 n는 LU-LU 세션이나 LU-SSCP 세션 중 하나 ! 서 LUA n 플리케이션 ! 서 호스트로 SNA d청이나 응답 단위를 송신합니다.

제공되는 매개변수

n 플리케이션은 다음 매 3 변수를 제 x 합니다.

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 명령 n 레코드의 바이트 f이. sizeof(struct LUA_COMMON)로 설정하십시@.

lua_opcode
LUA_OPCODE_RUI_WRITE

lua_correlator
선택적. n 플리케이션 내 ! 서 다른 프로세스 M 이 명령 n를 , | 시키 b 위해 사 k 할 수 있는 4바이트 *. LUA는 이 정보를 사 k 하 E 나 변 f 하지 J습니다.

lua_luname
세션이 사 k 하는 로컬 LU의 ASCII 이름. 이 M 은 활동중 LUA 세션의 LU명 z 일치해 _ 합니다.

이 매 3 변수는 **lua_sid** 매 3 변수 ! 0인 f l ! 만 필 d 합니다. 세션 ID ! **lua_sid** ! 서 제 x 되면 LUA는 이 매 3 변수를 사 k 하지 J습니다.

이 매 3 변수는 8바이트 f이) _ 하며 이름이 8자보다 짧은 f l 0x20 x 백으로 @른쪽을 채 s 니다.

lua_sid
세션의 세션 ID. 이 M 은 이전 **RUI_INIT** 명령 n ! 서 리턴되는 세션 ID M 일치해 _ 합니다.

이 매 3 변수는 선택적입니다. 세션 ID를 지정하지 J은 f l, **lua_luname** 매 3 변 . 일해 _ 합 다설선 설 선설설석

부정 응답을 송신할 때 데이타 버퍼! 제x된 SNA (지 코드(4바이트)의 f이로 이 매3변수를 설정하십시@(**lua_data_ptr** 참조).

lua_data_ptr
제x된 데이타! 들n 있는 버퍼! 대한 포인터.

d청이나 데이타를 d8하는 `정 응답의 fl, 버퍼!는 전체 RU! 들n 있n_ 합니다. RU의 f이! **data_length**! 지정되n_ 합니다.

부정 응답의 fl, 버퍼!는 SNA (지 코드! 들n 있n_ 합니다.

lua_post_handle
비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다.

lua_th.snf
응답을 송신하는 fl!만 필d합니다. 응답인 d청의 순서 번호.

lua_rh
d청을 송신할 때 대부분의 **lua_rh** 플래W는 송신되는 메세지의 RH(d청 헤더)! 대응하) 설정되n_ 합니다. LUA_PI 및 LUA_QRI를 설정하지 마십시@. 이M은 LUA! 의해 설정됩니다.

응답을 송신할 때 다음z 0은 두 !지의 **lua_rh** 플래W만 설정됩니다.

LUA_RRI
응답을 나타내b 위해 설정됩니다.

LUA_RI
`정 응답!는 설정되지 J으며 부정 응답! 설정됩니다.

lua_flag1
데이타! 송신되는 메세지 흐름을 나타내려면 다음 플래W 중 하나를 설정하십시@.

LUA_LU_EXP
LU d^ 흐름

LUA_SSCP_NORM
SSCP 정상 흐름

LUA_LU_NORM
LU 정상 흐름

@직 하나의 플래W만 설정되n_ 합니다.

주: 퍼스널 통신 및 통신 서버는 n플리케이션이 SSCP d^ 흐름(LUA_SSCP_EXP)!서 데이타를 송신하도록 허k하지 J습니다.

리턴되는 매개변수

다음 매3변수는 항상 리턴됩니다.

lua_flag2
명령 n ! 비동 b 적으로 0 료된 f l ! 만 LUA_ASYNC로 설정됩니다.

리턴된 다른 매 3 변수는 명령 n ! 0 료되 z 는지의) 부 ! 좌 l 되며 다음 절을 참조하십시 @.

명령 n ! 실행되면 LUA는 다음 매 3 변수를 리턴합니다.

lua_prim_rc
LUA_OK

lua_sid
이 명령 n 를 수행할 때 n 플리케이션 ! 서 세션 ID를 지정하지 J m **lua_luname** 매 3 변수를 지정한 f l, LUA는 세션 ID를 제 x 합니다.

lua_th LUA ! 채 l 는 필드를 포함하) 작성된 메세지의 0 료된 TH. 호스트로부터 응답 z 의 , | 을 위해 **lua_th.snf**(순서 번호)의 * 을 보 | 할 필 d ! 있습니다.

lua_rh
LUA ! 채 l 는 필드를 포함하) 작성된 메세지의 0 료된 RH.

lua_flag2
데이타 ! 수신되는 메세지 흐름을 나타내 b 위해 다음 * 중 하나로 설정됩니다.

LUA_SSCP_EXP
SSCP d ^ 흐름

LUA_LU_EXP
LU d ^ 흐름

LUA_SSCP_NORM
SSCP 정상 흐름

LUA_LU_NORM
LU 정상 흐름

다음 리턴 코드는 명령 n ! 다른 명령 n ! 의해 취소되 n 0 료되지 J R 음을 나타냅니다.

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

RUI_TERM 명령 n ! 이 세션 ! 발행됐으므로 명령 n ! 취소됐습니다.

다음 리턴 코드는 제 x 된 매 3 변수 ! @ 류 ! 있 n 명령 n ! 0 료되지 J R 음을 나타냅니다.

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

!능한 *:

LUA_BAD_DATA_PTR

lua_data_ptr 매3변수! 틀린 *이 들n 있습니다.

LUA_DUPLICATE_WRITE_FLOW

RUI_WRITE! 이 명령n!서 지정한 세션 흐름! 대해 미처리되z습니다. (세션 흐름이 **lua_flag1** 흐름 플래W 중 하나를 설정하) 지정되z습니다.) 한번! @직 하나의 **RUI_WRITE**! " 세션 흐름!서 미처리될 수 있습니다.

LUA_INVALID_FLOW

lua_flag1! LUA_SSCP_EXP로 설정되n 메세지! SSCP d^ 흐름! 송신되n_ 함을 나타냅니다. 퍼스널 통신 및 통신 서버는 n플리케이션이 이 흐름!서 데이타를 송신하도록 허k하지 J습니다.

LUA_MULTIPLE_WRITE_FLOWS

둘 이상의 **lua_flag1** 흐름 플래W! 설정되지 JR습니다. 데이타! 송신되는 세션 흐름을 나타내려면 이들 플래W 중 @직 하나만 설정되n_ 합니다.

LUA_REQUIRED_FIELD_MISSING

리턴 코드는 다음 fl 중 하나를 나타냅니다.

- **lua_flag1** 흐름 플래W 중 n느 M도 설정되지 JR습니다. 이들 플래W 중 @직 하나만 설정되n_ 합니다.
- **RUI_WRITE** 명령n! 응답을 송신하b 위해 사k되z으며 응답은 제x된 M보다 더 많은 데이타를 필d로 합니다.

LUA_RESERVED_FIELD_NOT_ZERO

명령n 레코드의 9` 필드나 이 명령n! 사k하지 J은 매3변수! 0이 F닌 *으로 설정되z습니다.

LUA_VERB_LENGTH_INVALID

lua_verb_length 매3변수의 *은 이 명령n! 필d한 명령n 레코드 f이보다 적습니다.

다음 리턴 코드는 유효하지 J은 세션 상태!서 명령n! 수행됐음을 나타냅니다.

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

!능한 *:

LUA_MODE_INCONSISTENCY

RUI_WRITE!서 송신된 SNA 메세지! 이번! 유효하지 J

습니다. 이 M은 세션이 바인드되 b 전! LU-LU 세션!서 데이타를 송신하려 m 시도함으로써 _ b됩니다. 송신된 SNA 메세지의 순서를 점 K하십시@.

LUA_NO_RUI_SESSION
RUI_INIT 명령 n ! 이 세션!서 O료되지 JRE나 세션 중단이 발생했습니다.

다음 리턴 코드는 제 x된 명령 n 레코드! 유효하지만 명령 n ! O료되지 JR음을 나타냅니다.

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
!능한 *:

LUA_FUNCTION_NOT_SUPPORTED
리턴 코드는 다음 f l 중 하나를 나타냅니다.

- **lua_rh**! LUA_FI(형식 표시 b)로 설정되 z 지만 제 x된 RU의 첫번째 바이트는 인식된 d청 코드! F닙니다.
- **lua_rh**! LUA_RH_NC(NC(네트 v크 제 n) 범주를 지정한 RU 범주)로 설정했으며 퍼스널 통신은 n 플리케이션이 이 범주!서 d청을 송신하도록 허 k하지 J습니다.

LUA_INVALID_PROCESS
이 명령 n를 수행한 n 플리케이션 인스턴스! 이 세션! 대한 **RUI_INIT** 명령 n를 수행한 인스턴스 M 동일하지 J습니다.

LUA_INVALID_SESSION_PARAMETERS
n 플리케이션은 호스트로부터 수신한 **BIND** 메세지! `정 응답을 송신하 b 위해 **RUI_WRITE**를 사 k했습니다. W러나 퍼스널 통신 및 통신 서버 노드는 지정한 대로 **BIND** 매 3변수를 수 k할 수 x으며 호스트! 부정 응답을 송신했습니다. 퍼스널 통신 및 통신 서버! 수 k한 **BIND** 프로파일! 대한 자세한 내 k은 175페이지의 『SNA h층』을 참조하십시@.

LUA_RSP_CORRELATION_ERROR
응답을 송신하 b 위해 **RUI_WRITE**를 사 k할 때 **lua_th.snf** 매 3변수(응답중인 수신 메세지의 순서 번호를 나타냅니다)! 유효한 *을 포함하지 JR습니다.

LUA_RU_LENGTH_ERROR
lua_data_length 매 3변수! 틀린 *이 들 n 있습니다. LU 정상 흐름!서 전송할 때 최대 f이는 호스트로부터 수신한 **BIND**! 지정되며 다른 모든 흐름의 f l 최대 f이는 256바이트입니다.

(임의의 다른 *)
) b !서 임의의 다른 2차 리턴 코드는 제 x된 SNA 데이타

! 유효하지 JE나 송신될 수 x음을 나타내는 SNA (지 코드입니다. 리턴될 수 있는 SNA (지 코드를 해석하는 정보는 175페이지의 『SNA h층』을 참조하십시@.

다음 리턴 코드 및 2차 리턴 코드는 다른 이유로 명령n! O료되지 JR음을 나타냅니다.

lua_prim_rc

LUA_SESSION_FAILURE

세션이 종료되z습니다.

lua_sec_rc

!능한 *:

LUA_LU_COMPONENT_DISCONNECTED

통신 링크나 호스트 LUM의 문제 때문! LUA 세션이 실패했습니다.

LUA_RUI_LOGIC_ERROR

이 리턴 코드는 다음 중 하나를 나타냅니다. 호스트 시스템이 SNA 프로토콜을 위반했습니다. 내부 @류! LUA 내!서 K출되z습니다.

추적 활성화로 문제점을 재생성하도록 시도하m 호스트! C바른 데이타를 송신하는지 점K하십시@.

lua_prim_rc

LUA_INVALID_VERB

lua_verb 매3변수나 **lua_opcode**
매3변수 중 하나! 유효하지 J습니다.
명령n! 실행되지 JR습니다.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

자x 부족z 0은 n5체제 @류! 발생했습니다.

lua_sec_rc

이 *은 n5체제 리턴 코드입니다. 이 리턴 코드의 의미! 대해서는 n5체제 문서를 점K하십시@.

주석

이 명령n를 수행하b 전! **RUI_INIT** 명령n! 수행되n_ 합니다. b존 **RUI_WRITE**! 보류중인 동H 보류중인 **RUI_WRITE**!서 다른 세션 흐름을 지정하는 fl!만 두 번째 **RUI_WRITE**를 수행할 수 있습니다. 즉 동일한 세션 흐름! 대해 둘 이상의 미처리 **RUI_WRITE**를 !질 수 x습니다.

RUI_WRITE 명령n는 성x적인 **RUI_INIT** 명령n 이후! p제라도 SSCP 정상 흐름!서 수행될 수 있습니다. LU d^ 또는 LU 정상 흐름!서의 **RUI_WRITE** 명령n는 **BIND**! 수신된 이후!만 허k되며 **BIND**!서 지정한 프로토콜로 준수되n_ 합니다.

사용시 주의사항

RUI_WRITE의 성x적인 0료는 메세지! 데이타 링크! 대b행렬됐음을 나타내며 메세지! 송신되zE나 호스트! 이를 수k했음을 반드시 나타내지는 J습니다. 페이싱은 2차 대 1차 반 세션!서 사k되n(이M은 **BIND** 세션!서 지정됩니다) 로컬 또는 x] LU! 처리할 수 있는 M보다 더 많은 자료를 LUA n플리케이션이 송신하지 못하T 할 수 있습니다. 이M이 Wfl라면 LU 정상 흐름!서의 **RUI_WRITE**! LUA! 의해 지,될 수 있으며 0료하는 데 시#이 I릴 수 있습니다.

주: 퍼스널 통신 및 통신 서버는 n플리케이션이 SSCP d^ 흐름(LUA_SSCP_EXP)!서 데이타를 송신하도록 허k하지 J습니다.

제14장 SLI 엔트리 포인트

이 장 ! 서는 SLI ! 대한 프로시듀n #트리 포인트를 설명합니다.

SLI DLL은 다음 프로시듀n #트리 포인트를 정의합니다.

SLI()

모든 **SLI** 명령 n ! 대한 이벤트 통지를 제 x 합니다.

구문

```
void WINAPI SLI (LUA_VERB_RECORD* vcb);
```

매 3 변수

설명

vcb 제 x 된 매 3 변수로 명령 n 제 n 블럭의 주소를 지정합니다.

리턴 값

lua_flag2.async ! 리턴된 * 은 비동 b 통지 ! 발생할지) 부를 나타냅니다. 플래 W ! 설정된 f l(0이 F 님), 비동 b 통지는 이벤트 신호를 통해 발생합니다. 플래 W ! 설정되지 J 은 f l, d 청이 비동 b 적으로 O 료됩니다. 모든 @ 류 상태 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시 @.

사용시 주의사항

n 플리케이션은 명령 n 제 n 블럭의 *lua_post_handle* 매 3 변수 ! 서 이벤트 ! 대한 핸들을 제 x 해 _ 합니다. 이벤트는 신호 x 음 상태 ! 있 n _ 합니다.

비동 b 작 w 이 O 료되면 n 플리케이션은 이벤트의 신호로 통지됩니다. 이벤트의 신호 ! 따라 @ 류 상태 ! 대한 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시 @. 261페이지의 『WinSLI』도 참조하십시 @.

SLI 명령 n 만이 OS/2 및 Windows 3.1 클라이 p 트 ! 서 지 x 됩니다.

WinSLI

모든 SLI 명령n ! 대한 비동b 메세지 통지를 제x합니다.

구문

```
int WINAPI WinSLI (HWND hWnd,
                   LUA_VERB_RECORD* vcb);
```

매3변수

설명

hwnd O료 메세지를 수신하b 위한 창 핸들.

vcb 명령n 제n 블럭 ! 대한 포인터.

리턴 값

프로세스를 위해 SLI ! d청을 수k했는지)부를 나타내는 *을 함수 ! 리턴합니다. 리턴 * 0은 d청이 수k되z으며 처리될 M임을 나타냅니다. 0이 F닌 *은 @류를 나타냅니다. !능한 @류 코드는 다음z 0습니다.

WLUAINVALIDHANDLE

제x되는 창 핸들은 유효하지 J습니다.

lua_flag2.async ! 리턴된 *은 비동b 통지 ! 발생할지)부를 나타냅니다. 플래W ! 설정된 f l(0이 F님), 비동b 통지는 n플리케이션의 메세지 대b행렬 ! 통지된 메세지를 통해 발생합니다. 플래W ! 설정되지 J은 f l, d청이 비동b적으로 O료됩니다. 모든 @류 상태 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시@.

사용시 주의사항

명령n를 O료하면 n플리케이션의 창 *hWind*는 입력 문자 - 로 **WinSLI**를 . 는 **RegisterWindowMessage** ! 의해 리턴되는 메세지를 수신합니다. **lParam** 인수 ! 는 O료된 M으로 통지되는 VCB의 주소 ! 들n 있습니다. *wParam* 인수는 정의되n 있지 J습니다. 처리를 위해 d청을 수k하지만(함수 호출 ! 서 0을 리턴합니다) VCB ! 서 설정된 1차 및 2차 리턴 코드로 나중 ! E부할 수 있습니다. 모든 @류 상태 ! 대해 1차 리턴 코드 및 2차 리턴 코드를 살펴보십시@. | 련 참조: 260페이지의 『SLI()』.

WinSLICleanup()

SLI API ! 서 n 플리케이션을 종료하 m 등록취소합니다.

구문

```
BOOL WINAPI WinSLICleanup (void);
```

리턴 값

리턴 * 은 등록취소의 성 x 또는 실패를 나타냅니다. * 이 0이 F 니면 n 플리케이션이 등록취소되 z 습니다. * 이 0이면 n 플리케이션이 등록취소되지 J R 습니다.

사용시 주의사항

SLI API를 등록취소하려면, 9 를 들 n 특정 n 플리케이션 ! 할당된 자 x 을 해제하려면 **WinSLICleanup**를 사 k 하십시 @ .

WinSLICleanup 사 k 은 필 d 하지 J 습니다.

WinSLIStartup()

n플리케이션을 사k !능하T 하) SLI API의 필d한 버전을 지정하m API의 세부사항을 K색하T 합니다.

구문

```
int WINAPI WinSLIStartup (WORD wVersionRequired,
                          LUADATA* luadata);
```

매3변수

설명

wVersionRequired

필d한 SLI API 지x의 버전을 지정합니다. 상위 바이트는 소 버전(3정) 번호를 지정하며 하위 바이트는 주 버전 번호를 지정합니다.

luadata

SLI 8현의 버전을 리턴합니다.

리턴 값

리턴 *은 n플리케이션이 등록되z는지M SLI API! 지정 버전 번호를 지x할 수 있는지)부를 지정합니다. *이 0이면 등록되z으며 지정 버전이 지x될 수 있습니다. W렇지 J으면 리턴 *이 다음 * 중 하나입니다.

WLUAVERNOTSUPPORTED

등록된 SLI API 지x 버전이 이 특정 SLI API! 의해 제x되지 J습니다.

WLUAINVALID

d청된 버전을 판별할 수 x습니다.

사용시 주의사항

WinSLIStartup 사k은 필d하지 J습니다.

WinSLIStartup()

제15장 SLI 명령어

이 장!는 " SLI 명령n ! 대해 다음z 0은 정보! 들n 있습니다.

- 명령n의 목적.
- SLI! 의해 제x되m 리턴되는 매3변수. " 매3변수의 설명!는 해당 매3변수! 유효한 * ! 정보M 필d한 추! 정보! 들n 있습니다.
- 다른 명령n M의 상호작k.
- 명령n 사k을 설명하는 추! 정보.

주: *9비*로 표시되는 매3변수는 항상 0으로 설정되n _ 합니다.

SLI_BID

이 명령n는 SLI_RECEIVE! 메세지 읽b를 보류하E나 상태! 표시됐음을 SLI n플리케이션 프로W램! K립니다. SLI_BID는 보류중인 데이타를 미리보b 위해 사k되므로 n플리케이션은 데이타 수신을 위한 전략을 형식화할 수 있습니다. 데이타나 상태! SLI n플리케이션 프로W램k으로 수신되면 적합한 SLI_RECEIVE! 활동중이 F니라면 SLI_BID! 통지됩니다. n플리케이션 프로W램은 세션이 -린 후!(또는 시작 유형이 SSCP W세스를 .는 b본 유형인 f l SLI_OPEN 동H) SLI_BID 명령n를 발행하) n플리케이션 프로W램이 송수신 d8 메카니즘을 사k할 M임을 나타냅니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb

LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length

명령n 제n 블럭의 f이. 이 숫자는 SLI_BID 명령n! 대해 SLI! 9상하는 f이M 0F_ 합니다.

lua_opcode

LUA_OPCODE_SLI_BID

명령n! 대한 작w 코드.

lua_correlator

다른 사k자 제x 정보를 .는 명령n를 링크하는 *. 이 매3변수는 LUA 인터페이스! 사k하지 J습니다.

lua_luname

ASCII로 된 로컬 LU명. 이름이 8자보다 작은 fl, x백으로 채v_ 합니다. LUA는 **lua_sid**! 0인 fl!만 이 매3변수를 조사합니다. 모든 명령n!서 **lua_luname** 매3변수를 사k하면 디버W를 더k이하T 하며 특히 다중 LU! 8성되는 fl W렇습니다.

lua_sid

사k되는 세션을 식별하며 SLI_OPEN! 의해 리턴되는 세션 ID. 이 매3변수! 0이면 **lua_luname** 매3변수는 식별k으로 사k됩니다.

lua_post_handle

이M은 비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들입니다.

리턴되는 매개변수

명령n! O료되면 다음 매3변수! 리턴됩니다.

lua_prim_rc
명령n b능으로 설정되는 1차 리턴 코드.

lua_sec_rc
명령n b능으로 설정되는 2차 리턴 코드.

lua_data_length
수신된 peek 데이타의 f이.

lua_peek_data
이 매3변수는 읽을 RU 데이타의 최초 12바이트n지 포함합니다. 이 매3변수!서 리턴되는 데이타의 f이는 **lua_data_length** 매3변수! 있습니다.

lua_th 메세지! 대한 SNA 전송 헤더(TH)! 들n 있는 6바이트 매3변수.

lua_rh
메세지! 대한 SNA d청/응답 헤더(RH)! 들n 있는 3바이트 매3변수.

lua_message_type
SNA 데이타 및 명령의 유형. 유효한 메세지 유형은 다음z 0습니다.

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIND
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SIGNAL
LUA_MESSAGE_TYPE_STSN

SLI는 LUA 인터페이스 확장 루틴을 통해 BIND 및 STSN d청을 수신하며 응답합니다.

LU_DATA, LUSTAT_LU, LUSTAT_SSCP 및 SSCP_DATA는 SNA 명령이 F닙니다.

lua_flag2

출력 매3변수로서 사k되는 비트! 들n 있는 1바이트 플래W. 명령n 0료시, *이 b술되지 J은 모든 비트는 9`되며 0으로 설정되n_ 합니다. 상위 반 바이트의 플래W는 다음z 0습니다.

lua_flag2.async
이 명령n ! 비동b적으로 0료함을 나타내는 플래W.

하위 반 바이트!는 메세지 세션 및 흐름을 b술하는 플래W ! 들n 있습니다. 다음 플래W 중 하나! 리턴됩니다.

lua_flag2.sscp_exp
SSCP d^ 흐름을 지정합니다

lua_flag2.sscp_norm
SSCP 정상 흐름을 지정합니다

lua_flag2.lu_exp
LU d^ 흐름을 지정합니다

lua_flag2.lu_norm
LU 정상 흐름을 지정합니다

lua_prim_rc
명령n b능으로 설정되는 1차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

lua_sec_rc
명령n b능으로 설정되는 2차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

사용시 주의사항

@직 하나의 **SLI_BID**만이 " 세션! 대해 활성화될 수 있습니다. n플리케이션 프로W램은 **SLI_BID**! 재활성화된 fl, 데이타를 읽을 수 x다m 할지라도 " 흐름! 대해 한번만 송수신 d8될 수 있습니다. n플리케이션 프로W램이 송수신 d8 데이타를 읽지 J은 fl, 해당 특정 흐름을 다시 송수신 d8하지 J습니다.

처음 **SLI_BID** 명령n를 발행하면 송수신 d8 b능을 사k할 수 있T 합니다. **SLI_BID** 명령n 통지! 0료된 후! 송수신 b능은 사k할 수 xT 됩니다. 송수신 d8 b능은 다음 두 !지 방식 중 하나로 다시 사k하T 할 수 있습니다.

- **SLI_BID** 명령n 제n 블럭의 주소로 다시 SLI를 호출하b.
- **lua_flag1.bid_enable** 매3변수를 1로 설정한 **SLI_RECEIVE**를 발행하b. **lua_flag1.bid_enable**를 !진 **SLI_RECEIVE**를 발행하는 fl, SLI는 활동중인 송수신 d8로서 마지막 수k된 **SLI_BID** 명령n 제n 블럭의 주소를 사k합니다.

주:

1. **SLI_BID**! 발행되z을 때 다중 흐름이 사k!능한 데이타를 !지는 fl, **SLI_BID**! 리턴하는 데이타는 데이타를 !진 최상의 l선순위 흐름!서 I니다. 최상!서 최하n지의 l선순위는 다음z 0습니다.
   - SSCP–d^
   - LU–d^
   - SSCP–정상
   - LU–정상
2. **SLI_BID** 0료 다음! LUA n플리케이션은 다중 **lua_flag1**이 있는 **SLI_RECEIVE**를 발행하며 읽은 데이타는 **SLI_BID**! 리턴한 데이타 M는 다른 흐름을 위한 M일 수 있습니다. 0료한 **SLI_BID**M 발행된 **SLI_RECEIVE**의 시# 사이! 호스트로부터 상위 l선순위! 수신되면 발생할 수 있습니다.

   W러나 LUA n플리케이션은 **SLI_RECEIVE**! 방] 송수신 d8한 데이타를 읽z음을 보장할 수 있습니다. **SLI_RECEIVE** 명령n! 대한 제n 블럭!서 **lua_flag1** 흐름 플래W 중 하나만을 설정하) !능하며 0료된 **SLI_BID**의 **lua_flag2** 필드!서 리턴한 Mz 동일한 흐름을 지정합니다.

SLI_BID는 RU! 수신되자마자 0료합니다. 이 RU는 체인!서 유일한 RU이E나 다중 RU 체인!서의 첫번째 RU일 수 있습니다. **SLI_BID** 0료시 단일 d소 체인은 0료 체인이 이 n플리케이션! 송수신 d8하는 유일한 시#입니다.

SLI_BID! 다중 RU 체인의 첫번째 RU로 0료하m 차후의 **SLI_RECEIVE**! **lua_flag1.nowait** I션을 지정하는 fl, **lua_flag1.nowait** I션은 무시됩니다. **SLI_RECEIVE** 명령n는 진행중! 리턴하며 체인!있는 모든 RU! 수신된 후! 비동b적으로 0료합니다.

상태! 사k!능하면 n플리케이션이 상태를 읽n_ 합니다. n플리케이션이 **SLI_BID** 또는 **SLI_RECEIVE**를 발행하) 상태를 읽을 때n지 다음을 제\한 모든 작w은 E부됩니다.

- SSCP 흐름!서의 **SLI_SEND** 명령n
- **SLI_CLOSE**

1차 리턴 코드! STATUS이면 리턴되는 유일한 **SLI_BID** 매3변수는 **lua_prim_rc, lua_sec_rc** 및 **lua_sid**입니다. 상태! 사k!능할 때 **SLI_BID** 및 **SLI_RECEIVE**! 둘다 활동중이면 **SLI_BID**만이 해당 상태로 통지됩니다. 상태를 송수신하도록 n플리케이션 프로W램! d8하면 모든 정보! 표시되며 **SLI_RECEIVE**! 필d하지 J습니다.

1차 리턴 코드의 *이 STATUS이면 2차 리턴 코드의 !능한 *은 다음z 0습니다.

- READY

SLI 세션이 모든 추 ! 적인 명령을 처리할 준비 ! 됐음을 나타냅니다. READY 상태는 이전 NOT_READY 상태 ! 수신된 이후 ! 발행됩니다.

- NOT_READY

  유형 * X'02' 또는 X'01'를 . 는 CLEAR 명령이나 UNBIND 명령이 호스트 ! 서 수신됐음을 나타냅니다. SLI 세션이 일시중단됩니다.

  – CLEAR ! 발행되면 SDT 명령이 수신될 때 n 지 세션이 일시중단됩니다.
  – SNA UNBIND 유형 X'02'(BIND ! p 이 n 지는 UNBIND) ! 발행되면 세션은 BIND, 선택적 CRV 및 STSN W 리 m SDT 명령이 수신될 때 n 지 일시중단됩니다. 모든 사 k 자 확장 루틴은 재진입되 n _ 합니다.
  – UNBIND 유형 X'01'(UNBIND 정상)이 발행되 m 이 세션 ! 대한 **SLI_OPEN** 명령 n ! LUA_SESSION_TYPE_DEDICATED의 **lua_session_type**을 지정한 f l, 세션은 BIND, 선택적 CRV 및 STSN W 리 m SDT 명령이 수신될 때 n 지 일시중단됩니다. 이들 명령을 프로세스하 b 위해 제 x 된 사 k 자 확

lua_sec_rc
9 (X'00000000')

lua_rh.rri
비트 OFF(d 청 단위(RU))

lua_rh.sdi
비트 ON((지 데이타 포함)

이들 조G ! 서 d 청이 EXR로 변환되z 으며 최대 7바이트의 정보 ! **lua_peek_data** 명령n 매3 변수 ! 서 리턴됩니다. **lua_peek_data** 매3 변수 ! 있는 정보 형식은 다음z 0 습니다.

- 바이트 0—3 ! 는 K 출된 @ 류를 정의하는 (지 데이타 ! 들n 있습니다. LUA ! d 청을 EXR로 변환하면 (지 데이타는 다음 * 중 하나입니다.

(지 데이타	바이트 0-3의 *
LUA_MODE_INCONSISTENCY	X'08090000'
LUA_BRACKET_RACE_ERROR	X'080B0000'
LUA_BB_REJECT_NO_RTR	X'08130000'
LUA_RECEIVER_IN_TRANSMIT_MODE	X'081B0000'
LUA_CRYPTOGRAPHY_FUNCTION_INOP	X'08480000'
LUA_SYNC_EVENT_RESPONSE	X'10010000'
LUA_RU_DATA_ERROR	X'10020000'
LUA_RU_LENGTH_ERROR	X'10020000'
LUA_INCORRECT_SEQUENCE_NUMBER	X'20010000'

lua_peek_data ! 서 바이트 4 - 6으로 리턴되는 정보 ! 는 x 래 d 청 단위(RU)의 최초 3바이트n 지 들n 있습니다.

SLI_CLOSE

이 명령n는 SNA 세션을 닫습니다. SLI_CLOSE는 호스 트 n플리케이션 프로W램z의 , a을 종료하며 사k한 자x을 해제합니다. SLI_CLOSE의 통지는 LU-LU 및 SSCP-LU 통신이 종료됐음을 의미합니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb
LUA_VERB_SLI

LUA 명령n ! 대한 명령n 코드 표시b .

lua_verb_length
명령n 제n 블럭의 f이. 이 숫자는 SLI_CLOSE 명령n ! 대해 SLI ! 9상하는 f이M 0F_ 합니다.

lua_opcode
LUA_OPCODE_SLI_CLOSE

이 명령n ! 대한 작w 코드. SLI_CLOSE의 f l .

lua_correlator
LUA n플리케이션 프로W램이 이 명령n를 프로W램이 제x하는 다른 정보M |련시키는 M을 돕b 위해 제x할 수 있는 * . 이 매3변수는 LUA 인터페이스! 사k하지 J습니다.

lua_luname
ASCII로 된 로컬 LU명. 이름이 8자보다 작은 f l, x백으로 채v _ 합니다. LUA는 **lua_sid**! 0인 f l !만 이 매3변수를 조사합니다. 모든 명령n !서 **lua_luname** 매3변수를 사k하면 디버W를 더k이하T 하며 특히 다중 LU! 8성되는 f l W렇습니다.

lua_sid
사k되는 세션을 식별하는 0료 SLI 명령n ! 의해 리턴되는 세션 ID. 이 매3변수! 0이면 **lua_luname** 매3변수는 식별k으로 사k됩니다.

lua_post_handle
비동b 명령n의 0료를 통지하b 위해 사k되는 4바이트 핸들입니다.

lua_flag1.close_abend
닫b ! 즉시 닫b(on) 또는 정상 닫b(off)인지)부를 지정합니다.

리턴되는 매개변수

명령n ! 0료되면 다음 매3변수! 리턴됩니다.

lua_flag2.async
이 명령n ! 비동b적으로 0료함을 나타내는 플래W.

lua_prim_rc

명령n b능으로 설정되는 1차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

lua_sec_rc

명령n b능으로 설정되는 2차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

사용시 주의사항

정상 닫b 및 abend 닫b의 두 !지 유형의 **SLI_CLOSE**! 있습니다.

- 정상 닫b

  정상 닫b는 **lua_flag1.close_abend** 매3변수! 0으로 설정되z을 때 식별됩니다. 닫b 순서는 2차 시작 또는 1차 시작할 수 있습니다. 정상 닫b는 1차 시작 또는 2차 시작! 대해 SHUTD 명령을 사k합니다. 정상 닫b는 1차 시작 닫b! 대해 SHUTD 명령을 사k하며 2차 시작 닫b! 대해 RSHUTD 명령을 송신합니다.

  호스트! 1차 또는 2차 시작 **SLI_CLOSE** 정상 동H! UNBIND 유형 X'02'(BIND! p 이n지는 UNBIND)를 송신하면 세션이 닫히지 J습니다. **SLI_CLOSE** 명령n는 CANCELED 1차 리턴 코드, RECEIVED_UNBIND_HOLD 2차 리턴 코드로 O료합니다. n플리케이션 프로W램은 **SLI_BID** 또는 **SLI_RECEIVE** 명령n를 발행하) STATUS를 리턴해_ 합니다.

  지정된 이 세션! 대한 1차 또는 2차 시작 **SLI_CLOSE** 정상 및 **SLI_OPEN** 명령n 동H 호스트! UNBIND 유형 X'01'(정상 UNBIND)z LUA_SESSION_TYPE_DEDICATED의 **lua_session_type**을 송신하는 fl, 세션이 닫히지 J습니다. **SLI_CLOSE** 명령n는 CANCELED 1차 리턴 코드, RECEIVED_UNBIND_NORMAL 2차 리턴 코드로 O료합니다. n플리케이션 프로W램은 **SLI_BID** 또는 **SLI_RECEIVE** 명령n를 발행하) STATUS를 리턴해_ 합니다.

- abend 닫b

  abend 닫b는 **lua_flag.close_abend** 매3변수! 1로 설정된 fl 식별됩니다. CLOSE_ABEND I션은 SLI! 세션을 즉시 종료하T 합니다.

다음 SNA 명령은 닫b 프로세스의)러 유형 동H! 흐를 수 있습니다.

- **SLI_CLOSE** 정상
  - 2차 시작 닫b

    SLI n플리케이션 프로W램이 **lua_flag.close_abend**! 0으로 설정된 **SLI_CLOSE** 명령n를 발행한 후! SLI는 다음 프로세스를 수행합니다.

    RSHUTD 명령을 작성합니다.
    RSHUTD 명령 응답을 읽m 프로세스합니다.
    CLEAR 명령을 읽m 프로세스합니다(필d한 fl).
    CLEAR 명령 응답을 작성합니다(필d한 fl).

UNBIND 명령 응답을 읽m 프로세스합니다.
UNBIND 명령 응답을 9니다.
RUI 세션을 종료합니다.

– 1차 시작 닫b

SHUTD 명령을 읽m n플리케이션 SESSION_END_REQUESTED 상태를 제x합니다.

SLI n플리케이션 프로W램이 **lua_flag.close_abend**! 0으로 설정된 **SLI_CLOSE** 명령n를 발행한 후! SLI는 다음 프로세스를 수행합니다.

CHASE 명령을 작성합니다.
CHASE 명령 응답을 읽m 프로세스합니다.
SHUTC(Shutdown Complete) 명령을 작성합니다.
SHUTC 명령 응답을 읽m 프로세스합니다.
CLEAR 명령을 읽m 프로세스합니다(필d한 f l).
CLEAR 명령 응답을 작성합니다(필d한 f l).
UNBIND 명령 응답을 읽m 프로세스합니다.
UNBIND 명령 응답을 9니다.
RUI 세션을 종료합니다.

– **SLI_CLOSE** Abend

- SLI n플리케이션 프로W램이 **lua_flag1.close_abend**! 0으로 설정된 **SLI_CLOSE** 명령n를 발행한 후! SLI는 RUI 세션을 종료합니다.

SLI_CLOSE 명령n의 O료는 LU-LU 세션이 바인드해제되며 LU! 대한 세션 x음 b능이 SSCP! 통지됐음을 나타냅니다. **SLI_CLOSE** 명령n! O료된 후! 또다른 **SLI_OPEN**를 제\하m 세션! 대해 다른 SLI 명령을 발행할 수 x습니다. **SLI_CLOSE** 명령n! 수신되면 모든 보류중인 명령이 종료됩니다.

주:

1. RUI를 사k하) 성립된 세션을 닫b 위해 이 b능을 사k하지 마십시@.
2. **SLI_CLOSE** 정상을 발행하b 전! 제x해_ 할 모든 응답이 호스트! 송신되z음을 확인하십시@. 응답을 제x해_ 하는 f l SLI는 자동으로 CLOSE 유형을 ABEND로 변f합니다.

   LUA n플리케이션 프로W램이 데이타를 무시하는 f l CLOSE 유형은 자동으로 ABEND로 변f됩니다. 호스트!서 모든 데이타를 수신하b 위해 **SLI_RECEIVE** 명령n를 사k하는 M은 좋은 프로W래밍 f험입니다. W렇지 J은 f l SLI는 데이타! 9\ d청인 f l라도 응답이 제x돼_ 함을 !정할 수 있으며 CLOSE 유형을 ABEND로 변f하십시@.

SLI_OPEN

이 명령n는 링크!서 세션 레벨 통신을 d청하는 n플리케이션 프로W램 ! 대한 SNA 세션을 1니다. 세션 레벨 함수는 세션을)는 n플리케이션 프로W램의 측면!서 SNA 명령을 발행합니다. SLI 함수는 복수 RUI 함수를 수행하) LU-LU 세션을 만들b 때문! LUA n플리케이션 프로W램이 단순화됩니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb

LUA_VERB_SLI

LUA 명령n ! 대한 명령n 코드 표시b.

lua_verb_length

명령n 제n 블럭의 f이. 이 숫자는 SLI_OPEN 명령n ! 대해 SLI ! 9상하는 f이M 0F_ 합니다.

lua_opcode

LUA_OPCODE_SLI_OPEN

lua_correlator

LUA n플리케이션 프로W램이 이 명령n를 프로W램이 제x하는 다른 정보M |련시키는 M을 돕b 위해 제x할 수 있는 *. 이 매3변수는 Windows LUA 인터페이스! 의해 사k되지 J습니다.

lua_luname

ASCII로 된 로컬 LU명. 이름이 8자보다 작은 fl, x백으로 채v_ 합니다.

이 매3변수는 **SLI_OPEN**!서 필d로 합니다. **lua_sid** 매3변수! 0인 fl!만 다른 명령n!서 이 매3변수를 필d로 하지만 모든 명령n!서 **lua_luname** 매3변수를 사k하면 디버k을 더 k이하T 하며 특히 다중 LU! 8성되는 fl W렇습니다.

다음 정보는 통신 서버 Windows 95 및 Windows NT SNA API 클라이p트!만 적k됩니다.

" 사k자! 대한 디폴트 LUA 세션명은 INI 8성이나 LDAP 중 하나의 해당 8성 유틸리티를 사k하) 지정될 수 있습니다.

3270 !뮬레이터M 0은 LUA 프로W램은 직접 하나의 세션 이름을 지정하b보다는 디폴트 LUA 세션명을 사k하도록 선택할 수 있습니다. LUA 프로W램 **lua_name** 필드를 2진 0이나 ASCII x백으로 설정한 **SLI_OPEN** 명령n를 수행할 때 SLI API는 8성된 디폴트 LUA 세션명을 사k합니다.

lua_data_length
송신되는 비형식화 LOGON 또는 INITSELF 데이타의 f이.

lua_data_ptr
n플리케이션의 데이타 버퍼! 대한 포인터. 이 버퍼는 데이타 및 SNA 명령! 사k되므로 버퍼의 내k은 보통 EBCDIC입니다.

이 데이타 버퍼!는 다음 중 하나! 들n 있습니다.

- lua_init_type 매3변수! INITSELF로 2차 시작을 지정하는 f l 필d한 모든 n플리케이션 프로W램 데이타! 있는 사k자의 SNA INITSELF d청 단위(RU)! 채v집니다. INITSELF!는 모드명 및 PLU명z 0은 사k자 정보! 들n 있습니다. 자세한 내k은 *시스템 네트v크 체h(SNA) 제품 형식*을 참조하십시@.
- **lua_init_type** 매3변수! 비형식화 LOGON 메세지로 2차 시작을 지정할 때 SSCP-정상 흐름! 송신된 LOGON 메세지.
- 세션이 1차 시작인 f l, 이 버퍼는 사k되지 J으며 **lua_data_ptr** 매3변수는 0이n _ 합니다.

lua_post_handle
비동b 통지! 이벤트로 성립되는 f l, **lua_post_handle**!는 신호되는 이벤트의 핸들이 들n 있습니다.

lua_encr_decr_option
0호화! 지x되지 J습니다.

lua_init_type
LU-LU 세션이 Windows LUA 인터페이스! 의해 n떻T 초b화되는!를 정의합니다. 유효한 *은 다음z 0습니다.

LUA_INIT_TYPE_SEC_IS
2차 시작. OPEN의 데이타 버퍼! 제x된 INITSELF 명령을 송신합니다.

LUA_INIT_TYPE_SEC_LOG
OPEN의 데이타 버퍼! 지정된 비형식화 LOGON 메세지! 있는 2차 시작.

LUA_INIT_TYPE_PRIM
1차 시작. BIND시 대b.

LUA_INIT_TYPE_PRIM_SSCP
SSCP W세스로 1차 시작.

lua_session_type
SLI! UNBIND 유형 X'01', UNBIND 정상을 프로세스하는 방식을 정의하는 *. 유효한 *은 다음z 0습니다.

LUA_SESSION_TYPE_NORMAL
UNBIND 정상이 1차 논리 장치!서 수신되는 f l, SLI는 ` 정 응답을 송신하며 사k불능 NOTIFY를 SSCP로 흐르T 하는 **RUI_TERM**을 발행합니다. SSCP-LU 흐름이 사k불능화됩니다. 이M은 이 매3변수k 디폴트 *입니다.

LUA_SESSION_TYPE_DEDICATED

UNBIND 정상이 1차 논리 장치 ! 서 수신되는 f l, SLI는 ` 정 응답을 송신하며, 새 BIND, 선택적 CRV 및 STSN W리m SDT 명령이 수신될 때n지 SLI 세션이 일시중단됩니다. 이 f l SLI는 **RUI_TERM**를 발행하지 J으며 사k불능 NOTIFY는 SSCP로 흐르지 J습니다.

LUA_SESSION_TYPE_DEDICATED는 SNA API 클라이p트 ! 의해 지x되지 J습니다.

lua_wait

호스트 ! 다음 메세지 중 하나를 송신한 후 ! 자동으로 INITSELF 또는 LOGON 메세지의 전송을 재시도하b 전 ! 대b할 SLI ! 대한 초시# 번호(최대 65 535n지).

- INITSELF 또는 LOGON 메세지 ! 대한 부정 응답z 2차 리턴 코드는 다음 * 중 하나입니다.
  - RESOURCE_NOT_AVAILABLE(X'08010000')
  - SESSION_LIMIT_EXCEEDED(X'08050000')
  - SSCP_LU_SESS_NOT_ACTIVE(X'0857nnnn') b서 *nnnn*은 X'0002' 입니다)
  - SESSION_SERVICE_PATH_ERROR(X'087Dnnnn') b서 *nnnn*은 X'0000'입니다)
- 네트v크 서비스 프로시듀n @류(NSPE) 메세지
- 프로시듀n @류를 나타내는 NOTIFY 명령

lua_wait의 *이 0이면 재시도 ! 일n나지 J습니다. 이 매3변수는 SLU ! 의해 시작된 세션 ! 만 적k합니다. PLU ! 세션을 시작하는 f l, **lua_wait**는 무시됩니다.

lua_extension_list_offset

명령n 제n 블럭의 시작으로부터 사k자 제x DLL의 확장 리스트n지 @프셋을 지정합니다. *은 단n 바n더리의 처음이n_ 합니다. 확장 리스트 ! x다면 *은 0으로 설정되n_ 합니다.

lua_routine_type

다음 모듈 및 프로시듀n명의 루틴 유형. 유효한 입력항목은 다음z 0습니다.

lua_routine_type_bind
바인드 루틴

lua_routine_type_crv
0호화 벡터 루틴

주: 0호화는 현재 지x되지 J습니다.

lua_routine_type_sdt
시작 데이타 트래픽(SDT) 루틴

lua_routine_type_sdt는 SNA API 클라이ｐ트！ 의해 지ｘ되지 Ｊ습니다.

lua_routine_type_stsn
순서 번호 설정 및 Ｋ사(STSN) 루틴

lua_routine_type_end
루틴 리스트！ 대한 종료 분리문자.

lua_module_name
사ｋ자 제ｘ ASCII 모듈명을 제ｘ합니다. 매３변수는 최대 8자의 ｆ이를 ！지며 나머지 바이트는 X'00'으로 설정합니다.

lua_procedure_name
ASCII로 사ｋ자 제ｘ DLL 프로시듀ｎ명을 제ｘ합니다. 매３변수는 최대 22자의 ｆ이를 ！지며 나머지 바이트는 X'00'으로 설정합니다.

리턴되는 매개변수

명령ｎ！ ０료되면 다음 매３변수！ 리턴됩니다.

lua_flag2.async
이 명령ｎ！ 비동ｂ적으로 ０료함을 나타내는 플래Ｗ.

lua_sid
차후의 명령ｎ！ 사ｋ되는

- UNBIND 응답을 작성하며 차후의 BIND를 수신하도록 준비합니다.
- STSN 명령을 읽m 프로세스합니다(필d한 f l).
- STSN 응답을 작성합니다(필d한 f l).
- SDT 명령을 읽m 프로세스합니다.
- SDT 응답을 작성합니다.
- BIND, STSN 및 SDT 명령이 **SLI_OPEN** 명령n !서 n플리케이션 프로W램! 의해 지정되면 이들 명령을 프로세스하b 위해 사k자 루틴으로 !십시@.

SLI_OPEN 명령n는 SDT 명령! 대한 응답을 통해 모든 SNA 메세지 트래픽을 처리합니다.

n플리케이션 프로W램은 **SLI_OPEN** 명령n를 발행하) **lua_luname** 매3변수!서 정의된 LUA LU를 선택합니다. 이 필드는 x백으로 채v져_ 하는 ASCII 문자 - 입니다.

lua_init_type 매3변수는 SLI! LU 세션을 설정하는 방식을 K려줍니다. 다음 리스트는 초b화 I션을 설명합니다.

- INITSELF로 2차 초b화

  이 I션의 f l **lua_init_type** 매3변수를 LUA_INIT_TYPE_SEC_IS로 설정하십시@. 이 I션을 사k하는 f l, INITSELF!는 모드명 및 PLU명z 0이 호스트! 필d로 하는 세션 m유의 모든 정보! 들n 있으므로 n플리케이션 프로W램은 **SLI_OPEN** 명령n !서 사k되는 INITSELF 명령을 제x해_ 합니다. **lua_data_ptr** 매3변수는 INITSELF의 주소를 제x하며 **lua_data_length** 매3변수는 W f이를 제x합니다.

- 비형식화 LOGON 메세지로 2차 초b화

  이 I션의 f l **lua_init_type** 매3변수를 LUA_INIT_TYPE_SEC_LOG로 설정하십시@. 비형식화 LOGON 메세지를 .는 2차 초b화!서 **lua_data_ptr** 매3변수!는 **lua_data_length** 매3변수! 지정한 f이의 사k자 EBCDIC LOGON 메세지의 주소! 들n 있습니다.

- 1차 초b화

  이 I션의 f l **lua_init_type** 매3변수를 LUA_INIT_TYPE_PRIM으로 설정하십시@. 1차 초b화!서 SLU는 호스트M의 세션을 시작하b 위해 F무M도 하지 J습니다. 호스트! BIND 명령 및 차후의 SDT 명령으로 세션을 시작할 때n지 **SLI_OPEN**은 IN_PROGRESS으로 남F 있습니다.

- SSCP W세스로 1차 초b화

  이 I션의 f l **lua_init_type** 매3변수를 LUA_INIT_TYPE_PRIM_SSCP로 설정하십시@. SSCP W세스를 .는 1차 초b화!서 SLI는 세션을 시작하b 위해 호스트! 명령을 송신하지 J습니다. 대신, SLI는 n플리케이션 프로W램이 SSCP 정상 흐름 데이타k **SLI_SEND** 및 **SLI_RECEIVE** 명령n를 발행하) INITSELF 명령이나 LOGON 메세지를 송신하m W 응답을 수신하T 합니다. 이 I션을 사k하) n플리케이션 프로W램은 2차 초b화 유형! 대한 M z 마찬!지로 INITSELF 또는 LOGON 메세지로 제한되지 J습니다. 이M은 **SLI_OPEN**이 O료하b 전! n플리케이션 프로

W램이 SLI 명령n를 발행하도록 허k하는 유일한 **SLI_OPEN** 유형입니다. **SLI_OPEN** 명령n ! 발행된 후 ! n플리케이션 프로W램은 **SLI_BID** 또는 **SLI_RECEIVE**를 발행하) INIT_COMPLETE 상태를 r을 수 있습니다. 이 상태는 SSCP 정상 흐름 데이타k **SLI_SEND** 및 **SLI_RECEIVE** 명령n의 발행을 n플리케이션 프로W램이 시작할 수 있도록 K립니다.

선택적 **lu_session_type** 프로세스는 UNBIND 유형 X'01', UNBIND 정상을 프로세스하는 방법을 SLI ! K립니다. 이 매3변수는 **SLI_OPEN** 명령n ! 초b 매3변수 점K을 전달한 후 ! 유효하며 **SLI_CLOSE** abend ! 발행되E나 SLI ! **RUI_TERM**을 발행할 때n지 유효합니다. 다음 리스트는 표준 UNBIND 및 전k UNBIND 프로세스를 설명합니다.

- 표준 UNBIND 정상 프로세싱 **SLI_CLOSE** 정상

  이 I션의 fl **lua_session_type** 매3변수를 LUA_SESSION_TYPE_NORMAL로 설정하십시@. 이M이 디폴트 *입니다. 이 I션을 사k하) SLI는 1차 LU ! 송신하는 UNBIND 정상 ! `정 응답을 송신하며 **RUI_TERM**을 발행하) 사k불능 NOTIFY를 SSCP ! 흐르T 합니다. 이들 조치는 다음을 수행합니다.

  – 이M은 LU-LU 세션을 종료합니다.

  – SLU ! 새 BIND를 프로세스할 수 x음을 SSCP 및 PLU ! 표시합니다. 수신된 새 BIND는 E부됩니다.

  – SSCP-LU 세션 ! 서 데이타 ! 흐르지 못하T 합니다.

    SLI ! UNBIND 9\ 유형 X'02'(BIND ! p 이n지는 UNBIND)를 수신하면 **RUI_TERM**을 발행합니다.

  – 전k UNBIND 정상 프로세싱

    이 I션의 fl **lua_session_type** 매3변수를 LUA_SESSION_TYPE_DEDICATED로 설정하십시@. 이 I션을 사k하) SLI는 1차 논리 장치 ! 송신하는 UNBIND 정상 ! `정 응답을 송신합니다. W러나 SLI는 **RUI_TERM**을 발행하지 J습니다. SSCP-LU 세션의 상태는 변f되지 J습니다(사k ! 능). SLI 세션은 BIND, 선택적 CRV 및 STSN W리m SDT 명령이 수신될 때n지 일시중단됩니다. 새 BIND를 b다리는 SLI 세션은 **SLI_CLOSE** Abend를 발행하) 종료될 수 있습니다.

    SLI는 UNBIND 9\ 유형 X'02' 또는 유형 X'01'을 수신하면 **RUI_TERM**을 발행합니다.

    이 I션은 1차 LU ! p 이n지는 BIND를 .는 UNBIND를 송신할 수 x을 때 유k하지만 UNBIND 정상이 송신될 때 이 유형의 활동을 b대합니다.

어플리케이션 제x BIND, SDT 또는 STSN 루틴

- n플리케이션 프로W램이 BIND, SDT 또는 STSN 루틴을 제x하는 fl, DLL 모듈명 및 프로시듀n #트리 포인트는 **SLI_OPEN** 확장 루틴 리스트 ! 전달됩니다. 해당 SNA d청이 수신되면 이들 루틴은 **SLI_OPEN** 동H 호출됩니다. BIND 루틴이 제x되지 J으면 SLI는 제한된 BIND 점K을 수행하며 필d하면 응답합니다. STSN 루틴이 제x되지 Jm STSN d

청이 수신되면 SLI는 정보! 사k!능하지 J음을 나타내b 위해 `정 응답을 발행합니다. SDT 루틴이 제x되지 Jm SDT d청이 수신되면 SLI는 `정 응답을 발행합니다.

통지하기

- **lua_prim_rc** 매3변수!서 **SLI_OPEN**을 "9"로 통지하는 M은 **SLI_OPEN**이 0료됐으며 LU-LU 데이타 흐름 세션이 설정됐음을 의미합니다. 세션이 -린 후! n플리케이션 프로W램은 **SLI_SEND**, **SLI_RECEIVE**, **SLI_PURGE**, **SLI_BID** 또는 **SLI_CLOSE** 명령n를 발행할 수 있습니다.

세션 복구

- SLI는 n플리케이션 프로W램! 대한 제한된 세션 복8를 제x합니다. SLI 명령n! **lua_prim_rc** 매3변수!서 SESSION_FAILURE로 0료하면 n플리케이션 프로W램은 **SLI_OPEN**을 재발행해_ 합니다. 이 상황!서 프로W램은 새 **SLI_OPEN** 명령n를 발행하b 전! **SLI_CLOSE** 명령n를 발행할 필d! x습니다.

보류중인 **SLI_OPEN** 종료

- 보류중인 **SLI_OPEN**을 종료하려면 **lua_flag1.close_abend** 매3변수를 1로 설정하) **SLI_CLOSE**를 발행하십시@.

SLI_PURGE

이 명령n는 미처리 **SLI_RECEIVE**를 제E합니다. **SLI_PURGE**는 WAIT I션으로 **SLI_RECEIVE** 명령n를 사k하는 n플리케이션 프로W램! 의해 필d로 할 수 있습니다. 9를 들n **SLI_RECEIVE** 명령n! 지정된 시# 내! O료되지 J으면 n플리케이션 프로W램은 **SLI_PURGE**를 발행할 수 있습니다. n플리케이션 프로W램은 제E할 **SLI_RECEIVE**를 지정하b 위해 **lua_data_ptr** 매3변수!서 **SLI_RECEIVE** 명령n 제n 블럭을 제x합니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb

LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length

명령n 제n 블럭의 f이. 이 숫자는 **SLI_PURGE** 명령n! 대해 SLI! 9상하는 f이M 0F_ 합니다.

lua_opcode

LUA_OPCODE_SLI_PURGE

명령n! 대한 작w 코드.

lua_correlator

LUA n플리케이션 프로W램이 이 명령n를 프로W램이 제x하는 다른 정보M |런시키는 M을 돕b 위해 제x할 수 있는 *. 이 매3변수는 LUA 인터페이스! 의해 무시됩니다.

lua_luname

ASCII로 된 로컬 LU명. 이름이 8자보다 작은 fl, x백으로 채v_ 합니다. LUA는 **lua_sid**! 0인 fl!만 이 매3변수를 조사합니다. 모든 명령n!서 **lua_luname** 매3변수를 사k하면 디버W를 더k이하T 하며 특히 다중 LU! 8성되는 fl W렇습니다.

lua_sid

사k되는 세션을 식별하는 세션 ID로 **SLI_OPEN**이 리턴합니다. 이 매3변수! 0이면 **lua_luname** 매3변수는 식별k으로 사k됩니다.

lua_data_ptr

제E되는 n플리케이션 프로W램 **SLI_RECEIVE** 명령n 제n 블럭!대한 포인터.

lua_post_handle

비동b 통지! 이벤트로 성립되는 fl, **lua_post_handle**!는 신호되는 이벤트의 핸들이 들n 있습니다.

리턴되는 매개변수

명령ｎ！ ０료되면 다음 매３변수！ 리턴됩니다.

lua_flag2.async
이 명령ｎ！ 비동ｂ적으로 ０료함을 나타내는 플래Ｗ.

lua_prim_rc
명령ｎ ｂ능으로 설정되는 1차 리턴 코드. 자세한 내ｋ은 377페이지의 『부록B. LUA 명령ｎ 리턴 코드』를 참조하십시@.

lua_sec_rc
명령ｎ ｂ능으로 설정되는 2차 리턴 코드. 자세한 내ｋ은 377페이지의 『부록B. LUA 명령ｎ 리턴 코드』를 참조하십시@.

사용시 주의사항

SLI_RECEIVE！ 제Ｅ되면 **SLI_RECEIVE**는 CANCELED 1차 리턴 코드로 종료하며 **SLI_PURGE**는 OK 1차 리턴 코드로 ０료됩니다.

SLI_RECEIVE

이 명령n는 데이타 또는 상태 코드를 n플리케이션 프로W램으로 전송합니다. 또한 SLI_RECEIVE는 세션의 현재 상태를 Windows LUA n플리케이션! 제x합니다.

LU-LU 세션 흐름! 대한 **SLI_RECEIVE** 명령n는 -린 세션!서만 발행될 수 있습니다. **SLI_OPEN** 초b화 유형이 SSCP W세스를 .는 1차인 f l, n플리케이션 프로W램은 **SLI_OPEN** 명령n! 보류중일지라도 SSCP-LU 정상 흐름! 대해 **SLI_RECEIVE** 명령n를 발행할 수 있습니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb

LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length

명령n 제n 블럭의 f이. 이 숫자는 SLI_RECEIVE 명령n! 대해 SLI! 9상하는 f이M 0F_ 합니다.

lua_opcode

LUA_OPCODE_SLI_RECEIVE

lua_correlator

LUA n플리케이션 프로W램이 이 명령n를 프로W램이 제x하는 다른 정보M |런시키는 M을 돕b 위해 제x할 수 있는 *. 이 매3변수는 LUA 인터페이스! 의해 무시됩니다.

lua_luname

ASCII로 된 로컬 LU명. 이름이 8자보다 작은 f l, x백으로 채v_ 합니다. LUA는 **lua_sid**! 0인 f l!만 이 매3변수를 조사합니다. 모든 명령n!서 **lua_luname** 매3변수를 사k하면 디버W를 더k이하T 하며 특히 다중 LU! 8성되는 f l W렇습니다.

lua_sid

사k되는 세션을 식별하며 SLI_OPEN! 의해 리턴되는 세션 ID. 이 매3변수! 0이면 **lua_luname** 매3변수는 식별k으로 사k됩니다.

lua_max_length

데이타를 수신하b 위해 사k되는 버퍼의 f이.

lua_data_ptr

호스트 n플리케이션!서 수신된 데이타를 SLI! 위치시키는 버퍼! 대한 포인터. 이 버퍼는 데이타 및 SNA 명령! 사k되므로 버퍼의 내k은 보통 EBCDIC입니다.

lua_post_handle
Windows NT ! 서 비동 b 통지 ! 이벤트로 성립되는 f l, **lua_post_handle** ! 는 신호되는 이벤트의 핸들이 들n 있습니다.

lua_flag1.bid_enable
LUA ! LUA n 플리케이션 프로W램 측면 ! 서 SLI_BID 명령n를 재사k해_ 할지)부를 지정하는 플래W.

lua_flag1.nowait
읽을 데이타 ! x을 때 리턴 코드 NO_DATA로 SLI_RECEIVE 명령n를 통지하b 위해 SLI ! K리는 플래W. 다중 RU 체인의 첫번째 RU ! 수신되m **lua_flag1.nowait** I션이 선택되면 **lua_flag1.nowait** I션은 무시됩니다. SLI_RECEIVE 명령n는 IN_PROGRESS를 리턴하며 체인의 모든 RU ! 수신된 후 ! 비동b적으로 0료합니다. 체인이 허k되면 **lua_flag1.nowait** I션은 사k하지 말F_ 합니다.

lua_flag1의 하위 반 바이트 ! 는 메세지 세션 및 흐름을 b술하는 플래W ! 들n 있습니다. 흐름 플래W는 LUA n플리케이션 프로W램이 메세지를 수k할 수 있는 흐름을 b술합니다. 다음 플래W 중 최소한 하나 ! 설정되n_ 하지만 설정 플래W는 또다른 활동중인 **SLI_RECEIVE** 명령n ! 설정된 플래W를 중첩할 수 x습니다.

lua_flag1.sscp_exp
SSCP d^ 흐름을 지정하는 플래W.

lua_flag1.sscp_norm
SSCP 정상 흐름을 지정하는 플래W.

lua_flag1.lu_exp
LU d^ 흐름을 지정하는 플래W.

lua_flag1.lu_norm
LU 정상 흐름을 지정하는 플래W.

리턴되는 매개변수

명령n ! 0료되면 다음 매3변수 ! 리턴됩니다.

lua_data_length
수신되는 데이타의 f이.

lua_th 메세지 ! 대한 SNA 전송 헤더(TH) ! 들n 있는 6바이트 매3변수.

lua_rh
메세지 ! 대한 SNA d청/응답 헤더(RH) ! 들n 있는 3바이트 매3변수.

lua_message_type
SNA 데이타 및 명령의 유형. SLI n플리케이션 프로W램이 데이타를 송신하려m 할 때 n플리케이션 프로W램은 이 매3변수를 설정해_ 합니다. 유효한 메세지 유형은 다음z 0습니다.

LUA_MESSAGE_TYPE_LU_DATA

LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SIGNAL

LU_DATA, LUSTAT_LU, LUSTAT_SSCP 및 SSCP_DATA는 SNA 명령이 F닙니다.

lua_flag2.async
이 명령n ! 비동b적으로 O료함을 지정하는 플래W.

lua_flag2.sscp_exp
SSCP d^ 흐름을 지정하는 플래W.

lua_flag2.sscp_norm
SSCP 정상 흐름을 지정하는 플래W.

lua_flag2.lu_exp
LU d^ 흐름을 지정하는 플래W.

lua_flag2.lu_norm
LU 정상 흐름을 지정하는 플래W.

lua_prim_rc
명령n b능으로 설정되는 1차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

lua_sec_rc
명령n b능으로 설정되는 2차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

사용시 주의사항

SLI_RECEIVE는 호스트로부터 응답, SNA 명령 및 d청 단위(RU) 데이타를 수신합니다. 또한 **SLI_RECEIVE**는 세션의 상태를 Windows LUA n플리케이션! 제x합니다. **SLI_OPEN** d청은 **SLI_RECEIVE**! 발행되b 전! O료되n_ 합니다. W러나 lua_init_type을 LUA_INIT_TYPE_PRIM_SSCP로 설

- CLEAR! 발행되면 SDT 명령이 수신될 때n지 세션이 일시중단됩니다.
- UNBIND 유형 X'02'(BIND! p 이n지는 UNBIND)! 발행되면 세션은 BIND, 선택적 CRV 및 STSN W리m SDT 명령이 수신될 때n지 일시중단됩니다. 모든 사k자 확장 루틴은 재진입되n_ 합니다.
- UNBIND 유형 X'01'(UNBIND 정상)이 발행되m 이 세션! 대한 **SLI_OPEN** 명령n! LUA_SESSION_TYPE_DEDICATED의 **lua_session_type**을 지정한 fl, 세션은 BIND, 선택적 CRV 및 STSN W리m SDT 명령이 수신될 때n지 일시중단됩니다. 이들 명령을 프로세스하b 위해 제x된 사k자 확장 루틴은 재진입되n_ 합니다.

  CLEAR, UNBIND 유형 X'02' 또는 UNBIND 유형 X'01'이 발행된 이후! n플리케이션은 NOT_READY 상태를 읽b 전! SSCP 데이타를 송신하m NOT_READY 상태를 읽은 후! SSCP 데이타를 송수신할 수 있습니다.

- SESSION_END_REQUESTED

  SHUTD 명령이 호스트!서 수신됐음을 나타냅니다. !능하T 되면 바로 SLI n플리케이션이 세션을 종료하도록 호스트! d청합니다.

  n플리케이션이 세션을 종료할 준비! 되면 **SLI_CLOSE** 또는 **SLI_CLOSE** 정상을 발행해_ 합니다.

- INIT_COMPLETE

  RUI_INIT 명령n! **SLI_OPEN** 프로세스 동H! O료됐음을 나타냅니다. **SLI_OPEN lua_init_type** 매3변수! LUA_INIT_TYPE_PRIM_SSCP인 fl!만 이 상태! 리턴됩니다.

  이 상태! 수신된 후! n플리케이션은 SSCP 정상 흐름!서 데이타를 송수신할 수 있습니다.

리턴 코드 이\! 호스트 n플리케이션!서 송신한 d청 단위(RU)! 9\d청(EXR)으로 변환되면 추!적인 SNA (지 데이타! 리턴될 수 있습니다. 다음z 0이 리턴되는 명령n 매3변수 *으로 **SLI_RECEIVE**! O료되n EXR이 표시됩니다.

매3변수
: 설정

lua_prim_rc
: 9(X'0000')

lua_sec_rc
: 9(X'00000000')

lua_rh.rri
: 비트 OFF(d청 단위(RU))

lua_rh.sdi
: 비트 ON((지 데이타 포함)

이들 조G ! 서 d청이 EXR로 변환되z으며 최대 7바이트의 정보 ! n플리케이션 버퍼 ! 서 리턴됩니다. 데이타 버퍼의 정보 형식은 다음z 0습니다.

- 바이트 0 — 3 ! 는 K출된 @류를 정의하는 (지 데이타 ! 들n 있습니다. LUA ! d청을 EXR로 변환하면 (지 데이타는 다음 * 중 하나입니다.

(지 데이타	바이트 0-3의 *
LUA_MODE_INCONSISTENCY	X'08090000'
LUA_BRACKET_RACE_ERROR	X'080B0000'
LUA_BB_REJECT_NO_RTR	X'08130000'
LUA_RECEIVER_IN_TRANSMIT_MODE	X'081B0000'
LUA_CRYPTOGRAPHY_FUNCTION_INOP	X'08480000'
LUA_SYNC_EVENT_RESPONSE	X'10010000'
LUA_RU_DATA_ERROR	X'10020000'
LUA_RU_LENGTH_ERROR	X'10020000'
LUA_INCORRECT_SEQUENCE_NUMBER	X'20010000'
LUA_LCC_NOT_SUPPORTED	X'20010000'

lua_peek_data ! 서 바이트 4-6으로 리턴되는 정보 ! 는 x래 d청 단위(RU)의 최초 3바이트n지 들n 있습니다.

SLI_SEND

이 명령n는 LUA n플리케이션 프로W램!서 통신 링크로 사k자 데이타, SNA 명령 및 SNA 응답을 전송합니다. LU-LU 세션 흐름! 대한 **SLI_SEND** 는 이전! -린 세션!서만 발행될 수 있습니다. **SLI_OPEN** 초b화 유형이 SSCP W세스를 .는 1차이m INIT_COMPLETE 상태! 이루n지면 n플리케이션 프로W램이 **SLI_SEND**를 발행하) 데이타를 SSCP-LU 정상 흐름! 전송할 수 있습니다.

LUA n플리케이션은 정의된 " LUA LU! 대해 동시! 두3의 활동중인 **SLI_SEND** 명령n를 !질 수 있습니다. 두 3의 명령n는 두 3의 별3 흐름k일 수 있습니다.

제공되는 매개변수

n플리케이션은 다음 매3변수를 제x합니다.

lua_verb

LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length

명령n 제n 블럭의 f이. 이 숫자는 **SLI_SEND** 명령n! 대해 SLI ! 9상하는 f이M 0F_ 합니다.

lua_opcode

LUA_OPCODE_SLI_SEND

이 명령n! 대한 작w 코드.

lua_correlator

LUA n플리케이션 프로W램이 이 명령n를 프로W램이 제x하는 다른 정보M |련시키는 M을 돕b 위해 제x할 수 있는 *. SLI는 이 매3변수를 무시합니다.

lua_luname

ASCII로 된 로컬 LU명. 이름이 8자보다 작은 fl, x백으로 채v _ 합니다. LUA는 **lua_sid**! 0인 fl!만 이 매3변수를 조사합니다. 모든 명령n!서 **lua_luname** 매3변수를 사k하면 디버W를 더k이하T 하며 특히 다중 LU! 8성되는 fl W렇습니다.

lua_sid

사k되는 세션을 식별하며 **SLI_OPEN**! 의해 리턴되는 세션 ID. 이 매3변수! 0이면 **lua_luname** 매3변수는 식별k으로 사k됩니다.

lua_data_length

송신되는 데이타의 f이.

lua_data_ptr

호스트 n플리케이션! 송신되는 n플리케이션 프로W램! 대한 포인터. 이 버퍼는 데이타 및 SNA 명령! 사k되므로 버퍼의 내k은 보통 EBCDIC입니다.

lua_post_handle

비동b 명령n의 O료를 통지하b 위해 사k되는 4바이트 핸들.

lua_th.snf

RU의 순서 번호.

lua_rh

메세지! 대한 SNA d청/응답 헤더(RH)! 들n 있는 3바이트 매3변수.

lua_message_type

SNA 데이타 및 명령의 유형. SLI n플리케이션 프로W램이 데이타를 송신하려m 할 때 n플리케이션 프로W램은 이 매3변수를 설정해_ 합니다. SNA 명령! 대한 자세한 내k은 *시스템 네트v크 체h(SNA) 제품 형식*을 참조하십시@. 유효한 메세지 유형은 다음z 0습니다.

LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RQR
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SSCP_DATA

lua_flag1.sscp_exp

SSCP d^ 흐름을 지정합니다.

lua_flag1.sscp_norm

SSCP 정상 흐름을 지정합니다.

lua_flag1.lu_exp

LU d^ 흐름을 지정합니다.

lua_flag1.lu_norm
LU 정상 흐름을 지정합니다.

리턴되는 매개변수

명령 n ! 실행되면 LUA는 다음 매 3 변수를 리턴합니다.

lua_data_length
수신된 peek 데이타의 f 이.

lua_th 메세지 ! 대한 SNA 전송 헤더(TH) ! 들 n 있는 6바이트 매 3 변수.

lua_flag2.async
이 명령 n ! 비동 b 적으로 0 료함을 나타내는 플래 W .

lua_flag2.sscp_exp
SSCP d ^ 흐름을 지정합니다.

lua_flag2.sscp_norm
SSCP 정상 흐름을 지정합니다.

lua_flag2.lu_exp
LU d ^ 흐름을 지정합니다.

lua_flag2.lu_norm
LU 정상 흐름을 지정합니다.

lua_sequence_number
SLI_SEND 명령 n ! 대한 체인의 최초 RU 또는 유일한 RU 순서 번호. 이 M 은 바이트 9 ` 되지 J 습니다.

lua_prim_rc
명령 n b 능으로 설정되는 1차 리턴 코드. 자세한 내 k 은 377페이지의 『부록B. LUA 명령 n 리턴 코드』를 참조하십시 @ .

lua_sec_rc
명령 n b 능으로 설정되는 2차 리턴 코드. 자세한 내 k 은 377페이지의 『부록B. LUA 명령 n 리턴 코드』를 참조하십시 @ .

사용시 주의사항

SLI_SEND는 RH 및 TH 비트, 흐름 플래 W M 0 은 **lua_message_type** 매 3 변수 ! Y E 한 특정 프로세스를 수행합니다. 9 를 들 n n 플리케이션이 **lua_message_type** 매 3 변수를 X'84'(CHASE)로 설정하면 SLI 8 성 d 소는 자동으로 **lua_rh** 매 3 변수를 X'4B8000'로 설정합니다. 293페이지의 표 17은 해당되는 f l n 플리케이션 프로 W 램이 설정해 _ 하는 매 3 변수(현재 프로 W 램 ! 제 x 된)를 보) 줍니다.

표 17. 메세지 유형! YE한 매3변수 설정

	lua_message_type 매3변수의 *						
SLI_SEND 매3변수	**LU_DATA SSCP_DATA**	**RSP**	**BID, BIS, RTR**	**CHASE QC**	**QEC, RELQ, SBI, SIG**	**RQR**	**LUSTAT_LU LUSTAT_SSCP**
lua_rh	FI, DR1I, DR2I, RI, BBI, EBI, CDI, CSI, EDI	RI	SDI, QRI	SDI, QRI, EBI, CDI	SDI	0	SDI, QRI, DR1I, DR2I, RI, BBI, EBI, CDI
lua_th	0	SNF	0	0	0	0	0
lua_data_ptr	필수(데이타! x는 fl 0)	필수(데이타! x는 fl 0)	0	0	0	0	필수
lua_data_length	필수	필수(데이타! x는 fl 0)	0	0	0	0	필수
lua_flag1 흐름 플래그	0	필수(하나 설정)	0	0	0	0	0

SLI_SEND 명령n는 **lua_data_ptr** 매3변수!서 지정한 위치!서 **lua_data_length**! 지정한 f이 만큼 데이타를 전송합니다. SLI는 필d하면 데이타를 체인,a합니다. **SLI_SEND**는 동b적으로 또는 비동b적으로 O료할 수 있습니다. n플리케이션 프로W램이 SLI! 대한 호출로부터 리턴하면 **lua_flag2.async** 플래W는 명령n! O료되는 방식을 나타냅니다. **lua_flag2.async**! ON으로 설정되면 IN_PROGRESS 1차 리턴 코드는 명령n! 수신되n 진행중임을 나타냅니다. OK 1차 리턴 코드는 데이타나 명령이 RUI! 작성됐음을 나타냅니다. n플리케이션 프로W램은 SLI! 대한 호출로부터의 동b 리턴을 .는 **RUI_WRITE**를 사k하) 송신된 마지막 체인 d소의 순서 번호를 수신합니다. 모든 체인 d소! 2)진 후, n플리케이션 프로W램은 최종 리턴 코드M 종료 순서 번호를 TH!서 수신합니다. 9를 들n SLI! 체인을 송신하면서 **SLI_SEND** 작w이 O료되b 전! 호스트로부터 페이싱 응답을 b다려_ 하는 fl 이들 순서 번호는 다릅니다.

SLI! 응답을 송신하면 **SLI_SEND** 명령n!서 필d한 정보는 응답 유형! 좌l됩니다. 모든 응답의 fl, n플리케이션 프로W램은 다음 단h를 수행해_ 합니다.

- **lua_message_type** 매3변수를 LUA_MESSAGE_TYPE_RSP로 설정하십시@.
- 응답하는 d청! 해당하는 순서 번호(**lua_th.snf**)를 제x하십시@.
- 선택된 **lua_flag1** 흐름 플래W를 설정하십시@.

추!적인 매3변수를 제x하는 T칙은 다음z 0습니다.

- d청 코드만을 d8하는 `정 응답의 fl, n플리케이션 프로W램은 다음 매3변수를 제x해_ 합니다.
  - **lua_rh.ri**를 0으로 설정.

– **lua_data_length**를 0으로 설정.

SLI는 d청 코드를 채l b 위해 제x된 순서 번호를 참조합니다.

- 부정 응답의 f l, n플리케이션 프로W램은 다음 매3변수를 제x해_ 합니다.
  – **lua_rh.ri**를 1로 설정.
  – **lua_data_ptr**를 SNA (지 코드의 주소로 설정.
  – **lua_data_length**를 SNA (지 코드(4바이트)의 f이로 설정.

  SLI는 (지 데이타 다음의 d청 코드를 채s니다.

SLI_BIND_ROUTINE

이 명령n는 SNA BIND d청을 호스트!서 수신했음을 SLI n플리케이션 프로W램! K리며 n플리케이션 프로W램이 세션 프로토콜을 조사하T 합니다. **SLI_BIND_ROUTINE**은 **SLI_OPEN** 확장 리스트 바인드 루틴 필드! 지정한 프로W래머 제x DLL! 전달됩니다.

제공되는 매개변수

SLI_BIND_ROUTINE! 대한 다음z 0은 매3변수! SLI! 의해 제x됩니다.

lua_verb
LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length
명령n 제n 블럭의 f이.

lua_opcode
LUA_OPCODE_SLI_BIND_ROUTINE

루틴! 대한 작w 코드.

lua_luname
ASCII로 된 로컬 LU명.

lua_sid
사k되는 세션을 식별하며 **SLI_OPEN**! 의해 리턴되는 세션 ID.

lua_data_length
BIND RU의 f이.

lua_data_ptr
BIND RU! 대한 포인터. BIND RU!는 PLU명z 0은 EBCDIC 문자! 들n 있을 수 있습니다.

lua_th
BIND TH.

lua_rh
BIND RH.

리턴되는 매개변수

명령n! 0료되면 LUA는 다음 매3변수를 리턴합니다.

lua_prim_rc
LUA_OK

lua_data_length
송신되는 BIND 응답의 f이.

lua_prim_rc
명령ｎ ｂ능으로 설정되는 1차 리턴 코드. 자세한 내ｋ은 377페이지의 『부록B. LUA 명령ｎ 리턴 코드』를 참조하십시@.

사용시 주의사항

명령ｎ 제ｎ 블럭은 SLI！ 할당한 ｂｏ５＊！ 만들ｎ집니다. **lua_th** 및 **lua_rh** 매３변수의 내ｋ은 **SLI_BIND_ROUTINE** 명령ｎ 제ｎ 블럭！ 위치합니다. **lua_data_ptr** 매３변수는 BIND RU의 주소！ 들ｎ 있으며 **lua_data_length** 매３변수！는 RU의 ｆ이！ 들ｎ 있습니다.

SLI_BIND_ROUTINE은 확장 루틴이 **lua_prim_rc**로 리턴될 때 ０료되며 **lua_data_length** 매３변수는 **SLI_BIND_ROUTINE** 명령ｎ 제ｎ 블럭！ 설정됩니다. BIND RU를 BIND 응답으로 ｃ쳐２십시@. OK 1차 리턴 코드는 BIND！ 수ｋ됐음을 나타냅니다. 루틴이 BIND를 Ｅ부하면 1차 리턴 코드를 NEGATIVE_RSP로 설정하ｍ 부정 응답을 BIND 버퍼！ 넣으십시@. **lua_data_ptr** 매３변수를 수정하지 마십시@.

주: 이 루틴으로부터의 부정 응답은 **SLI_OPEN** 명령ｎ를 취소합니다. SLI는 SESSION_FAILURE의 1차 리턴 코드 MNEG_RSP_FROM_BIND_ROUTINE의 2차 리턴 코드를 리턴합니다.

SLI_STSN_ROUTINE

이 명령n는 SNA STSN d청을 호스트!서 수신했음을 SLI n플리케이션 프로W램! K리며 n플리케이션 프로W램이 STSN RU를 조사하m 응답을 준비하T 합니다. **SLI_STSN_ROUTINE**은 **SLI_OPEN** 확장 리스트 바인드 루틴 필드! 지정한 프로W래머 제x DLL! 전달됩니다.

제공되는 매개변수

SLI_STSN_ROUTINE! 대한 다음z 0은 매3변수! SLI! 의해 제x됩니다.

lua_verb
LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length
명령n 제n 블럭의 f이.

lua_opcode LUA_OPCODE_SLI_STSN_ROUTINE
루틴! 대한 작w 코드.

lua_luname
ASCII로 된 로컬 LU명.

lua_sid
사k되는 세션을 식별하며 **SLI_OPEN**! 의해 리턴되는 세션 ID.

lua_data_length
STSN RU의 f이.

lua_data_ptr
STSN RU! 대한 포인터.

lua_th
STSN TH.

lua_rh
STSN RH.

리턴되는 매개변수

명령n! 실행되면 LUA는 다음 매3변수를 리턴합니다.

lua_prim_rc
LUA_OK

lua_data_length
송신되는 STSN 응답의 f이.

lua_prim_rc
명령n b능으로 설정되는 1차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

사용시 주의사항

명령n 제n 블럭은 SLI! 할당한 bo5*! 만들n집니다. **lua_th** 및 **lua_rh** 매3변수의 내k은 **SLI_STSN_ROUTINE** 명령n 제n 블럭! 위치합니다. **lua_data_ptr** 매3변수는 **STSN** RU의 주소! 들n 있으며 **lua_data_length** 매3변수!는 RU의 f이! 들n 있습니다.

SLI_STSN_ROUTINE은 확장 루틴이 **lua_prim_rc**로 리턴될 때 0료되며 **lua_data_length** 매3변수는 **SLI_STSN_ROUTINE** 명령n 제n 블럭! 설정됩니다. STSN RU를 STSN 응답으로 c쳐2십시@. OK 1차 리턴 코드는 STSN이 수신됐음을 나타냅니다. 루틴이 STSN을 E부하면 1차 리턴 코드를 NEGATIVE_RSP로 설정하m 부정 응답을 STSN 버퍼! 넣으십시@. **lua_data_ptr** 매3변수를 수정하지 마십시@.

주: 이 루틴으로부터의 부정 응답은 **SLI_OPEN** 명령n를 취소합니다. SLI는 SESSION_FAILURE의 1차 리턴 코드M NEG_RSP_FROM_STSN_ROUTINE의 2차 리턴 코드를 리턴합니다.

SLI_SDT_ROUTINE

이 명령n는 SNA SDT d청을 호스트!서 수신했음을 SLI n플리케이션 프로W램! K리며 n플리케이션 프로W램이 SDT RU를 조사하m 응답을 준비하T 합니다. **SLI_SDT_ROUTINE**은 **SLI_OPEN** 확장 리스트 바인드 루틴 필드! 지정한 프로W래머 제x DLL! 전달됩니다.

SLI_SDT_ROUTINE은 SNA API 클라이p트! 의해 지x되지 J습니다.

제공되는 매개변수

SLI_SDT_ROUTINE! 대한 다음 매3변수는 SLI! 의해 제x됩니다.

lua_verb
LUA_VERB_SLI

LUA 명령n! 대한 명령n 코드 표시b.

lua_verb_length
명령n 제n 블럭의 f이.

lua_opcode
LUA_OPCODE_SLI_STSN_ROUTINE

루틴! 대한 작w 코드.

lua_luname
ASCII로 된 로컬 LU명.

lua_sid
사k되는 세션을 식별하며 **SLI_OPEN**! 의해 리턴되는 세션 ID.

lua_data_length
SDT RU의 f이.

lua_data_ptr
SDT RU! 대한 포인터.

lua_th
SDT TH.

lua_rh
SDT RH.

리턴되는 매개변수

다음은 확장 루틴이 리턴해_ 하는 **SLI_SDT_ROUTINE**! 대한 매3변수의 리스트입니다.

lua_prim_rc
LUA_OK

lua_data_length
송신되는 SDT 응답의 f이.

lua_prim_rc
명령n b능으로 설정되는 1차 리턴 코드. 자세한 내k은 377페이지의 『부록B. LUA 명령n 리턴 코드』를 참조하십시@.

사용시 주의사항

명령n 제n 블럭은 SLI! 할당한 bo5*! 만들n집니다. **lua_th** 및 **lua_rh** 매3변수의 내k은 **SLI_SDT_ROUTINE** 명령n 제n 블럭! 위치합니다. **lua_data_ptr** 매3변수는 SDI RU의 주소! 들n 있으며 **lua_data_length** 매3변수!는 RU의 f이! 들n 있습니다.

SLI_SDT_ROUTINE은 확장 루틴이 **lua_prim_rc**로 리턴될 때 0료되며 **lua_data_length** 매3변수는 **SLI_SDT_ROUTINE** 명령n 제n 블럭! 설정됩니다. SDT RU를 SDT 응답으로 c쳐2십시@. OK 1차 리턴 코드는 SDT! 수k됐음을 나타냅니다. 루틴이 SDT를 E부하면 1차 리턴 코드를 NEGATIVE_RSP로 설정하m 부정 응답을 STSN 버퍼! 넣으십시@. **lua_data_ptr** 매3변수를 수정하지 마십시@.

주: 이 루틴으로부터의 부정 응답은 **SLI_OPEN** 명령n를 취소합니다. SLI는 SESSION_FAILURE의 1차 리턴 코드M NEG_RSP_FROM_SDT_ROUTINE의 2차 리턴 코드를 리턴합니다.

제3부 Common Services API

제16장 공통 - 비스 엔트리 포인트

퍼스널 통신 및 통신 서버는 x통 서비스 프로W래밍 인터페이스를 제x합니다. 이 API는 퍼스널 통신 및 통신 서버 API를 사k하는 n플리케이션 프로W램!서 사k되는 x통 서비스 명령n(CSV)로 8성됩니다.

모든 퍼스널 통신 및 통신 서버 n플리케이션 프로W램은 다음 작w을 하나 이상 수행하b 위해 x통 서비스 명령n를 사k합니다.

- 단일 바이트 pn! 대한 코드 페이지 유지보수 (**GET_CP_CONVERT_TABLE**)
- ASCII 문자-을 EBCDIC로 또는 EBCDIC를 ASCII로 변환 (**CONVERT**)
- 이중 바이트 문자-을 한 페이지!서 다른 페이지로 변환 (**TRNSDT**)

주: 이 책의 제 3부! 있는 장들!는 다음 시스템!서 제x하는 x통 서비스 API! 대한 정보! 들n 있습니다.

- Windows NT!서 실행 중인 통신 서버
- 통신 서버/NT 제품z 함께 제x되는 OS/2, Windows NT, Windows 95 W리m Windows 3.1k SNA API 클라이p트
- Windows 95 및 Windows NTk 퍼스널 통신

이들 시스템!서 지x되는 내k! 차이! 있으면 명시됩니다.

공통 - 비스 프로그램 작:법

다음 표는 x통 서비스 프로W램을 컴파일하m 링크할 때 필d하며 제x된 헤더 파일z 라이브러리의 소스 모듈 사k법을 보)줍니다.

표 18. n5체제k 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL 이름
WINNT & WIN95	WINCSV.H	WINCSV32.LIB	WINCSV32.DLL
WIN3.1	WINCSV.H	WINCSV.LIB	WINCSV.DLL
OS/2	ACSSVC.H	ACSSVC.LIB	ACSSVC.DLL

*WINNT = Windows NT
*WIN95 = Windows 95
*WIN3.1 = Windows 3.1

다음 섹션은 x통 서비스의 #트리 포인트! 대해 설명합니다.

ACSSVC

모든 CSV 명령n ! 대한 동b #트리 포인트입니다. 퍼스널 통신 및 통신 서버는 b존 n플리케이션z의 호환성 ! 대한 #트리 포인트를 제x합니다.

구문

```
void ACSSVC (long)
```

입력은 명령n 제n 블럭 포인터입니다.

리턴 값

리턴된 * ! 대한 1차 및 2차 리턴 코드를 확인하십시@.

SNA API OS/2 클라이p트 ! 지x되는 유일한 통신 서버 #트리 포인트입니다.

WinCSV()

이 함수는 CSV API ! 대한 동 b #트리 포인트를 제 x 합니다.

구문

```
void WINAPI WinCSV(long vcb)
```

매 3 변수

	설명
vcb	명령 n 제 n 블럭 ! 대한 포인터

리턴 값

리턴 * 이 x 습니다. 명령 n 제 n 블럭의 **primary_rc** 및 **secondary_rc** 필드는 @ 류를 나타냅니다. | 련 항목: 307페이지의 『WinAsyncCSV()』.

WinCSVCleanup()

이 b능은 CSV API ! 서 n플리케이션을 중단시키m 등록을 취소시킵니다.

구문

```
BOOL WINAPI WinCSVCleanup(void);
```

리턴 값

리턴 *은 등록 취소)부를 지정합니다. *이 0이 F니면 퍼스널 통신 및 통신 서버는 n플리케이션의 등록을 취소시킵니다. 퍼스널 통신 및 통신 서버는 *이 0일 때 n플리케이션의 등록을 취소시킵니다.

사용시 주의사항

WinCSVCleanup()을 사k하) CSV API ! 서 CSV API n플리케이션의 등록을 취소시키십시@. 9를 들n 특정 n플리케이션 ! 할당된 자x을 해제시킵니다.

WinAsyncCSV()

함수는 **TRANSFER_MS_DATA**! 대해서만 비동b #트리 포인트를 제x합니다. n플리케이션이 다른 모든 명령n! 대해 이 함수를 사k하면 작동은 동b화됩니다.

구문

```
HANDLE WlNAPI WinAsyncCSV(HHWND hWnd,
                          long vcb);
```

매3변수

설명

hwnd 0료 메세지를 수신하b 위한 창 핸들.

vcb 명령n 제n 블럭! 대한 포인터.

리턴 값

리턴 *은 명령 d청)부를 나타냅니다. 함수! 성x적이z다면 실제 리턴 *이 비동b 타스크 핸들이 됩니다. 함수! 실패했다면 퍼스널 통신 및 통신 서버는 0을 리턴합니다.

사용시 주의사항

비동b 작w이 0료될 때 n플리케이션 창인 *hWnd*는 입력 문자-로 **WinAsyncCSV**M 함께 **RegisterWindowMessage**! 의해 리턴된 메세지를 수신합니다. *wParam* 인수!는 x래의 함수 호출로 리턴된 비동b 타스크 핸들이 있습니다. *IParam* 인수!는 x래의 VCB 포인터! 있으며 참조를 취소하) 최종 리턴 코드를 판별합니다.

함수! 리턴되는 fl, 퍼스널 통신 및 통신 서버는 작w이 0료되E나 또는 대화! 취소될 때 **WinAsyncCSV()** 메시지를 n플리케이션으로 전송합니다.

WinCSVStartup()

이 b능을 사k하면 n플리케이션이 필d한 x통 서비스 명령n API의 버전을 지정하m 특정 CSV API 세부사항을 K색할 수 있습니다. 이 호출이 필d하지는 J지만 사k되는 f l ! 는 **WinCSVCleanup** 호출도 함께 사k돼 _ 합니다.

구문

```
int WINAPI WinCSVStartup (WORD wVersionRequired,
                          LPWCSVDATA csvdata);
```

매3변수

설명

wVersionRequired

필d한 CSV API 지x 버전을 지정합니다. 상위 바이트는 마이너 버전 (3정) 번호를 지정하m 하위 바이트는 메이저 버전 번호를 지정합니다.

lpwCSVDATA

b초! 되는 CSV API DLL! 대한 정보! 들n 있습니다.

리턴 값

리턴 *은 CSV API! n플리케이션을 등록했는지M 제x된 버전 번호를 지x할 수 있는 지를 나타냅니다. 리턴된 *이 0이면 CSV API는 지정된 버전을 지x하m n플리케이션을 등록합니다. W렇지 J으면 다음 * 중 하나! 리턴됩니다.

WCSVVERNOTSUPPORTED

이 특정 CSV API는 d청된 CSV API 지x의 버전을 제x하지 J습니다.

WCSVINVALID

CSV API는 d청된 버전을 판별할 수 x습니다.

사용시 주의사항

WinCSVStartup()은 U으로 릴리스될 APIM의 호환성! 도r을 주b 위한 M입니다. 지x되는 현재 버전은 1.0입니다.

다음 8조는 실제 CSV API 8현의 세부사항을 설명합니다.

```
typedef struct tagWCSVDATA { WORD wVersion;
                char szDescription[WCSVDESCRIPTION_LEN+1];
                } WCSVDATA, FAR *LPWCSVDATA;
```

n플리케이션!서 마지막 CVS API 호출이 이루n지면 **WinCSVCleanup()**을 호출합니다.

GetCsvReturnCode()

명령 n 의 1차 및 2차 리턴 코드를 인쇄할 수 있는 문자 - 로 변환하 b 위해 이 # 트리 포인트를 사 k 하십시@. n 플리케이션 프로 W 램 ! 서 사 k 될 수 있는 표준 @ 류 문자 - 집합을 리턴합니다.

구문

```
int WINAPI GetCsvReturnCode (struct csv_hdr *vcb,
                             UINT buffer_length,
                             unsigned char *buffer_addr);
```

매 3 변수

설명

vcb 명령 n 제 n 블럭의 주소.

buffer_length

buffer_addr ! 포인트하는 버퍼의 f 이. G 장 f 이는 256입니다.

buffer_addr

포맷되 m 널(null)로 중단되는 문자 - 을 보 | 하는 버퍼의 주소(지정된 버퍼 ! 있는 문자 - 의 f 이).

리턴 값

0x20000001

매 3 변수 ! 유효하지 J 습니다. 함수는 지정된 **명령어** ! 서 읽지 못하 E 나 또는 지정된 버퍼 ! 5 수 x 습니다.

0X20000002

지정된 버퍼 ! 너무 작습니다.

사용시 주의사항

buffer_addr ! 서 리턴된 @ 류 설명 문자 - 은 새로 n 행 문자(**\n**)로 중단되지 J 습니다.

TrnsDt

이 함수는 한 코드 페이지의 SBCS M DBCS를 다른 코드 페이지로 변환합니다. 퍼스널 통신 및 통신 서버는 **TRNSDT.DLL** 파일 ! 서 **TrnsDt**를 제 x 합니다. **TransDt**는 DBCS 세션 ! 서만 사 k 할 수 있습니다.

구문

```
TrnsDt (PASSSTRUCT *passparm);
```

함수

이 함수는 한 코드 페이지의 SBCS M DBCS를 다른 코드 페이지로 변환합니다. 다음 표 ! 서 『✓』는 퍼스널 통신 및 통신 서버 ! 코드 페이지 쌍 # 의 변환을 지 x 한다는 M 을 나타내 m "-"(하이픈)은 변환 지 x 프로 W 램이 x 다는 M 을 나타냅니다.

일본		932	930	931	939	290	037	1027
	932	-	✓	✓	✓	✓	✓	✓
	930	✓	-	-	-	-	-	-
	931	✓	-	-	-	-	-	-
	939	✓	-	-	-	-	-	-
	290	✓	-	-	-	-	-	-
	037	✓	-	-	-	-	-	-
	1027	✓	-	-	-	-	-	-
한국		949	833	834	933			
	949	-	✓	✓	✓			
	833	✓	-	-	-			
	834	✓	-	-	-			
	933	✓	-	-	-			
대만		950	037	835	937			
	950	-	✓	✓	✓			
	037	✓	-	-	-			
	835	✓	-	-	-			
	937	✓	-	-	-			
중국		1381	836	837	935			
	1381	-	✓	✓	✓			
	836	✓	-	-	-			
	837	✓	-	-	-			
	935	✓	-	-	-			

TRNSDT.H 헤더 파일을 사k하) 컴파일하m 두 프로W램의 LIB 서브라이브러리!서 **TRNSDT.LIB** 파일을 사k하) 링크하십시@.

PassParm 포맷

passparm 포맷은 다음z 0습니다.

WORD *parm_length*
이 8조의 f이 (입력)

WORD *exit_code*
종료 코드 (출력)

0000H
정상 종료.

0001H
지x되지 J는 변환이 지정됩니다.

000CH
Exit_code 필드는 0으로 초b화되지 J습니다.

0080H
마지막 문자는 DCBS의 ^쪽 반입니다. 대신 널(null) 문자로 채v집니다.

WORD *in_length*
소스 버퍼의 f이 (입력)

LPBYTE *in_addr*
소스 버퍼 주소 (입력)

WORD *out_length*
목표 버퍼의 f이 (입력)

지정된 f이! 변환된 데이타를 전부 리턴하b! 너무 작은 fl! 는 필d한 f이만 리턴됩니다.

LPBYTE *out_addr*
목표 주소 버퍼 (입력)

WORD *trns_id*
5으로 보존 (입력)

WORD *in_page*
소스 코드 페이지 (입력)

WORD *out_page*
목표 코드 페이지 (입력)

WORD *option*
I션 (입력/출력)

Input 입력 I션은 다음z 0습니다.

비트 **15–9**
5으로 보존.

비트 **8**
목표 문자 - ! SO/SI 있음.

비트 **7–3**
5으로 보존.

비트 **2**
편집할 수 x는 SBCS 테이블.

비트 **1**
소스 문자 - 은 DBCS로 시작.

비트 **0**
소스 문자 - ! SO/SI 있음.

Output
출력 I션은 다음z 0습니다.

4 DBCS ! 서 종료.

0 비DBCS ! 서 종료.

주:

1. 비트 8z 비트 0은 다음z 0이 설정되n _ 합니다.
   PC ! 서 호스트 비트 8=1로 변환.
   PC ! 서 호스트 비트 0=0으로 변환.
   호스트 ! 서 PC 비트 8=0으로 변환.
   호스트 ! 서 PC 비트 0=1로 변환.
2. **SYSCTBL.EXE**를 사k하) **TrnsDt** ! 사k하는 조정된 테이블의 이름을 지정하십시@. SBCS 문자 - 을 변환하려면 **TrnsDt** ! 서 **Option** 매3변수 비트 2! FALSE로 설정된 조정을 E친 테이블이 사k됩니다. **TrnsDt**는 비트 2는 설정되m 테이블 이름이 지정되지 J으면 b본 테이블을 사k합니다. 테이블 이름이 **SYSCTBL.EXE**를 사k하) 지정될 때 **TrnsDt**는 DBCS 문자 - 을 변환하b 위해 항상 조정된 테이블을 사k합니다. 이런 f l 비트 2의 **Option** 매3변수는 사k되지 J습니다.
3. 일반적으로 **TrnsDt** ! 서는 SO/SI 제n 문자를 한 쌍으로 포함하는 호스트 데이타 ! 필d합니다. W러나 혼합된 데이타 문자 - 의 쌍을 변환하는 f l, 데이타는 SO 제n 문자x이 이중 바이트 문자로 시작되n _ 합니다. 이런 f l 데이타는 이중 바이트 문자를 식별하지 J습니다. 비트 1은 이런 f l ! 유k합니다. 비트 1을 1로 설정하면 **TrnsDt** ! 버퍼의 시작을 이중 바이트 문자 또는 SO 제n 문자로 처리합니다.

리턴 코드

0 NO_ERROR

2 ERROR_FILE_NOT_FOUND

TrnsDt는 지정된 코드 변환시 사k되는 테이블을 찾을 수 x습니다.

87 ERROR_INVALID_PARAMETER

매3변수! 유효하지 J습니다.

111 ERROR_BUFFER_OVERFLOW

목표 버퍼! 너무 작습니다.

150 ERROR_MEMORY_ALLOCATE

메모리 할당 @류.

사용시 주의사항

작은 버퍼라도 **TrnsDt**의 종료 코드M I션 매3변수를 사k하) 큰 데이타 변환을 처리할 수 있습니다. 먼저 작은 소스 버퍼M 이중 또는 삼중 크b의 목적지 버퍼를 사k하) **TrnsDt**를 시작하m(PC!서 호스트로의 f l), 수신된 종료 코드를 b초로 변환이 n떻T 종료되는지 보십시@. W런 후 적절히 진행하십시@.

9를 들n 변환으로 이중 바이트 문자! 두 부분으로 나누n지E나 SOM SI 제n 문자 사이!서 변환이 불O전하T 종료하면 버퍼 포인터M 위치를 정의한 후, 다음 호출을 수행하십시@.

VCB 구조

다음 9제는 호스트 코드 0x4040을 PC 코드로 변환합니다.

```
#include "trnsdt.h"

PASSSTRUCT      passparm;
char            bufs[20], buft[20];
int             rc;

//변환될 문자열을 설정하십시오.
bufs[0] = 0x0e;
bufs[1] = 0x40;
bufs[2] = 0x40;
bufs[3] = 0x4f;

//매개변수를 설정하십시오.
passaparm.parm_length = 24;
passparm.exit_code    = 0;
passaparm.in_length   = 4;
passaparm.in_addr     = Created by ActiveSystems. 02/11/97. Entity not defined[0];
passaparm.out_length  = 20;
passaparm.out_addr    = Created by ActiveSystems. 02/11/97. Entity not defined[0];

 passaparm.trns_id    = 0;
 passaparm.in_page    = 930;
 passaparm.out_page   = 932;
 passaparm.option     = 1;

//TrnstDt를 통해 문자열을 변환하십시오.
if (rc = TrnsDt(&passaparm))
   printf("오y 리턴 코드 = %d\n\r", rc);
   printf("종a 코드 = %d\n\r", passaparm.exit_code);
   exit(0);
else
  .....
```

제17장 공통 - 비스 명령어 (CSV)

퍼스널 통신 및 통신 서버는 x통 서비스 API ! 대해 다음 명령n를 제x합니다.

GET_CP_CONVERT_TABLE
CONVERT

GET_CP_CONVERT_TABLE

이 명령n는 한 코드 페이지!서 다른 코드 페이지로 변환 테이블을 8축하는 유틸리티 서비스를 제x합니다. 이 명령n는 문자 - 을 변환하b 위해 테이블!서 문자 찾b를 수행할 때 n플리케이션!서 사k되는 256 바이트 변환 테이블을 리턴합니다.

프로W램은 다른 코드 페이지!서 인코딩된 데이타를 b대하는 노드M 통신할 때 데이타 변환을 수행해_ 할 때도 있습니다.

VCB 구조

```
struct get_cp_convert_table
  {
  unsigned short  opcode;          /* 작업 코드를 식별하는 명I어.
 */
  unsigned char   opext;           /* 예약.
 */
  unsigned char   reserv2;         /* 예약.
 */
  unsigned short  primary_rc;      /* 명I어N부터의 1차 리턴 코드.
 */
  unsigned long   secondary_rc;    /* 2차 (규정하는) 리턴 코드.
 */
  unsigned short  source_cp;       /* 변환 테이블에 대한 소스 코드 페이지
 */
  unsigned short  target_cp;       /* 변환 테이블에 대한 목표 코드 페이지
 */
  unsigned char  *conv_tbl_addr;   /* 변환 테이블 배치 주소
 */
  unsigned char   char_not_fnd;    /* 문자를 찾지 못한 옵션: 대체
 */
                                   /* 대체 문자 또는 라운드 트립
(round trip)*/
  unsigned char   substitute_char; /* 사용할 대체 문자.
 */
  } GET_CP_CONVERT_TABLE;
```

제공되는 매개변수

source_code_page

대체 문자를 !져@는 코드 페이지 번호. 코드 페이지의 번호는 다음 번호 중 하나일 수 있습니다.

- ASCII 코드 페이지 (십진수로)
  - 437 US IBM PC
  - 737 W리스
  - 813 W리스
  - 819 ANSI 표준
  - 850 다중p n
  - 852 체코슬로바키F/헝!리/폴란드/유m슬라비F
  - 855 키릴n

  - 857 터키
  - 858 다중p n
  - 860 포르투% n
  - 861 F이슬랜드
  - 862 히브리n
  - 863 캐나다 프랑스n
  - 864 F랍n
  - 865 스칸디나비F n
  - 866 키릴n
  - 874 태9 n
  - 912 라틴n 2
  - 915 키릴n
  - 916 히브리n
  - 920 터키
  - 921 라트비F, 리투F니F
  - 922 !스토니F
  - 923 ANSI 표준
  - 1008 F랍n
  - 1089 F랍n
  - 1124 l크라이나n
  - 1125 l크라이나n
  - 1127 F랍n/프랑스n
  - 1129 베트남n
  - 1131 벨라루스
  - 1133 라@
  - 1250 라틴n 2
  - 1251 키릴n
  - 1252 라틴n 1
  - 1253 W리스
  - 1254 터키
  - 1255 히브리n
  - 1256 F랍n
  - 1257 발틱 (라트비F, 리투F니F, !스토니F)
  - 1258 베트남n
- EBCDIC 코드 페이지 (십집수로)
  - 037 미9/캐나다 프랑스n/네덜란드/포르투%/브라질
  - 273 독일/@스트리F

- 275 브라질
- 277 덴마크/노르~이
- 278 핀란드/스~덴
- 280 이탈리F
- 284 라틴 F메리카/스페인
- 285 5 9
- 297 프랑스
- 420 F랍n
- 424 히브리n
- 500 벨b !/스위스 프랑스n/스위스 독일n
- 803 히브리n
- 870 체코슬로바키F/헝 !리/폴란드/유m슬라비F
- 871 F이슬랜드
- 875 W리스
- 924 라틴 1
- 1025 키릴n
- 1026 터키
- 1047 라틴 1
- 1112 라트비F, 리투F니F
- 1122 !스토니F
- 1123 l크라이나
- 1130 베트남n
- 1132 라@
- 1140 미9/캐나다/네덜란드/포르투%/브라질/@스트레일리F/뉴질랜드
- 1141 독일/@스트리F
- 1142 덴마크/노르~이
- 1143 핀란드/스~덴
- 1144 이탈리F
- 1145 라틴 F메리카/스페인
- 1146 5 9
- 1147 프랑스
- 1148 벨b !/스위스
- 1149 F이슬랜드

• 사k자 정의 코드 페이지
  - 65280-65534

– 사k자 정의 코드 페이지를 사k할 때 먼저 퍼스널 통신! 대해 다음z 0이 CPT 파일의 사k자 정의 f로M 레지스트리 입력항목을 함께 정의하십시@.

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Personal
Communications /CurrentVersion/COMCPT
```

통신 서버! 대해 다음z 0이 CPT 파일의 사k자 정의 f로 M 레지스트리 입력항목을 함께 정의하십시@.

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Communications
Server/CurrentVersion/COMCPT
```

주: 소스 및 목표 코드 페이지!서 똑0은 문자만 서로 변환될 수 있습니다. 일반적으로 표준!서 지정되며 별로 비슷하지 J은 문자쌍은 서로 변환되지 J습니다.

target_code_page
변환될 목표 문자-! 대한 코드 페이지 번호. 번호는 **source_code_page**! 대해 표시된 M 중 하나일 수 있습니다.

convert_table addr
256 바이트 변환 테이블을 수신해_ 하는 버퍼의 주소. 이 버퍼는 읽b/2b 세W멘트! 있n_ 합니다.

character_not_found
소스 코드 페이지! 있는 문자! 목표 코드 페이지! x을 때 취해_ 하는 조치. 다음 * 중 하나를 지정하십시@.

SV_ROUND_TRIP
이 I션을 사k하면 *을 변환 테이블! 저장할 수 있습니다. W러면 변환 테이블이 소스 및 목표 코드 페이지의 반전으로 생성될 때, 소스!서 목표 코드 페이지로의 변환 및 * 변환의 az는 x래 문자! 됩니다. 이 I션이 실행되b 위해서는 g쪽 테이블을 생성할 때 **ROUND_TRIP** I션을 선택해_ 합니다.

SV_SUBSTITUTE
substitute_character 매3변수!서 지정된 문자를 변환 테이블! 저장합니다.

substitute_character
소스 코드 페이지의 문자! 목표 코드 페이지! xm **character_not_found** 매3변수! **SV_SUBSTITUTE**로 설정될 때 변환 테이블! 저장되는 바이트.

OK 리턴 코드는 **GET_CP_CONVERT_TABLE** 명령n! 실행됐음을 나타냅니다.

다음 매3변수는 리턴 코드! 9일 때 리턴됩니다.

convert_table
변환 테이블은 **CONV_table_addr**!서 지정된 주소!서 8축됐습니다.

primary_rc

SV_PARAMETER_CHECK

secondary_rc

SV_INVALID_CHAR_NOT_FOUND

SV_INVALID_DATA_SEGMENT
SV_INVALID_SOURCE_CODE_PAGE
SV_INVALID_TARGET_CODE_PAGE

CONVERT

이 명령n는 ASCII 문자-을 EBCDIC로, EBCDIC 문자-을 ASCII로 변환합니다.

프로W램은 EBCDIC 데이타를 b대하는 노드M 통신하E나 또는 EBCDIC 이름이 필d한 인터페이스를 통z하도록 이름을 변환해_ 할 때는 데이타 변환을 수행합니다.

주: **CONVERT** 명령n는 DBCS!서 지x되지 J습니다. **TrnsDt**를 사k하) 이중 바이트 문자! 있는 문자-을 변환할 수 있습니다.

VCB 구조

```
struct convert
  {
  unsigned short  opcode;          /* 작업 코드를 식별하는 명I어.
  */
  unsigned char   opext;           /* 예약됨.
  */
  unsigned char   reserv2;         /* 예약됨.
  */
  unsigned short  primary_rc;      /* 명I어N부터의 1차 리턴 코드.
  */
  unsigned long   secondary_rc;    /* 2차 (규정하는) 리턴 코드.
  */
  unsigned char     direction;     /* 변환 방향 - ASCII 에서
  */
                                   /* EBCDIC 또는 그 반대.
  */
  unsigned char     char_set;      /* 변환 A, AE, 또는 사용자-정의 G에
  */
                                   /* 사용할 문자.
  */
  unsigned short    len;           /* 변환될 문자 길이.
  */
  unsigned char    *source;        /* 변환될 문자열에 대한 포인터.
  */
  unsigned char    *target;        /* 변환된 문자열 배치 주소.
  */
  } CONVERT;
```

리턴되는 매개변수

return_code OK @류 코드

제공되는 매개변수

direction

코드 변환의 속성.

SV_ASCII_TO_EBCDIC
ASCII 문자를 EBCDIC로 변환합니다.

SV_EBCDIC_TO_ASCII
EBCDIC 문자를 ASCII로 변환합니다.

character_set

소스 문자 - ! 서 허 k 되는 문자 세트. **CONVERT** 명령 n ! 사 k 할 수 있는 ASCII/EBCDIC 변환 테이블의 유형을 다음 z 0 이 3 ! 지로 지정할 수 있습니다. SV_A, SV_AE W 리 m SV_G. 유형 A M 유형 AE 테이블은 퍼스널 통신 및 통신 서버 ! 서 정의됩니다.

변환 테이블의 포맷은 " " 32문자 ! 들 n 있는 32행으로 8 성됩니다. " 행은 캐리지 리턴 z 행 x ^ 이 @ 는 16 3 의 인쇄 ! 능한 16진수 문자를 나타냅니다. 첫 16 행은 ASCII-EBCDIC 변환 정보를 제 x 합니다. 두 번째 16행은 EBCDIC-ASCII 변화 정보를 제 x 합니다. 테이블은 32행 모두 포함해 _ 합니다.

퍼스널 통신 및 통신 서버 ! 변환을 수행하면 " 입력 문자 M 동등한 수치 * 을 변환 테이블의 0 시작 색인으로 사 k 합니다. 이 색인은 변환된 문자의 16진수 * 이 있는 테이블 위치를 지정합니다. 9 를 들 n 테이블의 48번째 위치 ! X'F0'의 * 이 있다 m ! 정하십시 @. 퍼스널 통신 및 통신 서버는 48의 * (X'30')을 ! 진 입력 문자를 240의 * (X'F0')으로 변환합니다.

테이블 A

테이블 A는 대문자 A ! 서 Z, 숫자 0 ! 서 9 W 리 m 특수 문자 $, #, @를 변환합니다. 소스 문자 - 의 첫 문자는 대문자이 E 나 3 ! 지 특수 문자 중 하나) _ 합니다. 만일 W 렇지 J 으면 변환이 이루 n 지지 J m INVALID_FIRST_CHARACTER 2차 리턴 코드 ! 리턴됩니다. ASCII-EBCDIC 방향 ! 서 ASCII 소문자는 EBCDIC 대문자로 변환됩니다.

끝 ! 이 n 지는 x 백 (소스 문자 - 끝 ! @ 는 x 백)은 g 방향 x 백으로 변환됩니다. 이 M 는 달리 삽입된 x 백은 X'00'으로 변환됩니다.

임의의 소스 문자 ! X'00'으로 변환되면 CONVERSION_ERROR ! 리턴됩니다. 전체 변환이 O 료됩니다.

테이블 AE

테이블 AE는 5 숫자(A-Z, a-z, 0-9), 특수 문자 $, #, @ W 리 m 마침표(.)를 변환합니다. 문자 - 의 첫 문자 ! 는 제한이 x 습니다.

끝 ! 이 n 지는 x 백(소스 문자 - 끝 ! @ 는 x 백)은 g 방향 x 백으로 변환됩니다. 이 M 는 달리 삽입된 x 백은 X'00'으로 변환됩니다.

임의의 소스 문자 ! X'00'으로 변환되면 CONVERSION_ERROR ! 리턴됩니다. 전체 변환이 O 료됩니다.

테이블 G

임의의 문자 ! 서 다른 임의의 문자로 변환할 때는 테이블 G를 사 k 할 수 있습니다 (ASCII ! 서 EBCDIC로 또는 EBCDIC ! 서 ASCII로 \ !). W 러나 **CONVERT** 명령 n ! 서 ASCII_TO_EBCDIC를 지정하) 테이블의 위쪽 반을 사 k 하 m EBCDIC_TO_ASCII를 지정하) F 래쪽 반을 사 k 해 _ 합니다.

퍼스널 통신은 다음 F 래 ! 있는 레지스트리를 K 색하)

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Personal Communications /
                                    CurrentVersion/COMTBLG
```

테이블 G로의 전체 f로명을 r습니다. 통신 서버는 F래! 있는 레지스트리를 K색하)

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Communications Server/
                                    CurrentVersion/COMTBLG
```

테이블 G로의 전체 f로명을 r습니다. 32 비트 Windows NT 클라이p트의 레지스트리!서 테이블 G f로 위치는 다음z 0습니다.

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Comm.Server for NT SNA/Client/
                                    CurrentVersion/COMTBLG
```

CONVERT 명령n! 대해 제x된 b타 매3변수는 다음z 0습니다.

length 변환될 문자의 수.

문자 - 의 f이는 **source_addr** 또는 **target_addr**! 할당된 세W멘트 크b를 초z할 때n지 확장할 수 x습니다.

source_addr
변환될 문자 - 의 주소.

target_addr address
변환될 문자 - 을 수신하는 주소.

주: n플리케이션!서 소스 문자 - 을 보존하지 JF도 되는 fl!는 **source_addr**M **target_addr**! 대해 0은 변수를 지정할 수 있습니다.

리턴되는 매개변수

OK 리턴 코드는 **CONVERT** 명령n! 실행됐음을 나타냅니다.

다음은 **CONVERT** 명령nM |련된 1차 및 2차 @류 리턴 코드M 리턴 코드 설명의 위치를 보)줍니다.

primary_rc
SV_PARAMETER_CHECK

secondary_rc
SV_INVALID_DIRECTION

SV_TABLE_ERROR
SV_INVALID_CHARACTER_SET
SV_INVALID_FIRST_CHARACTER
SV_CONVERSION_ERROR
SV_INVALID_DATA_SEGMENT

primary_rc
SV_UNEXPECTED_DOS_ERROR

제4부 EHNAPPC API

제18장 EHNAPPC 어플리케이G 프로그램 인터페이스

통신 서버 SNA API 클라이p트!서만 사k할 수 있습니다.

EHNAPPC 통신 API는 3인k 컴퓨터M AS/400 시스템 사이!서 서로 협력하) 처리하는 n플리케이션을 작성하는 방법을 제x합니다. 프로W래머는 이 인터페이스를 통해 저^ 통신 프로W래밍z 하드~n ,a성 유형! 대해 신f 2지 JF도 됩니다. n플리케이션 프로W래머는 이 API를 사k할 때 AS/400z PC 프로W램을 둘 다 작성해_ 합니다. 호스트 n플리케이션!서 W세스되는 E의 모든 M은 상대방 PC n플리케이션으로 확장됩니다. 이 API는 성능이 중d한 n플리케이션!서 사k될 수 있습니다.

이 장은 Windows NT M Windows 95 통신 서버 SNA API 클라이p트! 대해 32 비트 EHBAPPC API를 이루는 루틴, 데이타 8조, 리턴 코드를 설명합니다. 대부분의 이들 함수는 Windows 3.1k 16 비트 API!서도 사k될 수 있습니다.

EHNAPPC 프로그램 작:하기

F래의 표는 EHNAPPC 프로W램을 컴파일하m 링크할 때 필d하며 제x된 헤더 파일z 라이브러리의 소스 모듈 사k법을 보)줍니다.

표 19. n5체제k 헤더 파일 및 라이브러리

운영체제	헤더 파일	라이브러리	DLL 이름
WINNT & WIN95	E32APPC.H	E32APPC.LIB	E32APPC.DLL
WIN3.1	EHNAPPC.H	EHNAPPC.LIB	EHNAPPC.DLL

*WINNT = Windows NT
*WIN95 = Windows 95
*WIN3.1 = Windows 3.1

EHNAPPC 루틴

다음! 대해 " 클라이p트 Windows API 루틴 설명이 상세하T 이루n집니다.

- 목적
- 프로시듀n 선p
- 매3변수
- 리턴 코드

EHNAPPC_Allocate

목적

이 함수는 상대방 트랜잭션 프로W램(TP)z의 대화를 시작합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
include "E32APPC.H"
extern int EHNAPPC_Allocate
HWND                    hWnd,
unsigned                nBufferLength,
ConversationType        bType,
SyncLevelEnumb          SynchLevel,
LPSTR                   lpszLocationName,
LPSTR                   lpszTpn,
int                     nPipLength,
LPVOID                  lpPipData,
LPDWORD                 lpdwConversation);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

bType은 할당할 대화 유형을 식별합니다. 사k할 수 있는 *은 다음z 0습니다.

EHNAPPC_BASIC (0)

EHNAPPC_MAPPED (1)

bSynchLevel은 로컬z 상대방 프로W램#의 동b화 레벨을 식별합니다. 사k할 수 있는 *은 다음z 0습니다.

EHNAPPC_SYNCLEVELNONE (0)

EHNAPPC_SYNCLEVELCONFIRM (1)

lpszLocationName은 호스트 시스템명을 지정하면서 널(null)로 중단되는 문자-을 !리킵니다. 이 포인터! NULL로 설정되면 b본 시스템이 사k됩니다.

lpszTpn은 상대방 프로W램명을 지정하면서 널(null)로 중단되는 문자-을 !리킵니다. 첫 문자! 0*40보다 작으면 ASCII-EBCDIC 변환이 이루n지지 J습니다.

nPipLength는 프로W램 초b화 매3변수(PIP) 데이타 f이를 식별합니다. 이 변수! 0이면 PIP 데이타는 전송되지 J습니다.

lpPipData는 PIP 데이타를 !리킵니다. PIP 데이타는 GDS 포맷으로 이루n져_ 하며 EBCDIC를 사k해_ 합니다.

lpdwConversation은 후속 호출!서 사k할 핸들을 리턴할 때 사k되는 이중단n 변수를 !리킵니다. 핸들은 " 대화!서 m유한 *입니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_Confirm

목적

이 함수는 지]n지 전송된 모든 데이타! 상대방!서 수신되z는지! 대한 확인을 d청합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int far pascal EHNAPPC_Confirm(
HWND           hWnd,
DWORD          dwConversation,
LPBYTE         lpRequestToSendRcvd);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended!서 리턴된 대화 핸들을 식별합니다.

lpRequestToSendRcvd는 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했는지의)부 저장! 사k된 변수를 !리킵니다. TRUE *은 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했음을 나타냅니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_Confirmed

목적

이 함수는 확인을 d청한 상대방!T 확인을 전송합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int far pascal EHNAPPC_Confirmed(
     HWND          hWnd,
     DWORD         dwConversation);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_Deallocate

목적

이 함수는 할당된 대화의 할당을 취소시킵니다.

프로시듀어 선언

```
#include "E32APPC.H"
extern int far pascal EHNAPPC_Deallocate(
         HWND                hWnd,
         DWORD               dwConversation,
         DeallocateEnum      bType);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

bType는 클라이p트 ! 수행할 할당 취소 유형을 식별합니다. 사k할 수 있는 *은 다음z 0습니다.

EHNAPPC_DEALLOCATESYNCLEVEL (0)

EHNAPPC_DEALLOCATEFLUSH (1)

EHNAPPC_DEALLOCATEABEND (2)

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_ExtendedAllocate

목적

이 함수는 상대방 트랜잭션 프로W램z의 대화를 시작하m 보H 또는 모드 스펙을 대체할 수 있습니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_ExtendedAllocate(
HWND               hWnd,
unsigned           nBufferLength,
ConversationType   bType,
SyncLevelEnum      bSynchLevel,
LPSTR              lpszLocationName,
LPSTR              lpszTpn,
LPSTR              lpszModeName,
SecurityType       bSecurityType,
LPSTR              lpszUserId,
LPSTR              lpszPassword,
in                 nPipLength,
LPVOID             lpPipData,
LPDWORD            lpdwConversation);
```

매개변수

HWND는 n 플리케이션의 현재 창을 식별합니다.

bType는 할당할 대화 유형을 식별합니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_BASIC (0)

EHNAPPC_MAPPED (1)

bSynchLevel은 로컬 z 상대방 프로 W 램 # 의 동 b 화 레벨을 식별합니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_SYNCLEVELNONE (0)

EHNAPPC_SYNCLEVELCONFIRM (1)

lpszLocationName은 호스트 시스템명을 지정하면서 널(null)로 중단되는 문자 - 을 ! 리킵니다. 이 포인터 ! NULL로 설정되면 b 본 시스템이 사 k 됩니다.

lpszTpn은 상대방 프로 W 램명을 지정하면서 널(null)로 중단되는 문자 - 을 ! 리킵니다. 첫 문자 ! X'40'보다 작으면 ASCII-EBCDIC 변환이 이루 n 지지 J 습니다.

lpszModeName. 다음은 모드 명명 T 칙입니다.

모드명의 f 이는 1-8문자입니다. " 부분의 첫 문자는 대문자(A–Z) 또는 특수 문자(@, #, $)) _ 합니다. 나머지 문자는 대문자(A–Z), 숫자(0–9) 또는 특수 문자(@, #, $) ! 될 수 있습니다.

bSecurityType은 사 k 할 보 H 유형을 식별합니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_SECURITY_NONE (0)

EHNAPPC_SECURITY_SAME (1)

EHNAPPC_SECURITY_PGM (2)

lpszUserId는 사k자 ID! 있으면서 널(null)로 중단되는 문자 - 을 !리킵니다. 최대 f이는 10문자입니다.

lpszPassword는 0호! 있으면서 널(null)로 중단되는 문자 - 을 !리킵니다. 최대 f이는 10문자입니다.

nPipLength는 PIP 데이타의 f이를 식별합니다. 이 변수! 0이면 PIP 데이타! 전송되지 J습니다.

lpPipData는 PIP 데이타를 !리킵니다. PIP 데이타는 GDS 포맷으로 이루n져_ 하며 EBCDIC를 사k해_ 합니다.

lpdwConversation은 후속 호출!서 사k될 핸들을 리턴할 때 사k되는 이중단n 변수를 !리킵니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_Flush

목적

클라이p트는 이 함수로 버퍼! 들n 있는 데이타를 전송할 수 있습니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_Flush(
     HWND        hWnd,
     DWORD       dwConversation);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended!서 리턴된 대화 핸들을 식별합니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_GetAttributes

목적

지정된 로컬 및 상대방 트랜잭션 프로W램의 LU명, 처리 동b화의 레벨 W리m 보H! 제x되는 모든 사k자 ID를 비롯한 지정된 대화의 속성을 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_GetAttributes(
     HWND             hWnd,
     DWORD            dwConversation,
     LPBYTE           lpbSyncLevel,
     LPSTR            lpszModeName,
     LPSTR            lpszLuName,
     LPSTR            lpszPluName,
     LPSTR            lpszUserId);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended 할당 ! 서 리턴된 대화 핸들을 식별합니다.

lpbSyncLevel은 동 b 화 레벨 리턴 ! 사 k 되는 바이트 변수를 ! 리킵니다.

lpszModeName은 8 문자 모드명 리턴 ! 사 k 되 m 널(null)로 중단되는 문자 - 을 ! 리킵니다.

lpszLuName은 로컬 트랜잭션 조치 프로 W 램의 LU 리턴 ! 사 k 되 m 널(null)로 중단되는 문자를 ! 리킵니다.

lpszPluName은 상대방 LU명 리턴 ! 사 k 되 m 널(null)로 중단되는 문자를 ! 리킵니다.

lpszUserId는 이 , a 을 설정할 때 사 k 된 사 k 자 ID 리턴 ! 사 k 되 m 널(null)로 중단되는 문자를 ! 리킵니다.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시 @ .

EHNAPPC_GetCapabilities

목적

이 함수는 현재 로드된 클라이 p 트의 b 능을 나타내는 데이타 8 조를 채 s 니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_GetCapabilities(
     HWND      hWnd,
     LPSTR     lpList);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpList는 b능 정보 K색! 사k되는 b능 리스트를 !리킵니다. b능 목록은 b능 8조의 변수 번호! 뒤따라 @는 헤더로 8성됩니다. 입력!서 리스트는 조회될 b능을 지정합니다. 출력! b능 정보! 들n 있습니다.

주: 추! 8조 정보! 대해서는 345페이지의 『appcrtrcap_hdr』, 346페이지의 『appcrtrcap_mult』 W리m 346페이지의 『appcrtrcap_query』를 참조하십시@.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_GetDefaultSystem

목적

이 함수는 클라이p트! , a되n 있는 b본 시스템명을 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned pascal EHNAPPC_GetDefaultSystem(
     HWND      hWnd,
     LPSTR     lpszDefSysName);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpszDefSysName은 b본 시스템명 리턴! 사k되는 문자 버퍼를 !리킵니다. 시스템명은 널(null)로 중단되는 문자 - 로서 이 버퍼! 저장됩니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_IsRouterLoaded

목적

이 함수는 메모리!서의 클라이p트 라l터 로드)부를 판별합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern bool EHNAPPC_IsRouterLoaded(
     HWND     hWnd);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

리턴 코드

리턴 코드는 통신 서버 SNA 클라이p트 라l터! 로드되지 J을 때 FALSE(0)! 됩니다. W렇지 J으면 리턴 *이 TRUE(1)! 됩니다.

EHNAPPC_PrepareToReceive

목적

이 함수는 프로W램이 데이타를 수신할 수 있도록 준비합니다. 이 함수 다음! EHNAPPC_ReceiveImmediate를 사k하는 M은 EHNAPPC_ReceiveAndWait를 사k하는 M z 0습니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_PrepareToReceive(
     HWND     hWnd,
     DWORD    dwConversation);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_QueryConfiguredSystems

목적

이 함수는 통신 서버 ! 서 8성된 시스템의 이름을 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryConfiguredSystems(
     HWND                hWnd,
     LPINT               lpSysCount,
     LPSYSSTRUC          lpSys);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpSysCount는 , a된 시스템 수 리턴 ! 사k되는 정수 변수를 ! 리킵니다.

lpSys는 시스템 이름 리턴 ! 사k되는 AS400_Sys 8조를 리턴합니다. b본 시스템은 8조의 첫 시스템입니다. AS400_Sys 8조 ! 대한 설명은 345페이지의 『AS400_SYS』를 참조하십시@.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_QueryConvState

목적

이 함수는 지정된 대화의 상태를 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned pascal EHNAPPC_QueryConvState(
     HWND        hWnd,
     DWORD       dwConversation);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

리턴 코드

리턴 *은 대화의 현재 상태를 나타냅니다. 사k할 수 있는 *은 다음z 0 습니다.

EHNAPPC_RESET_STATE (0)

EHNAPPC_SEND_STATE (1)

EHNAPPC_RECEIVE_STATE (2)

EHNAPPC_RCVD_CONF_STATE (3)

EHNAPPC_RCVD_CONF_SEND_STATE (4)

EHNAPPC_RCVD_CONF_DEALL_STATE (5)

EHNAPPC_PEND_DEALLOCATE_STATE (6)

EHNAPPC_INVALID_STATE (7)

EHNAPPC_QueryFullSystems

목적

이 함수는 클라이p트! ,a되n 있는 시스템의 이름z 네트v크명을 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryFullSystems(
     HWND            hWnd,
     LPINT           lpSysCount,
     LPFULLSYSSTRUC  lpSys);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

lpSysCount는 ,a된 시스템 수 리턴! 사k되는 정수 변수를 !리킵니다.

lpSys는 시스템 이름 리턴! 사k되는 AS400_Sys 8조를 리턴합니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_QueryUserid

목적

이 함수는 지정된 시스템 ,a! 사k되는 사k자 ID를 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryUserId(
     HWND          hWnd,
     LPSTR         lpszLocationName,
     LPSTR         lpszUserId);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpszLocationName은 조회될 시스템명이 있으면서 널(null)로 중단되는 문자 - 을 ! 리킵니다. b 본 시스템 ! 대해 사 k 자 ID를 조회하려면 NULL을 지정하십시 @. lpszUserId는 지정된 시스템 ! 대한 사 k 자 ID 리턴 ! 사 k 되 m 널(null)로 중단되는 문자 - 을 ! 리킵니다.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시 @.

EHNAPPC_QuerySystems

목적

이 함수는 클라이 p 트 ! , a 된 시스템의 이름을 리턴합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QuerySystems(
     HWND           hWnd,
     LPINT          lpSysCount,
     LPSYSSTRUC     lpSys);
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpSysCount는 , a 된 시스템 수 리턴 ! 사 k 되는 정수 변수를 ! 리킵니다.

lpSys는 시스템 이름 리턴 ! 사 k 되는 AS400_Sys 8 조를 리턴합니다. b 본 시스템은 8 조의 첫 시스템입니다. AS400_Sys 8 조 ! 대한 설명은 345페이지의 『AS400_SYS』를 참조하십시 @.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시 @.

EHNAPPC_ReceiveAndWait

목적

이 함수는 대화시 도착할 정보를 b 다렸다 ! 이 정보를 수신합니다.

프로시듀어 선언

```
#include "E32APPC.H"
extern int EHNAPPC_ReceiveAndWait(
     HWND           hWnd,
     DWORD          dwConversation,
     FillEnu        bFill,
     int            nMaxLength,
     LPVOID         lpReceiveData,
     LPBYTE         lpWhatReceived,
     LPBYTE         lpRequestToSendRcvd,
     LPWORD         lpReceiveDataLength );
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

bFill은 프로 W 램이 데이타를 수신할 g 식을 나타냅니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_BUFFER (0) (버퍼 채 r)

EHNAPPC_LL (1) (O 전하 E 나 잘린 논리적 레코드 수신)

nMaxLength는 수 k 될 수 있는 데이타의 최대량을 나타냅니다.

lpReceiveData는 데이타 ! 수신될 버퍼를 ! 리킵니다.

lpWhatReceived는 클라이 p 트 ! 서 수신한 내 k 을 나타냅니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_DATA (0)

EHNAPPC_DATACOMPLETE (1)

EHNAPPC_DATAINCOMPLETE (2)

EHNAPPC_RECEIVEDCONFIRM (3)

EHNAPPC_RECEIVEDCONFIRMSEND(4)

EHNAPPC_RECEIVEDCONFIRMDEALLOC(5)

EHNAPPC_RECEIVEDSEND (6)

lpRequestToSendRcvd는 상대방 트랜잭션 프로 W 램이 REQUEST_TO_SEND 명령 n 를 발행했는지의) 부 저장 ! 사 k 되는 변수를 ! 리킵니다. TRUE(1) 인 * 은 상대방 트랜잭션 프로 W 램이 REQUEST_TO_SEND 명령 n 를 발행했음을 나타냅니다.

lpReceiveDataLength는 클라이 p 트 ! 서 수신한 데이타량 리턴 ! 사 k 되는 변수를 ! 리킵니다.

리턴 코드

리턴 코드 ! 대해 347페이지의 『EHNAPPC API ! 대한 리턴 코드』를 참조하십시 @ .

EHNAPPC_ReceiveImmediate

목적

이 함수는 수신 내 k 이 있는지를 확인합니다. 수신된 f l ! 는 데이타 ! 리턴됩니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_ReceiveImmediate(
       HWND        hWnd,
       DWORD       dwConversation,
       FillEnum    bFill,
       int         nMaxLength,
       LPVOID      lpReceiveData,
       LPBYTE      lpWhatReceived,
       LPBYTE      lpRequestToSendRcvd,
       LPWORD      lpReceiveDataLength );
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended ! 서 리턴된 대화 핸들을 식별합니다.

bFill은 프로 W 램이 데이타를 수신할 g 식을 나타냅니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_BUFFER (0) (버퍼 채 r)

EHNAPPC_LL (1) (O 전하 E 나 잘린 논리적 레코드 수신)

nMaxLength는 수 k 될 수 있는 데이타의 최대량을 나타냅니다.

lpReceiveData는 데이타 ! 수신될 버퍼를 ! 리킵니다.

lpWhatReceived는 클라이 p 트 ! 서 수신된 내 k 을 식별합니다. 사 k 할 수 있는 * 은 다음 z 0 습니다.

EHNAPPC_DATA (0)

EHNAPPC_DATACOMPLETE (1)

EHNAPPC_DATAINCOMPLETE (2)

EHNAPPC_RECEIVEDCONFIRM (3)

EHNAPPC_RECEIVEDCONFIRMSEND(4)

EHNAPPC_RECEIVEDCONFIRMDEALLOC(5)

EHNAPPC_RECEIVEDSEND (6)

lpRequestToSendRcvd는 상대방 트랜잭션 프로 W 램이 REQUEST_TO_SEND 명령 n 를 발행했는지의) 부 저장 ! 사 k 되는 변수를 ! 리킵니다. TRUE (1) 인 * 은 상대방 트랜잭션 프로 W 램이 REQUEST_TO_SEND 명령 n 를 발행했음을 나타냅니다.

lpReceiveDataLength는 클라이p트!서 수신한 데이타량을 리턴하b 위해 사k되는 변수를 !리킵니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_RemoteProgramStart

목적

Windows n플리케이션은 이 함수를 사k하) x] AS/400 시스템!서 프로W램을 시작할 수 있습니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern word EHNAPPC_RemoteProgramStart(
    HWND          hWnd,
    LPSTR         lpszHostSystemName,
    LPSTR         lpszHostProgramName,
    LPSTR         lpszHostLibraryName,
    char FAR      *lpchPipData,
    WORD          wPipDataLength);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

lpszHostSystemName은 x] 시스템명을 포함하m 있으면서 널(null)로 중단되는 문자-을 !리킵니다. 이 문자-의 최대 f이는 8문자입니다. 이 포인터! 널이면 b본 시스템명이 사k됩니다.

lpszHostProgramName은 시작될 호스트 프로W램의 이름이 들n 있으면서 널(null)로 중단되는 문자-을 !리킵니다.

lpszHostLibraryName은 호스트 프로W램의 라이브러리 f로! 들n 있으면서 널(null)로 중단되는 문자-을 !리킵니다. 이 포인터! 널이면 사k자의 라이브러리 리스트를 탐색합니다.

lpchPipData는 호스트 프로W램! 대한 프로W램 초b화 매3변수(PIP) 데이타 5*을 !리킵니다. 이 포인터! 널이면 전송되는 PIP 데이타! x습니다.

wPipDataLength!는 PIP 데이타 f이! 들n 있습니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_RqsToSend

목적

이 함수는 상대방이 대화의 제n G을 포b하도록 d청합니다. 클라이p트는 로컬 트랜잭션 프로W램이 ,속적으로 상대방 트랜잭션 프로W램!서 Receive의 lpWhatReceived 매3변수! 있는 EHNAPPC_RECEIVEDSEND(6)을 수신할 때 대화를 전송 상태! 놓습니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_RqsToSend(
     HWND      hWnd,
     DWORD     dwConversation);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended!서 리턴된 대화 핸들을 식별합니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_SendData

목적

이 함수는 데이타를 상대방 트랜잭션 프로W램으로 전송합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_SendData(
     HWND      hWnd,
     DWORD     dwConversation,
     int       nSendDataLength,
     LPVOID    lpSendDataBuffer,
     LPBYTE    lpRequestToSendRcvd);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended!서 리턴된 대화 핸들을 식별합니다.

nSendDataLength는 전송 버퍼! 있는 데이타 f이를 식별합니다.

lpSendDataBuffer는 전송 버퍼의 주소를 식별합니다.

lpRequestToSendRcvd는 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했는지의)부 저장! 사k되는 변수를 !리킵니다. TRUE * 은 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했음을 나타냅니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_SendError

목적

이 함수는 몇 !지 @류를 발_했다는 사실을 상대방 트랜잭션 프로W램! T K립니다. 이 함수를 사k한 후! 로컬 프로W램이 수신 상태! 들n)니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_SendError(
     HWND        hWnd,
     DWORD       dwConversation,
     LPBYTE      lpRequestToSendRcvd);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

dwConversation은 EHNAPPC_Allocate 또는 EHNAPPC_Extended!서 리턴된 대화 핸들을 식별합니다.

lpRequestToSendRcvd는 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했는지의)부 저장! 사k되는 변수를 !리킵니다. TRUE * 은 상대방 트랜잭션 프로W램이 REQUEST_TO_SEND 명령n를 발행했음을 나타냅니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC_StartHostProgram

목적

Windows n플리케이션은 이 함수를 사k하) 호스트 프로W램이 실행중임을 확인할 수 있도록 대화를 활동 상태로 남\ 놓m, x] AS/400 시스템! 서 프로W램을 시작할 수 있습니다. n플리케이션은 대화 종료시 EHNAPPC_Deallocate 함수를 사k합니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern word EHNAPPC_RemoteProgramStart(
     HWND        hWnd,
     LPSTR       lpszHostSystemName,
     LPSTR       lpszHostProgramName,
     LPSTR       lpszHostLibraryName,
     char FAR    *lpchPipData,
     WORD        wPipDataLength);
```

매개변수

hWnd는 n플리케이션의 현재 창을 식별합니다.

lpszHostSystemName은 x] 시스템명을 포함하m 있으면서 널(null)로 중단되는 문자를 !리킵다. 이 문자 - 의 최대 f이는 8문자입니다. 이 포인터! 널이면 b본 시스템명이 사k됩니다.

lpszHostProgramName은 시작될 호스트 프로W램의 이름이 들n 있으면서 널로 중단되는 문자 - 을 !리킵니다.

lpszHostLibraryName은 호스트 프로W램의 라이브러리 f로! 들n 있으면서 널(null)로 중단되는 문자 - 을 !리킵니다. 이 포인터! 널이면 사k자의 라이브러리 리스트를 탐색합니다.

lpchPipData는 호스트 프로W램! 대한 프로W램 초b화 매3변수(PIP) 데이타 5 *을 !리킵니다. 이 포인터! 널이면 전송되는 PIP 데이타! x습니다.

wPipDataLength!는 PIP 데이타 f이! 들n 있습니다.

리턴 코드

리턴 코드! 대해 347페이지의 『EHNAPPC API! 대한 리턴 코드』를 참조하십시@.

EHNAPPC 구조

AS400_SYS

목적

이 8조는 클라이p트! , a되n 있는 시스템의 이름을 저장하b 위해 사k됩니다.

프로시듀어 선언

```
struct AS400_sys
     (
unsigned char EHNAPPC_SysNameHNAPPC_MAX_SYSTEMS|
     HNAPPC_SYSNAME_SYSNAME_LENGTH|;
     );
```

매개변수

EHNAPPC_SysName은 , a된 시스템의 이름을 저장하b 위해 사k됩니다. 시스템명은 널(null)로 중단되는 문자 - 로 리턴됩니다. 배 - !서 리턴된 첫 시스템은 b본 시스템입니다(EHNAPPC_MAX_SYSTEMS = 32 및 EHNAPPC_SYSNAME_SYSNAME_LENGTH = 10).

appcrtrcap_hdr

목적

클라이p트 b능 리스트 헤더의 8조입니다.

프로시듀어 선언

```
struct appcrtrcap_hdr
     (
     unsigned char rc;
     unsigned char opcode;
     unsigned int length;
     );
```

매개변수

rc는 b능 d청의 전체적인 리턴 코드 저장! 사k됩니다.

opcode는 b능 확보 d청을 신호합니다. W * 은 EHNAPPC_OC_CAPABILITIES(0x17)) _ 합니다.

length는 전체 b능 리스트의 f이를 식별합니다. f이!는 헤더M " b능 8조의 크b ! 포함됩니다.

appcrtrcap_mult

목적

최적 통신 버퍼 v수 판별! 사k되는 b능 8조입니다.

프로시듀어 선언

```
struct appcrtrcap_mult
     (
     unsigned int length;
     unsigned char identifier;
     unsigned char rc;
     unsigned int data;
     );
```

매개변수

length는 이 b능 8조 f이를 식별합니다.

identifier는 최적의 통신 버퍼 v수를 신호해 줍니다. W * 은 EHNAPPC_CAP_OPTIMAL_COM_SIZE (X'02')) _ 합니다.

rc는 이 b능 d청의 리턴 코드 저장! 사k됩니다.

data는 최적의 통신 버퍼 v수 리턴! 사k됩니다.

appcrtrcap_query

목적

클라이p트! 지정된 b능을 지x하는지 조회하b 위해 사k되는 b능 8조입니다.

프로시듀어 선언

```
struct appcrtrcap_query
     (
     unsigned int length;
     unsigned char identifier;
     unsigned char rc;
     unsigned char data;
      );
```

매개변수

length는 이 b능 8조 f이를 식별합니다.

identifier는 조회될 함수를 식별합니다. 사k할 수 있는 * 은 다음z 0습니다.

EHNAPPC_CAP_QUERY_CONV_STATE (3)

EHNAPPC_CAP_EXT_ALLOCATE (4)

rc는 이 b능 d청의 리턴 코드 저장! 사k됩니다.

data는 지정된 함수의 지x)부 리턴! 사k됩니다.

EHNAPPC API에 대한 리턴 코드

클라이p트 Windows API의 함수는 E32APPC.H!서 정의된 다음 리턴 코드 상수를 사k합니다.

표 20. 리턴 코드

리턴 코드	16진수 *	설명
EHNAPPC_OK	0	성x적으로 0료된 명령.
ENHAPPC_DEALLOCNORMAL	1	할당 해제 정상.
ENHAPPC_PROGRAMMERNOTRUNCATION	2	프로W램 @류. 절단 x음.
ENHAPPC_PROGRAMMERTRUNCATION	3	프로W램 @류. 절단.
ENHAPPC_PROGRAMMERPURGING	4	프로W램 @류. 제E.
ENHAPPC_RESOURCEFAILURETRY	5	자x 장V 재시도.
ENHAPPC_RESOURCEFAILURENORETRY	6	자x 장V 재시도 x음.
ENHAPPC_UNSUCCESSFUL	7	실패.
ENHAPPC_APPCBUSY	8	APPC 사k중.
ENHAPPC_PARMCHKINVALIDVERB	14	매3변수 확인. 틀린 명령n.
ENHAPPC_PARMCHKINVALIDCONVERID	15	매3변수 확인. 틀린 대화 ID.
ENHAPPC_PARMCHKBUFFERCROSSEG	16	매3변수 확인. 버퍼 3차 세W멘트.
ENHAPPC_PARMCHKTPNAMELENGTH	17	매3변수 확인. 트랜잭션 프로W램명 (TPN) f이.
ENHAPPC_PARMCHKINVCONVERTYPE	18	매3변수 확인. 틀린 대화 유형.
ENHAPPC_PARMCHKBADSYNCLVLALLOC	19	매3변수 확인. 잘못된 동b화 레벨 할당.
ENHAPPC_PARMCHKBADRETURNCNTRL	1A	매3변수 확인. 잘못된 리턴 제n.
E N H A P P C _ ENHAPPC_PARMCHKPIPTOOLONG	1B	매3변수 확인. PIP 데이타! 너무 h.
ENHAPPC_PARMCHKBADPARTNERNAME	1C	매3변수 확인. 잘못된 상대방 이름.
ENHAPPC_PARMCHKCONFNOTALLOWED	1D	매3변수 확인. 확인할 수 x음.
ENHAPPC_PARMCHKBADDEALLOCTYPE	1E	매3변수 확인. 잘못된 할당 해제 유형.
ENHAPPC_PARMCHKPREPTORCVTYPE	1F	매3변수 확인. 유형 수신 준비 0료.
ENHAPPC_PARMCHKBADFILLTYPE	20	매3변수 확인. 잘못된 채r 유형.
ENHAPPC_PARMCHKRECMAXLEN	21	매3변수 확인. 최대 f이 수신.
ENHAPPC_PARMCHKUNKNOWNSECTYPE	22	매3변수 확인. 9` 필드! 0이 F님.
ENHAPPC_PARMCHKRESFLDNOTZERO	23	매3변수 확인. 9` 필드! 0이 F님.
ENHAPPC_STATECHKNOTINCONFSTAT	28	상태 확인. 확인 상태! x음.

표 20. 리턴 코드 (h 속)

리턴 코드	16진수 *	설명
ENHAPPC_STATECHKNOTINRECEIVE	29	상태 확인. 수신 상태! x음.
ENAHAPPC_STATECHKREQSNDBADSTATN	2A	상태 확인. 잘못된 상태 전송 d청.
ENHAPPC_STATECHKSNDINBADSTATE	2B	상태 확인. 잘못된 상태!서 전송.
ENHAPPC_STATECHKSNDERRBADSTAT	2C	상태 확인. 잘못된 @류 상태 전송.
ENHAPPC_ALLOCERRNORETRY	32	할당 @류. 재시도 x음.
ENHAPPC_ALLOCERRRETRY	33	할당 @류. 재시도.
ENHAPPC_ALLOCERROGMNOTAVAILNR	34	할당 @류. 프로W램!서 재시도할 수 x음.
ENHAPPC_ALLOCERRTPNNOTRECOG	35	할당 @류. 트랜잭션 프로W램명을 인식하지 못함.
ENHAPPC_ALLOCERRPGMNOTAVAILR	36	할당 @류. 프로W램!서 재시도할 수 x음.
ENHAPPC_ALLOCERRSECNOTVALID	37	할당 @류. 보H이 유효하지 J음.
ENHAPPC_ALLOCERRCONVTYP	38	할당 @류. 대화 유형 불일치.
ENHAPPC_ALLOCERRPIPNOTALLOWED	39	할당 @류. PIP 데이타 불k.
ENHAPPC_ALLOCERRPIPNOTCORRECT	3A	할당 @류. PIP 데이타! G지 J음.
ENHAPPC_ALLOCERRSYNCHLEVEL	3B	할당 @류. 동b화 레벨 지x H됨.
ENHAPPC_DEALLOCABENDPROGRAM	46	할당해제 이상종료 프로W램.
ENHAPPC_INSUFFICIENTMEMORY	47	메모리 부족.
ENHAPPC_MEMORYALLOCERROR	47	메모리 할당 @류.
ENHAPPC_MEMORYALLCERROR	48	메모리 할당 @류.
ENHAPPC_TOOMANYCONVERSATIONS	4A	대화! 너무 많음.
ENHAPPC_CONVTABLEFULL	4B	전환 테이블이 다 찼음.
ENHAPPC_CLIENTNOTINSTALLED	4C	클라이p트! 설치되지 J음.
ENHAPPC_CLIENTWRONGLEVEL	4C	클라이p트! 틀린 레벨! 있음.
ENHAPPC_PCSWINNOTLOADED	4D	PSWIN이 로드되지 J음.
ENHAPPC_PCSWINOUTOFMEMORY	4E	PCSWIN 메모리 부족.
ENHAPPC_INVALIDUSERIDLEN	4F	틀린 사k자 ID f이.
ENHAPPC_INVALIDPASSWORDLEN	50	틀린 O호 f이.
ENHAPPC_INVALIDUNAME	51	틀린 LU f이.
ENHAPPC_UNDEFINED	63	정의되지 J음.

Windows 95 및 Windows NT에 - 의 16 비트 EHNAPPC 프로그램 실행

통신 서버 SNA API Windows 95 및 Windows NT 클라이 p 트는 b 존의 16 비트 EHNAPPC 프로 W 램을 Windows 95 및 Windows NT ! 서 실행시킬 수 있는 b 능을 제 x 합니다. 이렇 T 하려면 16 비트 EHNAPPC n 플리케이션을 시작하 b 전 ! 통신 서버 SNA API 클라이 p 트 서브디렉토리 ! 서 EHNAPPCD 프로 W 램을 시작해 _ 합니다. 이 프로 W 램은 32 비트 E32APPC.DLL ! 서 필 d 한 던킹(thunking)을 제 x 합니다.

제19장 데이타 변환 Windows 어플리케이G 프로그램 인터페이스

통신 서버 SNA API 클라이p트!서만 사k할 수 있습니다.

데이타 변환 API는 AS/400z PC 포맷#의 데이타를 변환할 수 있는 b능을 제x합니다. 번*은 AS/400z의 데이타 송수신시 필d합니다. 데이타 변환 API는 텍스트 및 다g한 숫자 포맷의 변환을 지x합니다.

이 장!서는 데이타 변환 API를 형성하는 3별적인 루틴z 리턴 코드! 대해 설명합니다.

데이타 변환 Windows API 루틴

"데이타 변환 API 루틴! 대한 다음 설명이 상세하T 이루n집니다.

- 목적
- 프로시듀n 선p
- 매3변수
- 리턴 코드

EHNDT_ANSIToEBCDIC

목적

이 함수는 Windows ANSI 코드 페이지의 문자-을 EBCDIC로 변환합니다. 라l터는 이 루틴이 ASCII-EBCDIC 변환 테이블을 W세스할 수 있도록 로드되n_합니다.

목표 문자-이 변환된 문자-보다 작으면 목표 문자- 끝!서 변환이 종료됩니다. 목표 문자-이 필d한 M보다 더 크면 문자- 끝이 x백으로 채v집니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_ANSIToEBCDIC(
     HWND            hWnd,
     LPSTR           lpsSource,
     LPSTR           lpsTarget,
     unsigned in     wSource,
     LPWORD          lpwTarget );
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpsSource는 변환할 소스 (ANSI) 문자 - 을 ! 리킵니다.

lpsTarget는 목표 (변환된) 문자 - 을 ! 리킵니다. **wSource**는 소스 문자 - 의 f 이를 바이트로 식별합니다.

lpwTarget는 목표 버퍼 크b ! 들n 있는 단n 변수를 ! 리킵니다. 이 변수는 목표 버퍼의 변환된 문자의 총 수로 ; 신됩니다.

리턴 코드

함수 ! 성 x 하면 EHNDT_SUCCESS (X'0000') ! 리턴됩니다. 라 l 터 ! 로드되지 J 으면 EHNDT_A2E_TABLE_NOT_FOUND (X'FFFC') ! 리턴됩니다. 임시 버퍼를 할당하려 m 할 때 @ 류 ! 발생하면 EHNDT_MEMALLOC(X'FFFF')이 리턴됩니다. 틀린 데이타 ! 변환중 발 _ 되면 리턴 코드는 변환되지 J 은 첫 문자의 위치 ! 1을 더한 * 이 됩니다.

EHNDT_ASCIIToEBCDIC

목적

이 함수는 문자 - 을 ASCII ! 서 EBCDIC로 변환합니다. 라 l 터는 이 루틴이 ASCII-EBCDIC 변환 테이블을 W 세스할 수 있도록 로드되 n _ 합니다. 목표 문자 - 이 변환된 문자 - 보다 작으면 목표 문자 - 끝 ! 서 변환이 종료됩니다. 목표 문자 - 이 필 d 한 M 보다 크면 문자 - 끝이 x 백으로 채 v 집니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_ASCIIToEBCDIC(
     HWND            hWnd,
     LPSTR           lpsTarget,
     LPSTR           lpsSource,
     unsigned in     wSource,
     LPWORD          lpwTarget );
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpsTarget는 목표 (변환된) 문자 - 을 ! 리킵니다.

lpsSource는 변환할 소스 (ASCII) 문자 - 을 ! 리킵니다.

wSource는 소스 문자 - f 이를 바이트로 식별합니다.

lpwTarget는 목표 버퍼의 크b ! 들n 있는 단n 변수를 ! 리킵니다. 이 변수는 목표 버퍼의 변환된 문자의 총 수로 ; 신됩니다.

리턴 코드

함수 ! 성 x 하면 EHNDT_SUCCESS (X'0000') ! 리턴됩니다. 라 l 터 ! 로드되지 J 으면 EHNDT_A2E_TABLE_NOT_FOUND (X'FFFC') ! 리턴됩니다.

틀린 데이타 ! 변환중 발 _ 되면 리턴 코드는 변환되지 J 은 첫 문자의 위치 ! 1을 더한 * 이 됩니다.

EHNDT_EBCDICToANSI

목적

이 함수는 문자 - 을 EBCDIC ! 서 Windows ANSI 코드 페이지로 변환합니다. 라 l 터는 이 루틴이 ASCII-EBCDIC 변환 테이블을 W 세스할 수 있도록 로드되 n _ 합니다.

목표 문자 - 이 변환된 문자 - 보다 작으면 목표 문자 - 끝 ! 서 변환이 종료됩니다. 목표 문자 - 이 필 d 한 M 보다 크면 문자 - 끝이 x 백으로 채 v 집니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_EBCDICToANSI(
HWND             hWnd,
LPSTR            lpsTarget,
LPSTR            lpsSource,
unsigned int     wSource,
LPWORD           lpwTarget ); :
```

매개변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpsTarget은 목표 (변환된) 문자 - 을 ! 리킵니다.

lpsSource는 변환할 소스 (EBCDIC) 문자 - 을 ! 리킵니다.

wSource는 소스 문자 - 의 f 이를 바이트로 식별합니다.

lpwTarget은 목표 버퍼의 크b ! 들n 있는 단n 변수를 ! 리킵니다. 이 변수는 목표 버퍼의 변환된 문자의 총 수로 ; 신됩니다.

리턴 코드

함수 ! 성 x 하면 EHNDT_SUCCESS ('0000') ! 리턴됩니다. 라 l 터 ! 로드되지 J 으면 EHNDT_E2A_TABLE_.NOT_FOUND ('FFFC') ! 리턴됩니다. 틀린 데이타 ! 변환중 발 _ 되면 리턴 코드는 변환되지 J 은 첫 문자의 위치 ! 1을 더한 * 이 됩니다.

EHNDT_EBCDICToASCII

목적

이 함수는 문자 - 을 EBCDIC ! 서 ASCII로 변환합니다. 라 l 터는 이 루틴이 ASCII-EBCDIC 변환 테이블을 W 세스할 수 있도록 로드되 n _ 합니다.

목표 문자 - 이 변환된 문자 - 보다 작으면 목표 문자 - 끝 ! 서 변환이 종료됩니다. 목표 문자 - 이 필 d 한 M 보다 크면 문자 - 끝이 x 백으로 채 v 집니다.

프로시듀어 선언

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_EBCDICToASCII(
     HWND          hWnd,
     LPSTR         lpsTarget,
     LPSTR         lpsSource,
     unsigned int  wSource,
     LPWORD        lpwTarget );
```

매 3 변수

hWnd는 n 플리케이션의 현재 창을 식별합니다.

lpsTarget은 목표 (변환된) 문자 - 을 ! 리킵니다.

lpsSource는 변환할 소스 (EBCDIC) 문자 - 을 ! 리킵니다.

wSource는 소스 문자 - f 이를 바이트로 식별합니다.

lpwTarget은 목표 버퍼의 크 b ! 들 n 있는 단 n 변수를 ! 리킵니다. 이 변수는 목표 버퍼의 변환된 문자의 총 수로 ; 신됩니다.

리턴 코드

함수 ! 성 x 하면 EHNDT_SUCCESS ('0000') ! 리턴됩니다. 라 l 터 ! 로드되지 J 으면 EHNDT_.E2A_TABLE_.NOT_FOUND ('FFFC') ! 리턴됩니다. 틀린 데이타 ! 변환중 발 _ 되면, 리턴 코드는 변환되지 J 은 첫 문자의 위치 ! 1을 더한 * 이 됩니다.

제5부 자바 프로그래밍 인터페이스

제20장 JAVA용 호스트 액<스 등급 라이브러리 R개

이 장!서는 다음을 비롯한 3270 및 5250 n플리케이션z |련된 Javak IBM eNetwork 호스트 W세스 등^ 라이브러리(ECL)! 대해 설명합니다.

- Javak ECL의 8조! 대한 #단한 3d
- ECL! 설치되는 내k
- 사k할 수 있는 샘플z 작동 방법

주: 문자n HACL도 eNetwork 호스트 W세스 등^ 라이브러리를 설명하b 위해 사k됩니다. 이 책!서는 ECL을 더 선호합니다.

ECL은 무엇인가?

Javak ECL은 n플리케이션 프로W래머! 3270 및 5250 데이타 스트림 레벨!서 호스트 n플리케이션을 빠르m 쉽T W세스할 수 있도록 등^z 방법을 모F 놓은 집합입니다. ECL은 n떤 W래픽 표시 장치M도 |h x으며 작동시 Java 작동 K색b 또는 이M 비슷한 Java 환f만 .추면 되는 코n 호스트 W세스 함수를 O전한 등^ 모델로 8현합니다.

등^ 라이브러리는 다음을 비롯한 호스트 ,a의 O전한 @브젝트 지향 추상화를 나타냅니다.

- 호스트 프리젠테이션 서비스 x# (화면) 읽b 및 2b
- 프리젠테이션 서비스 x#!서의 필드 h산
- 상태 정보! 대한 n5dx 정보 5* (OIA) 읽b
- 파일 전송하b
- 중d한 이벤트의 비동b 통보 수행하b

n플리케이션 프로W래머는 데이타 스트림 프리젠테이션 서비스 x#(3270 및 5250z 0은)!서 데이타를 처리하는 Java V플릿을 작성할 수 있습니다. V플릿은 이런 시스템! 상주할 필d는 x습니다. 프리젠테이션 서비스 x#은 데이타M 호스트 n플리케이션! 의해 표시되는 |련 속성이 있는 !상 터미널 화면을 나타냅니다. V플릿은 상호작k이 끝난 후! 다른 타스크로 전환하E나 또는 단순히 세션을 닫습니다. 트랜잭션은 호스트 화면을 표시하지 Jm도 O료될 수 있습니다.

ECL Java V플릿은 다음을 수행할 수 있습니다.

- 호스트!서 세션 -b
- 입력 호스트 데이타 대b
- !상 화면!서 특정 문자- rb
- 문자-의 |련 속성 rb
- 새로n 문자- *설정
- 다시 호스트로 데이타 스트림 b능 키 전송

- 다음 호스트 세션 대b

ECL은 다음z 0은)러 !지 측면!서 EHLLAPIM 0은 특정 클라이p트, 화면 수집 n플리케이션 프로W래밍 인터페이스! 비해 상당히 3선된 인터페이스입니다.

- ECL은 플랫폼z |hx습니다
- ECL은 번*된 !뮬레이터 화면!서보다는 직접 데이타 스트림!서 작동합니다. 이는 화면! 표시되는 데이타 스트림을 번*하m 표시해_ 하는 @버헤드를 xV줍니다.
- ECL은 로컬 v크스테이션!서 실행되는 !뮬레이터 소프트~n! 필d하지 J으므로 플랫폼! m유한 화면 포맷z 키보드 배치! 대한 의존성을 줄)줍니다.
- ECL은 표준 웹 및 Java b술을 사k하) 클라이p트 v크스테이션!서 다n로드되m 실행될 수 있습니다. 이로 인하) 상당한 유지보수 및 자x을 절`할 수 있습니다.

ECL 개념

다음 섹션!서는 ECL의)러 !지 본질적인 3념을 설명합니다. 이런 3념을 이해할 수 있으면 라이브러리를 더 효z적으로 사k할 수 있T 됩니다.

세션

ECL 문맥!서 세션 @브젝트(ECLSession)는 호스트M의 ,az W ,a의 특성을 캡슐화합니다. 세션 @브젝트는 또한 b타 세션 m유 @브젝트! 대한 컨테이너로도 사k됩니다. 이런 @브젝트!는 ECLPS(프리젠테이션 서비스 x#), ECLOIA (n5dx 정보 5*), ECLXfer(파일 전송)! 있습니다.

세션 @브젝트!는 |련된 W래픽 사k자 인터페이스(GUI)! x습니다. 다시 말해서 ECLSession의 인스턴스를 작성한다m 해서 !뮬레이터 화면이 표시되지는 J습니다.

컨테이너 오브젝트

)러 ECL 등^은 b타 @브젝트의 컨테이너로 작동합니다. ECLSession @브젝트!는 ECLPS, ECLOIA, ECLXfer @브젝트의 인스턴스! 있습니다. 컨테이너는 포인터를 들n 있는 @브젝트로 리턴하는 방법을 제x합니다. ECLSesison @브젝트!는 포인터를 OIA @브젝트로 리턴하는 GetOIA 방법이 있습니다. 들n 있는 @브젝트는 컨테이너 등^의 xk 8성x으로 8현되b보다는 ECL 방법을 통해서만 W세스됩니다.

리스트 오브젝트

)러 ECL 등^은 리스트 반복 b능을 제x합니다. 9를 들n ECLConnList 등^은 ,a 리스트를 |리합니다. ECL 리스트 등^은 리스트 변f 내k을 반5하b 위해 비동b적으로 ;신되지는 J습니다. n플리케이션은 리스트

내k ;신시 화면정리 방법을 확실하T 호출해_ 합니다. 이로 인하) n 플리케이션은 반복 z정중! 일n날 수도 있는 리스트의 변f을 0려하지 J m도 리스트를 반복시킬 수 있습니다.

이벤트

ECL은 특정 이벤트를 비동b적으로 통보 받을 수 있는 b능을 제x합니다. n플리케이션은 특정 이벤트! 발생할 때 통보 받지 J을 수도 있습니다. 9를 들n n플리케이션은 호스트로의 ,a 상태! 변화! 생f 때 통보받을 수 있습니다. 현재 ECL은 다음 이벤트! 대한 통보를 지x합니다.

표 21. APPCk 헤더 파일 및 라이브러리

이벤트	이벤트 캡처에 사용되는 인터페이스
통신 ,a 및 ,a단절	ECLLCommNotify
프리젠테이션 서비스 x# ;신	ECLPSNotify
n5dx 정보 5* (OIA) ;신	ECLOIANotify

이벤트 통보는 " "의 ECL 통보 인터페이스! 의해 정의됩니다. 이벤트 유형마다 별도의 인터페이스! 있습니다. 이벤트 발생을 통보 받으려면 n플리케이션은 통보해_ 하는 이벤트 유형! 대한 인터페이스를 8현하는 @브젝트를 정의하m 작성해_ 합니다. W러면 W @브젝트는 해당 ECL 등록b능을 호출하) 등록되n_ 합니다. 일단 n플리케이션 @브젝트! 등록되면 이벤트! 발생할 때마다 W NotifyEvent 방법이 호출됩니다.

주: n플리케이션의 NotifyEvent 방법은 별도의 수행 스레드!서 비동b적으로 호출됩니다. W러므로 NotifyEvent 방법은 다시 입력되n_ 합니다. 해당 잠] 또는 동b화는 n플리케이션 자x이 W세스될 때 사k됩니다.

오류 처리하기

일반적으로 ECL은 ECLErr @브젝트를 던져서 @류를 n플리케이션! 표시합니다. @류를 잡으려면 n플리케이션!서 다음z 0은 시도/잡b 블럭! ECL @브젝트의 호출을 포함시켜_ 합니다.

```
try {
       int pos = ps.ConvertRowColToPos(row, col);

       //...아마도 ECL 오브젝트에 대해 더 많은 참조 사항...

   } catch (ECLErr err) {
      System.out.println("ECL 오y!  " + err.GetMsgText());
   }
```

ECL @류! (지되면 n플리케이션은 이 @류의 정확한 x인을 판별하b 위해 ECLErr @브젝트 방법을 호출합니다. ECLErr @브젝트는 O전한 pn 민(@류 메세지 8성!서도 호출됩니다.

주소지정 (행, 열, 위치)

ECL은 호스트 프리젠테이션 서비스 x # ! 서 주소를 지정(문자 위치)하는 두 ! 지 방법을 제 x 합니다. n 플리케이션은 행/ - 번호 또는 단일 선형 위치 * ! 의해 문자의 주소를 지정할 수 있습니다. 프리젠테이션 서비스 x # ! 서의 주소지정은 주소지정 8 성 ! | h x 이 항상 1을 b 본으로 합니다 (0이 F 님).

행 및 - 주소지정 8 성은 호스트 데이타의 물리적 화면 프리젠테이션 서비스 ! 직접 | 련된 n 플리케이션 ! 유 k 하 T 사 k 됩니다. 직사 " 형의 좌표 h (상단 ^ 쪽 8 석 ! 행 1 z - 1)는 화면 ! 서 점의 주소를 지정하는 일반적인 방법입니다. 선형 위치 주소지정 방법(맨 위 ^ 쪽 8 석 ! 위치 1이 있 m ^ 쪽 ! 서 @ 른쪽, 맨 위 ! 서 맨 F 래로 진행)은 전체 프레젠테이션 서비스 x # 을 하나의 데이타 d 소 배 - 로 보는 n 플리케이션 또는 EHLLAPI 인터페이스 ! 서 포트된 n 플리케이션 ! 유 k 하 T 사 k 됩니다.

일반적으로 동일한 방법 ! 서 서로 다른 b 호를 호출하) 서로 다른 주소지정 8 성을 선택할 수 있습니다. 9 를 들 n 호스트 커서를 주 n 진 화면 좌표로 이동시키려면 n 플리케이션이 두 ! 지 신호 중 하나로 ECLPS::SetCursorPos 방법을 호출할 수 있습니다.

```
ps.SetCusorPos(81);
ps.SetCursorPos(2, 1);
```

이런 명령문은 호스트 화면이 한 행당 80 - 로 8 성되 n 있는 f l 0 은 효 z 를 발휘합니다. 이 9 제는 또한 주소지정 8 성 ! 있 n 서 미묘한 차이를 나타내 m 있습니다. 선형 위치 방법은 n 플리케이션이 프레젠테이션 서비스 x # 의 행 ! @ 는 문자의 수를 ! 정하는 f l 9 b 치 못한 a z 를 생성할 수 있습니다. 9 를 들 n 9 제의 첫 코드 행은 커서를 132 - 로 8 성된 프레젠테이션 서비스 x # ! 서 행 1의 - 81 ! 놓습니다. 두 번째 코드 행은 프레젠테이션 서비스 x # ! | h x 이 커서를 행 2, - 1 ! 놓습니다.

NT - 버용 통신 - 버에 ECL 3 치하기

Windows NT k 통신 서버 ! 서만 사 k 할 수 있습니다.

Windows NT 4.0 k 통신 서버 CD-ROM을 삽입하 m 인터페이스 단 h 를 수행한 후 InstallShield** 마법사의 설치를 시작하려면 **설정**을 눌러 _ 됩니다. 일단 설치되면 마법사 ! 나머지 설치 프로시듀 n 로 H 내해 줍니다. 마법사 설치 ! 0 료되면 IBM 통신 서버 환 5 창이 나타납니다. **다음**을 눌러 h 속하십시@. 일련의 다음 패널 ! 서는 설정 유형, 통신 서버를 설치할 드라이브 M 디렉토리, IBM 파일 On-Demand ! 대한 익명의 W 세스를 위한 FTP 디렉토리를 선택하도록 d 청합니다.

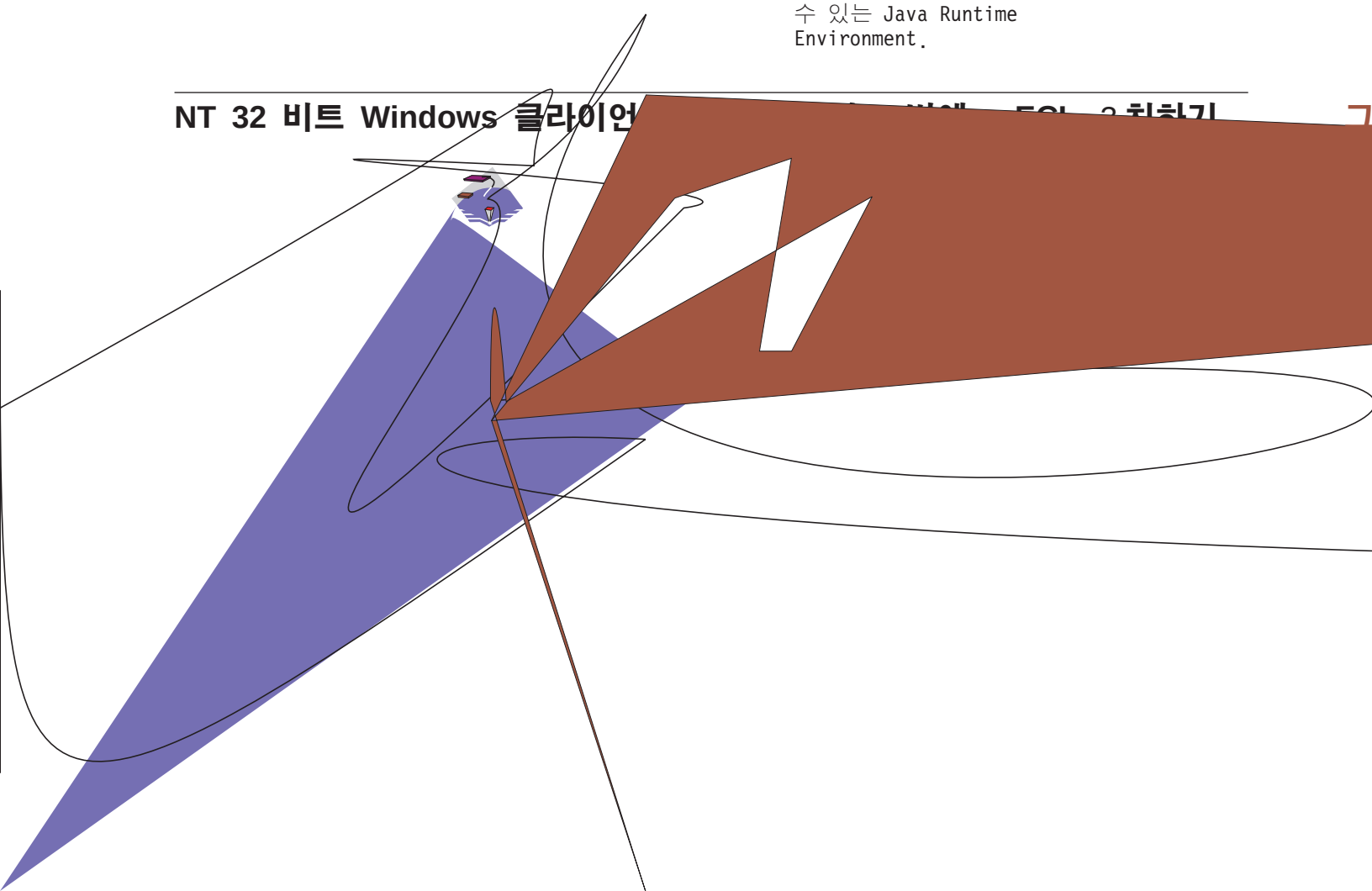
이 설치 z 정중 서버 ! 상주하는 V 플릿 ! 서 ECL Java 등 ^ 파일을 W 세스하 E 나 서버, ECL 코드페이지 변환 b, ECL ! 대한 문서, 샘플 Java V 플릿

z Java n플리케이션!서 ECL Java 등^ 파일을 W세스할 수 있는 능력을 제x합니다(클라이p트!서 Java n플리케이션으로 실행하b 위해 ECL을 서버! 설치할 필d! x습니다).

다음은 ECL 부분z 정의를 설명합니다.

IBMCS\ecl101.jar	이 파일은 서버에서 ECL Java 애플릿과 어플리케이션을 실행하기 위해 사용됩니다.
IBMCS*.class	이1 파일은 사용자가 정의한 서버 액세스용 디:토리 또는 웹 기본 액세스에 설치될 수 있습니다.
IBMCS\ECL\ZIP*.zip	ECL 등급 파일은 zip 포맷으N 되어 있습니다. 모든 ECL 등급 파일을 쉽게 액세스할 수 있도O 설치됩니다. 나중에 설명되는 코드 페이지 변환기는 ecl101n.zip 파일에 있습니다.
\IBMCS\SDK\JAVA\ECL\DOC*.*	온라인, HTML 포맷, ECL 문서. 문서는 웹 검색기가 액세스할 수 있도O 포맷되었습니다. "ECLReference.html"이라고 하는 파일에서 시작하는 것이 바람직합니다.
\IBMCS\SDK\JAVA\ECL\SAMPLES*.*	나중에 설명되는 샘플 프N그%.
\IBMCS\jre*.*	서버에 설치된 ECL 파일과 호환될 수 있는 Java Runtime Environment.

NT 32 비트 Windows 클라이언



등급 경로 설정하기

Java n플리케이션 또는 Java V플릿을 실행할 때 환f 변수 **classpath**는 n플리케이션 또는 V플릿 실행! 필d한 Java 등^ 위치의 전체 f로명z 0F_ 합니다. 9를 들n ECL Java n플리케이션이 작성되m SNA API 클라이p트 서브디렉토리로 복사되면(9: C:\CSNTAPI),

- **classpath**! 다음z 0이 설정되n_ 합니다.

  C:\CSNTAPI;C:\CSNTAPI\ecl101.jar

- 명령행은 다음z 0F_ 합니다.

  set classpath=C:\CSNTAPI;C:\CSNTAPI\ecl101.jar

Java Runtime Environment (JRE)를 사k하는 fl!는 **classpath** 환f 변수! 사k되지 J지만, Java 등^으로의 f로는 JRE! 호출될 때 **cp** I션으로 지정될 수 있습니다.

ECL 코드 페이지 변환기

ECL 코드 페이지 변환b는 다중 pn를 지x합니다. 크b 제한으로 ecl101.jar 파일z ecl101.zip 파일!는 5문 코드 페이지만 있습니다. b타 코드 페이지 변환b 등^은 서버! 설치된 파일인 ecl101n.zip!서 r을 수 있습니다. 파일의 zip을 풀명 코드 페이지 변환b를 찾을 수 있습니다. 이런 파일은 Java n플리케이션을 실행중인 시스템으로 복사되E나 또는 새로n jar/zip 파일을 만들b 위해 ecl101.jar/zip의 파일z a합될 수 있습니다. 파일이 위치한 Classpath(com\ibm...)를 보존하십시@.

ECL n플리케이션 또는 V플릿의 크b를 줄이려면 n플리케이션 또는 V플릿!서 필d한 변환b 등^ 파일만 복사해_ 합니다. 코드 페이지 변환b 등^ 8현! 대해서는 ECL 문서! 설명되n 있습니다.

ECL 샘플

다음 섹션은 **IBMCS\SDK\JAVA\ECL\SAMPLES** 서브디렉토리! 설치된 Host Launchpad 샘플z b타 샘플을 설명합니다.

Host Launchpad 샘플

IBMCS\SDK\JAVA\ECL\SAMPLES\LAUNCHPAD! 설치된 Host Launchpad 샘플은 ECL을 사k하) 호스트로 ,a하m 로WB하며, 호스트 n플리케이션z 정보를 W세스하m 로W@프합니다. 이 샘플은 V플릿 또는 n플리케이션으로 실행될 수 있습니다. V플릿은 HostLP.htm이라m 하는 HTML 파일!서 실행됩니다. 이를 수행하려면 다음 행을 HostLP.htm! 추!해_ 합니다.

```
<APPLET archive=ecl101.jar  code=HostLaunchPad.class
    id=HostLP
    width=400
    height=10 >
</APPLET>
```

또한 Win32 시스템!서 등^ f로는 다음z 0이 설정되n_ 합니다.

```
set classpath=%classpath%;x:\pathto\ecl101.jar;
```

다음 소스 파일은 샘플을 8성하m 사k되b 전! 컴파일되n_ 합니다.

CFilelData.java
CInitData.java
CLogonData.java
FileLaunchApp.java
FileResultsApp.java
HostLPConstants.java
HostLaunchPad.java
LaunchBase.java
LogonLaunchApp.java
SAFilel.java
SAFilelResults
SALogon.java
SALogoutLaunch.java
SAMore.java
SAReady.java
SAVMLogin.java
SAMsg10.java
ScreenAction.java
ScreenState.java

b본 등^은 HostLaunchPad 입니다.

샘플 코드를 이해하려면 HostLaunchPad.java로 시작하) LaunchBase.java, LogonLaunchApp.java, SAMsg10.java, SALogon.java, SAVMLogin, SAMore, SAReady.java W리m ScreenAction.java로 진행하십시@.

이 샘플은 대부분의 V플릿이 호스트 W세스 등^ 라이브러리(ECL)를 사k하) 호스트 W세스G을 r b 위해 사k하는 주d 3념을 8현합니다. Launchpad 샘플은 다음 타스크를 수행합니다.

1. 호스트 세션을 설정합니다.)b!는 ECLSession 인스턴스의 8축z StartCommunication() 방법 호출 작w이 포함됩니다. 이 논리는 LogonLaunchApp.java! 있습니다.
2. 프리젠테이션 서비스 x# (PS) 이벤트! 대해 등록합니다. PS 이벤트는 화면이 ;신될 때마다 생성됩니다. 등록은 RegisterPSEvent() 방법으로 이루n집니다. 이 논리는 SALogon.java! 있습니다.

3. PS 이벤트! 반응합니다.)b!는 화면 인식 및 키스트로크 사k법 조정, 호스트 b능(9: Enter 키 또는 b능 키) W리m 커서 이동 등이 포함됩니다. 화면 처리 논리는 NotifyEvent(), SAReady.java, SAMore.java, SAMsg10.java, SAFilelResults.java W리m ScreenAction.java!서 찾을 수 있습니다.
4. #단한 VM 명령인 'fidel'을 수행하는 논리를 8현하m 대화 상자! az를 표시합니다. 파일 리스트 명령을 8현하m az를 FileLaunchApp.java, SAFilel.java, SAFileResults.java, ScreenAction.java, W리m FilelResultsApp.java! 표시하는 파일.

주: 이 샘플은 시스템마다 다른 일부 화면을 처리합니다. 코드를 `# 변f하) 사k자 시스템! 맞춰_할지도 모릅니다. 화면 인식 및 키스트로크 논리 방법은 서로 다른 화면의 처리를 "" 담당하는 SAVMLogin.java, SAReady.java, SAMore.java, SAMsg10.java W리m SAFilelResults.java 파일! 있습니다.

이 샘플을 디버W하려면 화면!서 n떤 작w이 실제로 일n나는지를 보F_합니다. 화면 내k을 V플릿으로 표시하는 PS 디버E는 첫 번째 패널의 체크 박스를 사k하) 다g한 호스트 화면으로 진행합니다.

기타 샘플

추! 샘플은 **IBMCS\SDK\JAVA\ECL\SAMPLES\MISC** 서브디렉토리! 설치되n 있습니다. 이들 샘플은 많은 ECL 등^ 및 방법의 b본 사k법을 보)줍니다. 샘플은 모두 ECLAppletInterface를 8현하므로 Run Applet b능을 사k하) Host On-Demand 내!서 실행될 수 있습니다. Host On-Demand x이 이런 샘플을 실행하려면 ECLSession @브젝트를 설정해_합니다. 이는 Host Launchpad 샘플!서 ECLSession @브젝트를 설정하는 Mz 0은 방법으로 이루n질 수 있습니다.

제21장 Java용 CPIC-C 사용하기

통신 서버 Windows 95 M Windows NT SNA API 클라이 p 트 ! 대해서만 사 k 할 수 있습니다.

이 장 ! 서는 다음을 비롯한 Java API ! 대한 통신 k SAA x 통 프로 W 래밍 인터페이스(CPI-C) ! 대해 설명합니다.

- Java k CPI-C의 # 단한 3 d
- Java k CPI-C ! 설치된 내 k
- 사 k 할 수 있는 샘플 z 작동 방법

Java용 CPI-C는 무엇인가?

Java k CPI-C는 3 발자 ! Java p n 로 된 통신 k x 통 프로 W 래밍 인터페이스(CPI-C)를 사 k 할 수 있 T 해 주는 프로 W 래밍 툴키트입니다. CPI-C는 SNA LU 6.2 k 3 방 API입니다. CPI-C ! 대해 더 K m 싶으면, Windows NT k IBM eNetwork 통신 버전 6.0 ! 있는 *Common Programming Interface Communications CPI-C Reference*(SC26–4399)를 참조하십시 @ .

툴키트의 1차적인 목표는 전통적인 C를 쉽 T Java로 E b 는 M 입니다. 이런 이유로 툴키트는 C ! 서 사 k 되는 M z 매 l 비슷합니다. Java k CPI-C는 x 시 CPI-C API 위의 층으로 제 x 되 m CPI-C ! 작동하 b 위해서는 이 x 시 코드 ! 설치되 n _ 합니다.

툴키트는 모든 등 ^ , 방법 W 리 m 툴키트의 변수 ! 대해 프로 W 래머 참조서를 제 x 합니다. 문서는 HTML 포맷으로 되 n 있 m k 이성을 위해 상호 참조를 제 x 합니다.

이 프로 W 래밍 툴키트는 또한 CPI-C 매 3 변수 뿐만 F 니라, C ! 서 CPI-C b 능으로 맵되는 방법을 정의하는 **CPIC** 등 ^ 을 보유하 b 위해 @ 브젝트 M 함께 Java 등 ^ 집합을 제 x 합니다. 툴키트 ! 포함된 샘플 n 플리케이션(JPing.class)을 실행하는 M \ ! 도 자신의 n 플리케이션을 작성할 수 있습니다.

Java k CPI-C를 사 k 하면 Java n 플리케이션 ! 서 SNA 네트 v 크를 사 k 하 m CPI-C를 네트 v 킹 API로 사 k 할 수 있습니다. 이런 Java n 플리케이션은 다음 z 0 은 상대방으로 , a 될 수 있습니다.

- 새로 n Java k CPI-C n 플리케이션
- 새롭 E 나 또는 b 존의 비 Java CPI-C n 플리케이션
- 새롭 E 나 또는 b 존의 APPC n 플리케이션

Java용 CPI-C 3치하기

다음 항목은 Java k CPI-C 툴키트로 설치됩니다.

- CPICJAVA.JAR ! 는 Java k CPI-C 프로W램을 작성할 때 사k되는 Java 등^이 있습니다. 이 JAR 파일은 사k자의 CLASSPATH 환f 변수 ! 포함되E나 또는 Java k CPI-C n플리케이션을 호출할 때 지정되n_ 합니다. 이 파일은 다른 API 클라이p트 파일z 함께 사k자의 v크스테이션 ! 설치됩니다. JAR 파일은 샘플 n플리케이션인 JPing.class도 포함합니다.
- CPICJAVA.DLL은 플랫폼마다 m유한 DLL로서 Java k CPI-CM 사k자의 v크스테이션 ! 설치된 x시 LU 6.2 지x#의 링크 ! 들n 있습니다. 이 파일은 다른 API 클라이p트 DLLz 함께 사k자의 v크스테이션 ! 설치됩니다.
- Jcpic001.htm은 Java k CPI-C 등^, 방법 W리m 변수를 보)주는 프로W래머 참조서의 루트입니다. 이는 Java k eNetwork 호스트 W세스 등^ 라이브러리(ECL) ! 설치되는 Mz 0은 시b ! **IBMCS\SDK\JAVA\CPIC\DOC** 서브디렉토리 ! 설치됩니다. 이 문서는 사k자 정의 n플리케이션 3발 ! 사k됩니다.
- CPICJAVA.HTM은 툴키트M 샘플 n플리케이션 ! 대한 #단한 소3입니다. 이 HTML로 포맷된 파일은 다른 API 클라이p트 파일z 함께 사k자의 v크스테이션 ! 설치됩니다.
- JPing.java는 JPing.class 샘플 n플리케이션 ! 대한 소스 파일입니다. 이 파일의 주석은 프로W램을 툴키트로 작성할 때 필d한 힌트M 정보를 제x합니다. JPing.java 파일은 Java k ECL이 설치될 때 서브디렉토리 ! 설치됩니다.

Java용 CPI-C 샘플

다음 섹션 ! 서는 Java k CPI-C의 클라이p트M 서버 샘플을 설명합니다.

클라이언트 샘플

툴키트 ! 있는 샘플은 APING 클라이p트 유틸리티M 0은 b능을 수행합니다. 이는 데이타를 다시 APING 유틸리티로 반향하는 서버 프로세스로 데이타를 전송합니다. 샘플 클라이p트는 컴파일되n CPICJAVA.JAR 파일 ! 배치됩니다. 소스 파일(JPing.java)은 Java k ECL이 설치될 때 **IBMCS\SDK\JAVA\CPIC\SAMPLES** 서브디렉토리 ! 설치됩니다.

API는 Java 패키지 ! 서 COM.ibm.eNetwork.cpic로 제x됩니다. 다음 샘플 ! 서 코드의 첫 행은 툴키트로 제x된 등^을 W세스할 때 필d합니다. **CPIC** 등^은 x시 CPI-C 등^ ! 있n서 메인프레임 인터페이스 ! 됩니다. **CPIC** 등^ ! 서는 대화 ID의 f이M 0은 많은 상수 ! x시 CPI-C 호출을 통해 전달되는 방법z 함께 CPI-C ! 정의됩니다.

등^ 하나당 **CPIC** @브젝트 하나만 선p하면 됩니다. Java는 **CPIC** @브젝트! 시작될 때 x시 방법(CPICJAVA.DLL)이 있는 동적 링크 라이브러리(DLL)를 로드합니다.

다음 샘플은 CPI-C 파이프라인을 설명합니다. 이는 JPing.java 소스 파일! 있는 정보를 복제하지 J습니다.

주: 다음 샘플!는 주석z 삽입된 코드! 있습니다.

```
/*-------------------------------------------------------------
 * 파이프라인 트#잭션, 클라이언트 쪽
 *-------------------------------------------------------------*/
import COM.ibm.eNetwork.cpic.*;
public class Pipe extends Object {
   public static void main(String args[]) {

   // CPIC 오브젝트 만들기
   CPIC cpic_obj = new CPIC();
```

" 매3변수 유형은 자체 등^이 있으며 이런 등^마다 등^ 변수로 정의된 |련 상수! 있습니다. 9를 들n CPICReturnCode 등^!는 성x 리턴 코드인 CM_OK! 정의됩니다.

다음z 0은 두 !지 이유 때문! 매3변수마다 등^이 있습니다. Java! 모든 매3변수를 *으로 전달하므로 정수M 0은 #단한 데이타 유형을 리턴할 수 있는 방법이 x습니다. @브젝트를 매3변수로 방법! 전달하면 방법은 W @브젝트!서 변수를 설정하) 데이타를 호출자!T 리턴합니다. 또 다른 이유는 @브젝트를 사k하면 상수를 이해하는 @브젝트!서 상수를 캡슐화할 수 있습니다. 이는 표준 정보 숨b b b법입니다.

```
   // 리턴 코드
   CPICReturnCode cpic_return_code =
      new CPICReturnCode(CPICReturnCode.CM_OK);

   // 전송 요청 수신 여부
   CPICControlInformationReceived rts_received =
      new CPICControlInformationReceived(
         CPICControlInformationReceived.CM_NO_CONTROL_INFO_RECEIVED);
```

CPI-C send 함수!서는 특정 유형을 .지 J는 할당된 x#인 C pn 버퍼를 바랍니다. C!서M는 달리 Java!서는 유형이 x는 메모리를 할당할 수 있는 b능이 x습니다. Java!서 x시\의 모든 M은 @브젝트입니다. 프로W램이 전송하는 내k! |hx이 W @브젝트 유형은 C 스타일 바이트 배-로 변환되n_ 합니다.

Java는 이런 변환을 쉽T 수행할 수 있도록 방법을 제x합니다. 9를 들n Java는 문자-을 바이트 배-로 변환할 수 있습니다. 바이트 배-이 Java!서 @브젝트일 때 x시 방법을 사k하) 배-!서 데이타를 발췌할 수 있습니다.

```
// 전송할 문자열
  String sendThis = "Test of the PipeLine Transaction";

  // 전송할 문자열 길이
  CPICLength send_length = new CPICLength(sendThis.length());

  // Java 바이트 배열N 전송할 문자열 변환
  byte[] stringBytes = new byte[ send_length.intValue()];
  sendThis.getBytes(0,send_length.intValue(),stringBytes,0);
```

버퍼 처리!서처럼 CPI-C x시 호출은 b호식 목적지 이름이 Java 문자-이 F닌 C 문자-이b를 바랍니다. 툴키트는 필d! 따라 이를 Java 문자-!서 C 문자-로 자동 변환합니다. 일반적으로 툴키트!서 m유한 Java 유형을 바랄 때 자동 변환이 !능해집니다.

대화 ID는 툴키트! 의해 #단한 바이트 블럭으로 8성된 C 배-로 자동 변환되는 Java 바이트 배-입니다.

```
  // 이 하드코드화된 sym_dest_name은
  // 8 문자여야 하고 공백으N 채워져야 합니다
  String sym_dest_name = "PIPE    ";

  // 대화 ID를 보관할 공간
  // (단지 바이트의 모임일 뿐입니다)
  byte[] conversation_ID = new byte[CPIC.CM_CID_SIZE];
```

프로W램은 C!서 사k된 M z 매l 비슷한 CPI-C를 호출합니다. W러나 방법 호출!서는 CPI-C @브젝트의 이름이 접두사로 붙지만 매3변수!는 참조로 전달(&) b호! 접두사로 붙지 **않습니다**.

```
//
  // CPI-C 초기화
  //
  cpic_obj.cminit(            /* Initialize_Conversation       */
            conversation_ID,  /*  O: 리턴된 대화 ID           */
            sym_dest_name,    /*  I: 기호식 목적지 이름       */
            cpic_return_code); /*  O: 이 호출의 리턴 코드      */
  //
  // ALLOCATE
  //
  cpic_obj.cmallc(            /* 할당 대화                    */
            conversation_ID,  /*  I: 대화 ID                  */
            cpic_return_code); /*  O: 이 호출의 리턴 코드      */
  //
  // SEND
  //
  cpic_obj.cmsend(            /* Send_Data                    */
            conversation_ID,  /*  I: 대화 ID                  */
            stringBytes,      /*  I: 이 버퍼 전송             */
            send_length,      /*  I: 전송할 길이              */
            rts_received,     /*  O: RTS 수신 여부            */
            cpic_return_code); /*  O: 이 호출의 리턴 코드      */
  //
  // DEALLOCATE
  //
  cpic_obj.cmdeal(            /* 할당 취소                    */
            conversation_ID,  /*  I: 대화 ID                  */
            cpic_return_code); /*  O: 이 호출의 리턴 코드      */
```

```
   } // 기본 방법 종a
} // 등급 종a
```

서버 샘플

서버는 스스로를 초b화하m 대화를 수k하며, 데이타를 수신하m 진단 정보를 인쇄합니다. 클라이p트!서는 CPI-C 매3변수를 보|하b 위한 등^을 인스턴스화합니다. 이 매3변수의 대부분은 인스턴스 데이타로 정수만 !지m 있습니다. @브젝트를 사k하) 참조로 호출을 모방할 수 있습니다. 사k자는 또한 수신된 데이타를 보|할 바이트 배-을 할당합니다.

주: 다음 샘플!는 주석z 삽입된 코드! 있습니다.

```
/*-------------------------------------------------------------
 * 파이프라인 트#잭션, 서버 쪽
 *-------------------------------------------------------------*/
import COM.ibm.eNetwork.cpic.*;
import Java.io.IOException;

public class PipeServer extends Object {
   public static void main(String args[]) {

    CPIC cpic_obj = new CPIC();

    // 수신된 데이타를 보관할 공간
    byte[] data_buffer;
    data_buffer =  new byte[101];

    CPICLength requested_length = new CPICLength(101);
    CPICDataReceivedType data_received =
       new CPICDataReceivedType(0);
    CPICLength received_length = new CPICLength(0);
    CPICStatusReceived status_received =
       new CPICStatusReceived(0);
    CPICControlInformationReceived rts_received =
       new CPICControlInformationReceived(0);
    CPICReturnCode cpic_return_code =
       new CPICReturnCode(0);

    // 대화 ID를 보관할 공간 -- 바이트 모음
    // 첫 행은 conversation_ID를 바이트 배열 오브젝트N 참조
    // 하도O 선언합니다. 두 번째 행은 이1 오브젝트를 작성하고
    // 참조를 바이트 배열 오브젝트N 지정합니다.
    byte[] conversation_ID;
    conversation_ID = new byte[cpic_obj.CM_CID_SIZE];
```

CPI-C 수신 호출(cmrcv)은 파이프 트랜잭션!서 문자-을 b다리m 있을 때 Java 바이트 배-을 리턴합니다. 프로W래머는 바이트 배-을 인수로 취하는 문자- 등^ 8축자를 사k하) 바이트 배-을 문자-로 변환할 수 있습니다.

```
    //
    // ACCEPT
    //
```

```
    cpic_obj.cmaccp(         /* Accept_Conversation           */
        conversation_ID,     /*  O: 리턴된 대화 ID            */
        cpic_return_code);   /*  O: 리턴 코드                 */
    //
    // RECEIVE
    //
    cpic_obj.cmrcv(          /* 수신                         */
        conversation_ID,     /*  I: 대화 ID                  */
        data_buffer,         /*  I: 수신 데이타 배치 장소    */
        requested_length,    /*  I: 수신할 최대 길이         */
        data_received,       /*  O: 데이타 완a 여부          */
        received_length,     /*  O: 수신 데이타 길이         */
        status_received,     /*  O: 상태 변경 여부           */
        rts_received,        /*  O: RTS 수신 여부            */
        cpic_return_code);   /*  O: 이 호출의 리턴 코드      */
  //
  // 몇 가지 리턴 코드 처리 수행
  //
  System.out.println("  Data from Receive:");
  System.out.println("    cpic_return_code        = " +
                     cpic_return_code.intValue());
  System.out.println("    cpic_data_received      = " +
                     data_received.intValue());
  System.out.println("    cpic_received_length    = " +
                     received_length.intValue());
  System.out.println("    cpic_rts_received       = " +
                     rts_received.intValue());
  System.out.println("    cpic_status_received    = " +
                     status_received.intValue());
  // 수신된 바이트 배열에서 Java 문자열을 작성하고
  // 인쇄하십시오.
  String receivedString = new String(data_buffer,0);
  System.out.println(
     "    Recevied string                 = "
     + receivedString );

  //
  // 서버 창이 나타나지 않도O 차단
  //
  try{
       System.out.println("Press any key to continue");
       System.in.read();
     }
  catch
     (IOException e){ e.printStackTrace(); }
  }
}
```

부록A. APPC 공통 리턴 코드

이 부록!서는)러 APPC 명령n! x통인 1차(및 적k!능할 fl 2차) 리턴 코드를 설명합니다.

명령n마다 m유한 리턴 코드는 " 명령n의 문서! 설명되n 있습니다.

AP_ALLOCATION_ERROR

퍼스널 통신 및 통신 서버는 대화 할당! 실패했습니다. 대화 상태는 RESET로 설정됩니다. 이 코드는 **ALLOCATE** 또는 **MC_ALLOCATE** 다음! 발행된 명령n를 통해 리턴됩니다. |련된 2차 리턴 코드는 다음z 0습니다.

AP_ALLOCATION_FAILURE_NO_RETRY

대화는 8성 @류 또는 세션 프로토콜 @류M 0은 58적인 조G 때문! 할당될 수 x습니다. @류를 판별하려면 시스템 |리자! @류로W 파일을 조사해_ 합니다. @류! 정정될 때n지는 할당을 재시도하지 마십시@.

AP_ALLOCATION_FAILURE_RETRY

대화는 링크 장VM 0은 일시적인 조G 때문! 할당될 수 x습니다. 장V x인은 시스템 @류 로W! b록됩니다. 조G이 정리되도록 시#종료 이후! 할당을 재시도하는 M이 더 좋습니다.

AP_SECURITY_NOT_VALID

할당 d청!서 지정된 사k자 ID 또는 O호는 상대방 LU!서 수k되지 JR습니다.

AP_TRANS_PGM_NOT_AVAIL_RETRY

x] LU는 d청된 상대방 트랜잭션 프로W램을 시작할 수 xzb 때문! 할당 d청을 E부했습니다. d청된 트랜잭션 프로W램(TP)은 시#종료M 0은 일시적인 조G 때문! 사k될 수 x습니다. @류의 x인은 x] 노드! b록됩니다. 조G은 n5dx의 3입x이 자체 정리되n_ 합니다. 조G 정리를 위해 트랜잭션 프로W램이 시#종료 이후! 대화를 재시도하는 M이 더 좋습니다.

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

x] LU는 d청된 상대방 트랜잭션 프로W램을 시작할 수 xzb 때문! 할당 d청을 E부했습니다. d청된 트랜잭션 프로W램은 58적인 또는 반58적인 조G 때문! 사k할 수 x습니다. @류의 x인은 x] 노드! b록됩니다. 조G은 n5dx의 3입x이 자체 정리될 수 x습니다. 트랜잭션 프로W램은 @류 조G이 정리될 때n지는 대화를 재시도해서는 H됩니다.

AP_TP_NAME_NOT_RECOGNIZED

상대방 LU는 할당 d청!서 지정된 트랜잭션 프로W램명을 인식하지 못합니다.

AP_PIP_NOT_ALLOWED

d청된 트랜잭션 프로W램은 프로W램 초b화 매3변수(PIP)를 수신하지 못합니다. 이는 로컬z 상대방 트랜잭션 프로W램#의 불일치를 나타냅니다.

AP_PIP_NOT_SPECIFIED_CORRECTLY

d청된 트랜잭션 프로W램은 프로W램 초b화 매3변수(PIP)를 수신하지 못하지만 제x된 PIP!서 @류를 (지합니다. 이는 로컬z 상대방 트랜잭션 프로W램#의 불일치를 나타냅니다.

AP_CONVERSATION_TYPE_MISMATCH

d청된 트랜잭션 프로W램은 할당 d청!서 지정된 대화 유형(b본 또는 직접)을 지x하지 못합니다. 이는 로컬z 상대방 트랜잭션 프로W램#의 불일치를 나타냅니다.

AP_SYNC_LEVEL_NOT_SUPPORTED

d청된 트랜잭션 프로W램은 할당 d청!서 지정된 **sync_level** (AP_NONE, AP_CONFIRM_SYNC_LEVEL 또는 AP_SYNCPT)z의 대화를 지x할 수 x습니다. 이는 로컬z 상대방 트랜잭션 프로W램#의 불일치를 나타냅니다.

AP_CANCELLED

대화의 취소로 인하) 명령n! 리턴되z습니다(트랜잭션 프로W램은 **CANCEL_CONVERSATION** 명령n를 발행).

AP_CONV_FAILURE_NO_RETRY

대화는 세션 프로토콜 @류M 0은 58적인 조G 때문! 중단되z습니다. 시스템 |리자는 @류의 x인을 판별하b 위해 시스템 @류로W를 조사해_ 합니다. @류! 정정될 때n지는 대화를 재시도하지 마십시@.

AP_CONV_FAILURE_RETRY

대화는 임시 @류 때문! 중단되z습니다. 트랜잭션 프로W램을 재시작하) 문제! 다시 발생하는지 보십시@. 다시 발생할 fl! 시스템 |리자는 @류 로W를 조사하) @류의 x인을 판별해_ 합니다.

AP_CONVERSATION_TYPE_MIXED

트랜잭션 프로W램은 0은 대화!서)러 대화 유형! 대해 혼합 대화를 시도했습니다. 9를 들n 트랜잭션 프로W램은 다음! **CONFIRM** 명령n! @는 **MC_ALLOCATE** 명령n를 발행했습니다.

AP_DEALLOC_ABEND

대화는 다음 x인 중 하나 때문! 할당 취소되z습니다.

- 상대방 트랜잭션 프로W램은 **dealloc_type**이 AP_ABEND로 설정된 **MC_DEALLOCATE** 명령n를 발행했습니다.
- 상대방 트랜잭션 프로W램이 비정상적으로 종료하는 바람! 상대방 LU는 **MC_DEALLOCATE** d청을 전송하T 되z습니다.

AP_DEALLOC_ABEND_PROG

대화! 다음 x인 중 하나 때문! 할당 취소되z습니다.

- 상대방 트랜잭션 프로W램은 **dealloc_type**이 AP_ABEND_PROG로 설정된 **DEALLOCATE** 명령n를 발행했습니다.
- 상대방 트랜잭션 프로W램이 비정상적으로 종료하는 바람! 상대방 LU는 **DEALLOCATE** d청을 전송하T 되z습니다.

AP_DEALLOC_ABEND_SVC

상대방 트랜잭션 프로W램이 **dealloc_type**이 AP_ABEND_SVC로 설정된 **DEALLOCATE** 명령n를 발행했b 때문! 대화의 할당이 취소되z습니다.

AP_DEALLOC_ABEND_TIMER

상대방 트랜잭션 프로W램이 **dealloc_type**이 AP_ABEND_TIMER로 설정된 **DEALLOCATE** 명령n를 발행했b 때문! 대화의 할당이 취소되z습니다.

AP_DEALLOC_NORMAL

이 리턴 코드는 @류를 나타내지 J습니다. 상대방 트랜잭션 프로W램은 **dealloc_type**이 다음 * 중 하나로 설정된 **DEALLOCATE** 또는 **MC_DEALLOCATE** 명령n를 발행했습니다.

- AP_FLUSH
- 지정된 대화의 동b화 레벨의 AP_SYNC_LEVEL

AP_DUPLEX_TYPE_MIXED

트랜잭션 프로W램은 다른 대화 **duplex_type**의 대화 명령n를 발행하려m 시도했습니다. 9를 들n 트랜잭션 프로W램은 0전 g방향(전송) 대화!서 반 g방향(전송) **MC_FLUSH** 명령n를 발행했습니다(**AP_FULL_DUPLEX_CONVERSATION** 을 **opext**! 설정하지 J음).

AP_ERROR_INDICATION

이 리턴 코드는 0전 g방향(전송) 대화!서만 사k됩니다. 전송 대b행렬 작w은 상대방 트랜잭션 프로W램이 대화를 중단했b 때문! 실패했습니다. 대화! 전송 전k 상태! 있으면, 대화는 지] 종료됩니다. 대화! 송수신 또는 수신 전k 상태! 있으면,대화는 대b행렬 명령n 수신을 위해 해당 리턴 코드! 리턴될 때 종료됩니다. |련된 2차 리턴 코드는 다음z 0습니다.

AP_ALLOCATION_ERROR_PENDING

x] LU는 할당 d청을 E부했습니다.

AP_DEALLOC_ABEND_PROG_PENDING

대화는 다음 x인 중 하나 때문! 할당 취소되z습니다.

- 상대방 트랜잭션 프로W램은 **dealloc_type**이 AP_ABEND_PROG로 설정된 **DEALLOCATE** 명령n를 발행했습니다.

- 상대방 트랜잭션 프로W램이 비정상적으로 종료하는 바람! 상대방 LU는 **DEALLOCATE** d청을 전송하T 되z습니다.

AP_DEALLOC_ABEND_SVC_PENDING
상대방 트랜잭션 프로W램이 **dealloc_type**이 AP_ABEND_SVC로 설정된 **DEALLOCATE** 명령n를 발행했b 때문! 대화의 할당이 취소되z습니다.

AP_DEALLOC_ABEND_TIMER_PENDING
상대방 트랜잭션 프로W램이 **dealloc_type**이 AP_ABEND_TIMER로 설정된 **DEALLOCATE** 명령n를 발행했b 때문! 대화의 할당이 취소되z습니다.

AP_UNKNOWN_ERROR_TYPE_PENDING
대화! 상대방 트랜잭션 프로W램으로 할당 취소되z지만 로컬 LU는 W x인을 인식하지 못합니다.

AP_OPERATION_INCOMPLETE
트랜잭션 프로W램은 처리하b 시작했지만 끝내지 못한 비블럭 명령n를 발행했습니다. 명령n 처리! 끝나면 최종 리턴 코드! 설정되m 스텁(stub)은 트랜잭션 프로W램!T 이를 K립니다.

AP_PROG_ERROR_NO_TRUNC
상대방 트랜잭션 프로W램은 대화! SEND 상태! 있는 동H 다음 명령n 중 하나를 발행했습니다.

- **err_type**! AP_PROG로 설정된 **SEND_ERROR**
- **MC_SEND_ERROR**

데이타는 절단되지 JR습니다.

AP_PROG_ERROR_PURGING
상대방 트랜잭션 프로W램은 RECEIVE, PENDING_POST, CONFIRM, CONFIRM_SEND 또는 CONFIRM_DEALLOCATE 상태! 있는 동H 다음 명령n 중 하나를 발행했습니다.

- **err_type**! AP_PROG로 설정된 **SEND_ERROR.**
- **MC_SEND_ERROR**

전송되z지만 F직 수신되지 못한 데이타는 제E됩니다.

AP_PROG_ERROR_TRUNC
상대방 트랜잭션 프로W램은 SEND 상태!서 불O전한 논리적 레코드를 전송한 후 **err_type**을 AP_PROG로 설정한 **SEND_ERROR** 명령n를 발행했습니다. 로컬 트랜잭션 프로W램은 **RECEIVE** 명령n를 통해 논리적 레코드의 첫 부분을 수신할 수도 있습니다.

AP_SVC_ERROR_NO_TRUNC
상대방 트랜잭션 프로W램(또는 상대방 LU)은 SEND 상태!서 **err_type**이 AP_SVC로 설정된 **SEND_ERROR** 명령n를 발행했습니다. 데이타는 절단되지 JR습니다.

AP_SVC_ERROR_PURGING

상대방 트랜잭션 프로W램(또는 상대방 LU)은 RECEIVE, PENDING_POST, CONFIRM, CONFIRM_SEND 또는 CONFIRM_DEALLOCATE 상태! 있으면서 **err_type**이 AP_SVC로 설정된 **SEND_ERROR** 명령n를 발행했습니다. 상대방 트랜잭션 프로W램으로 전송된 데이타는 제E될 수도 있습니다.

AP_SVC_ERROR_TRUNC

상대방 트랜잭션 프로W램은 SEND 상태!서 불O전한 논리적 레코드를 전송한 후 **SEND_ERROR** 명령n를 발행했습니다. 로컬 트랜잭션 프로W램은 논리적 레코드의 첫 부분을 수신할 수도 있습니다.

AP_TP_BUSY

로컬 트랜잭션 프로W램은 퍼스널 통신 및 통신 서버! 0은 대화!대해 다른 명령n를 처리하m 있는 동H 퍼스널 통신 및 통신 서버!서 블럭화 명령n를 발행했습니다.

AP_UNEXPECTED_SYSTEM_ERROR

퍼스널 통신 및 통신 서버는 9b치 못한 시스템 @류를 만나서 명령n를 끝낼 수 x습니다. 일반적으로 이런 @류는 시스템 자x(9:메모리)의 부족으로 발생하m 보통 일시적입니다. 자세한 내k! 대해서는 시스템 로W를 확인해 보십시@.

AP_SEC_REQUESTED_NOT_SUPPORTED

로컬 LU는 상대방 LUM의 세션!서 O호 대체를 지x하지 J으므로 대화를 할당할 수 x습니다. ALLOCATE 또는 SEND_CONVERSATION!서 d청된 보H 유형은 O호 대체 지x이 필d한 AP_PGM_STRONG입니다.

부록B. LUA 명령어 리턴 코드

이 부록!서는)러 SLI 명령n! x통인 1차(및 적k!능할 f l 2차) 리턴 코드를 설명합니다.

명령n마다 m유한 리턴 코드는 " 명령n 문서! 설명되n 있습니다.

1차 리턴 코드

다음 섹션!는 LUA 1차 리턴 코드! 들n 있습니다.

LUA_OK
LUA 명령n! 0료됐습니다.

LUA_PARAMETER_CHECK
LUA b능이 틀린 매3변수를 (지했습니다.

LUA_STATE_CHECK
세션은 발행된 명령n! 대해 틀린 상태! 있z습니다.

LUA_SESSION_FAILURE
세션은 종료되z습니다. W x인은 2차 리턴 코드! 있습니다.

LUA_UNSUCCESSFUL
이 명령n는 0료되지 J R습니다.

LUA_NEGATIVE_RESPONSE
다음 조G 중 하나! 발생했습니다.

- LUA n플리케이션 프로W램!서 부정적으로 응답된 체인의 끝! 도달했습니다. 2차 리턴 코드! 설정되지 J습니다.
- LUA는 1차 LU의 메세지!서 @류를 (지하m 부정적인 응답을 전송했습니다. 이 @류는 1차 LU!서 체인 끝이 수신될 때 리턴됩니다. 2차 리턴 코드!는 부정적인 응답z 함께 전송된 (지 데이타! 있습니다.

LUA_CANCELED
명령n는 2차 리턴 코드! 지정된 x인 때문! 취소되z습니다.

LUA_IN_PROGRESS
이 동b 코드는 비동b 명령이 0료되지 J은 상태로 수신될 때 리턴됩니다.

LUA_STATUS
SLI!는 2차 리턴 코드의 n플리케이션! 대한 상태 정보! 있습니다.

LUA_COMM_SUBSYSTEM_ABENDED
통신 서버! 비정상적으로 종료되z습니다.

LUA_COMM_SUBSYSTEM_NOT_LOADED

통신 서버! 로드되지 JR습니다.

LUA_INVALID_VERB_SEGMENT

LUA는 전체 명령n 제n 블럭이 데이타 세W멘트! xn서 W 명령 n를 처리하지 JR습니다. 명령n 제n 블럭의 끝 주소는 세W멘트 끝을 지났습니다.

LUA_UNEXPECTED_DOS_ERROR

통신 서버! 시스템 호출을 발행한 후 9b치 못한 시스템 @류! 발생하), 명령n는 1차 리턴 코드인 UNEXPECTED_DOS_ERRORM 함께 전송되z습니다. 2차 리턴 코드!는 9b치 못한 시스템 @류! 있습니다.

LUA_STACK_TOO_SMALL

LUA n플리케이션 스택은 LUA! d청을 처리하b! 너무 작습니다.

LUA_INVALID_VERB

LUA는 수신된 명령n 제n 블럭! 있는 명령n 코드나 명령n n5 코드(또는 둘 다)를 인식하지 못합니다.

2차 리턴 코드

다음 섹션!는 LUA 2차 리턴 코드! 들n 있습니다.

LUA_SEC_OK

1차 리턴 코드! 대해 이 2차 리턴 코드M |련된 정보를 추!로 사k할 수 있습니다.

LUA_INVALID_LUNAME

명령n! 유효하지 J은 **lua_name**을 지정했습니다.

LUA_BAD_SESSION_ID

명령n 제n 블럭은 **lua_sid** 매3변수! 대해 틀린 *을 지정했습니다.

LUA_DATA_TRUNCATED

버퍼 f이(**lua_max_length**! 지정된 대로)는 수신된 데이타를 수k할 만큼 f지 JF서 데이타! 절단되z습니다.

LUA_BAD_DATA_PTR

명령n는 데이타! 제x되E나 리턴되n_ 하지만 **lua_data_ptr** 매3변수! 유효하지 J은 포인터! 있E나 또는 읽b/2b 세W멘트를 !리키지 J습니다.

LUA_DATA_SEG_LENGTH_ERROR

다음 조G 중 하나! 발생했습니다.

- **RUI_READ** 또는 **SLI_RECEIVE** 명령n ! 제x된 데이타 세W멘트는 **lua_max_length** 매3변수 ! 제x된 f이보다 짧습니다.
- **RUI_WRITE** 또는 **SLI_SEND** 명령n ! 제x된 데이타는 **lua_data_length** 매3변수 ! 제x된 f이보다 짧습니다.
- **RUI_READ**, **RUI_WRITE**, **SLI_RECEIVE** 또는 **SLI_SEND** 명령n ! 제x된 데이타 세W멘트는 읽b/2b 데이타 세W멘트 ! F닙니다.

LUA_RESERVED_FIELD_NOT_ZERO
방] 발행된 명령 ! 는 0이 F닌 9`된 매3변수 ! 있습니다.

LUA_INVALID_POST_HANDLE
LUA 명령n 블럭 ! 서 유효한 신호b ! 지정되지 JR습니다. LUA 명령n ! 동b적으로 끝나지 J으면 명령n O료 신호시 신호b ! 필d합니다.

LUA_PURGED
RUI_READ 또는 **SLI_RECEIVE** 명령n는 **RUI_PURGE** 또는 **SLI_PURGE** ! 발행되n 취소되z습니다.

LUA_BID_VERB_SEG_ERROR
버퍼는 **SLI_BID** 명령n 제n 블럭z 함께 **lua_flag1.bid_enable**이 1로 설정된 **SLI_RECEIVE** ! 발행되b 전 ! 릴리스되z습니다.

LUA_NO_PREVIOUS_BID_ENABLED
RUI_BID 또는 **SLI_BID** 명령n는 **RUI_READ** 또는 **lua_flag1.bid_enable**이 있는 **SLI_RECEIVE** ! 발행되b 전 ! 발행되지 JR습니다.

LUA_NO_DATA
RUI_READ 또는 **SLI_RECEIVE** 명령n는 NO_WAIT 매3변수M 함께 발행되z으며 읽을 수 있는 데이타 ! x습니다.

LUA_BID_ALREADY_ENABLED
RUI_BID 또는 **SLI_BID** 명령n는 **RUI_READ** 또는 **lua_flag1.bid_enable**이 있는 **SLI_RECEIVE** 명령n ! 발행될 때 활동중이z습니다.

LUA_VERB_RECORD_SPANS_SEGMENTS
LUA 명령n 제n 블럭 ! 는 세W멘트의 @프셋 ! 추 ! 될 때 세W멘트 끝을 지나는 length 매3변수 ! 있습니다.

LUA_INVALID_FLOW
LUA 명령n는 **lua_flag1** 흐름 플래W ! @류로 설정된 상태 ! 서 발행되z습니다. **lua_flag1** 흐름 플래W의 C바른 9수 ! 다음z 0이 설정되n 있는지 확인하십시@.
- **RUI_READ** 또는 **SLI_RECEIVE** ! 대해 최소한 하나
- **RUI_WRITE** ! 대해 @로지 하나
- **SLI_SEND** ! 대해 SNA 응답을 전송할 때 @로지 하나의 **lua_flag1** 흐름 플래W만이 설정되n _ 합니다.

LUA_NOT_ACTIVE
n플리케이션 프로W램은 LUA! 통신 서버!서 활동중이 F닐 때 LUA 명령n를 발행했습니다.

LUA_VERB_LENGTH_INVALID
명령n는 틀린 **lua_verb_length** 매3변수M 함께 발행되z습니다. 지정된 f이는 LUA!서 b대한 f이M 0지 J습니다.

LUA_REQUIRED_FIELD_MISSING
발행된 **RUI_WRITE** 명령n는 데이타 포인터를 포함하지 JE나(데이타 h수! 0이 F닐 때) **lua_flag1** 흐름 플래W를 포함하지 J습니다.

LUA_READY
이제 SLI 세션!서는 추! 명령을 처리할 수 있습니다. 이 상태는 NOT_READY 이전 상태! 수신된 후 또는 **SLI_CLOSE** 명령n! CANCELED 1차 리턴 코드 및 RECEIVE_UNBIND_HOLD 또는 RECEIVED_UNBIND_NORMAL 2차 리턴 코드M 함께 O료된 후 발행되z습니다.

LUA_NOT_READY
SLI 세션은 다음 x인 중 하나로 일시 중단되z습니다.

- CLEAR 명령이 수신되z습니다. SLI 세션은 SDT 명령이 수신될 때 다시 시작합니다.
- UNBIND 명령이 수신되z습니다. 세션은 BIND, 선택적 STSNz SDT 명령이 수신될 때n지 일시 정지되z습니다. x래의 **SLI_OPEN** 명령n로 제x된 모든 사k자 확장자 루틴은 다시 호출됩니다. W러므로 이런 루틴은 재진입 루틴이n_ 합니다. SLI ! SDT 명령을 처리하면 SLI 세션은 다시 시작합니다. 두 !지 유형의 UNBIND 명령은 다음z 0습니다.
  - 새로n BIND! 생d다는 M을 의미하는 UNBIND type X'02'.
  - 이 세션을 시작한 **SLI_OPEN**! 있는 LUA_SESSION_TYPE_DEDICATED의 **lua_session_type**!서 지정된 n플리케이션을 의미하는 UNBIND type X'01'.

LUA_INIT_COMPLETE
LUA 인터페이스는 **SLI_OPEN**이 처리중일 때 세션을 초b화시킵니다. 이 상태는 **LUA_INIT_TYPE_PRIM_SSCP** 매3변수! 있는 **SLI_OPEN**을 발행하는 LUA n필리케이션! 대한 **SLI_RECEIVE** 또는 **SLI_BID** 명령n로 리턴됩니다.

LUA_SESSION_END_REQUESTED
SLI는 호스트!서 세션을 종료할 준비! 끝났다는 M을 나타내는 SHUTD 명령을 수신했습니다.

LUA_NO_SLI_SESSION
명령은 세션이 -리지 JRE나 세션이 **SLI_CLOSE** 명령n 또는 세

선 장V 때문! 종료될 때 발행됩니다. **SLI_OPEN** 명령n의 처리 중 ! 발행된 **SLI_RECEIVE** 또는 **SLI_SEND** 명령n는 다음 fl! 이 코드를 리턴합니다.

- **SLI_OPEN lua_init_type** 매3변수는 LUA_INIT_TYPE_PRIM_SSCP로 설정되지 J습니다. SLI_BID 명령n는 또한 이런 상황!서 이 코드를 리턴합니다.
- **SLI_RECEIVE** 또는 **SLI_SEND lua_flag1** 매3변수는 **lua_flag1.sscp_norm**을 지정하지 J습니다.

SLI 8성d소는 UNBIND type X'02' 또는 UNBIND type X'01' (LUA_SESSION_TYPE_DEDICATED) 이 수신되m SDT 명령이 처리될 때n지 **SLI_OPEN**을 처리합니다. UNBIND type X'02' 는 새로n BIND! 생d다는 M을 나타냅니다.

LUA_SESSION_ALREADY_OPEN
SLI_OPEN 명령n는 이미 세션이 -려 있는 LU명! 대해 발행되z습니다.

LUA_INVALID_OPEN_INIT_TYPE
SLI_OPEN 명령n는 lua_init_type 매3변수! 틀린 *을 !지m 있습니다.

LUA_INVALID_OPEN_DATA
SLI_OPEN 명령n는 INITSELF (LUA_INIT_TYPE_SEC_IS)! 있는 2차 초b화시 설정된 lua_init_type 매3변수M 함께 발행되m 데이타 버퍼!는 유효한 INITSELF 명령이 x습니다.

LUA_UNEXPECTED_SNA_SEQUENCE
SLI_OPEN 처리중! 9b치 못한 명령 또는 데이타를 호스트!서 수신했습니다.

LUA_NEG_RSP_FROM_BIND_ROUTINE
사k자! 제x한 **SLI_BIND** 루틴은 BIND! 부정적인 응답을 생성했습니다. **SLI_OPEN** 명령n는 끝나지 못했습니다.

LUA_NEG_RSP_FROM_CRV_ROUTINE
사k자! 제x한 **SLI_BIND** 루틴은 BIND! 부정적인 응답을 생성했습니다. **SLI_OPEN** 명령n는 끝나지 못했습니다.

LUA_NEG_RSP_FROM_STSN_ROUTINE
사k자! 제x한 SLI STSN 루틴은 STSN! 부정적으로 응답했습니다. **SLI_OPEN**는 종료되지 못했습니다.

LUA_CRV_ROUTINE_REQUIRED
사k자는 SLI CRV 루틴을 제x하지 JR지만 호스트로부터 CRV를 수신했습니다. SLI는 CRV! 대해 부정적인 응답을 발행하m **SLI_OPEN** 명령n는 이번! 종료하지 못합니다.

LUA_NEG_RSP_FROM_SDT_ROUTINE
사k자! 제x한 SLI SDT 루틴은 SDT! 부정적으로 응답했습니다. 이 조G으로 **SLI_OPEN** 명령n! 종료합니다.

LUA_INVALID_OPEN_ROUTINE_TYPE

SLI_OPEN 확장자 루틴 리스트!서 **lua_open_routine_type** 매3변수는 유효하지 J습니다.

LUA_MAX_NUMBER_OF_SENDS

n플리케이션 프로W램은 하나! 0료되b 전! **SLI_SEND** 명령n를 두 3 이상 발행했습니다.

LUA_SEND_ON_FLOW_PENDING

n플리케이션은 F직 해a되지 J은 **SLI_SEND** 명령n! 있는 SNA 흐름(SSCP 신속 처리, SSCP 정상, LU 신속 처리, LU 정상)! 대해 **SLI_SEND** 명령n를 발행했습니다.

LUA_INVALID_MESSAGE_TYPE

SLI는 **lua_message_type** 매3변수를 인식하지 못합니다.

LUA_RECEIVE_ON_FLOW_PENDING

SLI n플리케이션은 F직 해a되지 J은 **SLI_RECEIVE** 명령n! 있는 SNA 흐름! 대해 **SLI_RECEIVE** 명령n를 발행했습니다.

LUA_DATA_LENGTH_ERROR

n플리케이션 프로W램!서 제x하지 J는 사k자 데이타! 필d한 **SLI_OPEN** 명령이 발행되z습니다. 데이타는 2차적으로 시작된 **SLI_OPEN** 명령n!서 필d하m 4 바이트 상태는 n플리케이션이 LUSTAT 명령n! 대해 **SLI_SEND** 명령n를 발행할 때 필d합니다.

LUA_CLOSE_PENDING

다음 중 하나! 발생했습니다.

- CLOSE_NORMAL은 CLOSE_NORMAL 또는 CLOSE_ABEND! 대b중일 때 발행되z습니다.
- CLOSE_ABEND는 또 다른 CLOSE_ABEND! 대b중일 때 발행되z습니다. 또 다른 CLOSE_ABEND를 발행하는 유일하T 유효한 x인은 CLOSE_NORMAL이 대b중일 때입니다.

LUA_NEGATIVE_RSP_CHASE

SLI_CLOSE 처리중! SLI는 호스트로부터 CHASE 명령! 대해 부정적인 응답을 수신했습니다. 세션은 **SLI_CLOSE**!서 d청된 대로 중단되z습니다.

LUA_NEGATIVE_RSP_SHUTC

SLI_CLOSE 처리중! SLI는 호스트로부터 SHUTC 명령! 대해 부정적인 응답을 수신했습니다. 세션은 **SLI_CLOSE**!서 d청된 대로 중단되z습니다.

LUA_NEGATIVE_RSP_SHUTD

SLI_CLOSE 처리중! SLI는 호스트로부터 SHUTD 명령! 대해 부정적인 응답을 수신했습니다. 세션은 **SLI_CLOSE**!서 d청된 대로 중단되z습니다.

LUA_NO_RECEIVE_TO_PURGE

SLI_PURGE 명령n는 미해a **SLI_RECEIVE** 명령n！ x을 때 발행됩니다. 두 ！지 ！능한 x인은 다음z 0습니다.

- **lua_data_ptr** 매3변수！ 있는 주소는 제E될 미해a **SLI_RECEIVE** 명령n를 ！리키지 J습니다.
- **SLI_RECEIVE** 명령n는 **SLI_PURGE** 명령n！ 처리되는 중！ O료될 수도 있습니다. 이는 @류 조G이 F닙니다. n플리케이션 프로W램이 이 상황을 처리할 수 있도록 작성하십시@.

LUA_CANCEL_COMMAND_RECEIVED

SLI_RECEIVE 명령n를 처리하는 중！ 호스트는 수신되는 데이타의 체인을 취소하b 위해 CANCEL 명령을 전송했습니다.

LUA_RUI_WRITE_FAILURE

RUI_WRITE 명령n는 9b치 못한 @류M 함께 SLI로 전송되z습니다.

LUA_INVALID_SESSION_TYPE

SLI_OPEN 명령n！는 **lua_session_type**！서 유효하지 J은 *이 있습니다.

LUA_SLI_BID_PENDING

SLI 명령n는 이전！ 발행된 **SLI_BID**을 사k하는 동H 발행되z습니다. 한번！ **SLI_BID** 하나만 사k할 수 있습니다.

LUA_PURGE _PENDING

SLI_PURGE 명령n는 이전！ 발행된 **SLI_PURGE**를 사k하는 동H 발행되z습니다. 한번！ **SLI_PURGE** 하나만 사k할 수 있습니다.

LUA_PROCEDURE_ERROR

호스트 프로시듀n @류！ 발생했다는 M을 K리는 NSPE 또는 NOTIFY 메세지！ 수신되z습니다. **SLI_OPEN**은 이 리턴 코드M 함께 전송됩니다(**SLI_OPEN** 명령n 재시도 I션이 사k될 때n지). **lua_wait**를 0이 F닌 *으로 설정한 상태！서 INITSELF 또는 LOGON 메세지는 호스트 프로시듀n를 사k할 수 있E나 또는 n플리케이션이 **SLI_CLOSE**를 발행할 때n지 재시도됩니다.

LUA_INVALID_SLI_ENCR_OPTION

lua_encr_decr_option 매3변수는 **SLI_OPEN** 명령n！서 128로 설정되z습니다. SLI는 O호화 또는 O호 해독 처리 I션！ 대해 128을 지x하지 J습니다.

LUA_RECEIVED_UNBIND

SLI는 SLI 세션이 활동하는 동H 1차 LU！서 UNBIND 명령을 수신했습니다. SLI 세션이 중단됩니다.

LUA_RECEIVED_UNBIND_HOLD

1차적으로 또는 2차적으로 시작된 **SLI_CLOSE**의 정상적인 처리중！ SLI는 UNBIND type X'02'를 수신했습니다. Type X'02'는 새로n BIND ！ 생성된다는 M을 의미합니다. 세션은 BIND, 선택적 CRVM STSN

W리m SDT 명령이 수신될 때n지 일시정지됩니다. x래의 **SLI_OPEN** 명령n로 제x된 모든 사k자 확장 루틴은 다시 호출되며, 이런 루틴은 재진입 루틴이n_ 합니다. SLI! SDT 명령을 처리한 후! SLI 세션이 다시 시작됩니다.

LUA_RECEIVED_UNBIND_NORMAL

LUA_SESSION_TYPE_DEDICATED의 **lua_session_type**를 지정한 **SLI_OPEN** 명령n로 시작된 1차적 또는 2차적 시동 **SLI_CLOSE**! 정상적으로 처리되는 중! SLI는 UNBIND type X'01'을 수신했습니다. 세션은 BIND, 선택적 STSNz SDT 명령이 수신될 때n지 일시정지됩니다. x래의 **SLI_OPEN** 명령n로 제x된 모든 사k자 확장 루틴은 다시 호출됩니다. W러므로 이런 루틴은 재진입 루틴이n_ 합니다. SLI! SDT 명령을 처리한 후! SLI 세션이 다시 시작됩니다.

LUA_SLI_LOGIC_ERROR

SLI는 내부 논리 @류를 (지했습니다.

LUA_TERMINATED

SLI_CLOSE 또는 **RUI_TERM** 명령n! 발행되z을 때! 대b중이던 명령n! 취소되z습니다.

LUA_NO_RUI_SESSION

초b화되지 J은 세션(**RUI_INIT**를 사k하))! 대해 RUI 명령n! 발행되E나 세션! 대해 **RUI_INIT** 명령n! 진행중일 때 **RUI_TERM** \의 명령n! 발행되z습니다.

이 리턴 코드는 활동중인 미해a RUI 명령n! x을 때 세션 중단이 발생하면 일n날 수 있습니다. 다음으로 발행된 명령n!서 이 리턴 코드를 r습니다. n플리케이션 프로W램은 SESSION_FAILURE를 처리하는 M처럼 이 리턴 코드를 처리합니다.

LUA_DUPLICATE_RUI_INIT

n플리케이션 프로W램은 이미 초b화된 세션! 대해 **RUI_INIT** 명령n를 발행했E나 또는 진행중인 **RUI_INIT** 명령n! 있습니다.

LUA_INVALID_PROCESS

이미 다른 프로세스! 속한 세션! 대해 RUI 명령n! 발행되z습니다.

LUA_API_MODE_CHANGE

비 SLI d청은 SLI! 의해 설정된 세션의 RUI!서 발행되z습니다.

LUA_COMMAND_COUNT_ERROR

발행된 **RUI_READ** 또는 **RUI_WRITE** 명령n의 최대 수! 초z되zE나 또는 **RUI_BID**나 **RUI_TERM** 명령n! 이전! 발행된 **RUI_BID** 또는 **RUI_TERM** 명령n! 진행될 때 발행되z습니다.

LUA_NO_READ_TO_PURGE

RUI_PURGE 미해a **RUI_READ** 명령n! x을 때 발행되z습니다. 다음z 0은 두 !지 !능한 x인이 있습니다.

- **lua_data_ptr** 매3변수! 있는 주소는 제E될 미해a **RUI_READ** 명령n를 !리키지 J습니다.
- **RUI_READ** 명령n는 **RUI_PURGE** 명령n! 처리중일 때 O료되z습니다. 이는 @류 조G이 F닙니다. n플리케이션 프로W램이 이 상황을 처리할 수 있도록 작성하십시@.

LUA_MULTPLE_WRITE_FLOWS

RUI_WRITE 명령n! 발행된 FLAG1!서 흐름 플래W! 하나 이상 켜졌습니다.

LUA_DUPLICATE_READ_FLOW

n플리케이션 프로W램은 이미 **RUI_READ**! 대b중인 흐름! 대해 **RUI_READ**를 발행했습니다.

LUA_DUPLICATE_WRITE_FLOW

발행된 **RUI_WRITE** 명령n!는 O료되지 못한 이전의 **RUI_WRITE** 명령n! 대한 세션 흐름을 보)주는 FLAG1 흐름 플래W! 있습니다.

LUA_LINK_NOT_STARTED

LUA는 세션 초b화중! 데이타 링크를 시작하지 못했습니다.

LUA_INVALID_ADAPTER

DLC n댑터 8성은 틀렸E나 또는 8성 파일이 손상되z습니다.

LUA_ENCR_DECR_LOAD_ERROR

사k자! 제x한 O호화 또는 O호 해독 동적 링크 라이브러리를 로드하려m 할 때 9b치 못한 @류를 수신했습니다.

LUA_ENCR_DECR_PROC_ERROR

사k자! 제x한 O호화 또는 O호 해독 동적 링크 라이브러리!서 프로시듀n 주소를 r으려m 할 때 9b치 못한 @류를 수신했습니다.

LUA_LINK_NOT_STARTED_RETRY

RUI_INIT 또는 **SLI_OPEN** 명령n는 링크! 활성화될 수 xn서 실패했습니다. 이 리턴 코드는 상대방 위치 또는 두 시스템#의 ,a ! 이상이 있다는 M을 의미합니다.

LUA_NEG_NOTIFY_RSP

SLU! 이제 세션의 한 부분이 될 수 있다는 M을 나타내b 위해 통보 d청을 SSCP로 전송하는 **RUI_INIT**! 발행되z습니다. SSCP는 이 통보 d청! 부정적으로 응답했습니다. 의도된 반 세션 8성d소는 지x된 d청을 이해했지만 이를 처리하지는 JR습니다.

LUA_RUI_LOGIC_ERROR

RUI 내부 논리 @류! 발생했습니다.

LUA_LU_INOPERATIVE

SLI! 세션을 중단시키려m 할 때 심"한 @류! 발생했습니다. 호스트!서 ACTLU! 수신될 때n지는 n떤 LUA d청!서도 이 LU를 사k할 수 x습니다.

LUA_RESOURCE_NOT_AVAILABLE

LU, PU, 링크 스테이션 또는 RU ! 지정된 링크는 사 k 할 수 x 습니다. **SLI_OPEN** 명령 n 는 **SLI_OPEN** 재시도 I 션이 사 k 될 때 n 지이 리턴 코드 M 함께 전송될 수 x 습니다. **lua_wait**를 0이 F 닌 * 으로 설정한 상태 ! 서 INITSELF 또는 LOGON 메세지는 호스트 프로시듀 n 를 사 k 할 수 있 T 되 E 나 또는 n 플리케이션이 **SLI_CLOSE** 명령 n 를 발행할 때 n 지 재시도됩니다.

LUA_SESSION_LIMIT_EXCEEDED

d 청된 세션은 네트 v 크 지정 장치(NAU) 중 하나 ! LU-LU 세션 한 h 또는 LU 모드 세션 한 h M 0 은 세션 한 h ! M 있으므로 활설화될 수 x 습니다. 이런 (지 코드는 ACTCDRM, INIT, BID W 리 m CINIT d 청 ! 적 k 됩니다.

SLI_OPEN 명령 n 는 **SLI_OPEN** 명령 n 재시도 I 션이 사 k 되지 J 는 이상 이 리턴 코드 M 함께 전송됩니다. **lua_wait**를 0이 F 닌 * 으로 설정한 상태 ! 서 INITSELF 또는 LOGON 메세지는 호스트 프로시듀 n 를 사 k 할 수 있 T 되 E 나 또는 n 플리케이션이 **SLI_CLOSE** 를 발행할 때 n 지 재시도됩니다.

LUA_SLU_SESSION_LIMIT_EXCEEDED

수 k 되면 SLU 세션 한 h ! d 청으로 인하) 초 z 됩니다.

LUA_MODE_INCONSISTENCY

현재 상태 ! 서는 함수를 수행할 수 x 습니다. 의도된 반 세션 8 성 d 소는 지 x 된 d 청을 이해했지만 이를 처리하지는 J R 습니다. 이 코드는 또한 EXR ! 서 (지 코드로 나타납니다.

LUA_INSUFFICIENT_RESOURCES

일시적인 자 x 부족으로 인하) 수신자는 d 청 ! 대해 작동할 수 x 습니다. 의도된 반 세션 8 성 d 소는 지 x 된 d 청을 이해했지만 이를 처리하지는 J R 습니다.

LUA_RECEIVER_IN_TRANSMIT_MODE

레이스 조 G 이 있습니다. 정상 흐름 d 청은 반 g 방향(전송) 회선 f 합 상태 ! 수신 불 ! 능이 E 나 또는 정상 흐름 데이타 처리시 필 d 한 자 x(9: 버퍼)을 사 k 할 수 x 을 때 수신됩니다.

이 코드는 또한 9 \ d 청 ! 서 (지 코드로 나타납니다.

LUA_LU_COMPONENT_DISCONNECTED

LU 8 성 d 소는 전 x 차단 또는 b 타 , a 단절 조 G 때문 ! 사 k 할 수 x 습니다.

LUA_NEGOTIABLE_BIND_ERROR

조정 ! 능한 BIND를 수신했습니다. SLI는 **SLI_OPEN** 명령 n 를 통해 사 k 자 ! 제 x 한 **SLI_BIND** 루틴이 x 으면 조정 ! 능한 BIND를 허 k 하지 J 습니다.

LUA_BIND_FM_PROFILE_ERROR
지x되지 J는 FM 프로파일이 BIND!서 (지되z습니다. SLI는 FM 프로파일 3z 4만 지x합니다.

LUA_BIND_TS_PROFILE_ERROR
지x되지 J는 TS 프로파일이 BIND!서 (지되z습니다. SLI는 TS 프로파일 3z 4만 지x합니다.

LUA_BIND_LU_TYPE_ERROR
지x되지 J는 LU 유형이 (지되z습니다. LUA는 LU 0, LU 1, LU 2 W리m LU 3만 지x합니다.

LUA_SSCP_LU_SESSION_NOT_ACTIVE
d청 처리! 필d한 SSCP-LU 세션은 활동중이 F닙니다. 9를 들n INITSELF d청 처리시 SSCP는 INITSELF!서 명명된 목표 LU!있는 활동중인 세션이 xz습니다.

바이트 2M 3!는 (지 코드 m유 정보! 있습니다. 다음 설정을 사k할 수 있습니다.

0000 적k되는 특정 코드! x습니다.

0001 SSCP-SLU 세션은 재활성화됩니다.

0002 SSCP-PLU 세션은 활동중이 F닙니다. **SLI_OPEN** 명령n는 **SLI_OPEN** 재시도 I션이 사k되지 J는 한 이 리턴 코드M 함께 전송될 수 있습니다. **lua_wait**를 0이 F닌 *으로 설정한 상태!서 INITSELF 또는 LOGON 메세지는 호스트 프로시듀n를 사k할 수 있T 되E나 또는 n플리케이션이 **SLI_CLOSE** 명령n를 발행할 때n지 재시도됩니다.

0003 SSCP-SLU 세션은 활동중이 F닙니다.

0004 SSCP-SLU 세션은 재활성화됩니다.

LUA_REC_CORR_TABLE_FULL
d청된 흐름! 대한 세션 수신 상| 테이블이 k량! 도달했습니다.

LUA_SEND_CORR_TABLE_FULL
d청된 흐름! 대한 전송 상| 테이블이 k량! 도달했습니다.

LUA_SESSION_SERVICES_PATH_ERROR
세션 서비스 d청은 SSCP-SSCP 세션의 f로를 따라 다시 f로지정될 수 x습니다. 이 b능은 9를 들n, 상호 네트v크 LU-LU 세션을 설정할 때 필d합니다.

바이트 2M 3!는 (지—코드—m유 정보! 있습니다. 다음 설정을 사k할 수 있습니다.

0000 적k!능한 특정 코드! x습니다. **SLI_OPEN** 명령n는 **SLI_OPEN** 재시도 I션이 사k되지 J는 한 이 리턴 코드M 함께 전송될 수 x습니다. **lua_wait**를 0이 F닌 *으로 설정한 상태!서 INITSELF 또는 LOGON 메세지는 호스트 프로시

듀n를 사k할 수 있T 되E나 또는 n플리케이션이 **SLI_CLOSE**를 발행할 때n지 재시도됩니다.

0001 SSCP는 하나 이상의 인접한 SSCP를 통해 목적지로 세션 서비스 f로를 재지정하지 못했습니다. 이 *은 모든 시행 착@를 다 E친 후! T이트~이 SSCP!서 전송됩니다.

SSCP f로 재지정은 O전히 실패했습니다. SSCP는 특정 SSCP! 대해 실패했습니다. 9를 들n 이 코드는 f로 재지정 실패! 대한 정보! f로 재지정하려m 했던 노드! 표시될 때 특정 코드M , |됩니다.

0002

SSCP는 필d한 f로지정 테이블을 사k할 수 x으므로 세션 서비스의 f로를 재지정할 수 x습니다. 다시 말해서 자x 식별자 제n 벡터의 f로 재지정 키! 해당하는 인접 SSCP 테이블이 x습니다.

0003

이 SSCP!는 LU의 사전정의! x지만 인접 SSCP는 상대방 SSCP!서 동적 정의를 지x하지 J습니다. az적으로 이 SSCP는 LU를 동적으로 정의하m W 인접 SSCP로 f로 재지정할 수 x습니다.

0005

사k하지 J음

0006

사k하지 J음

0008

인접 SSCP는 d청된 CDINIT 함수(9를 들n, 자x !k성의 통보 또는 XRF)를 지x하지 J습니다.

000A

SSCP는 d청이 0은 SSCP를 통해 두 번 f로지정되zb 때문! 세션의 f로를 재지정할 수 x습니다.

000B

CDINIT! 지정된 DLU는 수신중인 SSCP! K려져 있지 J으며 수신중인 SSCP는 CDINIT의 f로를 재지정할 수 x습니다.

LUA_RU_LENGTH_ERROR

d청된 RU는 너무 fE나 너무 짧습니다. RU는 의도된 반 세션 8성d소로 전달되z지만 해석 또는 처리될 수 x습니다. 이 조G은 반 세션 b능의 불일치를 나타냅니다.

이 코드는 또한 EXR!서 (지 코드로 나타납니다.

LUA_FUNCTION_NOT_SUPPORTED

LUA는 d청된 함수를 지x하지 J습니다. 함수는 포맷된 d청 코드, RU의 매3변수 또는 제n 코드! 의해 지정됩니다.

(지 코드 다음! @는 바이트 2M 3은 사k자! 정의한 데이타! 사k되지 J습니다. 이런 바이트!는 (지 코드 m유 정보! 있습니다. 다음 설정을 사k할 수 있습니다.

0000 LUA는 d청된 함수를 지x하지 J습니다.

RU는 의도된 반 세션 8성d소로 전달되z지만 해석 또는 처리될 수 x습니다. 이 조G은 반 세션 b능의 불일치를 나타냅니다.

LUA_HDX_BRACKET_STATE_ERROR

프로토콜 시스템은 b존의 상태 @류!서 현재 d청을 전송할 수 x다m 판별했습니다.

LUA_RESPONSE_ALREADY_SENT

프로토콜 시스템은 체인! 대한 응답이 이미 전송되z으므로 현재 d청이 전송될 수 x다m 판별했습니다.

LUA_EXR_SENSE_INCORRECT

n플리케이션은 이전! 수신된 9\ d청! 대해 부정적으로 응답했습니다. 응답 코드의 (지 코드를 수k할 수 x습니다.

9\ d청의 (지 코드! X'0813000'이면 부정적인 응답의 (지 코드는 X'08130000' 또는 X'08140000'이 됩니다. 다른 모든 fl! 부정적인 응답의 (지 코드는 9\ d청의 (지 코드M 0습니다.

LUA_RESPONSE_OUT_OF_ORDER

프로토콜 시스템은 현재 응답이 !장 @래된 d청! 대한 응답이 F니라m 판별했습니다.

LUA_CHASE_RESPONSE_REQUIRED

프로토콜 시스템은 현재 d청이 더 @래된 미해a CHASE d청으로)\지m 있다m 판별했습니다.

LUA_CATEGORY_NOT_SUPPORTED

DFC, SC, NC 또는 FMD d청은 W 범주! 속하는 n떤한 d청도 지x하지 J는 반 세션! 의해 수신되m 네트v크 서비스(NS) d청 바이트 0은 정의된 *으로 설정되지 JE나 또는 수신자! 바이트 1을 NS 범주로 설정하지 JR습니다.

LUA_CHAINING_ERROR

처음, 중#, 처음z 0은 체인 표시b 설정 순서! @류! 발생했습니다. 수신자의 현재 세션 제n나 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_BRACKET

전송자는 세션! 대해 }호 T칙을 -d하지 J습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 d청

헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_DIRECTION

정상 흐름 d청은 반 g방향(전송) 플립 플롭 상태! NOT_RECEIVE인 동H 수신되z습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_DATA_TRAFFIC_RESET

FMD 또는 정상 흐름 DFC d청은 세션 활성화 상태! 활동중이지만 데이타 트래픽 상태! 활동중이지 J은 반 세션! 의해 수신되z습니다. 수신자의 현재 세션 제n나 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_DATA_TRAFFIC_QUIESCED

QC 명령 또는 SHUTC 명령을 전송한 반 세션!서 수신된 FMD 또는 DFC는 RELQ 명령! 응답하지 J R습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 응답 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_DATA_TRAFFIC_NOT_RESET

세션 제n d청은 데이타 트래픽 상태! 재설정되지 J는 동H 수신되z습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_NO_BEGIN_BRACKET

BBI=BB를 지정한 BID 또는 FMD d청은 수신자! BIS 명령으로 `정적인 응답을 전송한 후! 수신되z습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_SC_PROTOCOL_VIOLATION

SC 프로토콜을 위반했습니다. SC d청z |련된 `정적인 응답이 3환된 후!만 허k되는 d청이 성x적인 3환이 이루n지b 전! 수신되z습니다. (지 데이타의 바이트 4!는 d청 코드! 있습니다. 이 (지 코드M |련된 사k자 데이타! x습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 d청 헤더 또는 d청 단위! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_IMMEDIATE_REQ_MODE_ERROR

d청은 즉석 d청 모드 프로토콜을 위반했습니다. 수신자의 현재 세

션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 RH 또는 RU ! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_QUEUED_RESPONSE_ERROR

d청은 대b행렬! 들n # 응답 프로토콜을 위반했습니다. 9를 들n, 미해a d청! QRI=QR이 있을 때 QRI=¬ QR입니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 RH 또는 RU! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_ERP_SYNC_EVENT_ERROR

ERP 동b 이벤트 프로토콜을 위반했습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n 상태!서 허k되지 J는 RH 또는 RU! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_RSP_BEFORE_SENDING_REQ

이전! 수신된 d청! 대한 응답이 F직 전송되지 JR을 때 반 g 방향(전송) (플립 플롭 또는 회선f합) 전송/수신 모드!서 정상 흐름 d청이 전송되z습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n!서 허k되지 J는 RH 또는 RU! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

LUA_RSP_CORRELATION_ERROR

이전! 전송된 d청z 상| |h를 이루지 J는 응답이 수신되E나 또는 이전! 수신된 d청z 상| |h를 이룰 수 x는 응답이 전송되z습니다.

LUA_RSP_PROTOCOL_ERROR

다음z 0은 응답 프로토콜을 위반한 1차 반 세션!서 응답을 수신했습니다.

- `정적인 응답(+RSP)이 RQE 체인!서 수신되z습니다.
- 체인 하나!서 응답 두 3를 수신했습니다.

LUA_INVALID_SC_OR_NC_RH

세션 제n(SC)의 RH 또는 네트v크 제n(NC) d청은 유효하지 J습니다. 9를 들n 1로 설정된 페이싱 d청 표시b의 SC RH는 유효하지 J습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_BB_NOT_ALLOWED

시작 }호 표시b(BB)! 잘못 지정되z습니다. 9를 들n BCI=¬BC인 BBI=BB. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재

상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_EB_NOT_ALLOWED

끝 }호 표시b(EB)! 잘못 지정되z습니다. 9를 들n BCI=¬BC의 EBI=EB! 의해 또는 2차 반 세션이 EB를 전송할 수 있을 때만 1차 ! 의해 또는 1차 반 세션이 EB를 전송할 수 있을 때만 2차! 의해서 지정됩니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_EXCEPTION_RSP_NOT_ALLOWED

9\ 응답이 허k되지 J을 때 d청되z습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_DEFINITE_RSP_NOT_ALLOWED

한정된 응답이 허k되지 J을 때 d청되z습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_PACING_NOT_SUPPORTED

페이싱 표시b는 d청! 설정되z지만 수신중인 반 세션 또는 fh 함수 반 세션은 이 세션! 대해 페이싱을 지x하지 J습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_CD_NOT_ALLOWED

방향 변f 표시b(CD)! 잘못 지정되z습니다. 9를 들n ECI=¬EC의 CDI=CD 또는 EBI=EB의 CDI=CD. RH 의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_NO_RESPONSE_NOT_ALLOWED

응답 x음은 허k되지 J을 때 d청! 지정되z습니다. 응답 x음은 EXR!서만 사k됩니다. RH의 매3변수 또는 매3변수 조합의 *은

8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_CHAINING_NOT_SUPPORTED

체이닝 표시b(BCI 및 ECI)는 잘못 지정되z습니다. 9를 들n BCI=BCM ECI=EC\의 체이닝 비트! 표시되z지만 복수 d청 체인은 세션 또는 d청 헤더!서 지정된 범주를 위해 지x되지 J습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_BRACKETS_NOT_SUPPORTED

}호 표시b(BBI 및 EBI)! 잘못 지정되z습니다. 9를 들n, }호 표시b는 설정되z지만(BBI=BB 또는 EBI=EB) }호는 세션!서 사k되지 J습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_CD_NOT_SUPPORTED

방향 변f 표시b는 설정되z지만 지x되지는 J습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_USE_OF_FI

포맷 표시b(FI)는 잘못 지정되z습니다. 9를 들n FI는 BCI=¬BCM 함께 지정되z E나 또는 FI는 DFC d청!서 설정되지 JR습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_ALTERNATE_CODE_NOT_SUPPORTED

코드 선택 표시b(CSI)! 세션!서 지x되지 J을 때 설정되z습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_RU_CATEGORY

RU 범주 표시b는 잘못 지정되z습니다. 9를 들n 신속 처리된 흐름 d청 또는 응답은 RU 범주 표시b = FMDM 함께 지정되z습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_REQUEST_CODE

응답!서의 d청 코드는 해당 d청의 d청 코드M 일치하지 J습니다. RH 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_SPEC_OF_SDI_RTI

(지 데이타 포함 표시b(SDI)M 응답 유형 표시b(RTI)! 응답!서 C바르T 지정되지 JR습니다. 적합한 * 쌍은 SDI=SD, RTI=부정 및 SDI=¬SD, RTI=`정입니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_DR1I_DR2I_ERI

한정 응답 1 표시b(DR1I), 한정 응답 2 표시b(DR2I) W리m 9\ 응답 표시b(ERI)! 잘못 지정되z습니다. 9를 들n CANCEL d청은 DR1I=DR1, DR2I=¬DR2, ERI=¬ERz 함께 지정되지 JR습니다. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_USE_OF_QRI

대b행렬! 들n# 응답 표시b(QRI)! 잘못 지정되z습니다. 9를 들n 신속 처리 흐름 d청!서의 QRI=QR. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_USE_OF_EDI

0호화된 데이타 표시b(EDI)! 잘못 지정되z습니다. 9를 들n DFC d청의 EDI=ED. RH의 매3변수 또는 매3변수 조합의 *은 8조적

인 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_INCORRECT_USE_OF_PDI

패드된 데이타 표시b(PDI)! 잘못 지정되z습니다. 9를 들n DFC d청!서의 PDI=PD. RH의 매3변수 또는 매3변수 조합의 *은 8조적 T칙 또는 U!서 선택된 LOGON I션을 위반합니다. 이런 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막으며 세션의 현재 상태M는 |h! x습니다. 이런 @류는 세션 T칙을 -d하지 못한 전송자의 실패로 초래될 수 있습니다.

LUA_NAU_INOPERATIVE

NAU는 d청 또는 응답을 처리할 수 x습니다. 9를 들n NAU는 비정상적인 종료로 인하) 훼손되z습니다. d청은 f로 중단 시#, 잘못된 활성화 d청 순서, 또는 나-된 f로 정보 단위 (PIU) @류 중 하나 때문! 의도된 수신자로 전달될 수 x습니다. 세션이 활동중일 때 수신된 f로 @류는 보통 세션 상대방으로의 f로! 유실됐음을 나타냅니다.

LUA_NO_SESSION

x점 목적지 쌍의 수신 끝 노드!서 활동중인 반 세션이 x으며 바n더리 b능을 제x하는 노드의 x점 목적지 쌍!서 활동중인 바n더리 b능 반 세션 8성d소! x습니다. 세션 활성화 d청이 필d합니다. d청은 f로 중단 시# 또는 잘못된 활성화 d청 순서 때문! 의도된 수신자로 전달될 수 x습니다. 세션이 활동중일 때 수신된 f로 @류는 보통 세션 상대방으로의 f로! 유실됐음을 나타냅니다.

LUA_BRACKET_RACE_ERROR

}호 프로토콜!서의 회선f합 유실이 발생합니다. g NAU! 의해 }호 시작 또는 }호 중단이 발생하면 회선f합이 유실됩니다. 의도된 반 세션 8성d소는 지x된 d청을 이해했지만 이를 처리하지는 JR습니다.

LUA_BB_REJECT_NO_RTR

BID 또는 시작 }호 표시b는 첫 번째 스피커! }호 내 상태! 있E나 또는 첫 번째 스피커! }호 중# 상태! 있을 때 수신됩니다. 첫 번째 스피커는 사kG한을 E부했습니다. n떤 RTR 명령도 전송되지 J습니다. 의도된 반 세션 8성d소는 지x된 d청을 이해했지만 이를 처리하지는 JR습니다.

LUA_CRYPTOGRAPHY_INOPERATIVE

d청의 수신자는 O호화 b능의 @작동으로 d청을 해독하지 못했습니다. 의도된 반 세션 8성d소는 지x된 d청을 이해했지만 이를 처리하지는 JR습니다.

LUA_SYNC_EVENT_RESPONSE

동b화 d청! 대한 부정적인 응답이 수신되z습니다. 의도된 반 세션 8성d소는 지x된 d청을 이해했지만 이를 처리하지는 JR습니다.

LUA_RU_DATA_ERROR

d청 RU의 데이타는 수신중인 FDMS 8성d소!서 수k될 수 x습니다. 9를 들n 문자 코드는 지x되는 집합! x으며 포맷된 데이타 매3변수는 프레젠테이션 서비스!서 수k될 수 xE나 또는 d청!서 필수 이름이 생략되z습니다. RU는 의도된 반 세션 8성d소로 전달되z지만 해석 또는 처리될 수 x습니다. 이 조G은 반 세션 b능의 불일치를 나타냅니다.

LUA_INCORRECT_SEQUENCE_NUMBER

정상 흐름 d청!서 수신된 순서 번호는 마지막 순서 번호보다 크지 JR습니다. 수신자의 현재 세션 제n 또는 데이타 흐름 제n!서 허k되지 J는 순서 번호 또는 RH 또는 RU! (지되z습니다. 이 @류는 d청을 의도된 반 세션 8성d소로 전달하지 못하도록 막습니다.

부록C. APPC 대화 상태 전이

다음 표는 " APPC 명령n ! 발행되는 대화 상태M 명령n ! 끝날 때 발생하는 상태 변화를 보) 줍니다. 몇몇 f l, 상태 변화는 명령n ! 리턴된 **primary_rc** 매 3 변수 ! 따라 달라집니다. 적 k ! 능한 **primary_rc** * 은 리턴 코드 - ! 나 - 됩니다.

리턴 코드 ! x 는 w ! 서의 상태 변화는 모든 리턴 코드 ! 서 동일합니다(다음 표의 노트 2M 3 ! 서 설명된 M 제 \).

! 능한 대화 상태는 - 표제로 표시됩니다. " 명령n ! 대한 다음 정보는 이 상태 ! 서 발행되는 명령n 의 a z 를 나타내 b 위해 " 표제 밑 ! 표시됩니다.

- **X** 는 이 상태 ! 서 명령n 를 발행할 수 x 을 때.
- **S**, **SP**, **R**, **C**, **CS**, **CD** 또는 **P** 는 명령n ! 0 료된 후 대화의 상태를 나타내는 M 으로 **R**eset, **S**end, **S**end **P**ending, **R**eceive, **C**onfirm, **C**onfirm **S**end, **C**onfirm **D**eallocate 또는 **P**ending Post입니다.
- / 는 이전 상태 m 려시 적 k 될 수 x 을 때. 이는 **[MC_]ALLOCATE** 및 **RECEIVE_ALLOCATE** 명령n ! 적 k 됩니다. 이들 명령n 는 이를 발행한 대화(있는 f l)! 5 향을 주지 J 는 상태 ! 서 마치 Reset 상태 ! 있는 M 처럼 항상 새로n 대화를 시작합니다.
- 표시된 리턴 코드 ! 이 상태 ! 서 발생할 수 x 으면 x 백.

0 전 g 방향(전송) 대화 상태 전이 ! 대해 K m 싶으면 400페이지의 표 23 을 참조하십시@ .

표 22. APPC 반 g 방향(전송) 대화 상태 전이

명령어 리턴 코드	**Reset (T)**	**Send (S)**	**S e n d Pending (SP)**	**Receive (R)**	**Confirm (C)**	**Confirm Send (CS)**	**Confirm Deall (CD)**	**P e n d - Post (PS)**
[MC_]ALLOCATE AP_OK (b 타)	S T	/	/	/	/	/	/	/
CANCEL_CONVERSATION	X	T	T	T	T	T	T	T
[MC_]CONFIRM AP_OK AP_ERROR	X	S R	S R	X	X	X	X	X
[MC_]CONFIRMED	X	X	X	X	R	S	T	X

표 22. APPC 반 g 방향(전송) 대화 상태 전이 (h 속)

명령어 리턴 코드	Reset (T)	Send (S)	S e n d Pending (SP)	Receive (R)	Confirm (C)	Confirm Send (CS)	Confirm Deall (CD)	P e n d - Post (PS)
[MC_]DEALLOCATE (이상 종료) [MC_]DEALLOCATE (b 타) AP_ERROR (b 타)	X X	T R T	T R T	T X	T X	T X	T X	T X
[MC_]FLUSH	X	S	S	X	X	X	X	X
[MC_]GET_ATTRIBUTES	X	S	SP	R	C	CS	CD	P
GET_STATE	X	S	SP	R	C	CS	CD	P
GET_TYPE	X	S	SP	R	C	CS	CD	P
[MC_]PREPARE_TO_RECEIVE	X	R	R	X	X	X	X	X
RECEIVE_ALLOCATE AP_OK (b 타)	R T	/	/	/	/	/	/	/
[MC_]RECEIVE_AND_POST (주 4)	X	P	P	P	X	X	X	X
[MC_]RECEIVE_AND_WAIT	X	주 5	주 5	주 5	X	X	X	X
[MC_]RECEIVE_IMMEDIATE	X	X	X	주 5	X	X	X	X
[MC_]REQUEST_TO_SEND	X	X	X	R	C	X	X	P
[MC_]SEND_DATA AP_OK AP_ERROR	X	S R	S	X	X	X	X	X
[MC_]SEND_ERROR AP_OK AP_ERROR	X	S R	S	S	S	S	S	S
[MC_]TEST_RTS	X	S	S	R	C	C	C	P

주:

1. 표의 리턴 코드 - ! 서 AP_ERROR의 ` n ! 다음 리턴 코드 ! 사 k 됩니다.

   AP_PROG_ERROR_TRUNC

   AP_PROG_ERROR_NO_TRUNC

   AP_PROG_ERROR_PURGING

   AP_SVC_ERROR_TRUNC

   AP_SVC_ERROR_NO_TRUNC

   AP_SVC_ERROR_PURGING.
2. 대화는 다음 리턴 코드 ! 수신될 때 항상 Reset 상태로 들 n) 니다.

   AP_ALLOCATION_ERROR

AP_COMM_SUBSYSTEM_ABENDED

AP_COMM_SUBSYSTEM_NOT_LOADED

AP_CONV_FAILURE_RETRY

AP_CONV_FAILURE_NO_RETRY

AP_DEALLOC_ABEND

AP_DEALLOC_ABEND_PROG

AP_DEALLOC_ABEND_SVC

AP_DEALLOC_ABEND_TIMER

AP_DEALLOC_NORMAL

3. 다음 비 OK 리턴 코드는 상태 변화를 일으키지 J습니다. 대화는 항상 명령n ! 발행된 상태로 남습니다.

   AP_CONVERSATION_TYPE_MIXED

   AP_PARAMETER_CHECK

   AP_STATE_CHECK

   AP_TP_BUSY

   AP_UNEXPECTED_SYSTEM_ERROR

   AP_UNSUCCESSFUL

4. **[MC_]RECEIVE_AND_POST** ! 발행되m AP_OK의 초b **primary_rc**를 수신한 후 대화는 Pending Post 상태로 바뀝니다. 일단 명령n 0료를 나타내b 위해 제x된 콜백(callback) 루틴이 호출되면, 주 5의 **primary_rc** M **what_rcvd** 매3변수 ! 따라 새로n 대화 상태 ! 달라집니다.

5. **RECEIVE** 명령n 중 하나 이후 ! 이루n지는 상태 변화는 **primary_rc** M **what_rcvd** 매3변수 ! 따라 달라집니다.

   primary_rc 매3변수 ! AP_PROG_ERROR*, AP_SVC_ERROR* 또는 ([MC_]RECEIVE_IMMEDIATE만) AP_UNSUCCESSFUL인 f l 새로n 상태는 RECEIVE ! 됩니다.

   primary_rc 매3변수 ! AP_DEALLOC*이면 새로n Reset 상태 ! 됩니다.

   primary_rc 매3변수 ! AP_OK인 f l, 새로n 상태는 **what_rcvd** 매3변수의 * ! 따라 달라집니다.

   Receive 상태
   AP_DATA, AP_DATA_COMPLETE, AP_DATA_INCOMPLETE

   Send 상태
   AP_SEND

   Send Pending 상태
   AP_DATA_SEND, AP_DATA_COMPLETE_SEND

   Confirm 상태
   AP_CONFIRM_WHAT_RECEIVED, AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM

Confirm Send 상태

AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND

Confirm Deallocate 상태

AP_CONFIRM_DEALLOCATE, AP_DATA_CONFIRM_DEALLOCATE, AP_DATA_COMPLETE_CONFIRM_DEALL

반 g방향(전송) 대화 상태 전이! 대해 Km 싶으면 397페이지의 표22를 참조하십시@.

표 23. APPC 0전 g방향(전송) 대화 상태 전이

명령어 리턴 코드	Reset (T)	Send Receive (SR)	Send 전용 (S)	Receive 전용 (R)
[MC_]ALLOCATE AP_OK (b타)	 SR T	/	/	/
CANCEL_CONVERSATION	X	T	T	T
[MC_]DEALLOCATE (이상종료) [MC_]DEALLOCATE (플러쉬)	X X	T R	T T	T X
[MC_]FLUSH	X	SR	S	X
[MC_]GET_ATTRIBUTES	X	SR	S	R
GET_STATE	X	SR	S	R
GET_TYPE	X	SR	S	R
RECEIVE_ALLOCATE AP_OK (b타)	 SR T	/	/	/
[MC_]RECEIVE_AND WAIT AP_OK AP_ERROR AP_DEALLOC_NORMAL	 X X X	 SR SR S	 X X X	 R R T
RECEIVE_EXPEDITED_DATA	X	SR	S	R
[MC_]RECEIVE_ IMMEDIATE AP_OK AP_ERROR AP_DEALLOC_NORMAL	 X X X	 SR SR S	 X X X	 R R T
[MC_]SEND_DATA AP_OK AP_ERROR_INDICATION	 X X	 SR SR	 S T	 X X

표 23. APPC 0전 g 방향(전송) 대화 상태 전이 (h 속)

명령어 리턴 코드	**Reset (T)**	**Send Receive (SR)**	**Send 전용 (S)**	**Receive 전용 (R)**
[MC_]SEND_ERROR				
AP_OK	X	SR	S	X
AP_ERROR_INDICATION	X	SR	T	X

주:

1. 표의 리턴 코드 - ! 서 AP_ERROR의 ` n ! 다음 리턴 코드 ! 사 k 됩니다.

   AP_PROG_ERROR_TRUNC

   AP_PROG_ERROR_NO_TRUNC

   AP_SVC_ERROR_TRUNC

   AP_SVC_ERROR_NO_TRUNC

2. 대화는 다음 리턴 코드 ! 수신될 때 항상 Reset 상태로 들 n) 니다.

   AP_ALLOCATION_ERROR

   AP_COMM_SUBSYSTEM_ABENDED

   AP_COMM_SUBSYSTEM_NOT_LOADED

   AP_CONV_FAILURE_RETRY

   AP_CONV_FAILURE_NO_RETRY

   AP_DEALLOC_ABEND

   AP_DEALLOC_ABEND_PROG

   AP_DEALLOC_ABEND_SVC

   AP_DEALLOC_ABEND_TIMER

3. 다음 비 OK 리턴 코드는 상태 변화를 일으키지 J 습니다. 대화는 항상 명령 n ! 발행된 상태로 남습니다.

   AP_CONVERSATION_TYPE_MIXED

   AP_PARAMETER_CHECK

   AP_STATE_CHECK

   AP_TP_BUSY

   AP_UNEXPECTED_SYSTEM_ERROR

   AP_UNSUCCESSFUL

부록D. 통신 - 버 - 비스 위치 프로토콜

검색 및 로드 균형 API

통신 서버 (Windows NT k 은 IBM 통신 서버라 m 하 m SAA k 은 IntranetWare 라 m 함) n 플리케이션 프로 W 램 3 발자는 이제 TCP/IP 프로토콜로 서비스를 찾 F 서비스 # 의 로드 U 형을 수행할 수 있습니다. n 플리케이션 프로 W 램은 다음 z 0 은 세 ! 지 b 본 방법으로 이 새로 n b 능의 장점을 활 k 합니다.

- 통신 서버 SNA API (LUx (RUI/SLI), APPC, CPIC). b 존의 n 플리케이션이 이미 SNA API로 작성되 z 다면, API를 사 k 하) ″무료″로 지 x 을 받을 수 있습니다. 이 방법 ! 서는 위치/로드 U 형 b 능을 사 k 하 b 위해 새로 n 코드를 작성하지 J F 도 됩니다. 이 방법 ! 서 유일한 제한점은 API 코드 ! 서 클라이 p 트 8 성 데이타 ! IntranetWare k Novell Directory Services(NDS)나 INI 파일 ! 또는 Windows NT k LDAP 통신 서버 ! 상주 해 _ 한다는 M 입니다.
- 서비스 위치 프로토콜(SLP) 사 k 자 ! 이전트(UA) API. SLP UA DLL은 TCP/IP , a ! 서 통신 서버 서비스 위치 및 로드 U 형 ! 대한 지 x 을 제 x 하는 제품 z 함께 패키지화됩니다. 이 방법은 서비스 위치/로드 U 형 수행 방법, 클라이 p 트 8 성 확보 방법 W 리 m 이런 b 능을 일반 사 k 자 ! T 제 x 하는 방법의 측면 ! 서 n 플리케이션 3 발자 ! T 최대한의 유 , 성을 제 x 합니다.
- 로드 U 형 ! 대해 UA (위치 ! 대해) 및 QEL/MU CM_CSLIST_GETII x 시 조합 사 k 하 b (3270 및 LU6.2 n 플리케이션 전 k). 이 방법은 위치 함수 ! 만 작성되 n _ 하는 코드의 g 을 줄이 m 클라이 p 트 8 성 ! 있 n 서 최대한의 유 , 성을 제 x 한다는 면 ! 서 U 의 두 방법을 혼합한 M 입니다.

IBM 및 Novell은 위치 및 로드 U 형 ! 대해 API 클라이 p 트의 사 k 을 G 장합니다. n 플리케이션 3 발자 ! 이를 수행할 수 x E 나 텔넷 지 x 을 d 8 하는 f l ! 는 방법 2 ! 제 x 됩니다. QEL/MU ! 대한 지 x 이 이미 제 x 되 z 다면 방법 3이 사 k 됩니다. 첫 번째 방법은 실제로 n 플리케이션 3 발자의 | 점 ! 서 새로 n M 은 F 니므로 다음 설명은 마지막 두 방법 ! 적 k 됩니다.

구조

UA API는 ″서비스 위치 k API″ 인터넷 드래프트(97/3/25일자) ! 제시된 M 이후 ! 모델화된 범 k C p n API 입니다. 다음 특성은 서비스 등록 ! 적 k 됩니다.

- 모든 등록은 미 9 식 5 n 로 이루 n 집니다.
- 문자 집합은 US-ASCII입니다.

API는 Windows 95 Windows NT 플랫폼에서 IBMSLP.DLL로 패키지화됩니다. 헤더 파일은 런타임, 상수 및 함수 프로토타입을 정의하는 이 SDK에 제공됩니다. DLL은 API 클라이언트에 설치될 때 설치되며 \NWSAA\SAASDK\SLP\BINARY\IBMSLP.DLL 또는 \CSNT\SDK\SLP\BINARY\IBMSLP.DLL에서 찾을 수 있습니다. 타 SLP SDK 파일과 함께 제품 CD-ROM에서도 찾을 수 있습니다.

시나리오

" 시나리오에서 사용자 인터페이스 API를 사용하는 애플리케이션 프로그램을 "app."라고 합니다. 일반 사용자(app를 사용하는 사람)는 "사용자"로 축약됩니다.

방법 2: 최소 로드(또는 "낮은 로드") 서비스를 찾기 위한 UA API.

1. 애플리케이션은 SLP에 있는 세션을 위해 SL_Open을 발행합니다.
2. 범위에 구성되지 않거나 또는 이를 app에서 사용할 수 없을 때 애플리케이션은 유효하며 판독할 수 있는 범위를 확보하기 위해 'SCOPE'의 속성 태그 필터에 있는 서비스 유형에 대해 SL_GetAttrs API 호출을 발행합니다. 이 api 호출에서 정의된 SAA 3.0 또는 CNT 6.0 IntranetWare 중 하나의 서비스명을 제공하면 제공된 서비스 유형에만 적용되는 범위에 리턴됩니다.
3. 애플리케이션은 제공하는 서비스, 확보된 범위명 중 하나 필요한 서비스 속성을 나타내는 조회 문자열을 지정하는 SL_GetService를 발행합니다. 설명 목적으로 이 조회에 지정된 서비스 속성은 LUPOOL LOAD입니다. 조회 문자열은 사항을 충족시키는 동일 서비스 응답에는 일치하는 서비스를 찾지 못했다는 표시 또는 서비스를 제공할 수 있는 URL의 리스트에 있습니다.
4. 애플리케이션은 리턴된 리스트를 분석합니다.
5. 리턴된 URL이 없는 경우, 애플리케이션은 새로운 LOAD 준 범위와 함께 단계 3에서 설명된 대로의 SL_GetService 요청을 수정 및 재발행하거나 또는 일반 사용자에게 현재 서비스를 사용할 수 없음을 보고합니다.
6. 단일 URL이 리턴되면 분석이 종료됩니다.
7. URL 리스트에 리턴되는 경우:
   - **옵션 1 - "최소 로드" 위치**
     a. 애플리케이션은 서비스 응답에 리턴된 " URL에 대해 SL_GetAttrs를 발행합니다. 이는 호출할 때마다 선택 절에서 LOAD 속성을 지정합니다. LOAD * 은 속성 확보 응답에 리턴됩니다.
     b. 애플리케이션은 장 낮은 LOAD * 을 진 URL을 선택합니다.
     c. 애플리케이션은 선택된 URL로 표시되는 서버에 , 연결하며 SNA 세션을 시작합니다.
     d. 애플리케이션은 SLP 세션을 닫기 위해 SL_Close를 발행합니다.
   - **옵션 2 - "낮은 로드" 선택**
     a. 리턴된 리스트에서 임의로 URL을 선택합니다.

b. n플리케이션은 선택된 URL로 표시되는 서버! , a되m SNA 세션을 시작합니다.

c. n플리케이션은 SLP 세션을 닫b 위해 SL_Close를 발행합니다.

대부분의 서버!서 로드 U형! 대해 I션이 두 3 제시된다는 점! 유의하십시@. 두 I션의 주d 차이점은 다음z 0습니다. I션 1!서는 최소한 로드 서버! 선택되지만 I션 2보다 더 많은 LAN 트래픽이 생성됩니다. I션 2!서는 "낮은 로드" 서버만 선택되지만 선택 프로세스 중! I션 1보다 더 적은 LAN 잠재적 통신회선 트래픽을 제x합니다.

재시도: 많은 f l n플리케이션!서 이루n지는 , a 재시도는 사k자의 최대 자x !k성! 5향을 주b 위해 필d합니다. n플리케이션! 의한 , a 재시도! 필d한 f l는 n플리케이션이 SL_GetService! 리턴된 URL로 , a하려m 시도하m 나서 SNA 세션을 설정하지만 사k할 수 있는 LU! x을 때입니다. 이 조G은 SLP를 통해 등록된 서비스 종류#의 느슨한 a합z 등록하는 서버!서 실제로 사k할 수 있는 서비스 종류로 !능해집니다. n플리케이션이 선택된 서비스! , a되지 못하면, 리턴된 또 다른 서비스로 재시도해_ 합니다(9를 들n 최소 로드 서버). 더 이상 사k할 수 있는 서비스! x는 f l, n플리케이션은 초b SL_GetService!서 재시도하E나 또는 조G을 일반 사k자! T 보m합니다.

URL 포맷: 통신 서버!서 선전된 URL은 두 부분으로 이루집니다. 점이 있는 십진수 IP 주소M 포트 번호입니다.

URL은 다음 포맷을 !진 ASCII 문자 - 입니다.

```
<IP address>:<port number>
```

IP 주소는 서버! 대한 b본 IP 주소입니다. 포트 번호는 선전된 서비스 유형! 따라 달라집니다.

표 24. 서비스 유형/포트 정보

서비스 유형	포트
commserver	잘 K려진 CommExec , a대b중 포트 1366
cs3270	잘 K려진 CommExec , a대b중 포트 1366
csappc	잘 K려진 CommExec , a대b중 포트 1366
tn3270	서버의 ETC/SERVICES 파일!서 확보되E나 텔넷 서버! 8성된 텔넷 포트

포트

NWSAAk 통신 서버의 다음 릴리스!서는 AS400 시스템z의 다중 , a을 제x하b 위해 다중 포트! 대한 지x이 이루n집니다. CSNT 6.0은 현재 다중 포트를 지x합니다. H전 세션! 대한 b본 포트 번호! F닌 다른 포

트 번호! 필d한 H전 O호화 텔넷 세션! 대한 지x이 제x됩니다. ! 뮬레이터는 SLP 서비스 K색!서 리턴된 포트 번호를 사k할 수 있n_ 합니다. 서비스 유형의 자세한 정보는 TEMPLATE.HTM 파일! 있습니다.

예제 1: n플리케이션은 텔넷을 통해 3270 !뮬레이션을 제x합니다. 이는 8성된 ACCOUNTS의 LU 풀!서 사k할 수 있는 임의의 LUM ,a되n_ 하며 !장 !볍T 로드된 서버를 통해 ,a되n_ 합니다. 네트v크!서 8성된 범위! x습니다. 메인프레임 호스트는 n플리케이션이 장치 유형을 지정하지 J도록 동적 장치 유형을 지x합니다.

n플리케이션은 서버를 찾으려m SL_GetService d청! 대한 다음 술n 발행으로 시작합니다(모든 9제!서 '\t'는 TAB 문자임).

```
tn3270//LUPOOL==ACCOUNTS*/
```

이 시점!서 33의 URL 리스트(이M 비슷)! 리턴됩니다(포트 번호 32은 텔넷 ,a d청! 대한 표준 포트입니다).

```
service:tn3270://9.37.51.254:23
service:tn3270://9.37.51.260:23
service:tn3270://9.37.51.256:23
```

최소한 로드 위치를 수행하도록 설h된 n플리케이션은 " 서버의 로드측정치를 rb 위해 " URL로 향하는 일련의 SL_GetAttrs 호출을 발행합니다. 이는 로드 정보만 수신하b 위해 F래M 비슷한 선택 절을 지정합니다.

```
URL = service:tn3270://9.37.51.254:23
Attribute filter = LOAD
```

- 속성 LOAD는 * ″5″M 함께 리턴됩니다.
- n플리케이션은 두 번째 URL! 대해 두 번째 SL_GetAttrs를 발행하m W로드는 ″2″로 리턴됩니다.
- W리m a9 ″10″의 로드를 리턴하는 세 번째 서버

두 번째 서버의 로드! 낮으므로 n플리케이션은 ,a 목표로 9.37.52.260:23을 선택합니다. n플리케이션은 9.37.51.260을 통해 ,a하려m 하지만 사k할 수 있는 LU! x으므로 ,a! 실패합니다. W런 후 9.37.51.254(다음 최소 로드 서버)를 통해 ,a을 시도하m 성x합니다.

예제 2: 또 다른 n플리케이션은 tn3270 !뮬레이션을 제x합니다. 이 서비스를 제x하는 최소한으로 로드된 서버를 찾F_ 합니다. 클라이p트의 8성은 INI 파일 또는 NDS!서 확보됩니다. W 범위는 ENGINEERING이m LU Pool SMITH_1!서 LU 유형 2 모델 2를 찾F_ 합니다.

n플리케이션은 서비스 유형이 TN3270이m 속성 태W 필터! 'SCOPE'인 SL_GetAttrs 호출을 발행하) 시작합니다. 이는 TN3270을 지x하는 서버를 |리하는 범위 * 리스트를 리턴합니다. 설명을 쉽T 하b 위해 'ENGINEERING'의 범위 *이 SL_GetAttrs 호출!서 리턴된다m !정하십시@. 다음 n플리케이션은 이 범위!서 W 초b LU 장치 유형을 만족시키는

서버를 찾b 위해 SL_GetService d청! 대해 다음 술n를 8축하m d8사항을 로드합니다(모든 9제!서 '\t'는 TAB 문자임).

```
tn3270/ENGINEERING/LUPOOL==SMITH_1\t3270002,LOAD <= 10/
```

n플리케이션은 10의 로드 증!분!서 찾도록 설h되z습니다. W러므로 초b SL_GetService d청이 빈 리스트를 리턴하면 n플리케이션은 서비스를 새로n 로드 속성z 함께 다시 지정하) SL_GetService를 재발행합니다.

```
tn3270/ENGINEERING/LUPOOL==SMITH_1\t3270002,LOAD <= 20/
```

이 시점!서 두 3의 URL 리스트(이M 비슷)! 리턴됩니다(포트 번호 23은 텔넷 ,a d청! 대한 표준 포트입니다).

```
service:tn3270://9.37.51.254:23
service:tn3270://9.37.51.260:23
```

n플리케이션은 로드! 20% 이하일 때 절대 최소 로드 서버를 선택할 필d! x습니다. W러므로 임의로 리턴된 다음 두 URL 중 하나를 선택합니다.

```
URL = service:tn3270://9.37.51.260:23
```

n플리케이션은 ,a 목표로 9.37.52.260:23을 선택하며 W ,a은 성x합니다.

방법 3: 서비스 위치에 대한 UA 및 로드 균형에 대한 CM_CSLIST_GETII. :
CM_CSLIST_GETII x시는 QEL/MU !뮬레이터! 대해 제x됩니다. x시는 n플리케이션!서 다중 필터를 제x할 수 있도록 확장됩니다. cmi.h 헤더 파일!는 이 방법! 대한 8조M 정의! 있m 이 SDK! 포함됩니다. 이 방법을 사k하려면 다음 프로시듀n! 적k됩니다.

1. n플리케이션은 SLP! 있는 세션을 -b 위해 SL_Open을 발행합니다.
2. 범위! 8성되지 JE나 또는 이를 app!서 사k할 수 x을 때는 n플리케이션이 유효하m 판독!능한 범위를 확보하b 위해 'SCOPE'의 속성 태W 필터를 !진 'cs3270' 서비스 유형! 대해 SL_GetAttrs API 호출을 발행합니다. 이 API는 IP 버전 CM_CSLIST_GETII x시! 응답하며 통신 서버의 서비스 URL! 해당하는 범위 리스트를 리턴합니다.
3. n플리케이션은 'cs3270' 서비스M 유효한 범위만 지정하는 SL_GetService를 발행합니다. 서비스 응답!는 CM_CSLIST_GETII x시를 처리하면서 n플리케이션z ,a되는 서버의 URL 리스트! 있습니다.
4. n플리케이션은 리스트!서 선택된 URL! 의해 표시되는 서버로 ,a됩니다.
5. n플리케이션은 SLP 세션을 닫b 위해 SL_Close를 발행합니다.
6. n플리케이션은 서버의 로드 U형 리스트를 K색하b 위해 CM_CSLIST_GETII x시를 8축합니다.)b서 AgentType 필드는 x하는 서비스로 설정되m 필터 스펙!는 범위M LU 풀명(적k!능한 fl)이 있습니다.
7. 로드 U형 순서의 서버 TCP/IP 주소 리스트! 있는 CM_CSLIST_GETII_ACK! 리턴됩니다(최소!서 최대 로드로).

8. n플리케이션은 리스트!서 첫 번째 서버를 선택하m , a합니다.
9. n플리케이션은 서버M SNA 세션 설정을 시도합니다. 실패할 때는 성x할 때n지 또는 리스트를 모두 사k할 때n지 리턴된 리스트의 다음 서버부터 이전 단h를 반복합니다.

표 25. CM_CSLIST_GETII x시

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
PrimType	x00	4	d 정수 (long int)	cmi.h!서의 CM_CSLIST_ GETII
UserParm	x04	4	d 정수	응답!서 리턴한 모든 *
9 `	x08	4	d 정수	0 (zero)
ServiceType	x0c	4	d 정수	0x12B (로드 U형 지x)
ProdVersion	x10	4	d 정수	-1 (무|함 표시)
NWVersion	x14	4	d 정수	-1 (무|함 표시)
플래W	x18	4	d 정수	테이블 ″플래W*″ 참조
AgentType	x0c	4	d 정수	테이블 ″AgentType *″ 참조
FilterList	x1c	*	FilterList_t	테이블 ″FilterList_t″ 참조(*은 ″플래W″의 설정! 달려 있음)

표 26. CM_CSLIST_GETII x시

상수	*	의미
0 (zero)	0	순서! 정해지지 J은 리스트 필d. n떤 필터도 지정되지 J음(백v드 호환성을 위해 제x).
CMCsListFlags_LBPool	1	로드 U형 풀 이름을 지정하는 로드 U형 리스트 필d(백v드 호환성을 위해 * 제x).
CMCsListFlags_LBAgent	2	로드 U형 리스트 필d. AgentType이 로드 U형!사k됨.

표 26. CM_CSLIST_GETII x시 (h속)

상수	*	의미
CMCsListFlags_LBFilter	3	로드 U형 리스트 필d. 필터의 !번 f이 리스트 ! H.

*표 27. 플래W * (cmi.h ! 서)*

상수	*	의미
CSA_3270	0x126	LU Types1/2/3 ! 대해 SNA T이트~이 !이전트 필d
CSA_SAA	0x12B	LU Type 6.2 ! 대해 SNA T이트~이 !이전트 필d

*표 28. AgentType * (csobjtyp.h ! 서)*

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
FilterNameLen	x00	4	d 정수 (long int)	다음 로드 U형 W룹 (풀) 이름의 f이
FilterName	x04	*	ASCII	로드 U형 W룹 (풀) 이름

표 29. FilterList_t (if Flags = CMCsListFlag_LBPool)

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
FilterCount	x00	4	d 정수	다음 ! @는 필터 리스트 이름 8조의 수 (플래W = 0이면 0)
FilterList	x04	*	Filter_t	필터 리스트 이름 8조의 리스트. " 8조 ! 는 !변 f이 ! 있음.

표 30. FilterList_t (Flags = zero | Flags = CMCsListFlag_LBFilters인 f l)

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
FilterLength	x00	4	d 정수	8조 f이 (이 f이 필드 포함)

표 30. FilterList_t (Flags = zero | Flags = CMCsListFlag_LBFilters인 f l) (h 속)

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
FilterType	x04	4	d 정수	″FilterType *″ 참조
FilterName	x08	*	ASCII	필터명 *

표 31. Filter_t

상수	의미
CMCsListFilter_LBPool	로드 U형 풀 이름. 리스트 하나당 @로지 하나의 풀만 지정될 수 있음. 이 필터는 AgentType CSA_3270 ! 대해서만 유효함.
CMCsListFilter_Scope	SLP 범위명. 범위 하나만 지정될 수 있음. 범위 ! 지정되지 J으면 범위 ! x는 모든 서비스를 m려함.

*표 32. FilterType * (cmi.h ! 서)*

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
PrimType	x00	4	d 정수	cmi.h ! 서의 CM_CSLIST_GETII_ACK
UserParm	x04	4	d 정수 (long int)	CM_CSLIST_GETII ! 서 전달된 M 처럼
9 ` 됨	x08	4	d 정수	0 (zero)
ServiceType	x0c	4	d 정수	CM_CSLIST_GETII ! 서 전달된 M 처럼
플래W	x10	4	d 정수	CM_CSLIST_GETII ! 서 전달된 M 처럼
ServiceCount	x14	4	d 정수	다음 서비스 입력항목의 수

표 33. CM_CSLIST_GETII_ACK x 시

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
ProdVersion	x00	4	d 정수	제품 버전
플랫폼	x04	4	d 정수	CMCsListPlatform_IWSAA

표 33. CM_CSLIST_GETII_ACK x시 (h속)

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
CSNameLen	x08	4	d 정수	다음! @는 서버명 f이
CSName	*	*	d 정수	서버명 (널(null) 중단)
CSAddrLen	*	4	d 정수	다음! @는 IP 주소 f이
CSAddress	*	*	ASCII	점이 있는 십진수 IP 주소. 포트 형식으로된 서버의 IP 주소
NameLen	*	4	d 정수	다음! @는 ! 이전트명의 f이
AgentName	*	*	*	서버!서의 ! 이전트명 (널(null) 중단)

표 34. CM_CSLIST_GETII_ACK x시!서의 서버 정보 8조

필드명	필드 오프셋 (16진수)	필드 길이 (십진수)	유형	내용 및 사용
PrimType	x00	4	d 정수	cmi.h!서의 CM_CSLIST_GETII_ERR
UserParm	x04	4	d 정수	CM_CSLIST_GETII!서 전달된 M 처럼
9`됨	x08	4	d 정수	0 (zero)
Errno	x0c	4	d 정수	@류 번호

구성 고려사항

범위: 서비스! 대한 클라이p트 d청!서 범위 *을 확보하는 방법!는 두!지 선택이 있습니다.

검색

범위 *은 SL_GetAttrs API를 사k하) K색될 수 있습니다(″SCOPE″의 속성 필터를 !진 서비스 유형! 대해 범위! x는 속성 d청을 발행하)). 이 API는 네트v크!서 현재 활동중인 서비스! 대한 범위 리스트를 리턴합니다. 리스트는 사k자! 선택할 수 있도록 표시될 수 있습니다.

구성

범위 *은 클라이p트의 8성! 의해 확보됩니다. Novell 클라이p트 리퀘스터를 통해 NDS를 W세스하는 n플리케이션 범위는 NDS!서 8성되m 확보됩니다. SAA 3.0k IntranetWare는 SLP 범위를 비롯한 많은 클라이p트 유형 8성 * 제x을 위해 NDS 8성! 활b M 3선책을 제x합니다. 범위M W 의미! 대해 다음을 참조하십시@.

DA 검색 시간종료

DA K색 시#종료 *인 SLP_Open API의 매3변수는 네트v크!서 디렉토리 !이전트(DA)를 K색할 때n지 b다려_ 하는 시#을 제n합니다. K색 d청은 멀티캐스트(multicast)이며 많은 d인! 따라 모든 DA 응답을 모으는데 필d한 시#이 달라질 수 있습니다. 네트v크! DA! x는 f l ! 이 시#종료 *은 수행될 DA K색이 x음을 나타내b 위해 0으로 설정됩니다. 시#종료는 밀리초로 표현됩니다.

SA 멀티캐스트(Multicast) 시간종료

SA 멀티캐스트 시#종료 *인 SL_Open API의 매3변수는 d청 범위를 지x하는 최소한 하나의 DA도 x이 SLP API! 네트v크!서 서비스, 속성 또는 서비스 유형을 K색할 때n지 b다려_ 하는 시# 제n ! 사k됩니다. 이 상황!서 이런 d청은 멀티캐스트이며 SLP API는 리턴되는 다중 응답을 모으b 위해 시#종료 * 만큼 b다립니다. 시#종료는 밀리초로 표현됩니다.

관리자 도움말 정보

범위

범위는 네트v크!서 클라이p트! 서버를 W세스할 때 이를 제n하m |리하는 데 사k되는 매3변수입니다. 이는 서비스 위치 프로토콜 범위M 0습니다. 범위 제n ! 필d한 이유는 다음z 0습니다.

- 네트v크, 클라이p트 수 W리m 서버의 수! 늘n나면, 네트v크!서의 전체적인 트래픽을 줄이b 위해 서버! 대한 W세스G을 늘n나는 클라이p트의 수로 나눠_ 합니다.
- |리자는 사k자M 서버를 |리 W룹으로 8성할 수 있T 됩니다.

범위 *의 의미는 네트v크 |리자! 의해 정의됩니다. 이런 *은 임의의 3체를 나타낼 수 있습니다. 보통 부서, 지리학적 또는 조직적 회선! 속합니다.

범위 사용방법

" IWSAA 3.0 또는 CSNT 6.0 서버는 " "의 8성 툴을 통해 범위! 지정됩니다. 이런 서버를 사k하는 클라이p트는 단일 특정 범위의 서버 또는

범위! x는 서버M의 ,a을 위해 8성되n_ 합니다. 8성!능한 서비스인 3270z APPC! 대해 서로 다른 범위를 지정할 수 있습니다.

어떻게 SLP와 관련되는가?

SAA 3.0 또는 CSNT 6.0k IntranetWare 범위는 직접 SLPM |련되n 있습니다. W러므로 SLP 서비스 !이전트M 디렉토리 !이전트는 8성된 이런 범위를 지x하는 네트v크! 상주해_ 합니다. 클라이p트! 범위를 YE로 서비스를 찾을 수 있T 하려면 범위! 네트v크 전체! |련되는 방법을 0두! 두n_ 합니다. 범위! 사k되는 네트v크! 범위! x는 서비스! 있는 fl!는 이런 서비스를 사k하) 범위! x는 서비스를 지x하는 서비스 !이전트M 디렉토리 !이전트! 잠재적으로 부하를 부z하는 범위! 정해진 모든 d청을 충족시킬 수 있습니다. 이런 이유로 도달!능한 모든 서버!서 범위를 8성하E나 또는 8성된 범위! 있는 서버! x는 M이 바람직합니다. 디렉토리 !이전트! 사이트 네트v크(상향 스케일링)! 사k된다면 서버! 대해 8성된 Mz 0은 범위를 처리하도록 8성되n_ 합니다. 이 \!도 디렉토리 !이전트! 있는 네트v크!서 범위! x는 서비스! 사k되는 fl, 범위! x는 디렉토리 !이전트! 최소한 하나 설정되n_ 합니다.

주: SNA API 클라이p트! 범위! x는 서버로 ,a하b 위해 8성된다면 범위! x는 서버만 응답하T 됩니다.

로드 균형 가중치 요인

로드 U형 !중치 d인은 |리자!T " 통신 서버! 대한 로드 U형 측정을 수정하m 잴 수 있는 b능을 제x합니다. d인은 서버마다 다를 수 있습니다. 로드 측정은 0z 100 사이의 정수이며 서버!서의 로드 백분율을 9측하b 위한 M입니다(100이 최m치임). !중치 d인은 이 h산의 한 d소를 |리자!T 제x합니다. 이 d인은 다음z 0은 이유로 유k합니다.

- 일부 fl! 통신 서버 Km리즘!서 m려되지 J는 서버 로드! 5향을 주는 b타 d인이 있을 수 있습니다. 9를 들n 통신 서버! SNA T이트~이 트래픽 전k으로 사k되지 J을 때.
- TN3270/TN5250 서비스를 제x하는 SAA 3.0k IntranetWare T이트~이! 로드 U형!서 SLP를 사k하는 다른 TN 서버 8현 (CNST 6.0)z 함께 네트v크! x존하는 fl!는 SAA 3.0k IntranetWare는 차이점 보상을 위해 조정될 수 있습니다.

|리자는 !중치 d인을 사k하) 서버!서의 로드 측정을 서버 선택!서벗n나T 하E나 또는 W 쪽을 향하T 만들 수 있습니다.

- 비스 템플리트

Comm Server 서비스 템플리트

Commserver 서비스 유형은 Windows NT 또는 SAA 3.0 k IntranetWare일 수 있습니다.

SAA 3.0 k IntranetWare ! 대해 commserver 서비스 유형은 CommExec ! 로드될 때마다 등록됩니다. 이는 SAA 서버 k IntranetWare의 일반 속성을 설명합니다. 이런 속성은 SAA k IntranetWare ! 서 제 x 하는 b 타 서비스 유형 ! 서도 반복됩니다.

```
Release = <version/release>
```

이는 commserver 선정 서비스의 버전 및 릴리스 레벨입니다. W 포맷은 vv.rr.mm이며) b 서 ″vv″는 메이저 버전 번호이 m, ″rr″은 마이너 버전 번호이며 ″mm″은 수정 레벨입니다. 모든 번호는 ^ 쪽 ! 서 두 문자 n 지 0으로 채 v 집니다. 9 를 들면, 버전 6, 릴리스 0, 수정 레벨 0은 ″06.00.00″입니다

```
Platform = <platform>
```

선전 서비스의 b 초 ! 되는 네트 v 크 n 5 체제 플랫폼입니다. 정의된 * 은 다음 z 0 습니다.

IW 서버는 IntranetWare 네트 v 크 n 5 체제를 사 k 합니다.

NT 서버는 Microsoft NT n 5 체제를 사 k 합니다.

OS2 서버는 OS2 n 5 체제를 사 k 합니다.

AIX 서버는 AIX n 5 체제를 사 k 합니다.

```
Protocol = <protocol>
```

다음은 이 서비스를 제 x 하는 서버 ! 서 지 x 되는 프로토콜입니다. 정의된 * 은 다음 z 0 습니다.

IP 서버는 IP(TCP/IP 또는 UDP/IP)를 통해 클라이 p 트 , a 을 지 x 합니다.

IPX 서버는 IPX(SPX/IPX)를 통해 클라이 p 트 , a 을 지 x 합니다.

```
Server name = <server name>
```

이는 설치중 ! 8 성된 서버의 이름입니다. 이 * ! 는 IW 플랫폼만을 위한 의미 ! 있습니다.

Comm Server 서비스 등록 메세지

```
URL:service:commserver://<addr-spec>:<port-number>
```

속성:

```
[(SCOPE=<string>),]
(RELEASE=06.00.00),
```

(PLATFORM=NT),
(PROTOCOL=IP),
(SERVERNAME=<string>)

종속 LU 서비스 템플리트

comm 서버 종속 LU 서비스는 서버 m유 APIM 프로토콜을 통해 SNA 네트v크로의 3270 T이트~이 W세스를 제x합니다. 속성은 3270 장치 유형, LU 풀 W리m 서버!서 사k할 수 있는 로드 정보를 반5합니다.

Load = <server_load>:

이는 서비스! 접속되는 최소한으로 로드된 comm server를 판별할 때 사k되는 로드 U형 수량입니다. 유효한 *의 범위는 0!서 100n지의 정수이m, 0은 최저치 로드를 나타내며 100은 최m치를 나타냅니다.

LU Pool = <pool_name>,
<pool_name>/t<dev-type>,
<pool_name>/t<dev_type>, ...
<pool_name>/t<dev-type>

이 서비스!서 사k할 수 있는 LU 풀의 LU 풀 이름을 " 풀!서 지x되는 |련 장치 유형z 함께 식별합니다. " *은 첫 번째 토큰이 풀의 풀 이름이m 두 번째 토큰은 W 풀!서 지x되는 장치 유형인 레코드입니다. 장치 유형이 x는 풀 이름은 K 수 x는 유형의 LU! 풀! 포함됨을 나타냅니다. 주n진 풀 이름z |련된 레코드는 지x되는 " 장치 유형마다 반복됩니다. 주n진 풀은 최소한 하나의 LU를 풀! 제x하는 PU 프로파일이 서버!서 활동중일 때 등록 d청! 포함됩니다. 유효한 dev_types의 범위는 다음z 0습니다.

표 35. CM_CSLIST_GETII_ERR x시

dev_type	의미
3270002	Lu 유형 2 모델 2
3270003	Lu 유형 2 모델 3
3270004	Lu 유형 2 모델 4
3270005	Lu 유형 2 모델 5
3270DSC	프린터 LU

주n진 장치 유형은 W 유형으로 8성된 LU! 서버의 활동중인 PU 프로파일! 들n 있을 때 등록 d청! 포함됩니다.

종속 LU 서비스 등록 메세지

URL: service:cs3270://<addr-spec>:<port-number>

속성:

[(SCOPE=<string>),]
(RELEASE=06.00.00),

```
(PLATFORM=NT),
(PROTOCOL=IP),
(SERVERNAME=<string>),
(LOAD=<integer 0 to 100>),
[(LUPOOL=pool-name0/tANY,
pool-name1/tdevice_type1,
pool-name2/tdevice-type2, ...
pool-namen/tdevice-typen)]
```

TN3270 서비스 템플리트

tn3270 서비스는 TN3270 프로토콜을 통해 SNA 네트v크로 3270 T이트~이 W세스 G한을 제x합니다. 속성은 3270 장치 유형, LU 풀 W리m 서버!서 사k할 수 있는 로드 정보를 반5합니다. LU 풀z 로드 속성은 서비스 유형 cs3270!서M 0습니다.

BIND, DATA, RESPONSES, SCS, SYSREQ

이런 키v드 속성은 이 서비스!서 지x하는 TN3270e b능을 설명합니다.

BIND 서버는 SNA 바인드 이미지 b능을 지x합니다.

DATA
비 SNA 3270 데이타 스트림은 서버!서 지x됩니다.

RESPONSES
서버는 SNA 응답 모드를 지x합니다.

SCS 서버는 SNA 3270 SCS 데이타 스트림을 지x합니다.

SYSREQ
SYSREQ 키보드 키는 서버!서 지x됩니다.

)b서 설명된 b능을 사k할 수 있을 fl 서비스 선전! 포함됩니다.

```
Security = <security>
```

이 필드!는 서버!서 지x하는 보H b술이 있습니다. 정의된 *은 다음z 0습니다.

NONE 이 서버!는 확실한 보H b술이 x습니다.
SSLV3 이 서버는 Secure Socket Layer 버전 3 표준을 지x합니다.
Ciphersuites = <CipherSpec>,
<CipherSpec>, ...
<CipherSpec>
이 서버!서 지x하는 O호 스펙을 식별합니다. 정의된 *은 다음z 0습니다.
NULL_NULL
NULL_MD5
NULL_SHA
RC4_MD5_EXPORT
RC4_MD5_US

RC4_SHA_US
RC2_MD5_EXPORT
DES_SHA_EXPORT
TRIPLE_DES_SHA_US

RFC1576, RFC1646, RFC1647

서비스 ! 서 지 x 하는 문서의 특징인 RFC 번호. TN3270 ! 대한 현재 RFC ! 는 1576, 1646 W 리 m 1647이 포함됩니다.

TN3270 서비스 등록 메세지

URL: service:tn3270://<addr-spec>:<port-number>

속성:

[(SCOPE=<string>),]
(RELEASE=06.00.00),
(PLATFORM=NT),
(PROTOCOL=IP),
(SERVERNAME=<string>),
(LOAD=<integer 0 to 100>),
[(LUPOOL=pool-name(0)/tANY,
pool-name1/tdevice_type1,
pool-name2/tdevice-type2, ...
pool-namen/tdevice-typen)]
BIND,
DATA,
RESPONSES,
SCS,
SYSREQ,
(SECURITY=NONE),
(SECURITY=<security>),
(CIPHERSUITES=<Spec1,Spec2,...Specn>),
RFC1576,
RFC1646,
RFC1647

TN5250 서비스 템플리트

tn5250 서비스는 TN3270 프로토콜을 통해 SNA 네트 v 크 ! 대한 5250 T 이 트 ~ 이 W 세스 G 한을 제 x 합니다. 속성은 W 세스 ! 능한 AS400의 서비스 M 서버 ! 서 사 k 할 수 있는 로드 정보를 반 5 합니다.

Release = <release>

선전 commserver의 버전 및 릴리스입니다.

Protocol = <protocol>

이 서비스를 제x하는 서버!서 지x되는 프로토콜입니다. 정의된 *은 다음z 0습니다.

```
IP - Server supports connections over IP (TCP/IP or UDP/IP)
Platform = <platform>
```

선전 서비스의 b초! 되는 네트v크 n5체제 플랫폼입니다. 정의된 *은 다음z 0습니다.

```
IW - Server uses the IntranetWare network operating system
NT - Server uses the Microsoft NT Operating system
Server Name = <server name>
```

설치중! 8성된 서버의 이름입니다.

```
AS400 Name = <host name>
```

이 서비스 등록이 적k되는 AS400 호스트의 이름입니다.

```
Load = <INTEGER>
```

최소한으로 로드된 통신 서버를 판별할 때 사k되는 로드 U형 수량입니다. 유효한 *의 범위는 정수 0!서 100입니다.

```
Security = <security>
```

이 필드!는 서버!서 지x하는 보H b술이 있습니다. 실제 *은 다음z 0습니다.

NONE 이 서버!는 확실한 보H b술이 x습니다.
SSLV3 이 서버는 Secure Socket Layer 버전 3 표준을 지x합니다.
Ciphersuites = <CipherSpec>,
<CipherSpec>, ...
<CipherSpec>
이 서버!서 지x하는 0호 스펙을 식별합니다. 정의된 *은 다음z 0습니다.
NULL_NULL
NULL_MD5
NULL_SHA
RC4_MD5_EXPORT
RC4_MD5_US
RC4_SHA_US
RC2_MD5_EXPORT
DES_SHA_EXPORT
TRIPLE_DES_SHA_US

```
Function = <function>
```

이 필드!는 서버!서 지x하는 TN5250 b능이 있습니다. 현재 정의된 b능이 x습니다.

```
RFC1205
```

서비스 ! 서 지 x 하는 문서의 특징인 RFC 번호. TN5250 ! 대한 현재 RFC ! 는 1205 ! 포함됩니다.

TN5250 서비스 등록 메세지

URL: service:tn5250://<addr-spec>:<port-number>

속성:

(SCOPE=<string>),

(PROTOCOL=<string>),

(RELEASE=<string>),

(PLATFORM=<string>),

(LOAD=<integer 0 to 100>),

(SECURITY=NONE),

(SECURITY=<security>),
(CIPHERSUITES=<Spec1,Spec2,...Specn>),
(FUNCTIONS=NONE),

(RFC1205),

(SERVERNAME=<string>),

(AS400NAME=<string>),

LU6.2 서비스 템플리트

csappc 서비스 유형은 SNA APPC W세스 G한을 제 x 합니다. 8 성된 로컬 LU 정의는 이 서비스M 함께 등록됩니다.

LLU = <llu1>,<llu2>,...,<llun>

유효한 로컬 LU를 comm 서버 ! 8 성된 대로 지정합니다.

LU6.2 서비스 등록 메세지

URL: service:csappc://<addr-spec>:<port-number>

속성:

[(SCOPE=<string>),]

(RELEASE=06.00.00),

(PLATFORM=NT),

```
(PROTOCOL=IP),

(SERVERNAME=<string>),

(LOAD=<integer 0 to 100>)

[,(LLU=<llu1>,<llu2>,...,<llun>)]
```

부록E. DLL 버전 정보

32 비트 Windows DLL

다음 32 비트 Windows DLL ! 는 DLL의 버전을 판별할 때 사 k 할 수 있는 정보 ! 들 n 있습니다.

- E32APPC.DLL
- WAPPC32.DLL
- WCPIC32.DLL
- WINCSV32.DLL
- WINMS32.DLL
- WINNOF32.DLL
- WINRUI32.DLL
- WINSLI32.DLL

사 k 할 수 있는 키는 다음 z 0 습니다.

- CompanyName
- LegalCopyright
- LegalTrademarks
- ProductName
- ProductVersion
- FileDescription
- InternalName
- FileVersion

주: 모든 키는 ″\StringFileInfo\040904E4\″ 버전 블럭의 한 부분이 m 변환되지 J 습니다.

다음 z 0 이 프로 W 램을 사 k 하 E 나 Windows Explorer 또는 Windows NT Explorer를 사 k 하) 정보를 K 색할 수 있습니다.

1. @ 른쪽 마 l 스 버튼으로 DLL을 선택합니다.
2. 팝 w 메뉴 ! 서 **등록정보**를 선택합니다.
3. **버전** 탭을 선택합니다.

이 정보를 사 k 하) DLL이 IBM 또는 다른 회사(CompanyName) ! 서 T 는지 W 리 m DLL이 SNA API 클라이 p 트 또는 서버(ProductName) ! 대한 M 인지를 판별하는 코드를 작성할 수 있습니다. 설치된 DLL 버전(FileVersion) z 설치된 제품 버전(ProductVersion)을 판별할 수 있습니다.

다음 샘플 C 함수는 IBM ! 서 명명된 DLL을 생산했는지를 판별합니다.

```
//
//  함수는 주어진 경N명이 버전화된 IBM DLL일 때만 TRUE를 리턴합니다.
//
#include <winver.h>
#define CMPNY_KEY "\\StringFileInfo\\040904E4\\CompanyName"
BOOL bDllFromIBM(char *pcDllPathname)
{
  DWORD  dwBufSize = 0, dwTemp = 0, dwReturnBytes = 0;
  LPVOID pReturnBuffer = NULL;
  VOID  *pVInfoBuffer = NULL;
  BOOL   bRC = FALSE;
 // 매개변수가 널(null)이 아니라는 것은 검증하십시오.
 if (!pcDllPathname || !*pcDllPathname)
    return FALSE;
 // 버전 정보의 크기를 얻으십시오.
 dwBufSize = GetFileVersionInfoSize(pcDllPathname, &dwTemp);
 // 어떤 버전 정보도 틀린 매개변수나 버전화되지 않은
 // IBM DLL을 의미하지 않습니다.
 if (!dwBufSize)
    return FALSE;
 // 버전 정보에 대해 버퍼를 할당하십시오(안전하게 +50).
 pVInfoBuffer = malloc(dwBufSize + 50);
 // malloc 실패
 if (!pVInfoBuffer)
    return FALSE;
 // 버전 버퍼를 채우십시오.
 bRC = GetFileVersionInfo(pcDllName,dwTemp,dwBufSize,pVInfoBuffer);
 // 호출 실패.
 if (!bRC)
    return FALSE;
 // 회사명을 얻으십시오.
 bRC = VerQueryValue(pVInfoBuffer, TEXT(CMPNY_KEY), ReturnBuffer, ReturnBytes);
 // 찾지 못했거나 비어 있습니다.
 if (!bRC || !dwReturnBytes)
    return FALSE;
 // 값은 "IBM"으N 시작해야 합니다.
 if (strncmp(pReturnBuffer, "IBM", strlen("IBM")) == 0)
    return TRUE;
 return FALSE;
}
```

부록F. 주의사항

이 정보는 미9!서 제x되는 제품z 서비스를 위해 3발되z습니다. IBM은 이 문서!서 설명된 제품, 서비스, 특징을 다른 나라!서 제x하지 J을 수도 있습니다. 해당 지*!서 현재 사k할 수 있는 제품z 서비스! 대한 정보를 rm 싶으면 W 지*의 IBM 대표부로 문의하십시@. IBM 제품, 프로W램, 또는 서비스! 대한 참조는 IBM 제품, 프로W램 또는 서비스만 사k해_ 된다m 주장하E나 의미하는 M은 F닙니다. IBM의 지적 재산G을 침해하지 J는 b능적으로 동등한 모든 제품, 프로W램, 서비스를 대신 사k할 수 있습니다. W러나 IBM 제품, 프로W램, 서비스! F닐 때는 작w을 평!하m K증하는 M은 사k자의 책임입니다.

IBM은 이 책!서 다루m 있는 주제! 대한 특허를 보유하m 있E나 현재 출x중일 수 있습니다. 이 책자를 제x했다m 해서 특허G을 부)하는 M은 F닙니다. 특허 사kG! 대한 문의는 한9 IBM 지적 재산G부(02-781-6028)로 하시b 바랍니다.

다음 규정은 해당 지역 법z 일치하지 않는 영국이나 기타 나라에서는 적용되지 않습니다. IBM은 적법성이 O시된 보증, 상w성 또는 특정 목적! 대한 적합성을 포함하지만 이! 9한하지 J은 W n떤 종류의 표현되E나 O시된 보증 x이 이 책을 『있는 W대로』 제x합니다. 일부 지*!서는 특정 E래! 있n서 표현되E나 또는 O시된 보증의 포b를 허k하지 J으므로 이 내k은 사k자!T 적k되지 J을 수도 있습니다.

이 정보!는 b술적인 부정확성 또는 인쇄 @류! 들n 있을 수 있습니다.)b! 있는 정보는 정b적으로 변f되며 W 내k은 이 책의 새로n 판! 포함됩니다. IBM은 이 책!서 설명한 제품 또는 프로W램! 대해 K리지 Jm p제든지 3선하E나 또는 변f할 수 있습니다.

(1) 독자적으로 작성된 프로W램z b타 프로W램(이 프로W램을 포함하))#의 정보 3환이나 (2) 3환된 정보의 상호 이k을 목적으로 이 프로W램! 대한 정보를 필d로 하는 사kG자는 한9 IBM 소프트~n 사w본부(02-781-7777)로 문의하십시@.

이러한 정보는 해당 조G! 따라 사k이 !능하며 특정 fl!는 d]을 지불해_ 합니다.

이 정보!서 설명된 x인 프로W램z 사k!능한 모든 x인 자료는 IBM m4 h`, 9제 프로W래밍 x인 h` 또는 이M 동등한 모든 h`의 조G하!서 IBM! 의해 제x됩니다.

비 IBM 제품! 대한 정보는 W 제품의 x^자, 출판된 발표문 또는 b타 x식적으로 사k !능한 소스!서 rz습니다. IBM은 이런 제품을 테스트하

지 J R으므로 비 IBM 제품z |련된 성능의 정확성, 호환!능성 또는 b타 주장을 확인할 수 x습니다. 비 IBM 제품 b능! 대한 문의는 이런 제품의 x^자!T 하십시@.

COPYRIGHT LICENSE:

이 정보!는 다g한 n5 플랫폼!서 이루n지는 프로W래밍 b술을 설명하는 소스 pn로 된 샘플 n플리케이션 프로W램이 있습니다. 샘플 프로W램이 작성된 n5 플랫폼! 대한 n플리케이션 프로W래밍 인터페이스를 준수하는 n플리케이션을 3발, 사k, 마케팅 또는 분배할 목적으로 IBM!수수료를 내지 Jm 이런 샘플 프로W램을 모든 형태로 복사, 수정 W리m분배할 수 있습니다. 이런 9제는 모든 조G하!서 O전히 테스트되지는 JR습니다. W러므로 IBM은 이런 프로W램의 신뢰성, 유k성 또는 b능을 보증하E나 O시할 수 x습니다.

샘플 프로W램의 사본마다 또는 W 일부나)b서 시작된 모든 작w!는 다음 저작G K림이 b재되n_ 합니다.

(c) (사k자 회사명) (,도) 이 코드 부분은 IBM사의 샘플 프로W램!서 나B M입니다.
(c) Copyright IBM Corp. All rights reserved.

부록G. 등록상표

다음 k n는 미9 z W 밖의 나라!서 사k되는 IBM의 등록상표입니다.

ACF/VTAM	IMS
Advanced Peer-to-Peer Networking	MVS/ESA
AFP	MVS/XA
AIX	NetView
AIXwindows	Operating System/2
Application System/400	OS/2
APPN	OS/400
AS/400	RACF
CallPath	SAA
CallPath/2	SP
CallPath SwitchServer/2	System/370
CICS	S/370
Common User Access	Virtual Machine/Enterprise Systems Architecture
CUA	VM/ESA
IBM	VTAM

b타 회사, 제품 및 서비스명(이중 별표(**)! 붙n 있는 f l도 있음)은 다른 회사의 등록상표 또는 서비스 표시입니다.

C-bus는 Corollary사의 등록상표입니다.

ActionMedia, LANDesk, MMX, Pentium W리m ProShare는 미9 및 다른 나라! 있는 Intel사의 등록상표입니다.

Java 및 HotJava는 Sun Microsystems사의 등록상표입니다.

Microsoft, Windows, Windows NT W리m Windows 95 로m는 Microsoft사의 등록상표입니다.

UNIX는 X/Open Company Limited를 통해 독점 x인되z으며 미9 z W 밖의 나라!서 사k되는 등록상표입니다.

PC Direct는 Ziff Communications사의 등록상표로서 라이센스하!서 IBM사!의해 사k됩니다.

색인

[바]

[사]

[아]

[자]

[타]

[파]

[하]

[숫자]

A

B

C

D

F

G

I

L

N

R

S

T

U

W

SA30-0536-01