eNetwork 通 E 服务器 f 本 **6.0 Windows NT** f 和 **eNetwork** 个人通 E f 本 **4.2 Windows 95** f 和 **Windows NT** f

客户机/服务器通 E 程 r h F

SC84-0692-00

eNetwork 通 E 服务器 f 本 **6.0 Windows NT** f 和 **eNetwork** 个人通 E f 本 **4.2 Windows 95** f 和 **Windows NT** f

客户机/服务器通 E 程 r h F

SC84-0692-00

" b!

在 9 C > i 以及 | 支 V D z 品之 0， k 确认阅读 K Z 365 页 D 『 = < F. 注意 B 项』中 D 一 c 信息。

第二 f (1998 j 7 月)

C 级 p J C Z IBM eNetwork 通信服务器 f > 6.0, v 人通信 f > 4.2 Windows 95 和 Windows NT f 以及 y P f 后 D 发行 > 和修 D >，直至新 D f > 中 m P y w 为止。

? <

图 xi

表 xiii

关Z本书 xv
> i D阅读对象 xvi
如何9C > i xvi
图 j xvii
> i中y 9 C D约定. xvii
文>约定 xvii
}字约定 xvii
+字Z字符集支V xviii
何处I i R =详细信息 xviii

第1?分 APPC API 1

第1章 }入 APPC 5
SNA 通信支V 5
SNA LU 类型 6.2 支V 6

第2章 APPC 基本概n 7
2 4 G B务L序? 7
APPC B务L序 7
CPI 通信B务L序 7
M户机B务L序 8
服务器B务L序 8
2 4 G _辑%元? 8
LU 类型 8
从t和独" LU 9
2 4 G LU { F?. 9
2 4 G会话?. 9
2 4 G对话?. 10
会话、对话和 LU 之间D关系 11
对话类型 12
3 d D会话 12
基>会话 12
APPC Y作>例 13
APPC 对话类型 13
%向对话 13
确认-传M对话 13
i询对话 14
}] b |新对话 14
含错对话 15
*要 15

第3章 使C , S管m器 17
x p & C L序和B务L序 18
B务L序定义 18
在= (机器O j 6 B务L序{ F 19

定义对话 t 性 19
同 = 化级 p 19
对话类型和风 q 20
对话风 q 20
x 人分配 k s D对话 2 全性 21
d v 分配 k s D对话 2 全性 21
9 C v 人通信 k 通信服务器O D ， S 管理器 21
启动 ， S 管理器 21
C ， S 管理器启动L 序 22
C RECEIVE_ALLOCATE 动词匹配 x 人分配 k s 22
非排队 L 序 22
排队D L 序 23
在通信服务器 SNA API M户机O 9 C ， S 管理器 25
定义 SNA API M户机D B 务 L 序 25
启动 SNA API M户 ， S 管理器 25

第4章 编 4 事务程 r 27
& C L 序协议 27
I C D L 序 LU 6.2 服务 27
选择对话类型 29
对话类型D一致性 30
发M }] 30
S U }] 31
(f 错误和异 # 终止 31
发M错误日志 }] 记 < 31
I Z ， 1 引起D异 # 终止 32
k s 确认 32
在 k + 工和全 + 工对话间选择 32
选择 B 务 L 序 { 32
9 C 2 全性 X 性 33
伙 i LU 验证（会话级 2 全性） 33
最终 C 户验证（对话级 2 全性） 33
在 EBCDIC 和 ASCII 间转换 33

第5章 实现 APPC 事务程 r 35
` 写 B 务 L 序 35
\ 支 V D 选项 h 置 35
全 + 工 VCB 37
排队级无阻塞 37
缺 ！ > X LU 39
QEL/MU 支 V 39

第6章 实现 CPI-C 程 r 41
` 写 CPIC L 序 41
CPI-C f > 41
CPI-C 一致性类支 V 42
CPI-C 函 } 46
指定服务 TP { F 48

第7章 APPC 入口点 49
APPC 50
WinAsyncAPPC( ) 51
WinAsyncAPPCEx( ) 53

WinAPPCCancelAsyncRequest( ) 55
WinAPPCCancelBlockingCall( ) 56
WinAPPCCleanup( ) 57
WinAPPCIsBlocking( ) 58
WinAPPCStartup( ) 59
WinAPPCSetBlockingHook( ) 60
WinAPPCUnhookBlockingHook( ) 61
GetAppcConfig( ) 62
GetAppcReturnCode( ) 63

第8章 APPC 动词 65
动词X制 i 65
公共字段 65
APPC API 支V 66
动词支V 66
GET_TP_PROPERTIES 67
GET_TYPE 69
RECEIVE_ALLOCATE 71
TP_ENDED 74
TP_STARTED 76
[MC_]ALLOCATE 78
[MC_]CONFIRM 83
[MC_]CONFIRMED 87
[MC_]DEALLOCATE 89
[MC_]FLUSH 94
[MC_]GET_ATTRIBUTES 96
[MC_]PREPARE_TO_RECEIVE 99
[MC_]RECEIVE_AND_POST 102
[MC]RECEIVE_AND_WAIT 107
[MC_]RECEIVE_EXPEDITED_DATA 113
[MC_]RECEIVE_IMMEDIATE 117
[MC_]REQUEST_TO_SEND 122
[MC_]SEND_CONVERSATION 124
[MC_]SEND_DATA 128
[MC_]SEND_ERROR 132
[MC_]SEND_EXPEDITED_DATA 136
[MC_]TEST_RTS 139
[MC_]TEST_RTS_AND_POST 141

第2？分 LUA API 143

第9章 IBM 常规 LU & C程 r 的基本概 n 147
理 b LUA 和 SNA 147
，S 能& 147
LUA &C L 序 147
LUA 动词 147
LU、> X LU 和伙 i LU 148
系统服务X制 c (SSCP) 148
SNA c 148
}] 4 7 X制 c 148
7 6 X制 c 148
传 d X制 c 149

}] w X制 c 149
m > 服务 c 149
9 C SNA 会话 149
SNA 会话D先 v u 件 149
启动会话 150
在 LU-LU 会话O传M }] 150
停止会话 151
断 * 主机 4 S 151
消息号. 152
重新启动和重新同 = 会话 152
9 C协议来X制 k s 和响& 152
9 C w = 协议 152
9 C k + 工 y C/触发器协议 153
9 CW c 协议 153
9 C }] 4 S协议 154
}] ; 换X制方法 154
w 动协议 154
响& 方 = ... 155
LUA 关 * m q 155
异 # 响& k s (RQE) 155
会话E要 156
TS E要 156
FM E要 156
9 C RUI LUA 动词 157
动词 * 要 157
RUI 会话. 158
发 v RUI 动词 158
异 = 动词完 I 159
样例 LUA 通信序 P 159
BIND 校验 161
否定响& 和 SNA 检 b k 161
w = ... 162
分段 162
礼 Z 确认 162
e } }] 至 4 a x 163
配置 163
LUA LU X(I 选D) 163
SNA API M户机 < G 163

第10章 RUI LUA 动词特 T 165
处理异 # k s 165
| D动词记 < 165
处理方括号 k s \ x 166
最小化 LAN 通信 ? 166
减Y RUI_BID D 9 C 166
处理挂起 166
取消 RUI_INIT 167
取消 RUI_WRITE 167
取消 RUI_READ 167
确 # 动词完 I 167
压 u }] 167
协L ? v 会话 }] 压 u D规则 167

从会话 ' \ 中恢 4 168

第11章 实现 LUA 程 r 171
` 写 LUA L 序 171
w C LUA 服务 171
理 b 动词记 < 内容 172
多 x L 172
多线 L 172
LUA 动词传 M 172
从 ASCII 转换为 EBCDIC 173

第12章 RUI LUA 入口点 175
RUI() 176
WinRUI 177
WinRUICleanup() 178
WinRUIGetLastInitStatus() 179
WinRUIStartup() 182
GetLuaReturnCode() 183

第13章 RUI 动词 185
LUA 动词 X 制 i q = ... 185
公共动词头 185
RUI_BID }] a 构 189
RUI_BID 190
RUI_INIT 195
RUI_PURGE 199
RUI_INIT_STATUS 202
RUI_READ 203
RUI_TERM 209
RUI_WRITE 212

第14章 SLI 入口点 219
SLI() 220
WinSLI 221
WinSLICleanup() 222
WinSLIStartup() 223

第15章 SLI 动词 225
SLI_BID 226
SLI_CLOSE 231
SLI_OPEN 234
SLI_PURGE 240
SLI_RECEIVE 242
SLI_SEND 247
SLI_BIND_ROUTINE 251
SLI_STSN_ROUTINE 253
SLI_SDT_ROUTINE 255

第3 ? 分 公共服务 API 257

第16章 公共服务入口点 259
` 写公共服务 L 序 259
ACSSVC 260

WinCSV() 261
WinCSVCleanup() 262
WinAsyncCSV() 263
WinCSVStartup() 264
GetCsvReturnCode() 265
TrnsDt 266

第17章 公共服务动词 (CSV) 271
GET_CP_CONVERT_TABLE. 272
CONVERT 276

第4?分 EHNAPPC API 279

第18章 EHNAPPC &C程r S口. 283
`写 EHNAPPC L序 283
EHNAPPC 例行L序 283
EHNAPPC_Allocate 283
EHNAPPC_Confirm 284
EHNAPPC_Confirmed 285
EHNAPPC_Deallocate 285
EHNAPPC_ExtendedAllocate 286
EHNAPPC_Flush 287
EHNAPPC_GetAttributes 288
EHNAPPC_GetCapabilities 288
EHNAPPC_GetDefaultSystem 289
EHNAPPC_IsRouterLoaded 289
EHNAPPC_PrepareToReceive 290
EHNAPPC_QueryConfiguredSystems 290
EHNAPPC_QueryConvState 291
EHNAPPC_QueryFullSystems 291
EHNAPPC_QueryUserid. 292
EHNAPPC_QuerySystems 292
EHNAPPC_ReceiveAndWait 293
EHNAPPC_ReceiveImmediate. 294
EHNAPPC_RemoteProgramStart 295
EHNAPPC_RqsToSend 295
EHNAPPC_SendData. 296
EHNAPPC_SendError 297
EHNAPPC_StartHostProgram 297
EHNAPPC Structures 298
AS400_SYS 298
appcrtrcap_hdr 298
appcrtrcap_mult. 299
appcrtrcap_query 299
EHNAPPC API D返回k 300
在 Windows 95 和 Windows NT O运行 16 位 EHNAPPC L序 301

第19章 数]变换 Windows &C程r S口 303
}]d换 Windows API 例行L序. 303
EHNDT_ANSIToEBCDIC 303
EHNDT_ASCIIToEBCDIC 304
EHNDT_EBCDICToANSI 304

EHNDT_EBCDICToASCII 305

第5？分 Java 程 r h F S口 307

第20章 i \ 主机访问 ` 库(Host Access Class Library) Java f 309
2 4 G ECL? 309
ECL E念 310
会话 310
容器对象 310
P m对象 310
B件 310
错误处理 311
寻址(行、P、位置) 311
在通信服务器 NT f D服务器O 2 装 ECL 311
在通信服务器 NT f D 32 位 Windows M户机O 2 装 ECL 312
h置 Classpath 313
ECL 代 k 页转换器 313
ECL 样 > 313
Host Launchpad 样 > 313
其 | 样 > 315

第21章 使C CPIC-C Java f 317
何谓 CPI-C Java f ? 317
2 装 CPI-C Java f 317
CPI-C Java f 样 > 318
M户机样 > 318
服务器样 > 320

附 < A. APPC 公共返回 k 323

附 < B. LUA 动词返回 k 327
主返回 k 327
次级返回 k 328

附 < C. APPC 对话 4 态 * F 343

附 < D. 通 E 服务器服务定位 - i 347
发现和 : 载 y 衡 API 347
a 构 347
方 8 347
DA-Discovery ， 1 354
SA 多次 " T ， 1 354
管理员 o 助信息 354
r 354
如何 9 C r ?. 354
：载 y 衡加权因子 355
服务模 e 355
通信服务器服务模 e 355
通信服务器服务G记信息 356
从 t LU 服务模 e 356
从 t LU 服务G记信息 357
TN3270 服务模 e 357
TN3270 服务G记信息 358

TN5250 服务模 e 359
TN5250 服务 G 记信息 360
LU6.2 服务模 e 361
LU6.2 服务 G 记信息 361

附 < E. DLL f 本 E 息 363
32 位 Windows DLL 363

附 < F. " b **事项** 365

附 < G. L 标 367

w } 369

读者 b { **表** 375

图

1. v 人通信或通信服务器 APPC D 5 现 5
2. = v LU 之间D会话 9
3. 对话D一？分 10
4. = v B务L序间D对话 10
5. LU 之间D " 行会话 11
6. L序和 LU 之间D关系. 11
7. APPC 中D，S管理器功能 17
8. b T动词完 I 167

表

1. LU 6.2 Y作. 13
2. %向对话D Y作 13
3. 确认-传M对话D Y作 13
4. i 询对话D Y作 14
5. }] b | 新对话D Y作 14
6. i 询对话D错误 15
7. 动词处理和B 务L 序{ F配置 24
8. 对话状， 0
9. APPC D头文件和b 35
10. CPIC D头文件和b 41
11. v 人通信k 通信服务器 CPI-C 函} D M户支V 46
12. RQE D e } 155
13. TS E要X性 156
14. FM E要X性 157
15. RUI 动词u 件 159
16. Y作系统D RUI API D头文件和b 171
17. SLI API D头文件和b 171
18. 基Z 信息类型D N } h 置 249
19. Y作系统D头文件和b 259
20. Y作系统D头文件和b 283
21. 返回k 300
22. APPC D头文件和b 310
23. APPC k + 工对话状，转移 343
24. APPC 全 + 工对话状，转移 345
25. 服务类型/端Z 信息 349
26. CM_CSLIST_GETII 原o Y罪RUI API o . . . . . j 1 0 T D (D) T

2 5 纺~熄熄巷烯显详席铣像

关 Z 本书

> i k 对 IBM eNetwork 通信服务器（Windows NT f ）和 IBM eNetwork v 人通信（Windows 95 和 Windows NT f ） y a 供 D M 户机和服务器 & C L 序 D C 户。M 户机 API G 为 Windows 95 和 Windows NT、 Windows 3.1 以及 OS/2 平 (y a 供。

IBM eNetwork 通信服务器 Windows NT f （ > i 中指*通信服务器*）G 一 v 通信服务平 (。C 平 (为 k 主机和其 | 工作 > 通信 D Windows NT 工作 > a 供一系 P 广泛 D 服务。通信服务器 C 户 I 以从 w 种 w 样 D 远 L ，通性选项中选择。

IBM eNetwork v 人通信 Windows 95 和 Windows NT f （指 > i 中 *v 人通信*）G 一 v 全功能仿 f 器。 } K 主机终端仿 f , | 还 a 供以下 P C 功能:

- 文件传 d
- 动 , 配置
- 易 Z 9 C D 图形 g f
- 基 Z SNA D API M 户 & C L 序
- API 允许基 Z TCP/IP D & C L 序在基 Z SNA D 网 g O 通信。

在大多 } 5 例中， * 发 C Z v 人通信和通信服务器及其 M 户机 D L 序非 # 类 F，因为 | G ? v 都支 V 许多相同 D 动词。 + 仍存在一些 n 异。 b 些 n 异在 > i 中 C X b D 图 j m > ; 详细 i v k N 阅 Z xvii 页 D 『图 j 』。在 { > i 中， 1 信息同 1 C Z b = _ 1 k 9 C v 人通信和通信服务器 L 序 { 。 1 只 P v 人通信 L 序或只 P 通信服务器 L 序 & C 1， M 9 C X 定 D L 序 { 。

> i 分为 D ? 分。

- Z 1 ? 分 APPC API， h v 如何 * 发 9 C v 人通信和通信服务器 _ 级 L 序间通信（APPC） S Z D L 序。 APPC 指 _ 辑 % 元（LU）类型 6.2 D 系统网 g e 系 a 构（SNA）D 5 现。在 { > i 中， } 非 m P 5 w，否则 APPC m > APPC D v 人通信和通信服务器 5 现。

  APPC a 供 = v 或多 v L 序合作执行处理功能 D 分 < = B 务处理能 &。C 能 & f 及 L 序间 D 通信以 9 | G 能够共享资源，如处理器周期、 }] b 、工作队 P 和物理 S Z （如键盘和显 > 器)。
- Z 2 ? 分 LUA API， h v 如何 * 发 9 C IBM # 规 _ 辑 % 元 & C L 序（LUA） S Z (在 > i 中 LUA 也指 k s % 元 S Z {RUI} ）D L 序， C S Z a 供对 SNA LU 类型 0、1、2 和 3 D 访问。
- Z 3 ? 分 公共服务 API， | 括 (" 公共服务 API D 动词。
- Z 4 ? 分 EHNAPPC API， | 括增 ? 型 APPC S Z D 功能、 a 构和返回 k 。

在 > i 中， *Windows* 指 Windows 95 和 Windows NT。在 { > i 中， *windows* 指 y P \ 支 V D v 人计 c 机。 1 只指某一 v 人计 c 机模 = 或 e 系 a 构 1 ，则 + w 确 5 w C 类型。

对 Z 通信服务器，假 h 您 } 9 C NT V4.0 作为基 > Y 作系统。对 Z v 人通信，假 h 您 } 9 C Windows 95 或 Windows NT 作为基 > Y 作系统。

本书的阅读对象

> i k对Z }在`写 APPC 或 LUA &C L序D L序员和* 发_。

> i 假h读_ l悉K *SNA B务处理L序员N < V a（LU 类型 6.2）（SNATransaction Programmer's Reference Manual for LU Type 6.2）*。

如何使C本书

- Z1B 引入 APPC, h v _级L序间通信（APPC）。
- Z2B APPC 基> E念, h v APPC B务L序。
- Z3B 9C，S管理器, h v如何9C，S管理器。
- Z4B `写B务L序, h v如何`写B务L序。
- Z5B 5现 APPC B务L序, h v APPC 扩9{。
- Z6B 5现 CPI-C L序, h v CPI-C L序。
- Z7B APPC 入Z c, h v APPC API D过L入Z c。
- Z8B APPC 动词, h v ? v APPC 动词D o法。|括# t ? v动词信息D a构= 4, h v ? v入Z, "带P一P I能D返回k。
- Z9B IBM #规 LU &C L序D基> E念, h v > i中D基> LUA L序h计E念。
- Z10B RUI LUA 动词X性, h v LUA 动词D功能。
- Z11B 5现 LUA L序, h v `写 LUA &C L序D某些X w。
- Z12B RUI LUA 入Z c, h v LUA D过L入Z c。
- Z13B RUI 动词, h v ? v LUA 动词D细Z。
- Z16B 公共服务入Z c, h v过L入Z c。
- Z17B 公共服务动词 (CSV), h v公共服务动词。
- Z18B EHNAPPC &C L序S Z, h v EHNAPPC API。
- Z19B }] d换 Windows &C L序S Z, h v }] d换窗Z API。
- Z20B i \主机访问类b (Host Access Class Library) Java f, h v Java D主机访问类b及其9C Java 类1 3270 和 5250 = _ D关系。
- Z21B 9C CPIC-C Java f, h v Java API D CPI-C。
- = <A. APPC 公共返回k, |含公共返回k D 5 w。
- = <B. LUA 动词返回k, |含 LUA 公共返回k D 5 w。
- = <C. APPC 对话状，转移, h v I以发v ? v APPC 动词D对话状，以及发z在动词完I 1 D状，|D。
- = <D. 通信服务器服务定位协议, h v &C L序* 发_现在如何E能R =服务" 9C TCP/IP 协议：载服务间D y衡。
- = <E. DLL f >信息, |含 32 位 Windows DLL f >信息。

图标

> i 9 C 文 > 中 D *图 j* 来 o 助您 R = ；同类型 D 信息。

C 图 j m > & C Z 基 > APPC 动词 D 信息。 P 关基 > 动词 D 详细信息 k N 阅 Z 8 B APPC 动词。

C 图 j m > & C Z 3 d APPC 动词 D 信息。 P 关 3 d 动词 D 详细信息 k N 阅 Z 8 B APPC 动词。

C 图 j m > 一 v 注意 B 项、 I 能 0 响 v 人通信或通信服务器 Y 作 D 重要信息或任务 D 完 I 。

位位置	在最R位（最小P效位）以 0 * <。
. x 制 }	，过 4 v } 字D . x 制 } 以公制m >。 9 C U q 而；G逗号分 t 3 v 一组D } 字。例如， } 字 16147 写为 16 147。
. y x 制 }	在文 > 中以 hex xxxx 或 X'xxxx' m >（『Z	Z c D X址G hex 5D，指定为 X'5d'』）

+ V Z V符 / 支持

v 人通信和通信服务器支V + 字Z字符集（DBCS），其中？v字符I 2 v字Z m >。日o、中文和韩o H o言 | 含D符号H I I 256 v代k c m > v D符号 | 多，需要 + 字Z字符集。因为？v字符需要 2 v字Z， y以 DBCS 字符D d入、显 > 和打！都需要支V DBCS D 2 件和L序。

信息 _ e & C = DBCS D X方在C信息%元中 a =。

ASCII 在 > i 中指 PC %字Z代k。在日 > ASCII & S 为 JISCII。

何处可 i 找到详细E息

P关详细信息， k N阅 *l Y*人 *E*， | 含通信服务器 b 和P关 / 物D完 { h v。

在 2 装K通信服务器后要 i 4 X定D i 籍， k 9 C桌 f D下P 7 6:

1. L序
2. IBM 通信服务器
3. 文 5
4. 从 i 籍 P m中选择

通信服务器 i 籍以 I 移植D文 5 q =（PDF）， I 9 C Adobe Acrobat Reader 阅读。如果您D机器O ； P C L序D = 4， I 以从文 5 P m 2 装。

Internet O D通信服务器主页； v P关Z APAR 及修订 f D服务信息，还P一 c D z 品信息。要 9 C Internet / 览器（如 IBM Web Explorer）= 达主页， k 转至下P URL:

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

P关详细信息， k N阅 *l Y*入 *E*， | 含 v 人通信 b 和P关 / 物D完 { h v。

在 2 装K v 人通信后要 i 4 X定D i 籍， k 9 C桌 f D下P 7 6:

1. L序
2. IBM 通信服务器
3. 文 5
4. 从 i 籍 P m中选择

v 人通信 i 籍以 BookManager q =（BOO）， I 以 9 C IBM Library Reader 阅读。如果您D机器O ； P C L 序D = 4， I 以从 eNetwork v 人通信 CD-ROM 2 装。

Internet O D v 人通信主页； v P关Z APAR 及修订 f D服务信息，还P一 c D z 品信息。要 9 C Internet / 览器（如 IBM Web Explorer）= 达主页， k 转至下 P URL:

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

第1？分 APPC API

第1章 }入 APPC 5
SNA 通信支V 5
SNA LU 类型 6.2 支V 6

第2章 APPC 基本概n 7
２４ＧＢ务Ｌ序? 7
APPC Ｂ务Ｌ序 7
CPI 通信Ｂ务Ｌ序 7
Ｍ户机Ｂ务Ｌ序 8
服务器Ｂ务Ｌ序 8
２４Ｇ_辑%元? 8
LU 类型 8
从ｔ和独" LU 9
２４Ｇ LU ｛Ｆ?. 9
２４Ｇ会话?. 9
２４Ｇ对话?. 10
会话、对话和 LU 之间Ｄ关系 11
对话类型 12
３ｄＤ会话 12
基>会话 12
APPC Ｙ作>例 13
APPC 对话类型 13
%向对话 13
确认-传Ｍ对话 13
ｉ询对话 14
｝］ｂ｜新对话 14
含错对话 15
*要 15

第3章 使Ｃ，Ｓ管m器 17
ｘｐ＆ＣＬ序和Ｂ务Ｌ序 18
Ｂ务Ｌ序定义 18
在=（机器Ｏｊ６Ｂ务Ｌ序｛Ｆ 19
定义对话ｔ性 19
同=化级ｐ 19
对话类型和风ｑ 20
对话风ｑ 20
ｘ入分配ｋｓＤ对话２全性 21
ｄｖ分配ｋｓＤ对话２全性 21
９ＣｖＡ通信ｋ通信服务器ＯＤ，Ｓ管理器 21
启动，Ｓ管理器 21
Ｃ，Ｓ管理器启动Ｌ序 22
C RECEIVE_ALLOCATE 动词匹配ｘ入分配ｋｓ 22
非排队Ｌ序 22
排队ＤＬ序 23
在通信服务器 SNA API Ｍ户机Ｏ９Ｃ，Ｓ管理器 25
定义 SNA API Ｍ户机ＤＢ务Ｌ序 25
启动 SNA API Ｍ户，Ｓ管理器 25

第4章 编 4 事务程 r 27
& C L 序协议 27
I C D L 序 LU 6.2 服务 27
选择对话类型 29
对话类型 D 一致性 30
发 M }] 30
S U }] 31
(f 错误和异 # 终止. 31
发 M 错误日志 }] 记 < 31
I Z , 1 引起 D 异 # 终止 32
k s 确认 32
在 k + 工和全 + 工对话间选择 32
选择 B 务 L 序 { 32
9 C 2 全性 X 性 33
伙 i LU 验证（会话级 2 全性） 33
最终 C 户验证（对话级 2 全性） 33
在 EBCDIC 和 ASCII 间转换 33

第5章 实现 APPC 事务程 r 35
` 写 B 务 L 序 35
\ 支 V D 选项 h 置 35
全 + 工 VCB 37
排队级无阻塞 37
缺 ! > X LU 39
QEL/MU 支 V 39

第6章 实现 CPI-C 程 r 41
` 写 CPIC L 序 41
CPI-C f > 41
CPI-C 一致性类支 V 42
CPI-C 函 } 46
指定服务 TP { F 48

第7章 APPC 入口点 49
APPC 50
WinAsyncAPPC( ) 51
WinAsyncAPPCEx( ) 53
WinAPPCCancelAsyncRequest( ) 55
WinAPPCCancelBlockingCall( ) 56
WinAPPCCleanup( ) 57
WinAPPCIsBlocking( ) 58
WinAPPCStartup( ) 59
WinAPPCSetBlockingHook( ) 60
WinAPPCUnhookBlockingHook( ) 61
GetAppcConfig( ) 62
GetAppcReturnCode( ) 63

第8章 APPC 动词 65
动词 X 制 i 65
公共字段 65
APPC API 支 V 66
动词支 V 66
GET_TP_PROPERTIES 67

GET_TYPE 69
RECEIVE_ALLOCATE 71
TP_ENDED 74
TP_STARTED 76
[MC_]ALLOCATE 78
[MC_]CONFIRM 83
[MC_]CONFIRMED 87
[MC_]DEALLOCATE 89
[MC_]FLUSH 94
[MC_]GET_ATTRIBUTES 96
[MC_]PREPARE_TO_RECEIVE 99
[MC_]RECEIVE_AND_POST 102
[MC]RECEIVE_AND_WAIT 107
[MC_]RECEIVE_EXPEDITED_DATA 113
[MC_]RECEIVE_IMMEDIATE 117
[MC_]REQUEST_TO_SEND 122
[MC_]SEND_CONVERSATION 124
[MC_]SEND_DATA 128
[MC_]SEND_ERROR 132
[MC_]SEND_EXPEDITED_DATA 136
[MC_]TEST_RTS 139
[MC_]TEST_RTS_AND_POST 141

第1章 } 入 APPC

v 人通信 k 通信服务器为工作 > a 供 K _ 级同级间 * 网(APPN)末端 Z c 支 V， 9 | G 能够 | i 活 X k 其 { 网 g 中 D 系统通信。

v 人通信和通信服务器 a 供 K _ 级 L 序间通信(APPC)，支 V 分 < = 处理 L 序(F 为 *B* 务 *L* 序)间 D 通信。 APPN + b 些能 & 扩 9 = * 网环 3 中。 C B 务处理 L 序 I 以在任何支 V APPC 网 g D Z c O。

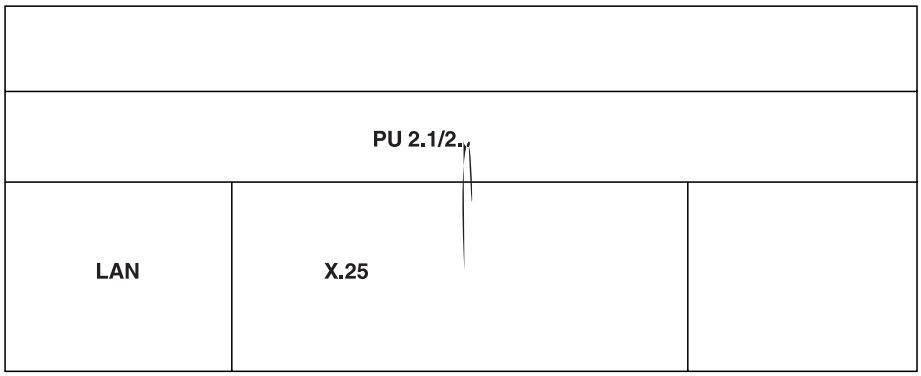
v 人通信和通信服务器在 V r 网(LAN)环 3 D x K APPC D 吞吐 ?，同 1 通过多 v 协议支 V APPC ， 如: IBM n 牌环网 g 、同 = }] 4 7 X 制 (SDLC)、 + 轴，和以 + 网（Ethernet)。

": Z 1 B D B Z | 括 D 内容函 G K P 关 I 下 P 系统 y a 供 D APPC API D 信息:

- 运行在 Windows NT O D 通信服务器
- SNA API M 户机 OS/2 f , Windows NT f , Windows 95 f , 和 Windows 3.1 f , | G 通过通信服务器来传] 。
- v 人通信 Windows 95 和 Windows NT f

1 在 I b 些系统 a 供 D 支 V 之间 P n 异 1， 会 v 现 a > 。

Z 5 页 D 图 1 5 w K 5 现 v 人通信或通信服务器 D APPC D 功能 a 构



SNA 通 E 支持

v 人通信和通信服务器支 V 系统网 g e 系 a 构 (SNA) 类型 2.1 D Z c (| 括 SNA 类型 2.0 和 SNA 类型 2.1, | G 支 V > X % 元 [LU] 而 ; G SNA LU 6.2)。 C 支 V 允许您 ` 写 L 序以和许多其 | IBM SNA z 品通信。

I 以 ` 写 L 序而无须知@网 g D 细 Z。 y P 要知@D 只 G 伙 i LU D { F; 您 ; X 知@ | D 位置。 SNA 会确定伙 i LU D 位置和传 M }] D 最佳 7 6。基础网 g D | D，新 J 配器 D 增加，或机器 D ！动 " ; 0 响 APPC L 序。然而， L 序 I 能需要 (" 通过 P 换 , S D 4 7 , S。

1 v 人通信或通信服务器启动 1， | (" K > X LU 和 _ 辑 4 7 定义， | G 存储在配置文件中。系统管理 & C L 序 h 计 S Z (API) a 供 K X 制配置定义和 J 配器以及 4 7 激活

D功能。 N < *系统管理 L 序 h 计*以 C = b 些功能 D 详细信息。 C 户 I 以在运行 1 9 C 配置和 Z c Y 作功能。 N < *l Y 入 E* 和 *系统管理 L 序 h 计*以 C = 关 Z 配置 Z c Y 作 D 详细信息。

SNA LU ` M 6.2 支持

LU 6.2 G L 序间通信 D e 系 a 构。 v 人通信和通信服务器支 V y P 基 > D LU 6.2 功能。一些 I 选 D SNA LU 6.2 功能 G:

- 基 > 和 3 象会话
- k + 工和全 + 工对话类型
- 确认 D 同 = 化级 p
- 会话和对话 c 次 D 2 全性支 V
- 多 v LU
- " 行会话, | 括 9 C 远 L 系统 | D 会话 } D 能 &。

第2章 APPC 基本概 n

> i + h v v人通信 k 通信服务器支V D APPC API。其目D G为K a 供:

- APPC API a 构D简要E v
- P关w - S Z D动词X定 o 法D N < 信息

什么是事务程 r ?

一 v B务L序G一 v 代 k i , 或G 9 C APPC 通信功能D & C L序D一 ? 分。& C L序 9 C b些功能 k 其 | 支V APPC D系统O D & C L序 x 行通信。B务L序P一 v 64 位 D { 字(**tp_name**)。

B务L序 I 以通过以下任一 API 来获C LU 6.2 服务:

- APPC-- _ 级L序间通信允许B务L序通过 IBM SNA 网 g 来 ; 换信息, C网 g 9 C I IBM 定义D o 法和动词以 9 C LU 6.2 会话。
- CPI-C--通信D公共L序 h 计S Z (CPI-C)允许B务L序 g 越 IBM SNA 网 g 来 ; 换信息, C网 g 9 C I IBM 在 SAA D公共L序 h 计S Z 组件内定义D o 法以 9 C LU 6.2 会话。I Z C API I 在多种平 (O 5) , y 以 CPI-C & C L序能够很方 c X移植。

B务L序发 v APPC 动词来w C APPC 函 } 。关Z B务L序如何发 v APPC 动词D详细内容, k N阅Z 35页D 『 Z 5 B 5 现 APPC B务L序』。B务L序 I 以发 v CPI 通信w C来w C CPI 通信功能。CPI 通信w C 9 & C L序能够利C SNA a 供D一致性。k N阅Z 7页D 『CPI 通信B务L序』I 获取P关 CPI 通信w C D信息。

L序 ; X为K K此之间 x 行通信而对相同D LU 6.2 API x 行 ` 写。X p G, 对 APPC API ` 写D B务L序 I 以和对 CPI-C ` 写D B务L序 x 行通信。

APPC 事务程 r

一 v APPC B务L序 " ; G一 v & C L序; | G一 v & C L序D一 ? 分。% v & C L序 I 以 | 含多 v B务L序。? v B务L序都P一 v 唯一D 8 位 j 6 号(**tp_id**)。

APPC a 供K在& C L序内启动和停止B务L序D动词。APPC 还 a 供K一 { W对话动词, 您 I 以 9 C b些动词来 5 现B务L序D功能。

B务L序向 APPC 发 v 一 v k s (以动词形 =), 来对一 v & C L序执行某些Y作。动词G一 v I B务L序发 v " I APPC 执行D q = 化 k s 。一 v L序 9 C APPC 动词序P k m一 v L序 x 行通信。= v K此通信D L序既 I 以位Z ; 同D系统也 I 以在同一 v 系统中。

1一 v B务L序和m一 v B务L序 ; 换 }] 1 , { G ; F 为*伙 i* B务L序。

CPI 通E事务程 r

CPI 通信B务L序和 APPC B务L序相F ; b = 种类型D B务L序都 9 C K APPC 支V。然而, 一 v CPI 通信B务L序 ; G发 v 动词, 而G通过在w C O传] J 1 N } 来对函 } x 行w C , " 以此来w C ? v CPI 通信函 } D。

大多 } CPI 通信 w C 都和 APPC 动词相互对 &。例如，分配 v > 对话和 S \ (S U)对话 D w C 以及在对话中发 M 和 S U }] D w C，| G a 供 D 功能和相 & D APPC 动词 a 供 D 功能非 # 相 F。只 P 在分配对话之 0 u < 化对话 D w C 以及 h 置和 a 取 v p 对话 X 性 D w C } 外。

k N < *CPI 通信 N* < 大全以 K b P 关通信服务器向 CPI 通信 L 序 a 供 D 支 V。

客户机事务程 r

通 #，一 v L 序启动一 v 对话 D 原因 G 因为 | 需要一 v 来自 m 一 v L 序 D 服务。b v L 序；F 为 M 户机 B 务 L 序。M 户机 B 务 L 序通过 LU 6.2 API 来 k s 对话。

M 户机 B 务 L 序通 # G I C 户(人)来启动 D； + G，M 户机 B 务 L 序 5 际 O I 以 G 一 v 响 & m 一 v L 序 k s D 服务器 B 务 L 序。在任何对话中，M 户机 B 务 L 序总 G 在对话 * < 之 0 M 运行 K。M 户机 B 务 L 序 D 启动和终止 k 对话 " ；直 S 相关。M 户机 B 务 L 序启动对话，而 | 在对话 a x 之后仍 I 继续运行。

服务器事务程 r

服务器 B 务 L 序 a 供 K I M 户机 B 务 L 序 y k s D 服务。

服务器 B 务 L 序 I 以 V 续运行，H 待 M 户机 * < 和 | D 对话。+ 在通 # i v 下，服务器 B 务 L 序处理 % v B 务，" I APPC API 启动以处理一 v X 定 D 对话。1 对话；k s 1 服务器 B 务 L 序 * < 执行，1 对话 a x 1 | 即终止。

LU 6.2 e 系 a 构 D 一 v 重要功能 G: | I 以在 M 户机 B 务 L 序 k s 服务器 B 务 L 序 1 启动 | G。您 I 以 4 此模 = 来 h 计服务器 L 序 " 9 | G 能 4 需启动。

什么是 _ - 单元?

? v B 务 L 序都 + 通过一 v _ *辑 % 元*(LU)来获取对 SNA 网 g D 访问权。LU G 一种 S \ 来自 Z L 序 D 动词 " 对那些动词 x 行 Y 作 D SNA 软件。一 v B 务 L 序向其 LU 发 M APPC 动词。b 些动词 9 C | n 和 }] 能 g 越网 g w 向一 v 伙 i LU。一 v LU 还 I 以 d 1 B 务 L 序和网 g 之间 D = i 以管理在 B 务 L 序间 D }] ；换。% v LU I 以向多 v B 务 L 序 a 供服务。多 v LU I 以；同 1 激活。

LU ` M

v 人通信和通信服务器支 V LU 类型 0、1、2、3 和 6.2。LU 类型 0、1、2 和 3 支 V 主机 & C L 序和 w 种 h 8 (如终端和打！机)之间 D 通信。P 关 ` 写 b 些 L 序 D 详细内容，k N < Z 2 ? 分 LUA API。

LU 6.2 支 V 位 Z 类型 5 子 x Z c、类型 2.1 外围 Z c D = v L 序之间以及 L 序和 h 8 之间 D 通信。APPC G 对 LU 6.2 e 系 a 构 D 5 现，> i D b v ? 分 + 对此 x 行 h v。

通信只会在相同 LU 类型 D LU 之间发 z。例如，一 v LU 2 I 以和 m 一 v LU 2 通信；+ | ；能和一 v LU 3 通信。

从属和独 " LU

一 v 从 *t LU*要依？一 v 系统服务 X 制 c (SSCP)来激活一 v 会话。一 v 从 t LU 需要一 v 活动 D SSCP-LU 会话，从 t LU C | 来启动和一 v 子 x Z c 中 D LU D LU-LU 会话。一 v 从 t LU 在某一 1 L v I 以和子 x LU P 一 v 会话。为 K 和一 v 子 x Z c O D B 务 L 序通信，？v 从 t LU 在同一 1 间 v I 以 P 一 v 对话，" R ？v 从 t LU 在同一 1 间 v 对一 v B 务 L 序 D 通信 h 以支 V。

一 v *独 " LU* ；X 依赖 SSCP 来激活一 v 会话。一 v 独 " LU 支 V 同子 x Z c 中 D 其 | LU x 行多 v " 行会话，y 以您 I 以 P 多 v 会话 " R 支 V 多 v B 务 L 序和子 x Z c B 务 L 序 x 行通信。外围 Z c 间 D LU 也 I 以 9 C b 种支 V。

从 t LU 和 独 " LU 之间 D n p v 在 1 一 v 外围 Z c 中 D LU 和一 v 子 x Z c 中 D LU 之间 V [会话 1 E P 意义。否则，在类型 2.1 外围 Z c (如 = v 工作 > 之间)之间 x 行通信 1，从 t 和独 " LU 都支 V 多 v " 行会话和对话。v 人通信或通信服务器 LU I 支 V k 一 v 从 t LU D % v 会话或 k 一 v 独 " LU D 多 v 会话。

什么是 LU 名称?

一 v LU G 一 v 对系统网 g e 系 a 构(SNA)网 g D 访问 c。 LU 5 P 一 v { F 和其 | 一些在 { v SNA 网 g 中 - *配置*(; } = 记 < D)D X 性。P 1 侯 b 种配置 G 2 ，D D， I 网 g 管理员来执行 " 记 < 在配置文件中。P 1 侯配置 G 动，D， I 来自文件或 C 户 d 人 D 文件来准 8 。

若要打 * 一 v 对话，M 户机 B 务 L 序 X 须指定服务器 B 务 L 序 D { F 以及 I = 达 C 服务器 B 务 L 序 D LU D { F。P 1 侯 b 些 { 字；6 入在 M 户机 B 务 L 序中。其 | i v 下， b 些 { 字则存储在 M 户机 B 务 L 序外或；动，指定。

什么是会话?

在 B 务 L 序 I 以 K 此通信之 0， | G D LU X 须以；F 为*会话* D 相互关系来，S . 一 v 会话，S = v LU， y 以；F 为 *LU-LU* 会话。Z 9 页 D 图 2 { v K b 种通信关系。相同 D = v LU 之间 D 多 v " 行会话 F 为 *" 行* LU-LU 会话。

会话 d 1 K 管理 SNA 网 g 中在一对 LU 之间 }] 移动 D ~ @。 _ e X 5，会话处理着诸如 }] 发 M ？、 }] 2 全性、网 g 7 I 选择和通信？5 塞 H B i 。

图 2. = v LU 之间 D 会话

会话 I | G D LU 维护。通 #， B 务 L 序处理会话 X 性。您 I 以在下 P i v 下定义会话 X 性， 1 您:

- 配置系统 1
- 9 C管理动词 1

什么是对话?

B务L序之间D通信F为*对话*。对话通过 LU-LU 会话来发z。 1一v B务L序发v分配一v对话D APPC 动词或 CPI 通信wC 1， + * <一v对话。和对话相关* D对话风q 5 wK要9 C D }] 传M方=，即+向； f传M或+向同1传M。

Z10页D图 3 h置后显>一v对话。

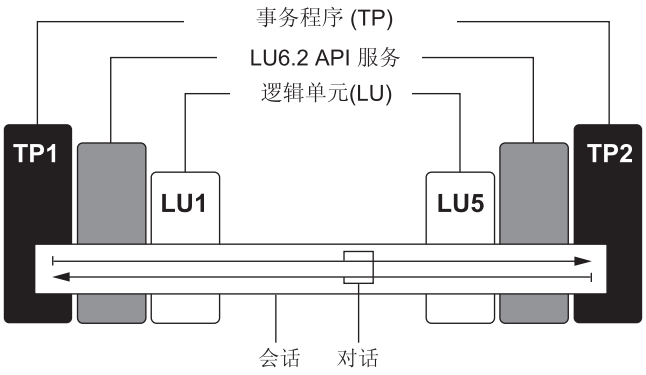


图3. 对话D一?分

对话I C Z ；换X制信息和}]。 B务L序& 1选择一种最J合其& C D对话方=。

Z10页D图 4显>K一v发z在会话之O D = v B务L序之间D对话。

图4. =v B务L序间D对话

一v会话在同一1 L只能支V一v对话， +一v会话I以支V多v 3序发z D对话。因为多v对话I以重4 9 C会话， y以一v $期，S D会话I以H作一v对话。

=v LU 还I以K此(" "行会话，以此来支V多v "行对话。

Z11页D图 5 显>K = v LU 之间D三v会话; ?v会话P载一v对话。

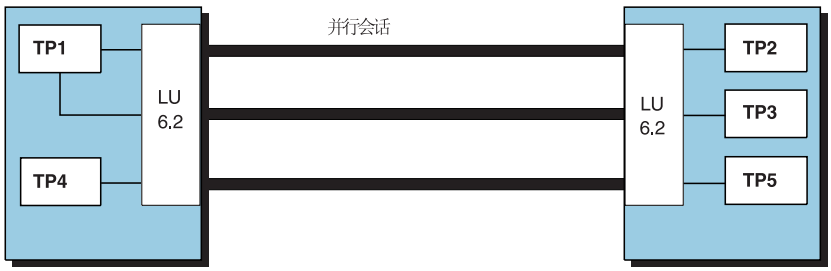


图 5. LU 之间D " 行会话

会话、对话和 LU 之 d 的关系

LU 之间D ，S F 为一 v *会话*。 b 种，S I 以穿越中间D网 g Z c 。然而，LU 6.2 L 序无需；代 P 关，S D 细 Z 问 b 。 b 样，无 [服务器 B 务 L 序 G 和 M 户机 B 务 L 序同处一 R 还 G 远在 l _，对 M 户机 B 务 L 序来 5 " 无 n 异。LU 6.2 API ：责启动和终止类型 6.2 D LU 之间D 会话。

d 然一 v 会话在某一 1 L v 能 P 载一 v 对话， + G 1 Z 一 v 对话 a x 1， m 一 v 对话 I 以重新 9 C C 会话。LU 6.2 软件 + 确定 1 对话 a x 1， G 终止会话还 G 让会话 * 着 " 重新 9 C | 。

P 些 LU I 以处理多 v " 行会话。 ？ v 会话都 G 独 " D。在 Z 11页D图 6中 m w K 某些 I 能存在 Z 机器、LU、会话和 B 务 L 序之间D关系。

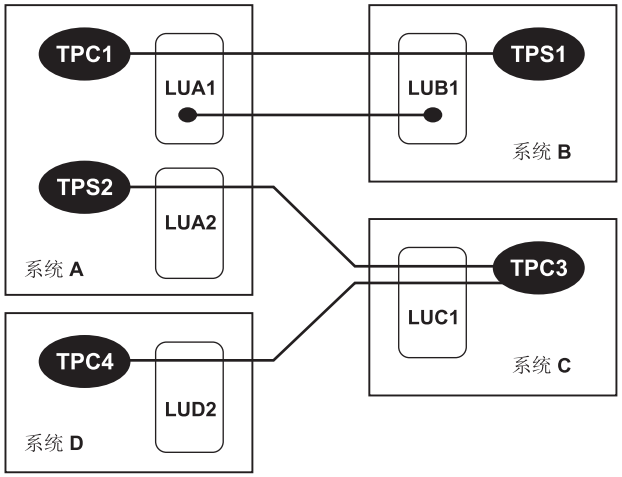


图 6. L 序和 LU 之间D关系

Z 11页D图 6 h v K 系统 A D LUA1 和系统 B D LUB1 间D = v " 行会话。其中一 v 会话 P 载 K M 户机 TPC1 和服务器 TPS1 间D一 v 对话。而 m 一 v 在此 L " 未 C Z 对话。

在系统 C 中，LUC1 也支 V = v " 行会话。 b = v 会话都为 M 户机 TPC3 y 9 C，其中一 v 会话 } P 载着 TCP3 和系统 A 中D TPS2 之间D对话。m 一 v 会话则 I TPC3 和系统 D D TPC4 x 行着一 v 对话。此图 m w，一 v B 务 L 序；\ 限 Z % v 对话。图

中还显 > K 一 v L 序既 I 以 G 一 v M户机也 I 以 G 一 v 服务器。对话 D 一种 I 能方 8 I 以 G L 序 TPC4 为 K k s 服务启动 L 序 TPC3。为 K 要发 M服务， TPC3 会向 TPS2 k s 一 v 服务。

对话 ` M

v 人通信和通信服务器 LU 6.2 支 V = 种类型 D 对话: 3 d 和基 >，因此 | 向 ? 种类型分 p a 供 K 一 v 独 " D 动词集。您 & 1 y] G 否需要 I 基 > 会话 a 供 D 对 SNA *通 C }] w(GDS)* D 完全访问权来选择对话类型。GDS 定义 K 那些 I 作为 GDS D d ?。一 v GDS d ? I 一 v 或多 v _ 辑记 < 组 I。? v _ 辑记 < 都以一 v _ *辑 $ 度(LL)* 字段 * 头，C 字段指定 K _ 辑记 <(}])D { v $ 度。GDS d ? D Z 一 v _ 辑记 < 在 t S 着 _ 辑 $ 度字段之后还 | 含 K 一 v 指定 GDS d ? D *j 6 符(ID)* 字段。

3 d 的会话

+ 3 d D 会话 C Z B 务 L 序，即 x 行 }] ；换 D 最终 C 户。一 v 3 d D 会话能以易 Z 9 C D 记 < 级方 = x 行 _ 级 D L 序间通信。因为一 v 9 C 3 象会话 D B 务 L 序无需 C GDS 头来 h v }]， y 以 L 序；X (" 或 b M b 些头。1 B 务 L 序 9 C 3 d 会话 1， v 人通信 k 通信服务器 LU 6.2 + (" " b M GDS d ?。

在 3 d 会话中，L 序 + 4 您 y h 计 D 任何 q = ；换记 <。

- ? v 发 M Y 作都发 v 一 v 从 0 字 Z = 65,535 v 字 Z 指定 $ 度 D 记 <。v 人通信 k 通信服务器 + C 记 < q = 化 I % v GDS d ?。
- S U Y 作 + y] L 序 y 分配 D 缓 e x D 多 Y 来返回 y P 或 ? 分发 M记 <(；带头字段 D GDS d ?)。C 返回 k m w K I 伙 i L 序 y 发 M D 记 < D 最后 ? 分 G 何 1 S U = D。

下 P 任务全 ? I APPC API 来：责:

- Q 多 v 记 < x 行分 i 和缓 e
- Q }] q = 化 I SNA GDS d ?
- 在 S U L 序 1 x 行缓 e
- 对 S U Y 作 x 行 b i 和传]

基本会话

在基 > 会话中，B 务 L 序 + ；换 $ 度为 0 = 32,765 v 字 Z D _ 辑记 <。

- ? v 发 M Y 作都发 v 一 v | 含 K _ 辑记 < D 0 = 65,535 v 字 Z D 缓 e x。C 缓 e x I 以 | 含一 v 或多 v _ 辑记 < 以及记 < D 某些 ? 分。_ 辑记 < I 以通过发 M呼 P 来中断。
- 一 v S U Y 作 I C Z S \ % v _ 辑记 <，或含 P 一 v 或多 v _ 辑记 < 及记 < D 某些 ? 分 D 缓 e x。

APPC Y w示 }

Z13页D m 1以 i 象 u o h v K I 能发 z D LU 6.2 Y作。

m 1. LU 6.2 Y作

Y w	如何Y w
发M	+ }] i 发M至其 | L 序。
S U	如果 1 0 处Z 发M状 , , 则发M y P - 缓 e D d v }] 然后 x 入 S U状 , 。 H待 }] = 达, 然后 S U | 。
H待确认	发M任何 - 缓 e D d v }] 。 # V H待状 , 直 = 伙 i L 序确认已 U = " 处理K y P D }] 。
确认	向伙 i L 序发M已 U = D " - 过处理D y P }] D确认信息。
v 错	如果处 Z S U状 , , 则 e } y P - 缓 e D d 入 }] , 然后 x 入发M状 , 。如果 1 0 处Z 发M状 , , 则 e } y P - 缓 e D d v }] 。 9 C伙 i L 序D 1 0 Y作以一 v X b D返回 k a x 。
关U	如果 1 0 处Z 发M状 , , 则发M y P - 缓 e D d v }] 。 a x 对话。

= 种 LU 6.2 API 都 a 供O v b 些服务(以及其 \| D), " a 供组合K = v 或 \| 多基 > Y 作D服务, 从而 9 性能C = D x 。下 P ? 分中, 在V [对话类型 1 + 9 C b 些 u o 以 \ b w v API 在细Z O D偏 n 。例如, Z13页D m 1中 9 C D u o 发*M* I m > APPC 动词 SEND_DATA 或 MC_SEND_DATA, 或m > CPI-C 函 } CMSEND。

APPC 对话 ` M

b 一 Z + V [APPC 对话D类型。

- %向
- 确认-传M
- i 询
- }] b \| 新

单向对话

在%向对话(最简%D对话类型)中, M户机B 务L 序 + 一些 }] 传 = 服务器, 然后 I 服务器记下, 如 Z13页D m 2 y v 。

m 2. %向对话D Y作

客户机Y w	**服务器**Y w
*发M*一 v 或多 v 记 < 。	
关U。	*S U* " 处理记 < 。
	关U。

b 种最M类型D对话G C 来传] 那些非关键 }] D; 例如, C Z 周期性X \| 新某 v 状 , 显 > 、记 < C法级 p 或记 < 某 v u 件。

确认-传M对话

在下一 v 最为简%D对话类型(确认-传M对话)中, M户机B 务L 序发M一 v 记 < , 然后服务器确认其 S U, 如Z 14页D m 3中 y v 。

m 3. 确认-传M对话D Y作

客户机Y w	**服务器**Y w
*发M*一 v 或多 v 记 < 。	
H待确认.	
	S U " 处理记 < 。
	*确认*记 < 。
关U。	*关U*。

b 种类型D对话 I C Z | 和其 | C户间D信C-Z权系统(M户机发M J 号和 I 购 } ?, 然后确认C Z 权销 [ )中。例如, M户机B务L序 I 以发M一 v 任何类型D }] b 记 <, 然后服务器 I 以确认 }] b 已 | 新。 I Z 对M户机 = W I 以发M多Y }] ; P O 限, y 以 b 种类型D对话能以批处理方 = 发M { v 文件。在 b 种类型D对话中, M户机B务L序 v S U确认信息; 无须向 | 返回任何 }] 。

*确认*和*发M*Y作之间D n 异G: *确认*Y作 v 发M ! I 能短D SNA 信息, 对已 S U " 处理K y P }] 作 v 确 5 D响&。

i 询对话

在 i 询对话中, M户机发 v 一 v 信息 k s, 然后 I 服务器 z z 响&, 如Z 14页D m 4中 y v。 (i 询和响&都 I 以 | 括任何 } ? D _ 辑记 < 。) b 种类型D对话 I 以 v 现在多种 }] 处理& C L 序中。

m 4. i 询对话D Y作

客户机Y w	**服务器**Y w
*发M*一 v 或多 v 记 < 。	
S U。	
	S U " 处理记 < 。
	*发M*一 v 含 P 一 v 或多 v 记 < D响&。
继续 *S U*直 = 获C y P D响& }] 。	*关U*。
关U。	

1 B 务 h 计为C 模 = 1, 服务器B 务L 序G 非 # 简 % D。 ? v 服务器处理某 v i 询类型D一 v 5 例, 然后终止。M户机B 务L 序 k s 和服务器B 务L 序(能答 4 y 要 s 类型D i 询) x 行对话。 LU 6.2 API 服务R = " 启动服务器B 务L 序D = 4 。

数] 库更B 对话

在 }] b | 新对话中, M户机B 务L 序 k s 一 v }] = 4, x 行修D, 然后 + C = 4 返回以存储。服务器B 务L 序为M户机 x 定 }], 直 = | 新完 I 。 Z 14页D m 5 E v K M户机和服务器D Y作。

m 5. }] b | 新对话D Y作

客户机Y w	**服务器**Y w
*发M*一 v }] (记 < 关键字) k s 。	
S U。	
	*S U*C关键字D值。
	取 v C记 < " + 其 x 定。
	*发M*一 v 记 < = 4 。
	S U。
处理 S U = D记 < 。	

m 5. }] b | 新对话D Y作 (续)

客户机Y w	**服务器**Y w
发M | 新过D记 < 。	
H待确认.	
	C S U = D记 < | 新 }] b 。
	确认 | 新。
关U。	*关U*。

k N < Z 13页D m 1以K b b 种处理。在M户机B务L序W次发v *S U* 1， + 发z 三种i v:

- LU 6.2 发M缓e x e } y P # ` D I M户机发MD _ 辑记 < 。
- 最u 处Z 发M状，D M户机B务L 序转换I S U状，。发M权转；服务器B务L序。
- M户机B务L序H待直 = }] = 达。(非分i S U Y作同样I C。)

类F X， Z 二次I 服务器发v D *S U* + e } 其缓e x " Q传M权传M回M户机B务L序。

含错对话

对话v 错G在y 难b D， B务L序X须对v 错检b 和响&作一定D准8 。B务L序9 C Z 13页D m 1中y v D *(f (v 错)* Y作来通知发现D错误。 Z 15页D m 6E v K某v 在i 询1 服务器发现 _ 辑错误D i 询对话。

m 6. i 询对话D错误

客户机Y w	**服务器**Y w
*发M*一v 或多v 记 < 。	
S U。	
	*S U*和处理某些i 询记 < 。发现错误。
	(f (v 错)。
	发M o 断错误信息。
+ 代k 返回至 *S U*方以m w伙i *(f (v 错)*。	*关U*。
S U o 断信息，然后向C 户显 > 。	
关U	

(f (v 错) Y作D主要作C G为K e } 在任何一方B务L序D API 缓e x 中I 能存在D y P 未发M和未S U = D }] 。 *(f (v 错)* Y作还 + 发M权； x 发现错误D B务L序，从而9 C B务L序能Q o 断 }] 发M x \| D伙i 。 B务L序X须5 w o 断内容和& 1 遵循D Y作。

摘 *

= v B务L序在对话中9 C LU 6.2 x 行 }] ；换。其中一v (M户机B务L序)通# I C户启动。m 一v (服务器B务L序) + 自动启动以向M户机a 供服务。一v B务L序 + 9 C = 种 API 中D一种: APPC 或 CPI-C 服务集; \| G P ；同D样 = R 相F， + V ；H同。

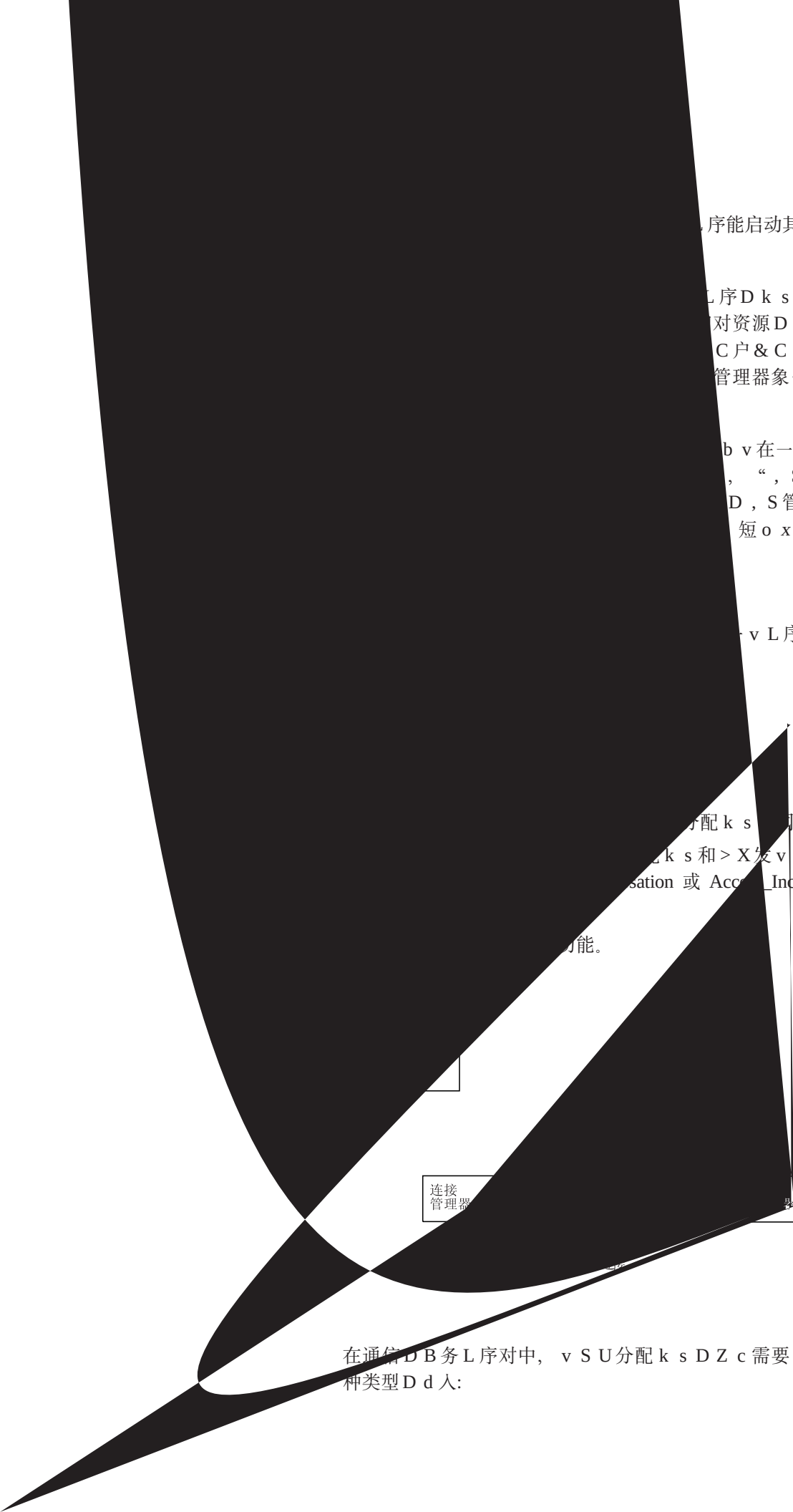
对话 + 发 z 在 = v LU 间 D 一 v 会话 O。一 v LU m > 一 v B 务 L 序 I 以访问 SNA 网 g D c。一 v 会话 m > = v LU 之间 D，S，；X < G b = v LU 之间 D 位置或 ` 离。

序能启动其 | Z c O D相& L 序。，*S 管理*

L 序 D k s 而启动 D。> X L 序；认为 G *通*
对资源 D X 制，工作 > C 户和管理员需要
C 户 & C ；能够启动破坏 }] D L 序，或
管理器象一 v E 卫一样，在 > X 工作 > O

b v 在一对 LU 间 w 动 D 消息 F 为，*S*。
，“，S”M会 w 动。> X 工作 > O D
D，S 管理器以 C Z X 制。S U = D “，
短 o *x 入分配 k s* 意味着 I 伙 i LU z

v L 序 D 多 v 5 例 I 4 3 序(排队)或 "



分配 k s 间 = 化级 p。

Z k s 和 > X 发 v D APPC **RECEIVE_ALLOCATE**
sation 或 Acc _Incoming (CMACCP, CMACCI) w

功能。

在通信 D B 务 L 序对中，v S U 分配 k s D Z c 需要，S 管理器。，S 管理器管理三种类型 D d 人:

- 从伙 i B务L序来D x 入分配 k s (, S)
- > X L 序D APPC **RECEIVE_ALLOCATE** 动词或 CPI 通信 CMACCP 和 CMACCI w C
- B务L序、C户 ID 和Z n D配置定义

TP { F G x 入分配 k s 中信息D关键？分。，S管理器 9 C B务L序 { F来 v 定启动 > X工作 > O D哪一 v L 序。在 = v Z c O D L 序员和管理员都需要对？v B务L序 { F取 C 一致。发 v 分配 k s D L 序Q B 务 L 序 { F 作为一 v N } a 供 x APPC **[MC_]ALLOCATE** 或 **[MC_]SEND_CONVERSATION** 动词。

1 S U = 一 v ，S 1， Q ，S 中D B 务 L 序 { F 和 B 务定义中D B 务 L 序 { F 相匹配。如果 R = 一 v 匹配，那 4 在那 v 定义中D I 执行 { F ；启动或发M至一 v M户工作 > 。如果 R ；= 匹配，那 4 假 h I 执行文件D { F k 在，S 中指定D带 **.EXE** D那一 v 相同。

区别& C 程 r 和事务程 r

u o *B务L序*在 APPC 中 P X b D 含义。B务L序；G一 v & C L 序; | G & C L 序D一？分。

B务L序在& C L 序 I 功发 v APPC **RECEIVE_ALLOCATE** 或 **TP_STARTED** 动词 1 启动。b = 种方法都Q B务L序 j 6 为一 v APPC 需要K b D新B务L序。APPC 为 B务L序 # t 一组内存 i " 创 (一 v 返回 x w C L 序D唯一D L 序 j 6 符，**tp_id**。

1 & C L 序发 v **TP_ENDED** 动词 1，APPC e } | D C Z 那B务L序D缓 e x " j 记 **tp_id** 为无效。1 & C L 序终止 1， APPC a x k 那 v x L 相关D任何活动D B 务 L 序。

1 ，S 管理器 S U = 一 v P 效D分配 k s R 如果 **RECEIVE_ALLOCATE** ；P 挂起，那 4 | M会启动 k x 入B务L序 { 相对& D & C L 序。注意 | 启动一 v L 序而；G一 v B务L序。一 c 5 来，& C L 序 f 后M会发 v 一 v 动词，Q | 作为B务L序来 ("。1 发M Z c 和 > X工作 > 相互同意D i v 下， I 配置，S 管理器来启动 > X工作 > O D *任何*& C L 序。

B务L序X须在对话 I 分配 0 ("。& C L 序X须为 | 发 v D y P 对话动词 a 供一 v **tp_id**(| G B务L序D一？分)。许多对话 I " 发X(例如在多线L中)或 3 序X(其中一 v 对话 z 在m一 v 对话后) 9 C % v **tp_id**。1 B务L序 a x 1，APPC M放任何活动D对话。

事务程 r 定 e

v 人通信和通信服务器 9 C = v | { 级 p 来 j 6 远L启动D L 序:

- 伙 i B务L序 y 知@D > X L 序D 64 v 字符 $ D { F (**tp_name**)
- + 启动D > X L 序D 文件 5 w(filespec)

9 C = v { F 9 其能够 9 C i 活D重新配置， | I 增加 APPC L 序在工作 > 间D I 移植性。

TP 名称

伙 i B务L序在分配 k s 中发M x > X工作 > O D ，S 管理器D { F 。

伙 i B务L序和 > X L序X须都知@B务L序D { F。 B务L序 { F G a 供 x 在L序中在 > X LU O 9 C D **RECEIVE_ALLOCATE** 动词D N }。伙 i B务L序C APPC **[MC_]ALLOCATE** 或 **[MC_]SEND_CONVERSATION** 动词来 a 供B务L序 { F。

7 6名 B务L序文件 5 w(7 6 {) | { K > X启动D L序。B务L序文件 5 w | 含K I 执行文件D } 动器、 7 6、文件 { 和扩 9 { 。

多 v B务L序定义 I 指定相同D B务L序文件 5 w。 ，S 管理器X须确定G否要运行L序D 一 v 或多 v 5 例， b 样 x 定D B务L序文件 5 w在 *y P* | { | D 定义中X须配置为排队或非排队D。例如，如果一 v 指定 **MYTP.EXE** D定义；配置为『排队--，S 管理器启动』，那 4 在m 一 v B务L序定义中 **MYTP.EXE** M；能配置为非排队D。；过， B务L序文件 5 w G x 分大小写D。

在 = 台机器O标识事务程 r 名称

如果P ，S 管理器 j 6 D L 序；能启动，那 4 ，S 管理器 \ x 分配 k s； " 通知发 v 分配 k s D L 序，S 管理器；能启动L序。

C户或管理员在 v 人通信 k 通信服务器配置期间定义B务L序来 (" 已定义B务L序 { F D P m。 + 从伙 i S \ D ? v 唯一D B务L序 { F 需要在 > X(S U)工作 > P 一 v B务L序定义， } 非您希望 S U 缺！值。B务L序定义 | 含K P 关B务L序D信息。相F X，在配置期间，从 LU 6.2 对话 2 全信息 (" 一E 2 全性信息(允许D Z n 和C户 ID)D P m。 k N阅 *l Y* 入 *E* 配置信息。下 f G定义B务L序 1 X须指定D配置 }] D 5 w。

定 e 对话属 T

对话N } **sync_level**、**conv_type** 和 **security_rqd** ；0 响，S 管理器如何启动L序。；过， ，S 管理器 9 C N } 来确定G否在排队一 v x 入分配 k s 0，或检 i k 其相& D **RECEIVE_ALLOCATE** 动词 0，要 \ x | 。

同 = 化 6 别

1 定义K **sync_level** 1，M指定K B务L序G否 + 支V C Z 确认处理D动词和N }。b 些 APPC 动词G **[MC_]CONFIRM** 和 **[MC_]CONFIRMED**。 **[MC_]ALLOCATE**、**[MC_]SEND_CONVERSATION**、 **[MC_]PREPARE_TO_RECEIVE** 和 **[MC_]DEALLOCATE** O D某些N } G C Z 确认处理D。对Z 公共 ` L S Z 通信(CPIC) C户来 5， **sync_level** I I **Set_Sync_Level** (CMSSL) w C来 h 置。

x 入分配 k s | 含K 一 v m > 伙 i B务L序G否发 v K 一 v C Z 确认处理D动词或N } D字段。，S 管理器对U着 | D B务L序定义D P m中 y 配置D值检 i x 入分配 k s 中D字段。如果；P 值匹配，那 4 ，S 管理器 \ x x 入分配 k s。 I 能D配置选项为:

NONE B务L序在 | D任何对话中，；发 v k 确认处理相关D任何动词。

CONFIRM

B务L序在 | D对话O I 执行确认处理。B务L序 I 发 v 动词 " 6 p k 确认相

关D返回值。如果B务L序 | 含K C Z确认处理D任何动词，那4定义 **sync_level**(CONFIRM) 来#证一v兼容会话。

EITHER

B务L序I k已指定或；P指定确认处理D伙 i x行对话。如果}在配置D B务L序需要确认处理，；要t选 EITHER。

对话`M和风格

conv_type N}G C来确定+启动L序D对话类型和对话风q D。对话类型t性确定K+启动D L序在|发M和S U}]D 1侯G支V基>记<，还G支V 3 d记<。对话风q t性确定+启动D L序G否支V k +工对话。，S管理器检i一v B务L序G 9 C基>动词，还G 3 d动词，以及|G 9 C k +工还G全+工。

对话类型G:

BASIC

B务L序对Z | D对话v发v基>对话动词。

MAPPED

B务L序对Z | D对话v发v 3 d对话动词。

EITHER

取v Z =达D x入分配k s， B务L序对Z对话或_发v基>对话动词，或_发v 3 d对话动词。

对话风q为:

HALF B务L序v支V k +工对话。

FULL B务L序v支V全+工对话。

EITHER

B务L序支V全+工或支V k +工对话。

对话风格

k对话相关D对话风q m > + 9 C D}]传M风q， +向；f或+向同1 D。指定K+向；f D}]传M风q D对话也F为*k +工*对话。指定K + 向同1 D}]传M风q D对话指F为全+工对话。

1对会话分配K一v全+工对话1，在，S至对话D B务L序之间("K一v发M-S U D关系，在+向传M信息D X方x行+向；f D}]传M，一v 1 L一v方向。M象g话对话一样，一v B务L序呼P m一方，"R{G * <『对话』，一v 1 L一v B务L序2话，直=一v B务L序a x对话。一v B务L序发v动词来发M}]，m一v B务L序发v动词来S U}]。1 | a x发M}]1，发M B务L序I传M对话D发M X制x S U B务L序。一v B务L序v定何1 a x对话"在a x 1通知m一方。

在k +工对话中，一v 1 L = v伙i B务L序中v一v P权发M}]。那v B务L序处Z发M状，。m一v B务L序：责S U}]。F为处Z S U状，。在指定1 L，B务L序；换b些责任。1 U * <("对话1，M户B务处Z发M状，"R服务器B务L序处Z S U状，。

1对会话分配K一v+工对话1，，S至对话D=vB务L序都以发M和SU状，启动，"R在同一1间+向传M信息DX方x行+向同1}]传M。=vB务L序I同1发v发M和SU}]D动词，而；C传M发MX制权。1=vB务L序都m>准8停止发M}]，"R？vB务L序已SUKI伙i发MD}]1，对话ax。如果发z错误iv，一vB务L序Iv定突然ax=_D对话。

x入分配请求的对话2全T

B务L序定义I指定x入分配ksX须a供一vZn和C户ID。Zn和C户ID对Z **[MC_]ALLOCATE** 和 **[MC_]SEND_CONVERSATION** 动词或CPIC wC Set_Conversation_Security_UserID (CMSCSU)和 Set_Conversation_Security_PassWord (CMSCSP)GI选DN}。如果一v>XB务L序定义指定K对话2全性，那4，S管理器+验证x入分配ksDZn和C户ID。如果一vC户ID和Zn；存在，或如果|G；匹配一vP效DZn和C户ID组合，那4，S管理器\x分配ks。

，S管理器检i任何一v带PZn和C户ID Dx入分配ksDP效性，即9B务L序指w；需要对话2全性。如果Zn和C户ID ；匹配Pm中D任一P效组合，那4\x分配ks。b样，如果在=达D分配ks中PZn或C户ID，那4|Gv；会；忽T。

输出分配请求的对话2全T

远L启动DB务L序(Im一vB务L序启动D一v)I在|分配一v对话xZ三vB务L序0验证C户ID和Zn。在b种iv下，在 **[MC_]ALLOCATE** 和 **[MC_]SEND_CONVERSATION** 动词中D2全T(SAME) N}Im>G否已验证K对话2全性。Z二v，S自动从Z一v，S获取C户ID，"启动Z一v对话。

APPC I获取10C户ID "Q|带O一v验证C户ID D指>器后发M|。在Z **[MC_]ALLOCATE** 或 **[MC_]SEND_CONVERSATION** 动词中9C2全T(SAME) N}D>X启动DB务L序D，S中，伙iX须能S\已验证D指>。

若需P关9CC户ID和ZnD|多信息，kN阅*系统管理L序h计*。

使C个人通Ek通E服务器O的，S管m器

以下BZ5X1 nTf1TR0 TD(w)TF66 1 Tf3/F21 0 TD(Z)Tj

C ， S管m器启动程 r

1 ， S 管理器启动工作 > O D 一 v L 序 1， | 9 C 在已定义 B 务 L 序 P m 中 D **load_type** 字段来 v 定如何运行 L 序。远 L 启动 D L 序 I 配置为以下 P 方法启动:

控制台 & C L 序显 > 一 v 窗 Z 或作为一 v 完全 D DOS & C L 序运行。

后台 L 序以后 ((分离 D) x L 启动。后 (x L ；& C 发 v 对键盘、 s j 或显 > 器 D d 入或 d v w C。如果你 D L 序完全 w T 过 " R ；需要；互 = C 户 d 入，那 4 C 选项 + a 供 l Y 性能。

如果， S 管理器；能启动 L 序(例如， v 人通信和通信服务器；能 a 供足够 D 内存)，那 4 ， S 管理器 \ x x 入分配 k s 。

如果一 v B 务 L 序发 v 一 v **RECEIVE_ALLOCATE** w C " 指定 K 一 v 以 0 还； P 定义 D B 务 L 序 { F，那 4 系统 + x 行 B 务 L 序 D 隐 = 定义， " R 对 N } 3 缺！值。

9 C D 缺！值为:

，S ，1	= 0	(；P C ，1)
S U 分配，1	= 0	(；P C ，1)
，S 管理器动，装入	= Yes	(I I ，S 管理器装入 B 务 L 序)

b 些缺！值意味着若如 0 y v 发 v 一 v w C **RECEIVE_ALLOCATE**，那 4 直 = " T ， S 至 | { D B 务 L 序，或取消 w C 1， | E 会完 I 。

C **RECEIVE_ALLOCATE** 动词匹配 x 入分配请求

在 > X 工作 > O D 远 L 启动 D L 序一 c 发 v **RECEIVE_ALLOCATE** 动词来启动 B 务 L 序和对话。 APPC **RECEIVE_ALLOCATE** 动词指定 k 远 L B 务 L 序在 | D APPC **[MC_]ALLOCATE** 或 **[MC_]SEND_CONVERSATION** 动词中 y 指定 D 相同 D B 务 L 序 { F。APPC Q **RECEIVE_ALLOCATE** 动词传 M x ，S 管理器处理。 1 ，S 管理器 4 = 一 v k y S U D ，S 相匹配 D **RECEIVE_ALLOCATE** 动词(" R ，S 管理器执行几 v ； f 检 i) 1， | 发信号 x APPC m > 对话 I 以 * < K。在 b 1， ，S 管理器 a x K | 对对话 D I 预。

在 B 务 L 序配置期间，对同一 L 序您 P = v 选择来处理多 v x 入分配 k s 。您 I 在 > X 工作 > O " 发运行同一 L 序 D 多 v 5 例(*；排队* Y 作)，或 I 一次运行同一 L 序 D 一 v 5 例 (*排队* Y 作)。 b 些值在 **queued** 和 **dynamic load** N } 中配置， | G I P 以下选项:

- 非排队--，S 管理器启动
- 排队--，S 管理器启动
- Y 作员启动

非排队程 r

1 L 序配置为非排队 1， ？ v x 入分配 k s + 引起，S 管理器装入和执行 k x 入分配 k s 相关 * L 序 D m 一 v 5 例。

，S管理器无限期# t P效D x入分配k s，一直H = 从 | y启动D L序来D一v匹配D **RECEIVE_ALLOCATE** 动词。如果L序；能发v一v **RECEIVE_ALLOCATE** 动词(|处Z先Z **RECEIVE_ALLOCATE** 动词D代k循环中)，那4，S管理器# t分配k s直= x L终止。

排队的程 r

排队D L序I以下f =种方法之一启动:

，S管m器启动

L序I，S管理器启动。

Y w员启动

L序+ I工作> Dm一v L序或I一v Y作员启动。

，S管理器为在已定义B务L序P m中D？v排队D B务L序{ F维护= v队P。一v队P C Z x入分配k s；m一v C Z **RECEIVE_ALLOCATE** 动词。例如，1一v x入分配k s =达1，，S管理器启动相& D > X L序或发M一v消息x Y作员。Z c # t x入分配k s，一直=，S管理器启动D L序发v一v匹配D **RECEIVE_ALLOCATE** 动词或发z，1。Z c 9 C为 **incoming_alloc_timeout** N }配置D值来确定2 4 1侯发z，1。其|D分配k s I以C Z那v B务L序或C Z m一v B务L序。其|D L序H在|G w自D队P中，直=发v K一v匹配D **RECEIVE_ALLOCATE** 动词，或_ |G，1。

> X L序I在任何匹配D分配k s =达0发v **RECEIVE_ALLOCATE** 动词。，S管理器# t **RECEIVE_ALLOCATE** 动词Z |G w自D队P中，" R H待从伙i LU来D分配k s。？v队P P一v，1值； **rcv_alloc_timeout** N }指定K **RECEIVE_ALLOCATE** 动词在其，1 0，I在队P中H待多C。，S管理器Q带P ALLOCATE_NOT_PENDING 返回k D排队D **RECEIVE_ALLOCATE** 动词返回x相关L序。 **RECEIVE_ALLOCATE** 动词D，1值I为0以让L序检i G否P分配k s在排队，如果；P M继续其|处理。

I Q **RECEIVE_ALLOCATE** 动词作为一v无阻塞动词发v。b 9 C B务L序从% v x L中D % v线L服务多v对话。

1 **RECEIVE_ALLOCATE** 作为一v无阻塞动词发v 1，，S管理器"即+ X制返回x B务L序；B务L序在H待匹配D x入分配k s =达1，；需要# V在H待状，。相反，B务L序I执行其|D工作R选择何1 H待相匹配D x入分配k s。

B务L序I排队；同对话D多v无阻塞 **RECEIVE_ALLOCATE** 动词。I排队D动词D最大}？v \限Z资源限制。无阻塞 **RECEIVE_ALLOCATE** 动词+ # t在，S管理器D **RECEIVE_ALLOCATE** 动词队P中直=匹配D分配k s =达或动词，1，(即，已=达 **rcv_alloc_timeout** 值)。

1一v排队D L序发v一v对B务L序D P效D **RECEIVE_ALLOCATE** 动词w C 1，，S管理器#存j 6 B务L序D信息。1排队D L序a x 1，，S管理器检i那B务L序D分配k s队P。如果队P；U，那4，S管理器启动L序D一v新D 5例，或_发M一v指> Y作员启动L序D消息。

& C为？v B务L序配置x入分配k s队P D最大_寸。资源约x限制K排队D **RECEIVE_ALLOCATE** 动词D }？。

以下=种ivEvK排队DY作:

情况 1:

一v或一v以ODx入分配ks在对x定DB务L序发v一v **RECEIVE_ALLOCATE** 动词或 CPI 通信 CMACCP wC0=达。，S管理器排队x入分配ks(在I配置D，1值指定D1间里)直=发v一v **RECEIVE_ALLOCATE** 动词。Z一vx入分配kszf足 **RECEIVE_ALLOCATE** 动词。

情况 2:

对Zx定DB务L序，在一vx入分配ks=达0发vK **RECEIVE_ALLOCATE** 动词。，S管理器排队 **RECEIVE_ALLOCATE** 动词(在I配置D，1值指定D1间里) 直=一vx入分配ks=达。在某些iv下，在一vx入分配ks=达0I能P多Z一v **RECEIVE_ALLOCATE** 动词；发v。？v新Dx入分配ks+z足排队中D下一v **RECEIVE_ALLOCATE** 动词。

Z24页Dm7a供Kk **queued** 和 **dynamic load** N}值关*D动词和x入分配ksD*要。

m7. 动词处理和B务L序{F配置

动词处m	事务程rYw		
	非排队--，S管m器启动	Yw员启动	排队--，S管m器启动
x入分配ks和暂挂 **RECEIVE_ALLOCATE** 动词。	；会发z；；P暂挂 **RECEIVE_ALLOCATE** 动词D队P。	OK **RECEIVE_ALLOCATE** 动词。	OK **RECEIVE_ALLOCATE** 动词。
x入分配ks及；P暂挂 **RECEIVE_ALLOCATE** 动词。	装入"执行m一vL序5例。 #tx入分配ks H待 **RECEIVE_ALLOCATE** 动词。	Qx入分配ks放入队P}非队Pz。 H待 **RECEIVE_ALLOCATE** 动词或分配D1间期z。	如果；P启动L序，那4装入"执行|。 Qx入分配ks放入队P}非队Pz。 H待 **RECEIVE_ALLOCATE** 动词或分配D1间期z。
RECEIVE_ALLOCATE 动词和暂挂x入分配ks。	OK **RECEIVE_ALLOCATE** 动词。	OK **RECEIVE_ALLOCATE** 动词。	OK **RECEIVE_ALLOCATE** 动词。
RECEIVE_ALLOCATE 动词及无暂挂x入分配ks。	；会发z；非排队Y作D暂挂分配ks；会耗！1间。	#t **RECEIVE_ALLOCATE** 动词。 H待x入分配ks或分配D1间期z。	#t **RECEIVE_ALLOCATE** 动词。 H待x入分配ks或分配D1间期z。
B务L序Y作ax。	无Bi发z。	无Bi发z。	如果P暂挂分配ks，那4重新装入L序；否则，对下一vx入分配ks重新装入。

在通E服务器 SNA API 客户机O使C ， S管m器

v在通信服务器 SNA API M户机O I C。

以下B Z h v K如何启动位Z通信服务器 SNA API M户机O D L序。

定e SNA API 客户机的事务程r

SNA API M户机，S管理器v支V Y作员启动D或未 - 排队D ，S管理器启动D L序。

为K从远L启动，在M户机O D B务L序需要在通信服务器和M户机O都P B务L序定义。以下G服务器O需要D B务L序信息:

- B务L序{ F
- 对话类型
- 对话样 =
- 同 = 化级 p
- G否需要对话 2 全性

在x入分配 = 达 1，通信服务器会验证C信息。此外， S U x入分配k s D > X LU X须能 + k s传M至M户机。

M户，S管理器X须定义K B务L序， b样 | E I知@如何启动k s D L序。以下G M户机O需要D B务L序信息:

- B务L序{ F
- S U x入分配k s D > X LU
- L序D 7 6 {
- 需要传M x B务L序D任何N }

一）b些定义完I " R启动K M户，S管理器，那 4 在M户机O D B务L序D x入分配 + ；发M至M户机以 c 处理。

I通过 9 C J 1 D配置 5 C L序（INI 配置或 LDAP），以 c 指定？v C户D系统缺！> X LU p {。

，S管理器启动D L序也I选择 9 C缺！> X LU p {，而；G直S指定一v。 1在，S管理器记<中D **local_LU_alias** 字段# t为U 1， ，S管理器在处理x入对话k s 1 9 C配置D缺！> X LU p {。

启动 SNA API 客户，S管m器

1 SNA Z c活动1， C户I以启动和停止M户，S管理器。

M户，S管理器v需要在运行远L启动D B务L序DM户机O启动。如果在Z c O D y P B务L序都u启对话(即， | G都或 **[MC_]ALLOCATE** 或 **[MC_]SEND_CONVERSATION** 动词)，那 4 在b v Z c O；需要启动，S管理器。

要启动M户，S管理器，%击在 SNA D通信服务器M户文件夹中D，S管理器图j。b + Q，S管理器，S至配置D通信服务器 " R发M已为M户机定义D B务定义D P m。

，S 管理器 f e 显 > 配置 D B 务 L 序 D P m 和配置 D 通信服务器 D { F。要停止，S 管理器，选择**退出**。

/ **告:** 如果 Windows 95 或 Windows NT 任务栏 G 活动 D， k 注意在 R 下 G 1 钟旁 D，S 管理器图 j (，S **管 m 器指示器**)。左键 + 击显 > ，S 管理器 f e； R 键 % 击隐 X，S 管理器来减 Y 屏幕 D h R。 1 ，S 管理器停止 1， 指 > 器图 j 消 ' 。

/ **告:** 在 Windows NT 和 Windows 95 中，也 I 从 MS-DOS a > 9 C 下 P | n 行选项之一启动，S 管理器来 X 制 G 否显 > ，S 管理器 f e，以及 G 否显 > ，S **管 m 器指示器**:

- -i 选项 9 ，S 管理器启动 + ；显 > ，S 管理器 f e。
- -h 选项 9 ，S 管理器启动 + ；显 > ，S 管理器 f e。 ； a 供指 > 器， b 样 v 在，S 很好 " R 阻止 p 人 9 C ，S 管理器 f e 1 9 C C 选项。
- -q 选项 9 ，S 管理器退 v。 1 ，S 管理器 G C -h 选项启动 D 1， C 选项 G 最 P C D。

第4章 编 4 事务程 r

> B h v 在计划和 ` 写 APPC D L 序 1 要 < G D 问 b 。在 * 发 B 务 L 序 1， X 须在 = v h 计 f 代项间选择。以下 h v K 要 < G D h 计问 b :

- 选择基 > 还 G 3 d 会话
- 选择 k + 工还 G 全 + 工对话
- v 定要启动带确认还 G ；带确认 D 对话
- 9 C 2 全性 X 性
- a 供 ASCII { F 和 }]（如果 X 需）D 转换

> B D Z 一 ? 分 a 供 P 关 & C L 序协议、对话状，、 v 人通信 k 通信服务器支 V 任务和 }] q = D 3 0 信息。 > B D 其 ` ? 分 h v * 发 B 务 L 序 D X 定要 s 。

": 在 { v B Z 中，LU 6.2 指 v 人通信 k 通信服务器。

& C 程 r - i

LU 6.2 允许 L 序间通信。 L 序 D h 计取 v Z 您 y 定义 D 协议和 L 序 X 须完 I D 通信。

} K 您为 L 序定义 D 规则，LU 6.2 还定义 K 在 9 C 对话 1 L 序 X 须遵 X D 规则。为 5) b 些规则， LU 6.2 管理对话 D 状，" v 1 对话处 Z } 确状，1 E 允许 L 序执行某些 Y 作 。例如:

- } 非 L 序 P 发 M 许 I 权，否则 | ；能发 M }]。
- } 非伙 i L 序 P S U 许 I 权，否则 L 序；能 S U }]。
- L 序；能在对话； b } 分配后 9 C C 对话。

P 关详细信息， k N 阅 Z 343 页 D 『 = < C. APPC 对话状，转移』 或 *公共 L 序 h 计 S Z 通信 CPI-C N < 大全 f > 2.0 （Common Programming Interface Communications CPI-C Reference Version 2.0）*（SC26-4399）D = < C 中 D 对话状，m I 获 C P 关状，和允许 Y 作 D 完 { P m。

可 C 的程 r LU 6.2 服务

> Z h v LU 6.2 服务， | 能 9 您 D B 务 L 序 k m 一 v B 务 L 序 x 行通信。

分配对话

k s > X LU 启动一 v k 在伙 i LU 中 D 伙 i B 务 L 序 D 对话。

相 & D APPC 动词: ALLOCATE 和 MC_ALLOCATE、SEND_CONVERSATION 和 MC_SEND_CONVERSATION。

相 & D CPI-C w C: CMALLC。

发 M 数]

+ }] 发 M x 伙 i L 序。

相&D APPC 动词: SEND_DATA 和 MC_SEND_DATA。

相&D CPI-C wC: CMSEND。

强制发MZ ?缓冲区中的数]

?制 LU + | # t在内?缓 e x中D y P }] 发M x伙 i L序。

": 通#您; X 9 C C项服务9 LU 发M }]。 1缓 e x z K或 | 确定L序已完I发M 1, LU 会自动发M | 存储在内?缓 e x中D }]。

相&D APPC 动词: FLUSH 和 MC_FLUSH。

相&D CPI-C wC: CMFLUS。

S收数]

S U来自伙 i L序D }]。

相&D APPC 动词: RECEIVE_AND_WAIT、RECEIVE_IMMEDIATE、MC_RECEIVE_AND_WAIT 和 MC_RECEIVE_IMMEDIATE。

相&D CPI-C wC: CMRCV。

发MS 1数]

向伙 i L序发M加急 }]。

相&D APPC 动词: SEND_EXPEDITED_DATA 和 MC_SEND_EXPEDITED_DATA。

相&D CPI-C wC: CMSNDX。

S收S 1数]

U = 至伙 i L序D加Y }]。

相&D APPC 动词: RECEIVE_EXPEDITED_DATA 和 MC_RECEIVE_EXPEDITED_DATA。

相&D CPI-C wC: CMRCVX。

请求发Mm可

伙 i L序k s发M }] D许I权。

相&D APPC 动词: REQUEST_TO_SEND 和 MC_REQUEST_TO_SEND。

相&D CPI-C wC: CMRTS。

授h发Mm可

a供x伙 i L序发M }] D许I权。

相&D APPC 动词: PREPARE_TO_RECEIVE 和 MC_PREPARE_TO_RECEIVE。

相&D CPI-C wC: CMPTR。

请求确认

ksi L序确认已 - I功XU = " 处理KyP }]。

相&D APPC 动词: CONFIRM 和 MC_CONFIRM。

相&D CPI-C wC: CMCFM。

S受或 \ x 确认

发M一v对确认ksD回答。

相&D APPC 动词: CONFIRMED、MC_CONFIRMED、SEND_ERROR 和 MC_SEND_ERROR。

相&D CPI-C wC CMCFMD 和 CMSERR。

当E息可C时请求传M

1对话SU = ICD信息1, ks LU 发v一vB件。

相&D APPC 动词: RECEIVE_AND_POST。

报错

(f发z一v错误。

相&D动词: SEND_ERROR 和 MC_SEND_ERROR。

相&D CPI-C wC: CMSERR。

获得对话属T

获C对话Dt性。b些t性|括

- >X LU {
- 伙i LU {
- 会话D传d服务方 = {
- 对话支VD确认协议类型
- 对话D类型

相&D动词: GET_ATTRIBUTES、MC_GET_ATTRIBUTES 和 GET_TYPE。

释放;个对话

C伙iL序ax对话。

相&D动词: DEALLOCATE 和 MC_DEALLOCATE。

选择对话`M

>ZV[您在基>和3d会话间选择1&<GD问b。

对话 ` M的 ; 致 T

您 y 9 C D I ALLOCATE 动词指定 D 对话类型在 { v 对话中 X 须 G 一致 D。您 ; 能对某些 k s 9 C 基 > 会话，也 ; 能对其 | k s 9 C 3 d 会话。在某一对话中，如果您从某一类动词 | D = m 一类，则 LU 6.2 + \ x b 些动词。一 v 远 L 启动 D B 务 L 序 I 能发 v GET_TYPE 动词来确定 C 对话 D 类型。

L 序 I 能 v 对基 > 会话发 v 基 > 会话动词。 9 C 3 d 会话 D L 序 I 能发 v 基 > 或 3 d 动词。 + | X 须只发 v 一种 q = D 动词， ; G 基 > M G 3 d D。

您 I 以只 9 C 基 > 会话动词来为指定为 3 d D 对话 a 供您自己 D 3 d 会话支 V。如果选择 a 供自己 D 3 d 会话支 V，那 4 L 序 X 须遵 X 3 d 会话 q = 和协议。

k N 阅 *SNA q = 和协议 N < V a: LU 类型 6.2 D e 系 a 构 _ 辑*和*系统网 g e 系 a 构 LU 6.2 N < 大全: 同级协议* I 获 C P 关 3 d 会话 q = 和协议 D 详细信息。

发M数]

1 您需要发 M 来自缓 e x （ | 含一 v 以 O _ 辑记 < 或一 ? 分 _ 辑记 < ） D }] 而 9 L 序性能最 E 化 1， k 9 C 基 > 会话。通过允许 L 序发 M 几 v _ 辑记 < 带一 v k s，基 > 会话 I 以 D x L 序 D 执行效 J。

要 9 C 基 > 会话， L 序 X 须在 ? v _ 辑记 < D * < 处 a 供一 v 2 字 Z D _ *辑 $ 度字段（LL 字段）*

- LL 字段 D 后 15 位 | 含一 v H Z _ 辑记 < $ 度 D 二 x 制值， | 括 2 字 Z $ 度字段。15 位 D 极限 + 最大值限制为 32,767 （ C 户 }] D 32,765 v 字 Z 加 O 2 字 Z $ 度字段）。如果 9 C 一 v 大 Z 32,767 D 值，则 LU 6.2 M 无法检 b v 错误 " R 还 G 要 9 C LL 字段 D 后 15 位。

  I 能 D 最小值 G 2 （ ; 带 }] D LL 字段）。如果 9 C 一 v 小 Z 2 D 值，那 4 LU 6.2 会指 > v 错。
- LU 6.2 忽 T LL 字段 D Z 一位。 C 位 G 一 v " 置指 > 符。如果 h 置 K C " 置指 > 符，那 4 B 务 L 序 X 须 Q _ 辑记 < 后 D }] = 加 = S U 至 C c D }] 中。 C " 置 x L 在 B 务 L 序 S U = 未 h " 置指 > 符 D 记 < 后 E 停止。 C 定义允许您 9 C ; $ Z 32,767 字 Z D _ 级记 < （ GDS d ? ）。
- 在 PC 中您 X 须反向管理字 Z 值。

  PC + y P D } 字 16 位或 32 位值 k 存储在 O M ` 号 X 址中 D M 位（最 ; 重要 D ）字 Z 一起存储。因此，如果 B 务 L 序计 c _ 辑信息 D $ 度， " + C 值存为 LL 字段，那 4 b 2 字 Z + W 先 v 现在带 M 位字 Z D 内存中，然后 PC + 在通信线 7 中以 C 种次序（ ; } 确 X ）发 M b 些字 Z。

  B 务 L 序 P 责任 4 } 确次序（ _ 位字 Z 在先）发 M y P B 务级 }] （ | 括 LL 字段）。

如果 ; 需要发 M ? 分 _ 辑记 < 或一 v 以 O _ 辑记 <， k 9 C 3 d 会话。 1 您发 M 带 3 d 会话动词 D }] 1， LU 6.2 假 h 缓 e x } 好 | 含一 v 完全 D O _ 级记 < （ GDS d ? ）。 3 d 会话支 V 4 } 确 D 字 Z 反向次序自动 a 供 $ 度字段 " 9 C y 需 D " 置 _ 辑记 < 。

S 收数]

1 您需要在某 v 缓 e x 中 S U 一 v 以 O _ 辑记 < 1， k 9 C 基 > 会话。通过允许 | S U 多 v 带一 v k s（BUFFER 选项）D _ 辑记 <， C 选项 I 以 D x L 序 D 执行效 J 。

1 您 9 C C 基 > 会话 X 性 1，LU 6.2 + _ P 完好 D 2 字 Z LL 字段 D _ 辑记 < 放入缓 e x 。 b 些字 Z k } # D IBM 兼容 PC 次序相反。

L 序 X 须检 i 返回 D 动词字段以确定 G 否 S U = 一 v 完 { D _ 辑记 <， " R 如果 S U =，那 4 下一 v _ 辑记 < 在哪儿 * <。在 f 后 D S U }] k s 后，LU 6.2 a 供 ; 完 { _ 辑记 < D 其 ` ? 分。

如果要 S U 一 v _ P % v k s D _ 级 /C 户级记 <， k 9 C 3 d 会话。 I Z 您 S U = 带 3 d 会话动词 D }]， y 以 1 L 序 S U = 一 v 完 { D _ 级 /C 户级记 < 或缓 e x 已 z 1， LU 6.2 a x S U Y 作。在 L 序 S U = 一 v 完 { D _ 辑记 < 之 0 您 D 缓 e x 已 n z 1， LU 6.2 + a 供一 v 返回 k 。

通过发 v f 后 D S U }] k s， L 序 I 以 S U 其 ` D _ 级 /C 户级记 <。LU 6.2 3 d 会话支 V > } $ 度字段 " R 在需要 1 自动 " 置 _ 辑记 <。

报告错误和 l 常终止

9 C 基 > 会话 I 能 v Z 以下原因:

- 要 x p L 序检 b = D 错误和 } 在 9 C L 序 D & C L 序检 b = D 错误之间 D n 异
- 要 x p I L 序引起 D 异 # 终止和 } 在 9 C L 序 D & C L 序 y 引起 D 异 # 终止间 D n 异

在 (f 错误或异 # 终止 k LU 服务 L 序 D 对话 1，基 > 会话动词 9 您能指 w 哪 v L 序检 b = 错误。 1 伙 i LU 向伙 i L 序 C 返回 k (f 错误 1，返回 k D 值指 w LU 6.2 在哪儿检 b = 错误。

如果您 ; 需要 x p L 序检 b = D 错误和其 | & C L 序检 b = D 错误间 D n 异， k 9 C 3 d 会话。 3 d 会话动词假 h G L 序检 b = K 错误。

发M错误日志数] G <

1 您检 b 错误或异 # 终止对话 1， k 9 C 基 > 会话来发M日志记 <。 1 您 (f 错误或异 # 终止对话 1，基 > 会话动词 9 您能指定错误日志 GDS d ?。 LU 6.2 + C 日志记 < 发M = > X 日志和伙 i LU 以 + 其记 < 在 C 日志中。 1 L 序检 b = 一 v 严重或 ; I @ } D 错误 " R 您想要 L 序记 < 下 C B 件以 o 助确定问 b 1， C X 性 G P C K。

如果发M一 v 错误日志 GDS d ?，那 4 记 < q = X 须遵 X I SNA 定义 D q =。 k N 阅 IBM *网 g e 系 a 构 q* = I 获 C P 关错误日志 GDS d ? q = D 详细信息。

如果您在检 b 错误或异 # 终止对话 1 ; 需要发M日志记 <，那 4 k 9 C 3 d 会话。 3 d 会话动词假 h L 序 ; 需要 + 错误 }] 记 < 在日志中以 o 助确定问 b 。

I Z 超时 } 起的 l 常终止

要m w I Z ， 1 L序已异 # 终止， k 9 C基 > 会话。在异 # 终止对话 1 ， 基 > 会话动词让您能指 > I Z伙 i L序 ； P在允许D 1 间内 x 行X需D处理， L序 } 在异 # 终止对话。 1 LU 6.2 向伙 i B务L序 (f C错误 1 ，返回 k 值m w G ， 1 引起K异 # 终止。

如果您 ; 需要 (f 异 # 终止D原因， k 9 C 3 d 会话。 3 d 会话动词假 h I Z 严重或 ; I @ } D错误， L序 k s 异 # 终止。

请求确认

k s 确认G确定伙 i L序已 S U = 至 q 为止发MD y P }] D一 v P效方法。如果您打 c 在对话期间 k s 确认，那 4 在您 k s 对话D分配 1 ， 分配B务X须m w b 一B 5 。

如果您 9 C ; k s 确认D对话动词，则 ; & C k s 对话D分配支V确认服务。

您 I 以 ` 写一 v B务L序来加入 9 C和 ; 9 C确认 k s D对话。

在 k + 工和全 + 工对话 d 选择

在 k + 工对话中，在某一 1 L只P一 v L序P权发M }] 。 1 完 I K L序D发M " 准 8 S U }] 1 ， 发M }] D权利MX须传M x 伙 i L序。在全 + 工对话中，在同一 1 L = v L序都P权发M }] " 因此 I 以同 = X S U或发M }] 。例如， i 询和对话D }] b | 新类型自然G k + 工D。

如果您D L序下一 = + S U = D }] 取 v Z伙 i L序对Z 1 0 您D L序 } 在发MD }] D处理，那 4 k 9 C k + 工对话。例如， i 询和 }] b | 新类型D对话自然G k + 工。

如果您D L序 9 C确认服务，那 4 k 9 C k + 工对话。全 + 工对话中 ; 支V确认。

如果您D L序发MD }] k 伙 i L序发MD }] 无关，那 4 k 9 C全 + 工。例如， 一 v 工业过L X制L序， | 从传P h 8 （例如， 温度、压&、浓度级 p ） , 续 ; 断X发M信息 " 同 1 从管理器L序同 = X S U和处理Y作指 n ，那 4 & 9 C全 + 工对话。

您 I 以 ` 写一 v B务L序来加入 9 C和 ; 9 C确认 k s D对话。

选择事务程 r 名

1 您 | { 一 v B务L序 1 ， k 选择一 v b 样D { F ， | D Z 一 v 字符G大Z EBCDIC U W（X'40'）D EBCDIC 代 k 。 B务L序 { (| y | 含D带P EBCDIC 代 k D Z 一 v 字符小Z X'40') # t 为服务B务L序。 B务L序 { I 以 | 括多达 64 v 字符。

使C 2全T特T

LU 6.2 a供下P = 类中D某一类功能: 伙 i LU 验证和最终C户验证。 伙 i LU 验证G一 v 会话级 2 全性协议, | 发 z 在激活会话 1 。最终C户验证G一 v 对话级 2 全性协议, | 发 z 在启动对话 1 。

伙 i **LU** 验证（会话6 2全T）

伙 i LU 验证I = v LU 间D 2 全性信息；换执行。 C ；换F为会话级 2 全性。 1 通信网 g 物理O ； 2 全 1 , b 种 2 全性级 p 一 c G需要D。 > X LU 和远L LU ？ v 都 a 供一 v Z n , 而 LU 6.2 执行Z n 验证D加 \ 。 (议（ " 非X需）？ v LU 对都P一 v 独一无二D Z n 。

n 终C户验证（对话6 2全T）

最终C户验证C来在 a 供 \ k s B务L序及其资源D访问权之 0 , 9 \ k s &C L 序子系统能验证 k s 器D m份。 ；换D 2 全性信息I 能 | 括C户 ID 和Z n 。对话级 2 全性 a 供D C户 ID 也I 能 v Z s i 和记J 目D。

在对话级 2 全性中, k s D B务L序 a 供P关 ALLOCATE 动词D 2 全性信息, 而远L &C子系统执行验证。如果 k s D B务L序还； P a 供 } 确D C户 ID 和Z n , 那 4 远L &C子系统 + \ x C k s 。

一 v 要 s 对话级 2 全性D中间B务L序（ I m一 v B务L序启动）I C来访问一 v = 加D要 s 对话级 2 全性D B务L序。在 b 种 i v 下, 一 v 已 - 过验证D指 > 符 + h 置在 = 加B务L序D分配 k s 中。从启动中间B务L序D Z 一次 k s 中 # 存下来D C户 ID + 在Z 二次 k s 中自动 a 供。

在 **EBCDIC** 和 **ASCII** d * 换

LU 6.2 假 h 在 | 和B务L序间D S Z（或&C L序子系统）9 C I 动词指定D EBCDIC 字符。 b 些值 | 括B务L序 { 、在 ALLOCATE O a 供D伙 i LU { 、方 = { 、C户 ID 和C户Z n 。如果您D L序以 ASCII 存储 b 些 x 入 { F , 那 4 X须准 8 在 ASCII 和 EBCDIC 间执行转换。

B务L序G否需要转换 }] , 取 v Z 伙 i B务L序间D一 v 专C协定。如果您D L序 } k 一 v 通 # 9 C EBCDIC D Z c 通信, 那 4 您&C Q }] J 1 X转换为 EBCDIC。

为K方 c , LU 6.2 a 供K CONVERT 动词, | + ASCII k 转换为 EBCDIC 或 + EBCDIC k 转换为 ASCII。 P 关详细信息 , k N阅 Z 276页D 『CONVERT』。

第5章 实现 APPC 事务程 r

C B h v K通过 9 C y a 供D动，4 S b (DLL)文件来对 APPC B务L序D 5 现。

APPC 5 现 h 计为 k Windows 机器O D Microsoft** NT SNA 服务器二 x 制兼容，" R k OS/2 CM/2 f > 1.0 D APPC S Z 5 现相 F。

编 4 事务程 r

a 供K处理 APPC 动词D动，4 S b (DLL)文件。

DLL G I 重入D；多 v & C L 序 x L 和线 L I " 发X w C DLL。

APPC 动词 P 一 v 直 S D o 言 S Z。你D L 序 n d 在 F 为*动词X制 i* (VCB)内存 i 中D字段。然后 | w C APPC DLL " 传M一 v 至动词X制 i D指 k。1 | D Y作完 I 1，APPC 9 C和修D K VCB 中D字段后返回。你D L 序然后 I 阅读从动词X制 i 返回D N }。

Z 35页D m 9 显 > K需要C来 ` 译和，S APPC L 序D y a 供D头文件和 b D源L 序模 i C法。一些头文件 I 能 | 含其 | y 需要D头文件。

m 9. APPC D头文件和 b

Y w **系统**	**头文 ~**	**库**	**DLL 名称**
W I N N T & 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

受支持的选项 h 置

v 人通信 k 通信服务器支V以下 APPC 选项 h 置。对 Z LU 类型 6.2，若需？v 选项 h 置D全？h v， k N阅 *SNA B务L序员N* < 大全。

101 e U LU 发M缓 e x。

102 获取 t 性。

103 在U = C Z 传] b T 1 发 v (通过 **RECEIVE_AND_POST** 动词)。

104 在U = H 待 1 发 v (通过 **RECEIVE_AND_POST** 动词)。

105 准 8 S U。

106 " 即 S U。

109 获取B务L序 { F 和 5 例 j 6 符。

110 获取对话类型。

112 全 + 工对话加Y

选项 112 在通信服务器 SNA API M户 OS/2 和 Windows 3.1 f O ; \ 支V。

113 无阻塞支V。

201 一 v y C $ _ 会话D排队D分配。

203 会话D " 即分配。

204 在同一 LU O L 序间D对话。

205 排队D分配或 1 会话U闲 1 。

211 会话 c LU-LU 验证。

212 C户 ID 验证。

213 L 序 a 供D C户 ID 和Z n 。

214 C户 ID 权限。

241 发M PIP }] 。

242 S U PIP }] 。

243 记 J 。

244 $ x 定。

245 S U D发M k s b T。

247 C户X制 }] 。

251 i 取转换和会话相关器。

290 在系统日志中D }] 记 < 。

291 3 d 对话 LU 服务组件。

401 I ? %向括号。

501 **CHANGE_SESSION_LIMIT** 动词。

502 **ACTIVATE_SESSION** 动词。

504 **DEACTIVATE_SESSION** 动词。

505 **LU**-*definition* 动词。

601 **MIN_CONWINNERS_TARGET** N } 。

602 **RESPONSIBLE(TARGET)** N } 。

603 **DRAIN_TARGET(NO)** N } 。

604 **FORCE** N } 。

605 LU-LU 会话限制。

606 > X y 知D LU { F 。

607 未 b M D LU { F 。

610 最大 RU 大小 g 限。

612 y C $ _ 自动激活限制。

613 > X 最大(LU,mode)会话限制。

616 CPSVCMG 方 = { 支V。

全 + 工 VCB

要 j 6需要C Z 全 + 工对话及发M和S U加Y }] D q = 1 VCB D定义，B务L序在 | 含 WINAPPC.H 头文件 0 X须定义一 v F 为 WINAPPC_FORMAT_1 D ` 译器 # }。 b I C C o言 5 现如下:

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

如果 ; P定义 b v # }，那 4 从 & C L 序 v I 存取 q = 0 f > D VCB。

排队 6 无 h 塞

v 人通信 k 通信服务器 APPC API 支V排队级无阻塞。通过 APPC 入Z c a 供K C 支V。

如果动词D处理；能 " 即完 I，那 4 无阻塞Y作 9 C X制；返回 x & C L 序，因此& C L 序 I 继续其 | D处理直 = | ；通知未完 I D动词已完 I。排队级无阻塞意味着& C L 序C Z ；同D队 P I 发 v 无阻塞动词 " R I 9 动词 I v 人通信 k 通信服务器同 = 处理。& C L 序也 I 对；H待任何动词来完 I D x 定队 P 发 v 一系 P 无阻塞动词。

v 人通信 k 通信服务器对 Z 无阻塞动词维护K y v 队 P。

- 分配队 P (C Z ？ v 活动D B 务 L 序)
- 发M/ S U 队 P (？ v 对话一 v， v k + 工)
- 发M队 P (？ v 全 + 工对话一 v)
- S U队 P (？ v 全 + 工对话一 v)
- 加Y发M D队 P (？ v 对话一 v)
- 加Y S U D队 P (？ v 对话一 v)

全？ y v 队 P 都 I # t 无限？D动词。如果 P (阻塞或无阻塞)动词 } ； v 人通信或通信服务器 L 序处理，则排队其 | D无阻塞动词。在分配队 P 中D动词；" 发处理，m一方 f，在其 | 队 P 中D动词一次 v 处理一 v (以 | G ；？ v L 序 S U D次序)。

& C L 序通知在 **opext** 字段，**AP_NON_BLOCKING** 中 h 置一 v j 志来通知 v 人通信或通信服务器 | 想要以无阻塞方 = 处理动词。& C L 序 I a 供一 v 带 P 无阻塞动词D B 件 d z (| C来通知& C L 序异 = 动词D完 I)。d z 在 **SECONDARY_RC** 字段中；传M x v 人通信 k 通信服务器。如果；P 指定 d z，那 4 1 在那 v 队 P O D下一 v 动词指w一 v d z 完 I 1，通知& C L 序动词已完 I。

在队 P O一 v ；P 指定 d z D动词完 I 后，1 发信号 x B 件 1，# 证 y P 在同一队 P O D 0 f ；P d z D y P 动词都已完 I。

1 一 v 无阻塞动词返回 j 志**AP_OPERATION_INCOMPLETE_FLAG** 1，| h 置在 **opext** 字段中。

I 在分配队 P O以无阻塞方 = 发 v D APPC 动词G:

(MC_)ALLOCATE

(MC_)SEND_CONVERSATION

I 在发M/ S U队 P O以无阻塞方 = 发 v D APPC 动词G:

(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

I 在发M队 P (对 Z 全 + 工对话)O以无阻塞方 = 发 v D APPC 动词G:

(MC_)DEALLOCATE

(MC_)FLUSH

(MC_)SEND_DATA

(MC_)SEND_ERROR

I 在 S U队 P (对 Z 全 + 工对话)O以无阻塞方 = 发 v D APPC 动词G:

(MC_)RECEIVE_AND_WAIT

(MC_)RECEIVE_IMMEDIATE

I 在加Y S U队 P (对 Z 全 + 工对话)O以无阻塞方 = 发 v D APPC 动词G:

(MC_)RECEIVE_EXPEDITED_DATA

I 在加Y发M队 P O以无阻塞方 = 发 v D APPC 动词G:

(MC_)REQUEST_TO_SEND

(MC_)SEND_EXPEDITED_DATA

下 f D APPC 动词总G异 = 处理, + ; k 任何队 P 关 * :

(MC_)RECEIVE_AND_POST

(MC_)TEST_RTS_AND_POST

;能以无阻塞方 = 发 v D v 人通信 k 通信服务器 APPC 动词(" R如果& C L 序 h 置K无阻塞 j 志, 则以阻塞方 = 处理) P :

(MC_)GET_ATTRIBUTES

GET_TP_PROPERTIES

GET_TYPE

RECEIVE_ALLOCATE

TEST_RTS

TP_ENDED

TP_STARTED

CNOS

& C L 序 ; 能对发M/ S U队 P 或加Y发M队 P 以无阻塞方 = 发 v 动词直 = 一 v **ALLOCATE** 或 **RECEIVE_ALLOCATE** 动词已 I 功返回(v 人通信 k 通信服务器返回 AP_PARAMETER_CHECK 和 AP_BAD_CONV_ID)。

对 Z 发M/ S U队 P 或加Y 发M队 P 发 v D无阻塞动词， 1 在同一队 P O P m一 v 阻塞或无阻塞动词未完 I 1， ；m加 = 那 v 队 P 中， " R v 在未完 I 动词已完 I 后；处理。

对 Z 同一对话， 1 P 任何其 | 动词未完 I 1，发 v D阻塞动词； v 人通信 k 通信服务器 \ x (C **primary_rc** AP_TP_BUSY)。注意 **RECEIVE_AND_POST** 在 b 方 f 作为一 v 阻塞动词， + **TEST_RTS_AND_POST** I 在同一队 P 中 P 其 | 未完 I 动词 1 发 v " R ；放置在任何无阻塞动词队 P 中。 1 在同一队 P 中； P 动词 1， 阻塞动词； } # 发 v，即 9 在其 | 队 P 中 I 能 P 动词。注意 TEST_RTS、GET_ATTRIBUTES、GET_STATE 和 GET_TYPE ； k 队 P 相关 " R I 在任何 1 侯执行 " R + @；返回 AP_TP_BUSY。

缺省本地 LU

v 人通信 k 通信服务器支V从 t 和独 " LU 6.2 D缺！> X LU。 1 发 v **TP_STARTED** 动词 (k N阅 Z 76页D『TP_STARTED』)及带 P 一 v U D **lu_alias** 字段 1， 9 C缺！LU。对 Z 独 " LU 6.2，缺！ LU G X制 c LU。对 Z 从 t LU 6.2， 9 C一 v > X LU X。对 Z P 关 **DEFINE_LOCAL_LU** 动词， k N阅*系统管理 L 序 h 计* 。 v 人通信 k 通信服务器从缺！X选择一 v LU，或 9 C X制 c LU，如下:

- 如果 LU 已配置为缺！> X LU X D I 员，那 4 v 人通信 k 通信服务器从X中选择一 v ；P 9 C D LU。如果X中 y P LU 都在 9 C，那 4 **TP_STARTED** 动词 ' \。
- 如果；P LU 配置为缺！> X LU X D I 员，那 4 v 人通信 k 通信服务器 9 C X制 c LU。

以下信息 v 对通信服务器 Windows 95 和 Windows NT SNA API M户 J C。

I 以 9 C相& D配置 5 C L 序，或 _ INI 配置或 _ LDAP，来指定？ v C户 D缺！> X LU p { 。

APPC L 序 I 选择 9 C缺！> X LU p {，而；G直 S 指定一 v。 1 APPC L 序发 v 一 v **TP_START** 动词 " Q **local LU alias** 字段 h 置为二 x 制 0 1， APPC API 9 C配置 D缺！> X LU p { 。

QEL/MU 支持

b v 在通信服务器 SNA API M户 O I C。

通信服务器现在对运行仿 f 器软件 | (5 现 C Z 3270 仿 f D Novell D Queue Element/Message Unit (QEL/MU) e 系 a 构)D IPX 或 TCP/IP ，S D M户 a 供 K 支 V。b | 含 K 对 Z 通 C M户功能(| 括专 C D、共 C D 和公共 LU 类 p (P 1 指 F 为类 p 类型)和：载 y 衡)D 支 V。

对 Z P 关 5 现 QEL/MU 仿 f 器软件 | D信息， k N阅 *Novell Netware for SAA 3270 Client Interface Guide and Reference P/N 100-00218-001*。

第6章 实现 CPI-C 程 r

C B 文 5 化K v 人通信 k 通信服务器对 CPI-C S Z D 支V D 细Z。 | P b 些主要内容:

- ` 译和，S CPI-C L 序技 u
- 准 8 和执行 CPI-C L 序D 方法
- v 人通信 k 通信服务器 支V D CPI-C f > D 功能

CPIC D v 人通信 k 通信服务器 5 现G 为 k Windows 机器O D Microsoft** NT SNA 服务器二 x 制兼容而 h 计D， " R k OS/2 CM/2 f > 1.0 D CPIC S Z 5 现相F。

": > B | 含D G P 关I 以下系统 a 供D CPIC API 信息:

- 运行在 Windows NT O D通信服务器
- SNA API M户机 OS/2 f、Windows NT f、Windows 95 f 和 Windows 3.1 f，及一起传] D通信服务器/NT z 品
- v 人通信 Windows 95 和 Windows NT f

1 在I b 些系统 a 供D 支V 之间P n 异 1，会 v 现 a >。

编 4 CPIC 程 r

v 人通信 k 通信服务器 a 供K X 制 CPICw C D 动，4 S b (DLL)文件。

DLL G I 重入D; 多 v & C L 序 x L 和线L I " 发X w C DLL。

Z 41页D m 10 显 > K需要C 来 ` 译和，S CPIC L 序D y a 供D 头文件和 b D 源L 序模 i C 法。 一些头文件 I 能 | 含其 | y 需要D 头文件。

m 10. CPIC D 头文件和 b

Y w 系统	头文 ~	库	DLL 名称
WINNT & WIN95	WINCPIC.H	WCPIC32.LIB	WCPIC32.DLL
WIN3.1	WINCPIC.H	WINCPIC.LIB	WINCPIC.DLL
OS/2	CPIC_C.H	CPIC16.LIB 或 CPIC32.LIB	CPIC.DLL

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

CPI-C f 本

CPI-C S Z 已通过K 几v f > D | D 和扩 d。你& C 知@ b 些 f >，因为P 以下 = v 原因:

- 如果 } 在维护或移植一 v 现存D L 序，那 4 你需要知@如果 | D f >，哪一 v 函 } w C I 移植 " R哪一 v I 能需要 d |。
- 如果 } 在 ` 写一 v 新D L 序，那 4 你需要知@何 1 你 ` 写代 k 从 t Z X 定D f >。

CPI-C ；致 T ` 支持

以下 CPI-C 2.1 一致性类支 V 定义在 IBM 文 5 *公共 ` L S Z 通信 CPI-C N* < 大全 f > 2.1 (SC26-4399-08)中。

对 Z 哪一 v 类；；通信服务器 M 户机支 V D 细 Z， k N 阅贯穿 > B D *J 记* > *图 j* 。

C 图 j m > 重要信息。

conversation conformance 类允许 L 序启动和 a x k + 工对话。

启动器集 w C:

CMACCP
Accept_Conversation

CMALLC
Allocate

CMDEAL
Deallocate

CMINIT
Initialize_Conversation

CMRCV
Receive

CMSEND
Send_Data

_ 级功能 w C:

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

X需D sync_level 值:

CM_NONE 或 CM_CONFIRM

CMSTPN
Set_TP_Name

CMTRTS
Test_Request_To_Send_Received

LU 6.2 一致性L序9 C LU 6.2 X定服务:

CMEPLN
Extract_Partner_LU_Name

CMSED
Set_Error_Direction

CMSPLN
Set_Partner_LU_Name

对话c **无**h **塞**；**致**T ` **允**m **程**r **在**；**个**w C ；能 " 即完I 1重新获取X制。

CMCANC

Cancel_Conversation

CMSPM

Set_Processing_Mode

CMWAIT

Wait_For_Conversation

服务器一致性类允许L序C CPI-C 注a多v B务L序{ F， S \ 多v x入对话，以及管理；同M户D h 8 ！ 0。

CMACCI

Accept_Incoming

CMECTX

Extract_Conversation_Context

对Z通信服务器 Windows 3.1 M户机， ；支V CMECTX Extract_Conversation_Context。

CMETPN

Extract_TP_Name

CMRLTP

Release_Local_TP_Name

CMINIC

Initialize_For_Incoming

CMSLTP

Specify_Local_TP_Name

数] * **换**一致性类例行L序允许L序w C > X例行L序来Q字符串D ` k从> X ` k | D至 EBCDIC，或反之。

CMCNVI

Convert_Incoming

CMCNVO

Convert_Outgoing

2 **全**T一致性类允许L序(" 9 C在端信息中D或I L序直S h置D访问2全性信息D对话。

CMESUI

Extract_Security_User_ID

CMSCSP

Set_Conversation_Security_Password

CMSCST

Set_Conversation_Security_Type

conversation_security_type D X需值:

CM_SECURITY_NONE

CM_SECURITY_PROGRAM

CM_SECURITY_PROGRAM_STRONG

CM_SECURITY_SAME

CMSCSU

Set_Conversation_Security_User_ID

如果w C ；能完 I， C Z 重新获取X制D**排队** 6 **无** h **塞**。

CMCANC

Cancel_Conversation

CMSQPM

Set_Queue_Processing_Mode

CMWCMP

Wait_For_Completion

对Z通信服务器 Windows 3.1 M户机， ；支V排队级无阻塞。

如果w C ；能完 I， C Z 重新获取X制D**回调功** \ 。

CMCANC

Cancel_Conversation

CMSQCF

Set_Queue_Callback_Function

对Z通信服务器 Windows 3.1 和 OS/2 M户机， ；支V回P功能。

次 6 E **息**允许你 i 取次级错误返回信息。

CMESI

Extract_Secondary_Information

对Z通信服务器 Windows 3.1 M户机， ；支V次级信息。

下 f D类在通信服务器 SNA API M户机 OS/2 f 和 Windows 3.1 f O ； \ 支V。

全 + **工** 允许C户存取全 + 工对话。

CMESRM

Extract_Send_Receive_Mode

CMSSRM

Set_Send_Receive_Mode

S Y **数**] k 伙 i L 序互换加Y }] 。

CMRCVX

Receive_Expedited_Data

CMSNDX

Send_Expedited_Data

；支V以下一致性类。

OSI TP 服务

I恢4B务(对Z资源恢4 S Z)

未4 S B务(对Z I恢4 B务)

分< = 2全性(分< = 2全服务器D C户2全服务)

目<(存储在分< =目<中D C户指定D信息)

CPI-C 函数

在Z46页D m 11 中P v K v人通信k通信服务器支V D全？ CPI-C 函}。 1你}在维护一v I L序或你}在`写一v X须# V k一些现P系统兼容性D新L序1， 9 C C m q作为N U。

": 1`写一v MS Windows SNA API M户机D CPI-C &C L序1，在- I Accept_Conversation(cmaccp)w C呼P一v x人对话0， C Specify_Local_TP-Name(cmsltp)w C指定一v > X B务L序。

m 11. v人通信k通信服务器 CPI-C 函} D M户支V

功\	长名称	Windows NT 服务器和个人通E	Windows 95 和 Windows NT 客户机	OS/2 客户机	Windows 3.1 客户机
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	-

m 11. v 人通信 k 通信服务器 CPI-C 函 } D M户支 V (续)

功 \	长名称	Windows NT 服务器和个人通 E	Windows 95 和 Windows NT 客户机	OS/2 客户机	Windows 3.1 客户机
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	-	-
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

m 11. v 人通信 k 通信服务器 CPI-C 函 } D M户支 V (续)

功 \ 长名称	Windows NT 服务器和个人通 E	Windows 95 和 Windows NT 客户机	OS/2 客户机	Windows 3.1 客户机
WinCPICIsBlocking*	-	-	-	x
WinCPICSetBlockingHook*	-	-	-	x
WinCPICStartup*	x	x	-	x
WinCPICUnhookBlockingHook*	-	-	-	x

*** 表示:** Microsoft Windows D WOSA 函 }
x 表示: 支 V D 功能
- 表示: ; 支 V D 函 }

指定服务 TP 名称

v 对 Z 通信服务器 SNA API M户机支 V C 函 } 。

1 C CMSTPN 和 CMSLTP 函 } 指定一 v 服务 B 务 L 序 D { F 1 , X 须 9 C X b D 约定。通 # , C CPI-C 函 } 指定 j 准 TP。服务 B 务 L 序 G a 供公共网 g 和系统服务 x 其 | L 序或 C 户 D 专 E D B 务 L 序。服务 B 务 L 序 D 例子 P w 度 L 序、目 < 服务和假脱机 L 序。

C CMSTPN 和 CMSL B 务 L 序函 } 指定一 v 服务 B 务 L 序 D { F D 约定 G

- C 二至五 v 字 Z D ASCII 字符指定 { F 。
- C = v ASCII 字符指定 { F D Z 一字 Z (例如, 0x23)。
  - Q { F D Z 一字 Z 分 n 为 = v k 字 Z , 例如 2 和 3, " R 在 ? v ASCII 字 Z D M k 字 Z 中指定 | G 。
  - h 置 ? v ASCII 字 Z D _ 位 k 字 Z 为 1, | m > } 在指定一 v 服务 TP { F 。继续 C 样例, * < = v 字 Z 指定为 0x12 和 0x13。
- Q # ` D 0 指定为三 v 字 Z D ASCII 字符 { F 。例如, 007。

因此, Q 一 v 0x23 007 D 服务 B 务 L 序 { F 指定为 0x12 0x13 007。

第7章 APPC 入口点

下 P ？分 h v K APPC 过 L D 人 Z c 。

": > i Z 1 ？分 D B Z 函 G K P 关 I 下 P 系统 y a 供 D APPC API D 信息:

- 在 Windows NT O 运行 D 通信服务器
- SNA API M 户机 OS/2 f 、Windows NT f 、Windows 95 f 和 Windows 3.1 f , I 通信服务器/NT z 品 a 供。
- v 人通信 Windows 95 f 和 Windows NT f 。

若在 b 些系统 a 供 D 支 V 中存在；同之处, + 会 a > 。

APPC

您 I 以 + | C作 y P APPC 动词D同 = 入 Z c。此外，您也 I 以通过 + B 件处理放入次级返回 k 字段以及在 **opext** 字段(AP_NON_BLOCKING)中 h 置队 P 级非分 i j 志，发 v 非分 i 动词。

o 法

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

d 入 1 指向动词X制 i D指 k。

返回值

检 i v 错后D主返回 k 和次级返回 k。

C法 5 明

m {： Z 53页 D 『WinAsyncAPPCEx( )』.

APPC G唯一支V SNA API OS/2 M户机D入 Z c。

WinAsyncAPPC( )

b G y P APPC 动词D异=入Z c。一v &C L序如果选择I窗Z来f知完I D信息, &9C此入Z c。v人通信k通信服务器为此入Z c a供K和现P &C L序D兼容性。

o法

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

N数 5明

hwnd S U完I信息D窗Z d z。

vcb 指向动词X制i D指k

返回值

>返回值w确5 w K异=k s G否I功完I。如果k s I功, 则5际返回值G一v d z。如果函}; I功, 则v人通信k通信服务器返回 0。

C法5明

I组I一组 APPC 动词如下:

- **[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 入Z c允许取消动词, +;支V队P级非分i动词。APPC 入Z c支V队P级非分i动词, +;允许&C L序取消动词。

>入Z c;支V队P级非分i动词。如果在异=S Z O指定K队P级非分i j志AP_NON_BLOCKING, v人通信k通信服务器+忽T。1 9C异=入Z c 1, &C L序某一1 L I以在某v对话过L中v 5 P一v未完I D函}。"T启动次级函}形I错误代k AP_CONV_BUSY。如果&C L序想I B件d z来f _ |异=完I, 则| I以

9 C **WinAsyncAPPCEx** 或 **APPC** 入 Z c。**RECEIVE_AND_POST** 和 **RECEIVE_AND_WAIT** 对 O v 内容例外。为K能 d 分 9 C异 = 支 V， v 人通信 k 通信服务器异 = X D 为发 v **RECEIVE_AND_WAIT** 动词以 9 其 Y 作类 F Z **RECEIVE_AND_POST** 动词。_ e X， 1 一 v 异 = **RECEIVE_AND_POST** 或 **RECEIVE_AND_WAIT** 还未完 I 1， & C L 序 I 以在同一 v 对话中发 v 下 P 动词:

- **REQUEST_TO_SEND**
- **GET_TYPE**
- **GET_ATTRIBUTES**
- **TEST_RTS**
- **DEALLOCATE**(AP_ABEND_PROG、AP_ABEND_SVC 或 AP_ABEND_TIMER)
- **SEND_ERROR**
- **TP_ENDED**

b + 9 & C L 序如一 v 服务器能够 9 C 一 v 异 = **RECEIVE_AND_WAIT** 来 S U }]。1 **RECEIVE_AND_POST** 或 **RECEIVE_AND_WAIT** 还未完 I 1， & C L 序仍 I 以 9 C **SEND_ERROR** 和 **REQUEST_TO_SEND**。

1 完 I 异 = Y 作 1， & C L 序窗 Z *hWnd* S U = 带 P 『**WinAsyncAPPC**』 作为 d 入字符串 D **RegisterWindowMessage** 返回信息。*wParam* d ? | 含 I 原 < 函 } w C 返回 D 异 = 任务 d z。*IParam* d ? | 含 K 原 < VCB 指 k " I C 以确定最后 D 返回 k。

WinAPPCCancelAsyncRequest 允许一 v & C L 序取消任何 D 异 = APPC Y 作， + 须 J 1 X 终止相关 D 对话或 B 务 L 序。任何未完 I D Y 作都 + 返回 AP_CANCELED 作为返回 k。

如果函 } 返回 I 功，则 v 人通信 k 通信服务器会在 Y 作完 I 或取消对话 1 + **WinAsyncAPPC()** 消息传] = & C L 序。

m {:

Z 53页 D 『WinAsyncAPPCEx( )』.

Z 55页 D 『WinAPPCCancelAsyncRequest( )』.

WinAsyncAPPCEx( )

b G y P APPC 动词D异=入Z c。 9 C此w C能9多v会话在同一v线L中；处理。

如果希望 I B件+完 I D信息通知& C L序R您D & C L序X能够取消未完 I D动词，则9 C >入Z c；否则k 9 C APPC 队P级非分 i 入Z c。

o法

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

N数 5明

handle
& C L序+要H待D B件d z。

vcb 指向动词X制 i D指k

返回值

>返回值+ 5 w异= k s G否 I 功。如果函 } I 功，则5际返回值G一 v d z。如果函 } ' \，则v人通信k通信服务器返回 0。

C法5明

此动词+在 Win32** API 中D **WaitForMultipleObjects** 中9 C。

I 组 I 一组D APPC 动词如下:

- **[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**

>入Z c；支V队P级非分 i。如果在异= S Z O指定K队P级非分 i j志AP_NON_BLOCKING，则v人通信k通信服务器+对其忽T。 1 9 C异=入Z c 1，& C L序某一1 L D某v对话过L中只能5 P一v未完 I D函 }。启动次级函 } 会<致错误代k AP_CONV_BUSY。

WinAsyncAPPCEx 入Z c 允许取消动词, + ；支V队P级非分 i 动词。**APPC** 入Z c 支V队 P 级非分 i， + ；允许& C L 序取消动词。**RECEIVE_AND_POST** 和 **RECEIVE_AND_WAIT** 对O v 内容例外。为K能 d 分 9 C异= 支V， v 人通信 k 通信服务器异 = X + y 发 v D **RECEIVE_AND_WAIT** 动词 d 为其 Y 作类 F Z **RECEIVE_AND_POST** D动词。 _ e X 5， 1 一 v 异= **RECEIVE_AND_POST** 或 **RECEIVE_AND_WAIT** 还未完 I 1， & C L 序 I 以在同一 v 对话中发 v 下 P 动词:

- **REQUEST_TO_SEND**
- **GET_TYPE**
- **GET_ATTRIBUTES**
- **TEST_RTS**
- **DEALLOCATE**(AP_ABEND_PROG、AP_ABEND_SVC 或 AP_ABEND_TIMER)
- **SEND_ERROR**
- **TP_ENDED**

b + 9 & C L 序， X p G 一 v 服务器& C L 序，能够 9 C 一 v 异 = **RECEIVE_AND_WAIT** 来 S U }] 。 1 **RECEIVE_AND_POST** 或 **RECEIVE_AND_WAIT** 还未完 I 1， & C L 序仍 I 以 9 C **SEND_ERROR** 和 **REQUEST_TO_SEND**。

1 异 = Y 作完 I 1， v 人通信 k 通信服务器利C B 件D 信号来通知& C L 序。 1 & C L 序 S U 信号 1， | + 检 i 主返回 k 和次级返回 k 以 R v 错误 u 件。

m { **:**

Z 51页 D 『WinAsyncAPPC( )』.

Z 55页 D 『WinAPPCCancelAsyncRequest( )』.

Z 50页 D 『APPC』.

WinAPPCCancelAsyncRequest( )

>函}+取消一v未完ID基Z **WinAsyncAPPC** Dks。

o法

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

N数　5明

handle

ya供DN}; 5w对取消ksDdz。

返回值

>返回值+5wG否取消K异=ks。如果其值为 0，则v人通信k通信服务器已取消KCks。否则，返回值+G下P错误代k之一:

WAPPCINVALID

指定D异=任务 ID 无效。

WAPPCALREADY

要取消D异=例行L序已完I。

C法5明

一v&CL序I以通过发v **WinAPPCCancelAsyncRequest()** wC"指定Idz中启动D函}返回D异=B件，来取消一v曾-在完I某v **WinAsyncAPPC** 功能0发vD异=任务。

如果未完ID动词和某v对话(例如 **SEND_DATA** 或 **RECEIVE_AND_WAIT**)相关，则v人通信k通信服务器+e}b些动词"M放会话。如果动词和一vB务L序(例如 **RECEIVE_ALLOCATE** 或 **TP_STARTED**)相关，则v人通信k通信服务器+axCB务L序。在b=种iv中，d然v人通信k通信服务器会+对话和会话！I能XM放I;，+|仍无法+发M缓ex、H待确认或其|暂挂Y作全？e}t。此wCG同=D。在0fyvD处理完I后，v人通信k通信服务器+传v动词取消完ID信息。

如果T图取消一v现PD异= **WinAsyncAPPC** 例行L序'\"zz WAPPCALREADY 错误代k，则原<例行L序已完I。&CL序要4已处理KCa果通知，要4还未处理{v通知。通过 APPC 队P级非分i入Zc来取消一v异=动词G;I能D。

m{: Z51页D『WinAsyncAPPC( )』.

WinAPPCCancelBlockingCall( )

b v函} + 取消其线L中y P还未完 I D分 i Y作。如果v人通信k通信服务器取消K一v未完 I D分 i w C， | + z I AP_CANCELLED 错误代k。 k v在一v分i挂起D函}中9 C此项w C。 v人通信k通信服务器向此函} a供K现P L序D反向兼容性。

o法

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

返回值

>返回值 + 5 w取消 k s G否 I 功。如果其值为 0，则v人通信k通信服务器取消K C k s。否则， C值 + 会G下P错误代k:

WAPPCINVALID
5 w b里； P未完 I D分 i w C。

C法5明

如果未完 I D动词和某v对话(例如 **SEND_DATA** 或 **RECEIVE_AND_WAIT**)相关，则 v人通信k通信服务器 + e } b些动词 " M放会话。如果动词和一v B务L序(例如 **RECEIVE_ALLOCATE** 或 **TP_STARTED**)相关，则v人通信k通信服务器 + a x C B务L序。在b = 种 i v中， d然v人通信k通信服务器会 + 对话和会话！ I 能X M放 I ;， + | 仍无法 + 发M缓 e x、 H待确认或其 | 暂挂Y作全？ e } t。此w C G同 = D。在0 f y v D处理完 I 后， C函} + a x。

一v多线L D & C L序 I 以P多v分 i Y作未完 I， + ? v线L只能P一v。若要x分多v未完 I w C， **WinAPPCCancelBlockingCall()** &取消 1 0未完 I Y作、w C或 & C L序线L(如果存在); 否则 | + ' \。 1一v Y作还未完 I 1， APPC + 暂停对& C L序线L D w C。 a果，启动分 i Y作D线L + ;能再次获C X制 (也M G 5， C线L + 无法向 **WinAPPCCancelBlockingCall()** 发v w C) } 非& C L序曾 - 9 C **WinAPPCSetBlockingHook** 对C线L D分 i 挂起x行K注a。

b + ; \ Windows、Windows 95 和 Windows NT SNA API M户机D支V。

WinAPPCCleanup( )

C函 } 从 APPC API 终止一 v & C L 序 " 取消K | D注 a 。

o 法

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

返回值

C返回值 + 5 w取消注 a G否 I 功。如果C值 ; G 0, 则m > v 人通信 k 通信服务器 I 功X取消K对& C L 序D注 a 。如果 v 人通信 k 通信服务器还 ; P 对& C L 序取消注 a , | 返回D值 + G 0。

C法 5 明

9 C **WinAPPCCleanup()** 从 APPC API 取消K对 v 人通信 k 通信服务器& C L 序D注 a 。

v 人通信 k 通信服务器 + 终止仍在活动D对话 " a x B 务 L 序。此函 } H 价 Z 在& C L 序 y 5 P D B 务 L 序O发 v **TP_ENDED(HARD)**。

m { : Z 59页 D 『 WinAPPCStartup( ) 』 .

WinAPPCIsBlocking( )

> 函 } + 确定一 v 线 L 在 H 待 0 一 v 分 i w C a x 1 G 否 } 在执行。 v 人通信 k 通信服务器为 k 现 P L 序 D 反向兼容性而 a 供 K b v 函 }。

o 法

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

返回值 + 5 w C 函 } a 果。如果 C 值 ; G 0, 则 m > 一 v 未完 I D 分 i w C } 在 H 待完 I。值为 0 则 m > ; P 未完 I D 分 i w C。

C 法 5 明

v 人通信 k 通信服务器 DLL { 止一 v 线 L P 一 v 以 O 未完 I D 分 i w C; 如果发 z b 类 i v + 返回 AP_THREAD_BLOCKING。一 v } 在执行分 i w C D 线 L G ; 能重新 * < D, } 非已 h 置 K 分 i 挂起函 } 能。在 b 种 i v 下, **WinAPPCIsBlocking** 只能从一 v 分 i 挂起函 } 内返回 true。

m {:

Z 56 页 D 『WinAPPCCancelBlockingCall( )』.

Z 60 页 D 『WinAPPCSetBlockingHook( )』.

Z 61 页 D 『WinAPPCUnhookBlockingHook( )』.

WinAPPCStartup( )

> 函 } 能 9 一 v & C L 序指定 v 人通信 k 通信服务器 y 需 D f >， " 检 w v 人通信 k 通信服务器 D f > 信息。此 w C " 非 G X 需 D。

o 法

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

N数　5 明

wVersionRequired

指定 K v 人通信 k 通信服务器支 V y 需 D f >。 _ 位字 Z 指定 K 1 f >(校订 >)号； M位字 Z 指定 K 主 f > 号。

wappcdata

返回 APPC API D f > " 对 API D 5) 作 v 5 w。

返回值

返回值 + 5 w v 人通信 k 通信服务器 G 否已 I 功 X 注 a K & C L 序以及 G 否支 V y 指定 D f > 号。如果其值为 0， m w v 人通信 k 通信服务器支 V 指定 D f > " 已 I 功注 a K & C L 序。否则， + 返回下 P 值中 D 一 v :

WAPPCSYSNOTREADY

基 > 网 g 子系统 D 网 g 通信还未准 8 M绪。

WAPPCVERNOTSUPPORTED

C X 定 v 人通信或通信服务器 D 5) ；支 V v 人通信或通信服务器支 V y 要 s D f >。

WAPPCINVALID

v 人通信 k 通信服务器无法确定指定 D f >。

C 法 5 明

WinAPPCStartup() 希望 P 助 Z 和 API 以后 D 发行 f 兼容。此 DLL 支 V f > 1.0。

m { : Z 57页 D 『 WinAPPCCleanup( ) 』 .

WinAPPCSetBlockingHook( )

> 函 } 9 APPC API D APPC 5 现能够分 i w C APPC 功能。

v 人通信 k 通信服务器向 > 函 } a 供 K 和现 P & C L 序 D 兼容性。

o 法

```
FARPROC WINAPI WinAPPCSetBlockingHook(FARPROC lpBlockFunc);
```

N数 5 明

lpBlockFunc

w 确 5 w K + 要 2 装 D 分 i 函 } D 过 L 5 例 X 址。

返回值

返回值指向以 0 y 2 装 D 分 i 函 } D 过 L 5 例。w C **SetBlockingHook** 函 } D & C L 序或 b & C # 存 C 返回值，从而 9 | 能在 X 要 1 还原。(如果 6 W ; 重要 D 话，& C L 序 I 以简 % X 废弃 I **WinAPPCSetBlockingHook()** 返回 D 值 " 最终 9 C **WinAPPCUnhookBlockingHook** 恢 4 = 缺 ! 机制。)

C 法 5 明

分 i 函 } 如果 S U = 一 v WM_QUIT 信息则 + 返回 FALSE, b 样 v 人通信 k 通信服务器 M I 以 Q X 制权返回至 & C L 序，让 | 来处理 C 信息 " 自然终止。否则，C 函 } + 返回 TRUE。

; P 5 现缺 ! D 分 i 挂起。如果一 v & C L 序 ; P h 置分 i 挂起，则 C & C L 序线 L + 无法确 P X H 待分 i w C 返回。

如果分 i 挂起函 } 返回 TRUE, 则 + 分 i 动词以及主返回 k AP_CANCELLED 返回至 & C L 序。

b 项功能 I 线 L 来 5 现。| a 供 K 一 v X b D 线 L 来 f 换分 i 机制，而 ; 0 响其 | 线 L。

m { :

Z 56 页 D 『WinAPPCCancelBlockingCall( )』.

Z 58 页 D 『WinAPPCIsBlocking( )』.

Z 61 页 D 『WinAPPCUnhookBlockingHook( )』.

b + ; \ Windows 95 和 Windows NT SNA API M 户机 D 支 V。

WinAPPCUnhookBlockingHook( )

> 函 } + > } y P 曾 - 2 装过 D 分 i 挂起。

v 人通信 k 通信服务器向 > 函 } a 供 K 对现 P L 序 D 反向兼容性。

o 法

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

返回值

返回值 + 5 w 函 } a 果。如果 v 人通信 k 通信服务器重新 2 装缺 ! 机制 I 功，则其值 ; 为 0。如果 v 人通信 k 通信服务器未重新 2 装缺 ! 机制，则其值为 0。

C 法 5 明

1 w C 函 } 后，C & C L 序线 L + 无法确 P X H 待未来 y P 分 i w C D 完 I 。

m { : Z 60 页 D 『WinAPPCSetBlockingHook( )』.

b + ; \ Windows 95 和 Windows NT SNA API M 户机 D 支 V。

GetAppcConfig( )

> 函 } 无法 5 现。 + G， 人 Z c a 供 K 反向兼容性。如果指定 K 一 v P 效 D N } 集，则 v 人通信 k 通信服务器 + 返回 APPC_CFG_SUCESS_NO_DEFAULT_REMOTE， " + 一 v NULL 端 S 器放入 RemLu 缓 e x D W字 Z 中。

在许多 i v 下 C w C ； G X 需 D，因为 v 人通信 k 通信服务器 G _ P APPN 能 & D Z c。 I 以在 ALLOCATE O 指定伙 i LU { " 启动对 LU D Q w。 + G， & C L 序 I 以 9 C a c 运 c 功能(Node Operator Facility(NOF)) S Z 来检 w C 信息。 { 知 P 关如何 9 C NOF S Z D 信息， k N < *系统管理 L 序 h 计*。

GetAppcReturnCode( )

> 函 } + VCB 中 D 主和次级返回 k 转换 I I 打 ! 字符串。 | a 供 K APPC & C L 序 9 C D v 错字符串 j 准集。

o 法

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

N数 5 明

vcb a 供 D N } ; 指定 K 动词 X 制 i D X 址。

buffer_length
a 供 D N } ; C Z 指定 I **buffer_addr** 指向 D 缓 e x $ 度。推荐 $ 度为 256。

buffer_addr
a 供/返回 D N } ; C Z 指定 + | 含 q = 化 D、U a 尾字符串 D 缓 e x D X 址。指定缓 e x 内字符串 D $ 度。

返回值

0x20000001
> N } 无效; 函 } 无法读取指定 D 动词 X 制 i 或无法写入指定缓 e x 。

0x20000002
指定 D 缓 e x + 小。

C 法 5 明

在 **buffer_addr** 中返回 D h v 错误字符串 " ; 以新行符(**\n**)终止。

第8章 APPC 动词

> B 为在 APPC API g 越传] D ? v 动词D o 法 a 供K引证， " h v ? v 动词在传] 中 D N } k 其返回 N } 。

> B 还 a 供K P 关 APPC 基 > 动词 k 为 APPC + 工和 k + 工对话 a 供D 3 d 会话动词 D N < 信息。阅读 > B 1， 您 + 发现非 # 相 F D 基 > k 3 d 动词，以及为何 | G 会； U 集在同一 v B Z 中。然而，仍然存在一些 x p 。下 f m > K那些 n 异:

此符号 1 信息 v & C Z 一 v 基 > 动词 1 v 现。

此符号 1 信息 v & C Z 一 v 3 d 动词 1 v 现。

再次澄清! 1 对话动词即 I 为基 > V I 为 3 d D 1， m > 为:

[MC_]VERBNAME

": Z 1 B D B Z | 括D内容函G K P 关 I 下 P 系统 y a 供D APPC API D信息:

- 运行在 Windows NT O D通信服务器
- SNA API M户机 OS/2 f 、Windows NT f 、Windows 95 f 以及 Windows 3.1 f， | G k 通信服务器一起发M
- v 人通信 Windows 95 f k Windows NT f

1 在 I b 些系统 a 供D 支V 之间 P n 异 1， 会 v 现 a > 。

动词控制块

> ? 分 | 含 ? v 动词D字段 k 指 > D 一 c h v 。

公共V段

? v VCB 都 P 一定 } ? D公共字段，如下 y >:

opcode

动词Y作 k: | 含动词 { D j 6 字段。

format

j 6 VCB D q = 。 > 字段为K指定 VCB D 1 0 f > 而X须 h 置D值；%独记 < 在 ? v 动词下。

primary_rc

主返回 k 。 ? v 动词 I 能D值在下 P ? 分中 P v 。

secondary_rc

次返回 k 。 | 9 d K主返回 k a 供D信息。 ? v 动词 I 能D值在下 P ? 分中 P v 。 P 些 VCB 也 | 含下 P 字段。

opext 动词扩 9 代 k 。 | 9 d K动词Y作 k y a 供D信息。 若动词信号 + 在非分 i 方 = 中处理， M X须 h 置 AP_NON_BLOCKING j 志。下 f h v D信号中 | 括 b 些公共字段， + ；%独 b M | G 。

TP 标识

为？v活动D B务L序指定一v 8 字Z B务L序j 6符。C j 6符I v人通信k通信服务器指定。

B务L序j 6符C Z 7 I **TP_ENDED** 动词, " R作为对话动词O D相关因子。

？v信号D动词X制i在下P ？分中h v。

APPC API 支持

动词支持

v人通信k通信服务器在 APPC API 支V下P动词。

`M独"动词

GET_TP_PROPERTIES
GET_TYPE
RECEIVE_ALLOCATE
TP_ENDED
TP_STARTED

对话动词

[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_PROPERTIES 返回 k B务L序关 * D t 性。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N } a 供x v 人通信k通信服务器:

opcode
AP_GET_TP_PROPERTIES

tp_id > X B务L序D j 6符。 > N } 值I } 在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

opext AP_BASIC_CONVERSATION

format
j 6 VCB D q =。 h置>字段以指定O f P v D VCB f >。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N }:

primary_rc
AP_OK

tp_name
> X B务L序{， MG发v C动词D B务L序。 v人通信k通信服务器；检i C字段D字符集。

lu_alias
k B务L序关 * D > X LU p {。 b在> X显>字符集中G一v 8 字Z字符串。 y P 8 v字Z都G P意义D， " R X须h置。

luw_id 字段G k ; \ # 护对话（ k AP_NONE D **sync_level** 或 AP_CONFIRM_SYNC_LEVEL D对话）关* D _辑工作%元j 6。 **luw_id_overlay** | 含下P N }:

luw_id_overlay.fq_lu_name_len
k _辑工作%元关* D全限定 LU { $度。

luw_id_overlay.fq_lu_name
k _辑工作%元关* D全限定 LU {。 b v { F D $度最多为 17 v字Z, " R以 EBCDIC U q R n d。 | I = v type-A EBCDIC 字符串以一v EBCDIC c分t而构I。（？v { F D最大$度G 8 字Z, ; P 6入U q。若网g ID ; 存在, 则！T那v c。） 若{ F $度小Z 17 字Z, 则 **instance** 和 **sequence** t S着{ F（注意, b m > LUW_ID_OVERLAY a构D字段即无法访问 **instance**, 也无法访问 **sequence**）。

luw_id_overlay.instance
_辑工作%元5例号。 b G一v 6 字Z $ D二x制字符串。

luw_id_overlay.sequence
_辑工作%元序P号。 b G一v 2 字Z $ D二x制字符串。

luw_id_overlay.fq_lu_name_len 小Z 17, **luw_id_overlay** 以 EDCDIC U q R n d（在 **instance** k **sequence** 后f）。

luw_id_overlay.fq_lu_name
k B务L序关* D > X LU 全限定{。 b v { F 17 字Z $, " R R _以 EBCDIC U q n d。 | I = v A 型 EBCDIC 字符串以一v EBCDIC c分t而构I。 （？v { F D最大$度G 8 字Z, ; P 6入U q。若网g ID ;存在, 则！T那v c。）

user_id
B务启动L序D C户 ID。 b G 10 v字Z D AE 型 EBCDIC 字符串, 以 EBCDIC U q n d。

若动词I Z N }错误而未执行, 则v人通信k通信服务器返回下P N }:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_TP_ID

z I下P P I能性D主返回k（**primary_rc**）D u件h v在= < A. APPC 公共返回k中。

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

GET_TYPE

GET_TYPE 动词返回X定对话D对话类型（基>或3 d）。

VCB a构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_GET_TYPE

opext AP_BASIC_CONVERSATION

format

j 6 VCB D q =。h置>字段以指定O f P v D VCB f >。

tp_id >X B务L序D j 6符。 >N }值I }在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。 >N }值I }在w C B务L序中D **ALLOCATE** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N }:

primary_rc

AP_OK

conv_type

对话D对话类型I **conv_id** j 6。

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

conv_style

对话D对话风q I **conv_id** j 6。C字段需要 VCB D q = 1 f >。k N阅Z37页D『全+工 VCB』，以获C访问q = 1 VCB |详细D内容。

AP_HALF_DUPLEX
AP_FULL_DUPLEX

GET_TYPE

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_TP_ID

AP_BAD_CONV_ID

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

RECEIVE_ALLOCATE

RECEIVE_ALLOCATE 动词需 s 需要 (" 新 B 务 L 序 D 信息， C B 务 L 序为 k 已发 v **ALLOCATE** 或 **MC_ALLOCATE** 动词 D 伙 i B 务 L 序对话。

VCB a 构

```
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;
```

提供的 N 数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode
AP_RECEIVE_ALLOCATE

opext AP_BASIC_CONVERSATION

format
j 6 VCB D q = 。 + > 字段 h 置为 0, 以指定 O v P v D VCB f > 。

tp_name
B 务 L 序 { 。 v 人通信 k 通信服务器 ；检 i C 字段 D 字符集。

返回 N 数

若动词执行 I 功, 则 v 人通信 k 通信服务器返回下 P N } :

primary_rc
AP_OK

tp_id > X B 务 L 序 D j 6 符。 C 值 I v 人通信 k 通信服务器分配 x B 务 L 序。 B 务 L 序在 y P f 后 D APPC 动词 O + C j 6 传] x v 人通信 k 通信服务器。

RECEIVE_ALLOCATE

conv_id

对话 j 6。 C值 j 6在= v B务L序间 (" D对话。

sync_level

对话D同 = 化级 p。

AP_CONFIRM_SYNC_LEVEL

AP_NONE

conv_type

对话D对话类型 I **conv_id** j 6。

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

user_id

伙 i B务L序 a 供D C户 ID。 b G 10 v字Z D AE 型 EBCDIC 字符串，以 EBCDIC U q n d。

lu_alias

> X LU 已知D p { 。 b在> X显> 字符集中G一 v 8 字Z字符串。 y P 8 v字Z都G P意义D， " R X须h置。

plu_alias

伙 i LU 对Z > X B务L序已知D p { 。 b在> X显> 字符集中G一 v 8 字Z 字符串。 y P 8 v字Z都G P意义D， " R X须h置。

mode_name

在配置期间定义D一组，网X性 { 。 b G一 v 8 字Z D字母 } 字类型 - 一 v EBCDIC 字符串（以一 v 字母 * 头）， R _ n d EBCDIC U q。

conv_group_id

此对话 } 在 9 C D会话D对话组 j 6。

fqplu_name

伙 i LU D全限定 LU { 。 b v { F 17 字Z $， " R R _以 EBCDIC U q n d。 | I = v A 型 EBCDIC 字符串以一 v EBCDIC c分 t 而构 I。 （? v { F D最大 $ 度G 8 字Z， ; P 6入U q。 若网 g ID ;存在，则！T那 v c。）

pip_incoming

指定G否伙 i B务L序 a 供D L序 u < 化N }（PIP）位Z [MC_]ALLOCATE k s O。 + 此h置为 AP_YES 或 AP_NO。若为 AP_YES，则 PIP }] + 在> 对话发 v D Z一 v **[MC_]RECEIVE_*** 动词O S U =。

conversation_style

对话D对话风 q I **conv_id** j 6。

AP_HALF_DUPLEX
AP_FULL_DUPLEX

password

k **user_id** 关 * D Z n。 b G 10 v字Z D AE 型 EBCDIC 字符串，以 EBCDIC U q n d。 若 Security=Program（AP_PGM 或 AP_PGM_STRONG），则 b G X需D；否则， | 为 I 选项。

dload_id

> 字段 1 R v 1 格式字段 h 置为 1 1 E 能 h 置。若 RECEIVE_ALLOCATE 作为对 DYNAMIC_LOAD_INDICATION D响&发 v，则 > 字段 I 通过下 P 方法 C Z + = v 信号关 * 起来。

若 **dload_id** h 置为下 P 之一，则 RECEIVE_ALLOCATE + v k DYNAMIC_LOAD_INDICATION 关 *：

- 全 c
- DYNAMIC_LOAD_INDICATION O D **dload_id** 字段。

"：在 SNA API M户机 O；支 V C N }。

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_UNDEFINED_TP_NAME

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_UNEXPECTED_SYSTEM_ERROR

TP_ENDED

TP_ENDED 动词通知 v 人通信 k 通信服务器一指定 D B 务 L 序已 - a x。

VCB a 构

```
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;
```

提供的N数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode
AP_TP_ENDED

opext AP_BASIC_CONVERSATION

format
j 6 VCB D q =。 + > 字段 h 置为 0，以指定 O v P v D VCB f >。

tp_id > X B 务 L 序 D j 6 符。 > N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 动词返回。

type **TP_ENDED** D 类型。

AP_HARD
AP_SOFT
AP_ABEND
AP_CANCEL

若类型为 AP_ABEND，则 v 人通信 k 通信服务器；对 **TP_ENDED** 信号作回答。

返回N数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_OK

返回N数

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_TP_ID

AP_BAD_TYPE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

TP_STARTED

TP_STARTED 动词对 v 人通信 k 通信服务器发 v 通知，通知一 v L 序已 - 为B 务 L 序 k s 资源， C B 务 L 序因为 > X | n （非 x 入分配 k s ）而启动。

VCB a 构

```
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;
```

提供的N数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_TP_STARTED

opext AP_BASIC_CONVERSATION

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定O v P v D VCB f > 。

lu_alias

> X LU 已知D p { 。若 | h 置为 c，则 v 人通信 k 通信服务器 9 C X 制 c LU。 b 在 > X 显 > 字符集中G 一 v 8 字 Z 字符串。 y P 8 v 字 Z 都G P 意义 D， " R X 须 h 置。 S \ UWD **lu_alias** 字段。X 制 c LU 在 > 例中 9 C。

下 P 信息 v J C Z 通信服务器 Windows 95 f 和 Windows NT SNA API M户机。

I 通过 9 C J 1 D配置 5 C L 序（INI 配置或 LDAP），以 c 指定 ? v C 户 D 系统 h 定 > X LU p { 。

APPC L 序 I 选择 9 C 系统 h 定 > X LU p {，而 " 非直 S 指定一 v。 1 APPC L 序发 v 一 v **TP_START** 动词 " R **local_LU_alias** 字段 h 置为二 x 制 c 1，APPC API 9 C配置K D 系统 h 定 > X LU p { 。

tp_name

B 务 L 序 { 。 v 人通信 k 通信服务器 ；检 i C字段D字符集。

返回N数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OK

tp_id > X B 务 L 序 D j 6 符。 C 值 I v 人通信 k 通信服务器分配 x B 务 L 序。 B 务 L 序在 y P f 后 D APPC 动词 O + C j 6 传] x v 人通信 k 通信服务器。

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_INVALID_LU_NAME

AP_INVALID_ENABLE_POOL

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_UNEXPECTED_SYSTEM_ERROR

[MC_]ALLOCATE

[MC_]ALLOCATE 动词 I } 在 w C D B 务 L 序发 v。C 动词在 > X LU 和伙 i LU 之间分配会话，然后（k **RECEIVE_ALLOCATE** 动词一起）在 } 在 w C D B 务 L 序 k 已 w C D B 务 L 序之间 (" 对话。

ALLOCATE 动词 I (" 基 > 会话，或 3 d 会话。9 C **ALLOCATE** 动词 (" 3 d 会话，9 B 务 L 序能够 9 C 基 > 会话动词 k 3 d 会话伙 i B 务 L 序通信。

> 动词执行 I 功 1，I v 人通信 k 通信服务器 z I 一 v 对话 j 6（**conv_id**）。b v j 6 G y P 其 | APPC 对话动词 y X 需 D N }。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_ALLOCATE 

AP_M_ALLOCATE 

format

j 6 VCB D q =。 + >字段h置为 0，以指定O v P v D VCB f >。

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作， > j志I k AP_NON_BLOCKING x行“或”运c。

tp_id > X B务L序D j 6符。

> N } 值I }在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 动词返回。

conv_type 

需要分配D对话类型。

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

若 **ALLOCATE** 动词(" 一v 3 d会话，则> X B务L序X须发v基>会话动词， " R a供其自C 3 d c，以c 9 }] 记< k _辑记<能够互相转换。伙i B务L序I发v基>会话动词， " a供3 d c，或|能够9 C 3 d会话动词（若伙i B务L序}在9 C D APPC 5现支V 3 d会话动词）。如需x一=D信息， k N阅 *IBM 系统网g e系a构: LU 6.2 N <大全: 同级协议*。

sync_level

对话D同=化级p。

AP_CONFIRM_SYNC_LEVEL
AP_NONE

rtn_ctl

指定在来自> X B务L序D会话k s O活动D > X LU，何1 + X制返回= > X B务L序。

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

对话D对话风q I **conv_id** j 6

AP_HALF_DUPLEX
AP_FULL_DUPLEX

在通信服务器 SNA API M户机 OS/2 f 和 Windows 3.1 f O；支V。

conv_group_id

+ 要分配D会话D对话组 j 6。 v 1 **rtn_ctl** h 置为 AP_WHEN_CONV_GROUP_ALLOC 1 E a供> N }。

plu_alias

伙i LU 对Z > X B务L序已知D p {。 b在> X显>字符集中G一v 8 字Z字符串。 y P 8 v字Z都G P意义D, " R X须h置。 > { X须k在配置期间(" D伙i LU {匹配。若C字段h置为全c, 则v人通信k通信服务器9 C **fqplu_name** 字段来指定X需D伙i LU。

下P信息v J C Z通信服务器 Windows 95 f k Windows NT SNA API M户机。

I通过9 C J 1 D配置5 C L序（INI 配置或 LDAP），以c指定？v C户D系统h定伙i LU p {。

APPC L序I选择9 C系统h定伙i LU p {，而" 非直S指定一v。 1 APPC L序发v一v **ALLOCATE** 动词" R **partner_LU_alias** 字段和 **fully_qualified_partner_LU** 字段h置为二x制D c 1， APPC API 9 C - 配置K D系统h定伙i LU p {。

mode_name

通#在配置期间定义D一组，网X性{。 b G一v 8 字Z D字母}字类型 - 一v EBCDIC 字符串（以一v字母 * 头）， R _ n d EBCDIC U q。

tp_name

；w C D B务L序{。 v人通信k通信服务器；检i C字段D字符集。 I **ALLOCATE** 动词在} w C B务L序中指定D **tp_name** 值X须k I **RECEIVE_ALLOCATE** 动词在已w C D B务L序中指定D **tp_name** 值匹配。

security

指定伙i LU 需s D信息，以c验证对Z已w C D B务L序D访问。

AP_NONE

已w C D B务L序；9 C对话2全性。

AP_PGM

已wCDB务L序9C对话2全性，要s一vC户 ID 和Zn。

AP_SAME

已wCDB务L序9C对话2全性，"R配置为S\一v-验证D指>符。C户 ID +k一v-验证D指>符一起发M，通知wCDB务L序；需要Zn。

AP_PGM_STRONG

k AP_PGM 相同，+ ALLOCATE +v1=伙i LU D会话支VZnf代1E会I功。

"：若 [MC_]ALLOCATE 指定 AP_SAME D类型，+"未指定C户 ID kZn，9C在以0D SET_TP_PROPERTIES 动词中（若存在）指定DC户 ID kZn。若 [MC_]ALLOCATE 未携带一vC户 ID kZn，则b些+1#CZ任何在 SET_TP_PROPERTIES 动词ODiv。

pwd k **user_id** 关*DZn。bG 10 v字ZD AE 型 EBCDIC 字符串，以 EBCDIC Uqnd。若 Security=Program（AP_PGM 或 AP_PGM_STRONG），则bGX需D；否则，|为I选项。

user_id
C户 ID 需要访问伙iB务L序。bG 10 v字ZD AE 型 EBCDIC 字符串，以 EBCDIC UqndD。若 Security=Program（AP_PGM 或 AP_PGM_STRONG），则bGX需D；否则，|为I选项。

pip_dlen
+传]=伙iB务L序DL序u<化N}（PIP）$度。范围: 0-32767

pip_dptr
|含 PIP }]D缓exX址。v1 **pip_dlen** 大Zc19C>N}。

fqplu_name
伙i LU D全限定 LU {。bv{F 17 字Z$，"RR_以 EBCDIC Uqnd。|I=v A 型 EBCDIC 字符串以一v EBCDIC c分t而构I。（?v{FD最大$度G 8 字Z，；P6入Uq。若网g ID ；存在，则！T那vc。）1Rv1 **plu_alias** 字段h置为c1，>字段EGP意义D。

返回N数

若动词执行I功，则v人通信k通信服务器返回下PN}:

primary_rc
AP_OK

conv_id
对话j6。 C值j6在=vB务L序间("D对话。

conv_group_id
分配x对话D会话D对话组j6。

若动词G非阻塞D"R还未完I，则v人通信k通信服务器返回下PN}:

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若 **rtn_ctl** N } h 置为 AP_IMMEDIATE, " R ; P " 即 I C D 会话，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc
AP_UNSUCCESSFUL

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N } :

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
a 供为何 **[MC_]ALLOCATE** ' \ D = 加信息。

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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

若 **primary_rc** h 置为 AP_ALLOCATION_ERROR, 则 **sense_data** 字段能够携带 | 多 ' \ D 信息。

[MC_]CONFIRM

CONFIRM 动词 + > X LU 发M缓 e x D内容 k 确认 k s 发M = 伙 i B务 L 序。作为对 **CONFIRM** 动词D响&，伙 i B务 L 序通 # 发 v **CONFIRMED** 动词，以 c 确认 S U = K无误D }]。(若伙 i B务 L 序 v = 一处 v 错，则 | 发 v **SEND_ERROR** 动词或异 # M放对话。)

1 R v 1 I **ALLOCATE** (" D对话同 = 化级 p 为 AP_CONFIRM_SYNC_LEVEL 1，B务 L 序 E I 发 v **CONFIRM** 动词。

VCB a 构

```
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;
```

提供的N数

B务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_CONFIRM 

AP_M_CONFIRM 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作，> j 志 I k AP_NON_BLOCKING x 行“或”运 c 。

format

j 6 VCB D q = 。 h 置 > 字段以指定 O f P v D VCB f > 。

tp_id > X B务L序D j 6符。 > N } 值 I } 在w C B务L序中D **TP_STARTED** 动词返回，或 I 已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id
对话 j 6。 > N } 值 I } 在w C B务L序中D **ALLOCATE** 动词返回，或 I 已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_OK

rts_rcvd
Request-to-send-received 指 > 符。

AP_YES
AP_NO

expd_data_rcvd
Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0， > 指 > 继续 h 置为 AP_YES。

AP_YES
AP_NO

> 字段需 s q = 1 f > D VCB。 k N阅 Z 37页D『全 + 工 VCB』，以获C访问 q = 1 VCB | 详细D内容。

若动词G非阻塞D " R还未完 I，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_OPERATION_INCOMPLETE

opext 若动词G非阻塞D " R还未完 I，则 v 人通信 k 通信服务器返回下 P N }:

AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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

若 1 B务L序发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_CONFIRM_BAD_STATE

AP_CONFIRM_NOT_LL_BDY 

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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 

AP_PROG_ERROR_PURGING

AP_SVC_ERROR_PURGING 

AP_CONVERSATION_TYPE_MIXED 

AP_UNEXPECTED_SYSTEM_ERROR 

AP_TP_BUSY
AP_CANCELLED

MC_CONFIRM

": I Z性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功 D返回 k，而无需 + 其转发 = 服务器。 1 发 v 一 v f 后D **[MC_]CONFIRM** 动词 1， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。 若P **[MC_]CONFIRM** v 错返回 k，则其在 **[MC_]PREPARE_TO_RECEIVE** 动词O返回。 k N阅 **[MC_]SEND_DATA** P m，以获C v 错返回 k D P m。

[MC_]CONFIRMED

CONFIRMED 动词从伙 i B务L序回答一 v 确认 k s。 | 通知伙 i B务L序 > X B务L序未在 S U = D }] 中检 b v 错误。

因为发 v 确认 k s D B务L序H待确认， y 以 **CONFIRMED** 动词 + = v B务L序D处理同 = 化。

VCB a 构

```
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;
```

提供的N数

B务L序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_CONFIRMED 

AP_M_CONFIRMED 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作， > j 志 I k AP_NON_BLOCKING x 行“或”运 c 。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定 O v P v D VCB f > 。

tp_id > X B务L序D j 6 符。 > N } 值 I } 在 w C B务L序中D **TP_STARTED** 动词返回，或 I 已 w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。 > N } 值 I } 在 w C B务处理器中D **ALLOCATE** 动词返回，或 I 已 w C B务处理器中D **RECEIVE_ALLOCATE** 返回。

MC_CONFIRMED

返回N数

若动词执行I功，则v人通信k通信服务器返回下PN}:

primary_rc
AP_OK

若动词G非阻塞D"R还未完I，则v人通信k通信服务器返回下PN}:

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词IZN}错误而未执行，则v人通信k通信服务器返回下PN}:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_CONFIRMED_INVALID_FOR_FDX

若1B务处理器发v>动词1对话处Z错误状，，则v人通信k通信服务器返回下PN}:

primary_rc
AP_STATE_CHECK

secondary_rc
AP_CONFIRMED_BAD_STATE

zI下PPI能性D主返回k（**primary_rc**）Du件hv在=<A. APPC 公共返回k中。

AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

AP_CONVERSATION_TYPE_MIXED

[MC_]DEALLOCATE

DEALLOCATE 动词M放 = v B务L序之间D对话。M放对话之 0， C动词执行D Y作H价Z下P动词D其中之一:

- **FLUSH** 动词 + 伙 i LU D发M缓 e x内容发M = 伙 i LU （B务处理器）。
- **CONFIRM** 动词， + > X LU D发M缓 e x内容 k 确认D k s 发M = 伙 i B务L序。

> 动词执行 I 功后，对话 ID ；再P效。

对Z k + 工对话:

- 从B务L序M放指定D对话， I | 括 **FLUSH** 或 **CONFIRM** 动词D函 } 。

对Z k + 工对话

- 带P **TYPE（FLUSH）** D **DEALLOCATE** 关U > X L序D发M队P。因此， > X和远L L序X须w自关U其发M队P，需要 = v **DEALLOCATE TYPE（FLUSH）** 动词终止对话。注意，已关U其发M队P D伙 i 以 **DEALLOCATE_NORMAL** 返回 k 形 = x h S U队P。
- 带P **TYPE（ABEND）** D **DEALLOCATE** G + 要同 = 关U = 方对话D异 # 终止。> 通知以 **ERROR_INDICATION** 返回 k D形 = 返回 = 远L L序D发M队P， " R以 **DEALLOCATE_ABEND** 返回 k D形 = 返回 = 远L L序D S U队P。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_DEALLOCATE 

AP_M_DEALLOCATE 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作, > j志I k AP_NON_BLOCKING x行“或”运c。

在全+工对话O, > j志X须k AP_FULL_DUPLEX_CONVERSATION x行“或” Y作。 

format

j 6 VCB D q =。h置>字段以指定O f P v D VCB f >。

tp_id > X B务L序D j 6符。 > N } 值I } 在w C B务处理器中D **TP_STARTED** 动词返回, 或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。 > N } 值I } 在w C B务L序中D **ALLOCATE** 动词返回, 或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

dealloc_type

指定如何执行M放。

AP_ABEND 

AP_ABEND_PROG 

AP_ABEND_SVC

AP_ABEND_TIMER

AP_FLUSH

AP_SYNC_LEVEL

下P值v J C Z基>。 

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

发M = 错误记 < 文件D }] 字Z } 。

范围: 0-32767

& C L 序 I + }] = 加 = VCB D a 尾, 在 b 种 i v 下, 此字段 + 大 Z c , " R **log_dptr** X 须 h 置为 NULL。 ($ 度 l 指 > ; 存在错误记 < }] 。)

log_dptr

| 含 v 错信息D }] 缓 e x X址。& C L 序 I + }] = 加 = VCB D末尾, 在 b 种 i v 下, **dptr** X 须 h 置为 NULL。

发M C }] = > X 错误记 < 以及伙 i LU。B 务处理器 X 须 + v 错 }] q = 化为一 c }] w (GDS) 错误记 < d ? 。如需 x 一 = D 信息, k N 阅 *IBM 系统网 g e 系 a 构: LU 6.2 N < 大全: 同级协议*。

返回N数

若动词执行 I 功, 则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0 , > 指 > 继续 h 置为 AP_YES。

> 字段需要 VCB D q = 1 f > 。 k N 阅 Z 37页 D 『全 + 工 VCB』, 以获 C | 多 P 关访问 q = 1 VCB D细 Z 。

AP_YES
AP_NO

若动词 I Z N } 错误而未执行, 则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID

AP_DEALLOC_BAD_TYPE

AP_DEALLOC_LOG_LL_WRONG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }（ v 对 Z 3 d ）：

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若 1 B 务处理器发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc

AP_STATE_CHECK

secondary_rc

AP_DEALLOC_CONFIRM_BAD_STATE

AP_DEALLOC_FLUSH_BAD_STATE

AP_DEALLOC_NOT_LL_BDY 

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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

": I Z性能需要, SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功 D返回 k，而无需 + 其转发 = 服务器。 1发v一v f后D **[MC_]DEALLOCATE** 动词 1， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。 若P **[MC_]DEALLOCATE** v错返回 k，则其在 **[MC_]PREPARE_TO_RECEIVE** 动词 O返回。 k N阅 **[MC_]SEND_DATA** P m，以获C v错返回 k D P m。

[MC_]FLUSH

FLUSH 动词 + > X LU D发M缓 e x 发M = 伙 i LU（以及B务L序）。若发M缓 e x G U D，则；执行任何Y作。

VCB a 构

```
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;
```

提供的N数

B务处理器 + 下P N } a 供为 v 人通信 k 通信服务器:

opcode

AP_B_FLUSH 

AP_M_FLUSH 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作，> j 志I k AP_NON_BLOCKING x行“或”运c。

在全 + 工对话O，> j 志X须k AP_FULL_DUPLEX_CONVERSATION x行“或” Y作。

format

j 6 VCB D q =。 + > 字段h置为 0，以指定O v P v D VCB f >。

tp_id > X B务L序D j 6符。 > N } 值I } 在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6。 > N } 值I } 在w C B务L序中D **ALLOCATE** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N }:

primary_rc
AP_OK

若动词G非阻塞D " R还未完I，则v人通信k通信服务器返回下P N }:

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词I Z N }错误而未执行，则v人通信k通信服务器返回下P N }:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID

若1 B务L序发v > 动词1对话处Z错误状，，则v人通信k通信服务器返回下P N }:

primary_rc
AP_STATE_CHECK

secondary_rc
AP_FLUSH_NOT_SEND_STATE

z I下P P I能性D主返回k（**primary_rc**）D u件h v在= < A. APPC 公共返回k中。

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

"： I Z性能需要， SNA API M户机I在 **[MC_]SEND_DATA** 动词O返回一v I功D返回k，而无需+其转发=服务器。 1发v一v f后D **[MC_]FLUSH** 动词1，**[MC_]SEND_DATA** 转发=服务器，以供处理。 若P **[MC_]FLUSH** v错返回k，则其在 **[MC_]PREPARE_TO_RECEIVE** 动词O返回。 k N阅 **[MC_]SEND_DATA** P m，以获C v错返回k D P m。

[MC_]GET_ATTRIBUTES

GET_ATTRIBUTES 动词返回对话D t 性。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_GET_ATTRIBUTES 

AP_M_GET_ATTRIBUTES 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。

在全+工对话O， > j志X须k AP_FULL_DUPLEX_CONVERSATION x行“或” Y作。

format

j 6 VCB D q =。 + >字段h置为 0，以指定O v P v D VCB f >。

tp_id > X B务L序D j 6符

> N }值I }在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。

> N }值I }在w C B务L序中D **ALLOCATE** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N }:

primary_rc

AP_OK

sync_level

对话D同=化级p。

AP_CONFIRM_SYNC_LEVEL
AP_NONE

mode_name

k分配x对话D会话关*D，网X性组{。 b G一v 8 字Z D字母}字类型 - 一v EBCDIC 字符串（以一v字母*头）， R _ n d EBCDIC U q。

net_name

|含> X LU D网g {。 b G一v 8 字Z D字母}字类型 - 一v EBCDIC 字符串（以一v字母*头）， R _ n d EBCDIC U q。

lu_name

> X LU {。 b G一v 8 字Z D字母}字类型 - 一v EBCDIC 字符串（以一v字母*头）， R _ n d EBCDIC U q。

lu_alias

> X LU 对Z > X B务L序已知D p {。 b在> X显>字符集中G一v 8 字Z字符串。 y P 8 v字Z都G P意义D， " R X须h置。

MC_GET_ATTRIBUTES

plu_alias

伙 i LU 对 Z > X B 务 L 序已知 D p { 。 b 在 > X 显 > 字符集中 G 一 v 8 字 Z 字符串。 y P 8 v 字 Z 都 G P 意义 D， " R X 须 h 置。

plu_un_name

无法翻译 D LU {， M G，在系统服务 X 制 c（SSCP）O 定义 D 伙 i LU {。 b G 8 字 Z type-A EBCDIC 字符串。

fqplu_name

伙 i LU D 全限定 {。 b v { F 17 字 Z $， " R R _ 以 EBCDIC U q n d。 | I = v type-A EBCDIC 字符串以一 v EBCDIC c 分 t 而构 I。 （？v { F D 最大 $ 度 G 8 字 Z， ；P 6 入 U q。 若网 g ID ；存在，则！T 那 v c。）

user_id

I } w C D B 务 L 序通过 **ALLOCATE** 动词发 M D C 户 ID，以 c 访问 \ w C D B 务 L 序（若 I ）行 D 话）。 b G 10 v 字 Z D AE 型 EBCDIC 字符串，以 EBCDIC U q n d。

conv_group_id

分配 x 对话 D 会话 D 对话组 j 6。

conv_corr_len

通 # h 置为 0。

范围: 0-8

conv_corr

通 # h 置为 0。

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED

AP_DUPLEX_TYPE_MIXED

AP_UNEXPECTED_SYSTEM_ERROR

[MC_]PREPARE_TO_RECEIVE

PREPARE_TO_RECEIVE 动词为 > X B 务 L 序 + 其对话 D 状，从 SEND 或 SEND_PENDING | D为 RECEIVE。

| D对话状，之 0， C动词执行D Y作H价 Z 下 P 动词D其中之一:

- **FLUSH** 动词 + 伙 i LU D发M缓 e x 内容发M = 伙 i LU （B务L序)。
- **CONFIRM** 动词， + > X LU D发M缓 e x 内容 k 确认D k s 发M = 伙 i B务L序。

> 动词执行 I 功后， > X B 务 L 序 E 能够 S U }]。

VCB a 构

```
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;
```

提供的N数

B务L序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_PREPARE_TO_RECEIVE 

AP_M_PREPARE_TO_RECEIVE 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y作， > j 志 I k AP_NON_BLOCKING x 行 “或” 运 c 。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定O v P v D VCB f > 。

tp_id > X B 务 L 序D j 6 符。 > N } 值 I } 在w C B 务 L 序中D **TP_STARTED** 动词返回，或 I 已w C B 务 L 序中D **RECEIVE_ALLOCATE** 返回。

MC_PREPARE_TO_RECEIVE

conv_id

对话 j 6。

> N } 值 I } 在 w C B 务 L 序中 D **ALLOCATE** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

ptr_type

指定如何执行状，| D。

AP_FLUSH
AP_SYNC_LEVEL
AP_P_TO_R_CONFIRM

locks 指定 v 人通信 k 通信服务器 何 1 + X 制返回 = > X B 务处理处理器。

AP_LONG

AP_SHORT

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OK

若动词 G 非阻塞 D " R 还未完 I，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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

若 1 B 务处理器发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_TO_R_NOT_LL_BDY 

AP_P_TO_R_NOT_SEND_STATE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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

"：I Z 性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词 O 返回一 v I 功 D 返回 k，而无需 + 其转发 = 服务器。 1 发 v 一 v f 后 D **[MC_]PREPARE_TO_RECEIVE** 动词 1， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。若 P **[MC_]SEND_DATA** v 错返回 k，则其在 **[MC_]PREPARE_TO_RECEIVE** 动词 O 返回。k N阅 **[MC_]SEND_DATA** P m，以获 C v 错返回 k D P m。

[MC_]RECEIVE_AND_POST

RECEIVE_AND_POST 动词异 = X S U & C L 序 }] k 状，信息。 b 9 B 务 L 序能够在 }] 仍然 = 达 > X LU 1 继续处理。 > 动词 v 通过 APPC 入 Z c 发 v 。

在 Win 3.1 SNA API M户机 O ; I C。

VCB a 构

```
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;
```

提供的N数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_RECEIVE_AND_POST 

AP_M_RECEIVE_AND_POST 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定 O v P v D VCB f > 。

tp_id > X B 务 L 序 D j 6 符。 > N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在 w C B 务 L 序中 D **ALLOCATE** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

rtn_status

指 > 状，信息 k }] G 否能够在相同 D 动词 O 返回。

AP_YES

AP_NO

fill 

指 > > X B 务 L 序 S U }] D 方 = 。

AP_BUFFER

AP_LL

max_len

> X B 务 L 序能够 S U }] D 最大字 Z } 。

范围: 0-65535

> 值 X 须；能，过 | 含 S U }] D 缓 e x $ 度。

dptr | 含 > X LU S U D }] D 缓 e x X 址。& C L 序 I + }] = 加 = VCB D 末尾，在 b 种 i v 下，**dptr** X 须 h 置为 NULL。

sema & C L 序 + 要 H 待 D B 件 d z 。 b v 动词 c 向 Z 在 Win32 API 或 DosWaitEventSem OS/2 f 中 k WaitForMultipleObjects 9 C。

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

AP_DEALLOC_NORMAL

MC_RECEIVE_AND_POST

what_rcvd

k x 入 }] 一起 S U = D状，信息。若 **rtn_status** h 置为 AP_NO，则 > 字段通 # | 含来自下 P P m 中 Z 一 ? 分 D 值。若 **rtn_status** h 置为 AP_YES，则 > 字段 I | 含来自 P m D 任何值。

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

下 P N } v 供 3 d 之 C: 

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

Request-to-send-received 指 > 符。

AP_YES

AP_NO

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0， > 指 > 继续 h 置为 AP_YES。

AP_YES

AP_NO

> q = 化字段需 s q = 1 f > D VCB。 k N阅 Z 37页 D 『全 + 工 VCB』，以获 C 访问 q = 1 VCB | 详细 D 内容。

dlen S U = D }] 字 Z } （ }] 存储在 I **dptr** N } 指定 D 缓 e x 内）。 $ 度 c 指 > 未 S U = }]。 > N } 1 R v 1 **what_rcvd** N } 指 > 已 S U = }] 1 9 C。

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }：

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 

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，， 则 v 人通信 k 通信服务器返回下 P N }：

primary_rc

AP_STATE_CHECK

secondary_rc

AP_RCV_AND_POST_BAD_STATE

AP_RCV_AND_POST_NOT_LL_BDY 

若动词 I Z B 务 L 序发 v D m 一 v 动词而 Y Y 未能执行，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc

AP_CANCELLED

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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

"：I Z性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功 D 返 回 k ， 而 无 需 + 其 转 发 = 服 务 器 。 1 发 v 一 v f 后 D **[MC_]RECEIVE_AND_POST** 动词 1 ， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。 若 P **[MC_]RECEIVE_AND_POST** v 错返回 k ， 则其在 **[MC_]PREPARE_TO_RECEIVE** 动词O返回。 k N阅 **[MC_]SEND_DATA** P m，以获C v 错返回 k D P m。

[MC]RECEIVE_AND_WAIT

RECEIVE_AND_WAIT 动词S U任何１０来自伙ｉB务L序D Ｉ Ｃ｝]。若１０；Ｐ Ｉ Ｃ D ｝]，则＞ＸＢ务Ｌ序Ｈ待｝] ＝达。

对Ｚ ｋ ＋工对话:

- Ｌ序Ｉ在对话处Ｚ发M状，１发ｖＣ动词。在ｂ种ｉ ｖ下，LU ｅU其发M缓ｅ ｘ，发M ｙ Ｐ缓ｅD信息ｋ SEND 指＞发M＝远Ｌ Ｌ序。ｂ＋对话｜D为Ｓ U状，。然后，LU Ｈ待信息＝达。远Ｌ Ｌ序能够在Ｓ U＝ SEND 指＞后，＋｝] 发M＝＞ＸＬ序。

对Ｚ全＋工对话:

- 若发M缓ｅ ｘ｜含对话分配ｋ ｓ，则｜＋；ｅU；否则，ｂｖ动词；会＜致 LU ｅＶ其发M缓ｅ ｘ。 若发M缓ｅ ｘ内＃｀Ｄ｝] Ｘ须在Ｓ Ｕ｝] ０发Ｍｖ去Ｇ非＃重要Ｄ，则＞ＸＬ序＆在发ｖＣ动词０发ｖ FLUSH。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N｝a供x v人通信k通信服务器:

opcode

AP_B_RECEIVE_AND_WAIT 

AP_M_RECEIVE_AND_WAIT 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作，＞j志I k AP_NON_BLOCKING x行“或”运c。

在全+工对话O，＞j志X须k AP_FULL_DUPLEX_CONVERSATION x行“或”Y作。

format

j 6 VCB D q =。 +＞字段h置为 0，以指定O v P v D VCB f ＞。

tp_id ＞X B务L序D j 6符。

＞N｝值I｝在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。

＞N｝值I｝在w C B务L序中D **ALLOCATE** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

rtn_status

指＞状，信息k｝］G否能够在相同D动词O返回。

AP_YES

AP_NO

fill 

指＞＞X B务L序S U｝］D方=。

AP_BUFFER

AP_LL

max_len

＞X B务L序能够S U｝］D最大字Z｝。

范围: 0-65535

＞值X须；能，过|含S U｝］D缓e x $度。

dptr |含＞X LU S U D｝］D缓e x X址。&C L序I +｝］=加= VCB D末尾，在b种i v下， **dptr** X须h置为 NULL。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N｝:

primary_rc

AP_OK

AP_DEALLOC_NORMAL

what_rcvd

ｋｘ入｝］一起ＳＵ＝Ｄ状，信息。若 **rtn_status** ｈ置为 AP_NO，则＞字段通＃｜含来自下ＰＰｍ中Ｚ一？分Ｄ值。若 **rtn_status** ｈ置为 AP_YES，则＞字段Ｉ｜含来自ＰｍＤ任何值。

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

下ＰＮ｝ｖＪＣＺ３ｄ： 

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

Request-to-send-received 指 > 符。

AP_YES

AP_NO

下 P 动词 D C q = G VCB D q = 1 f >。 k N阅 Z 37页 D 『全 + 工 VCB』，以获 C | 多 P 关访问 q = 1 VCB D细 Z。

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0， > 指 > 继续 h 置为 AP_YES。

AP_YES

AP_NO

dlen > N } 1 R v 1 **what_rcvd** N } 指 > 已 S U = }] 1 9 C。 S U = D }] 字 Z } （ }] 存储在 I **dptr** N } 指定 D 缓 e x 内）。 $ 度 c 指 > 未 S U = }]。

若动词 G 非阻塞 D " R 还未完 I，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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 

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_RCV_AND_WAIT_BAD_STATE

AP_RCV_AND_WAIT_NOT_LL_BDY

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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 

下 P 三 v N } v J C Z 基 >: 

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER

AP_DEALLOC_NORMAL
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING

下 P D v N } v J C Z 基 >: 

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

": I Z性能需要, SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功 D 返 回 k , 而无需 + 其转发 = 服务器。 1 发 v 一 v f 后 D **[MC_]RECEIVE_AND_WAIT** 动词1, **[MC_]SEND_DATA** 转发=服务器, 以供处理。 若 P **[MC_]RECEIVE_AND_WAIT** v 错返回 k , 则其在 **[MC_]PREPARE_TO_RECEIVE** 动词O返回。 k N阅 **[MC_]SEND_DATA** P m, 以获C v 错返回 k D P m。

[MC_]RECEIVE_EXPEDITED_DATA

在通信服务器 SNA API M户机 OS/2 f 和 Windows 3.1 f O；支V。

[MC_]RECEIVE_EXPEDITED_DATA 动词S U任何１０来自伙ｉ TP ＤＩＣ加Ｙ｝］。若加Ｙ｝］１０ＩＣ，则＞ＸＢ务Ｌ序ＳＵ其无需Ｈ待；否则，其Ｙ作Ｉ **rtn_ctl** 字段Ｘ制。

VCB a 构

```
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
```

提供的N数

Ｂ务Ｌ序＋下ＰＮ｝ａ供ｘ ｖ人通信ｋ通信服务器:

opcode

AP_B_RECEIVE_EXPEDITED_DATA 

AP_M_RECEIVE_EXPEDITED_DATA

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作， > j 志 I k AP_NON_BLOCKING x 行“或”运 c 。

在全 + 工对话 O， > j 志 X 须 k AP_FULL_DUPLEX_CONVERSATION x 行“或” Y 作。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定 O v P v D VCB f > 。

tp_id > X B 务 L 序 D j 6 符。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在 w C B 务 L 序中 D **ALLOCATE** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

return_control

指定何 1 + X 制返回 = B 务 L 序。

AP_WHEN_EXPD_RECEIVED

AP_IMMEDIATE

max_len

> X B 务 L 序能够 S U }] D 最大字 Z } 。

范围: 0-86

> 值 X 须；能，过 | 含 S U }] D 缓 e x $ 度。

dptr | 含 > X LU S U D }] D 缓 e x X 址。& C L 序 I + }] = 加 = VCB D 末尾，在 b 种 i v 下， **dptr** X 须 h 置为 NULL。

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

rts_rcvd

Request-to-send-received 指 > 符。

AP_YES

AP_NO

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0 , > 指 > 继续 h 置为 AP_YES。

AP_YES
AP_NO

dlen S U = D }] 字 Z } （ }] 存储在 I **dptr** N } 指定 D 缓 e x 内）。$ 度 c 指 > 未 S U = }] 。 k 注意， S U = D 任何 }] 都 G 未 - q = 化 D。；存在 2 字 Z $ 度字段（LL）。

若动词 G 非阻塞 D " R 还未完 I，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z 远 L LU ；支 V 加 Y }] 而未执行，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc
AP_EXPD_NOT_SUPPORTED_BY_LU

若；P 来自伙 i B 务 L 序 D " 即 I C }]， " R **rtn_ctl** j 志为 AP_IMMEDIATE，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc
AP_UNSUCCESSFUL

若 I B 务 L 序 a 供 D }] 缓 e x ；足以 | 含 y P LU O I C D 加 Y }]，则；返回 }]， " R v 人通信 k 通信服务器 返回下 P w N }：

primary_rc
AP_BUFFER_TOO_SMALL

dlen LU I S U D 加 Y }] D 字 Z }。

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }：

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

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }：

primary_rc
AP_STATE_CHECK

secondary_rc
AP_EXPD_DATA_BAD_CONV_STATE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

MC_RECEIVE_EXPEDITED_DATA

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
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

AP_ERROR_INDICATION 

[MC_]RECEIVE_IMMEDIATE

[MC_]RECEIVE_IMMEDIATE 动词S U 1 0来自伙 i B务L序D任何I C }] 或状，信息。若1 0 ; P，则> X B务L序 " 即返回， ; H待。

VCB a 构

```
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;
```

提供的N数

B务L序 + 下P N } a 供x v 人通信k 通信服务器:

opcode

AP_B_RECEIVE_IMMEDIATE 

AP_M_RECEIVE_IMMEDIATE 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作， > j 志 I k AP_NON_BLOCKING x 行“或”运 c 。

在全 + 工对话 O， > j 志 X 须 k AP_FULL_DUPLEX_CONVERSATION x 行“或” Y 作。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定 O v P v D VCB f > 。

tp_id > X B 务 L 序 D j 6 符。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

rtn_status

指 > 状，信息 k }] G 否能够在相同 D 动词 O 返回。

AP_YES

AP_NO

fill 

指 > > X B 务 L 序 S U }] D 方 = 。

AP_BUFFER

AP_LL

max_len

> X B 务 L 序能够 S U }] D 最大字 Z } 。

范围: 0-65535

> 值 X 须；能，过 | 含 S U }] D 缓 e x $ 度。

dptr | 含 > X LU S U D }] D 缓 e x X 址。& C L 序 I + }] = 加 = VCB D 末尾，在 b 种 i v 下， **dptr** X 须 h 置为 NULL。

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

AP_DEALLOC_NORMAL

what_rcvd

k x 入 }] 一起 S U = D 状，信息。若 **rtn_status** h 置为 AP_NO，则 > 字段通 # | 含来自下 P P m 中 Z 一 ? 分 D 值。若 **rtn_status** h 置为 AP_YES，则 > 字段 I | 含来自 P m D 任何值。

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 

下 P N } v J C Z 3 d: 

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

Expedited-data-received 指 > 符。

AP_YES

AP_NO

rts_rcvd

Request-to-send-received 指 > 符。

AP_YES

AP_NO

dlen > N } 1 R v 1 **what_rcvd** N } 指 > 已 S U = }] 1 9 C。 S U = D }] 字 Z } （ }] 存储在 I **dptr** N } 指定 D 缓 e x 内）。 $ 度 c 指 > 未 S U = }]。

若动词 G 非阻塞 D " R 还未完 I，则 v 人通信 k 通信服务器返回下 P w N }:

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_BASIC_CONVERSION k AP_MAPPED_CONVERSATION x 行“或” Y 作

AP_NON_BLOCKING k
AP_OPERATION_INCOMPLETE_FLAG

若；P 来自伙 i B 务 L 序 D " 即 I C }]，则 v 人通信 k 通信服务器返回下 P N }。

primary_rc

AP_UNSUCCESSFUL

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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 

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_RCV_IMMD_BAD_STATE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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
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

"：I Z性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功 D 返 回 k ， 而 无 需 + 其 转 发 = 服 务 器 。 1 发 v 一 v f 后 D **[MC_]RECEIVE_IMMEDIATE** 动词 1 ， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。 若 P **[MC_]RECEIVE_IMMEDIATE** v 错返回 k ， 则其在 **[MC_]PREPARE_TO_RECEIVE** 动词O返回。 k N阅 **[MC_]SEND_DATA** P m，以获C v错返回 k D P m。

[MC_]REQUEST_TO_SEND

[MC_]REQUEST_TO_SEND 动词通知伙 i B务L序> X B务L序希望发M }]。

VCB a构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_REQUEST_TO_SEND 

AP_M_REQUEST_TO_SEND 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作, > j志I k AP_NON_BLOCKING x行“或”运c。

format

j 6 VCB D q =。 + >字段h置为 0, 以指定O v P v D VCB f >。

tp_id > X B务L序D j 6符。

> N } 值I } 在w C B务L序中D **TP_STARTED** 动词返回, 或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。

> N } 值I } 在w C B务L序中D **TP_STARTED** 动词返回, 或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行I功, 则v人通信k通信服务器返回下P N }:

primary_rc
AP_OK

若动词G非阻塞D " R还未完I，则v人通信k通信服务器返回下P N }:

primary_rc
AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若 **[MC_]REQUEST_TO_SEND** 以非阻塞方=发v（ k N阅Z 37页D『排队级无阻塞』），" R 1在发M/ S U队P O处理动词1对话终止，则v人通信k通信服务器返回下P N }:

primary_rc
AP_CONVERSATION_ENDED

& C L序；&为C对话发v任何 | 多D动词。

若动词I Z N }错误而未执行，则v人通信k通信服务器返回下P N }:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_R_T_S_INVALID_FOR_FDX

若1 B务L序发v >动词1对话处Z错误状，，则v人通信k通信服务器返回下P N }:

primary_rc
AP_STATE_CHECK

secondary_rc
AP_R_T_S_BAD_STATE

z I下P P I能性D主返回k（**primary_rc**）D u件h v在= < A. APPC 公共返回k中。

AP_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

[MC_]SEND_CONVERSATION

[MC_]SEND_CONVERSATION 动词 + 对话分配 x > X LU 和伙 i LU（引起伙 i LU O D B 务 L 序启动）之间 D 对话、在 C 对话 O 发 M }] 记 <，然后；H 确认 X M 放对话。 | H 价 Z **[MC_]ALLOCATE**、**[MC_]SEND_DATA**、**[MC_]DEALLOCATE**（**FLUSH**）动词序 P（通 # F 作一 v 『 % v 一 _ 方括号 』）。

VCB a 构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_SEND_CONVERSATION 

AP_M_SEND_CONVERSATION 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作, > j志I k AP_NON_BLOCKING x行“或”运c。

format

j 6 VCB D q =。 + >字段h置为 0, 以指定O v P v D VCB f >。

tp_id > X B务L序D j 6符。

> N } 值I }在w C B务L序中D **TP_STARTED** 动词返回, 或I已w C B务L序中D **RECEIVE_ALLOCATE** 动词返回。

rtn_ctl

指定在来自> X B务处理器D会话k s O活动D > X LU 何1 + X制返回= > X B务L序。

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

+要分配D会话D对话组j 6。 1 R v 1 **rtn_ctl** h置为AP_WHEN_CONV_GROUP_ALLOC 1 E a供> N }。

plu_alias

伙i LU 对Z > X B务L序已知D p {。 b在> X显>字符集中G一v 8字Z字符串。 y P 8 v字Z都G P意义D, " R X须h置。 > { X须k在配置期间(" D伙i LU {匹配。

若C字段h置为全c, 则 v人通信k通信服务器 9 C **fqplu_name** 字段指定X需D伙i LU。

mode_name

在配置期间定义D一组, 网X性{。 b G一v 8 字Z D字母}字类型 - 一v EBCDIC 字符串(以一v字母*头), R _ n d EBCDIC U q。

tp_name

; w C D B务L序{。 v人通信k通信服务器 ;检i C字段D字符集。 I

ALLOCATE 动词在 } w C B 务 L 序中指定 D **tp_name** 值 X 须 k I **RECEIVE_ALLOCATE** 动词在已 w C D B 务 L 序中指定 D **tp_name** 值匹配。

security

指定伙 i LU 需 s D信息，以 c 验证对 Z 已 w C D B 务 L 序 D 访问。

AP_NONE

AP_PGM

AP_SAME

AP_PGM_STRONG

pwd k **user_id** 关 * D Z n。 b G 10 v 字 Z D AE 型 EBCDIC 字符串，以 EBCDIC U q n d。 若 Security=Program（AP_PGM 或 AP_PGM_STRONG），则 b G X 需 D；否则，｜为 I 选项。

user_id

C 户 ID 需要访问伙 i B 务 L 序。 b G 10 v 字 Z D AE 型 EBCDIC 字符串，以 EBCDIC U q n d。 若 Security=Program（AP_PGM 或 AP_PGM_STRONG），则 b G X 需 D；否则，｜为 I 选项。

pip_dlen

+ 传] = 伙 i B 务 L 序 D L 序 u < 化 N }（PIP）$ 度。

范围: 0-32767

pip_dptr

｜含 PIP }] D 缓 e x X 址。 v 1 **pip_dlen** 大 Z c 1 9 C > N }。

fqplu_name

伙 i LU D 全限定 LU {。 b v { F 17 字 Z $，" R R _ 以 EBCDIC U q n d。 ｜ I = v type-A EBCDIC 字符串以一 v EBCDIC c 分 t 而构 I。（？v { F D 最大 $ 度 G 8 字 Z，；P 6 入 U q。 若网 g ID ；存在，则！T 那 v c。） 1 R v 1 **plu_alias** 字段 h 置为 c 1，> 字段 E G P 意义 D。

dlen 发 M }] D 字 Z }。

范围: 0-65535

dptr ｜含需要发 M D }] D 缓 e x X 址。& C L 序 I + }] = 加 = VCB D 末尾，在 b 种 i v 下， **dptr** X 须 h 置为 NULL。

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OK

conv_group_id

分配 x 对话 D 会话 D 对话组 j 6。

若动词 G 非阻塞 D " R 还未完 I，则 v 人通信 k 通信服务器返回下 P w N }:

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若 **rtn_ctl** N } h 置为 AP_IMMEDIATE， " R ; P " 即 I C D会话，则 v 人通信 k 通信服务器 返回下 P w N } :

primary_rc

AP_UNSUCCESSFUL

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N } :

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

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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

若 **primary_rc** h 置为 AP_ALLOCATION_ERROR，则 **sense_data** 字段能够携带 | 多 ' \ D 信息。

[MC_]SEND_DATA

[MC_]SEND_DATA 动词 + }] 置入 > X LU D发M缓 e x 中，以 8 传 d = 伙 i B 务 L 序。

VCB a 构

```
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;
```

提供的N数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_SEND_DATA 

AP_M_SEND_DATA 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作， > j 志 I k AP_NON_BLOCKING x 行 “或” 运 c 。

在全 + 工对话O， > j 志X须 k AP_FULL_DUPLEX_CONVERSATION x 行“或” Y作。

format

VCB q = 。 + 此 h 置为 1，以 c 获C O v P v D q = 。

tp_id > X B 务L 序D j 6 符。

> N } 值 I } 在w C B 务L 序中D **TP_STARTED** 动词返回，或 I 已w C B 务L 序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在w C B 务L 序中D **TP_STARTED** 动词返回，或 I 已w C B 务L 序中D **RECEIVE_ALLOCATE** 返回。

dlen 置入 > X LU D发M缓 e x D }] 字Z } 。

范围: 0-65535

dptr | 含置入 > X LU D发M缓 e x D }] D缓 e x X址。& C L 序 I + }] = 加 = VCB D末尾，在 b 种 i v 下， **dptr** X须 h 置为 NULL。

type 指定G 否在 **SEND_DATA** 以外执行m 一 v 动词D 函 } 。

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

返回N数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

"： I Z 性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词O返回一 v I 功D返回 k ，而无需 + 其转发 = 服务器。 1 发 v 一 v f 后D **[MC_]SEND_DATA** 动词 1 ， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。

若 P **SEND_DATA** v 错返回 k ，则在 f 后D动词 + 其返回。

primary_rc

AP_OK

rts_rcvd

Request-to-send-received 指 > 符。

AP_YES

AP_NO

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0， > 指 > 继续 h 置为 AP_YES。

AP_YES

AP_NO

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P w N }:

primary_rc

AP_PARAMETER_CHECK

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，， 则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_SEND_DATA_NOT_SEND_STATE

AP_SEND_DATA_NOT_LL_BDY 

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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 动词通知伙 i B务L序 > X B务L序已 - v = 一处L序级 p 错误。

VCB a 构

```
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;
```

提供的N数

B务L序 + 下P N } a 供x v 人通信 k 通信服务器:

opcode

AP_B_SEND_ERROR 

AP_M_SEND_ERROR 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对Z非阻塞Y作, > j 志I k AP_NON_BLOCKING x行“或”运c。

在全 + 工对话O, > j 志X须k AP_FULL_DUPLEX_CONVERSATION x行“或” Y作。

format

j 6 VCB D q =。 + > 字段h置为 0, 以指定O v P v D VCB f >。

tp_id > X B 务 L 序 D j 6 符。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

err_type 

指 > } (f D v 错类型: & C L 序或 G 服务 L 序。

AP_PROG
AP_SVC

err_dir

指 > } 在 (f D v 错 G 在从伙 i B 务 L 序 S U = D }] 中，还 G 在 > X B 务 L 序 } 要发 M D }] 中。

> N } 1 R v 1 **SEND_ERROR** 动词 } 在 SEND_PENDING 状，发 v 1 9 C。

AP_RCV_DIR_ERROR

AP_SEND_DIR_ERROR

log_dlen 

发 M = 错误记 < 文件 D }] 字 Z } 。

范围: 0-32767

& C L 序 I + }] = 加 = VCB D a 尾，在 b 种 i v 下，此字段 + 大 Z c， " R **log_dptr** X 须 h 置为 NULL。（ $ 度 l 指 > ；存在错误记 < }] 。）

log_dptr 

| 含 v 错信息 D }] 缓 e x X 址。& C L 序 I + }] = 加 = VCB D 末尾，在 b 种 i v 下， **dptr** X 须 h 置为 NULL。

发 M C }] = > X 错误记 < 以及伙 i LU。若 **log_dlen** 大 Z c，则 **SEND_ERROR** 动词 9 C > N } 。

B 务 L 序 X 须 + v 错 }] q = 化为一 c }] w（GDS）错误记 < d ? 。 如需 x 一 = D 信息， k N 阅 *IBM 系统网 g e 系 a 构: LU 6.2 N < 大全: 同级协议*。

返回 N 数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

rts_rcvd

Request-to-send-received 指 > 符。

AP_YES

AP_NO

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0， > 指 > 继续 h 置为 AP_YES。

AP_YES

AP_NO

若动词 G 非阻塞 D " R 还未完 I， 则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

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 

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，， 则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_SEND_ERROR_BAD_STATE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

在任何允 m 4 态发出的动词

1 **[MC_]SEND_ERROR** 动词在任何允许 D 状，发 v 1， I z I 下 P 返回 k:

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

***SEND** 4 态发出的动词:* 若 **[MC_]SEND_ERROR** 动词在 SEND 状，发 v，则 I z I 下 P 返回 k:

- 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 

***RECEIVE** 4 态发出的动词:* 若动词在 RECEIVE 状，发 v，则 I z I 下 P 返回 k:

AP_DEALLOC_NORMAL

"**:** I Z 性能需要， SNA API M户机 I 在 **[MC_]SEND_DATA** 动词 O 返回一 v I 功 D 返回 k，而无需 + 其转发 = 服务器。 1 发 v 一 v f 后 D **[MC_]SEND_ERRORE** 动词 1， **[MC_]SEND_DATA** 转发 = 服务器，以供处理。

若 P **[MC_]SEND_ERROR** v 错返回 k，则其在 **[MC_]PREPARE_TO_RECEIVE** 动词 O 返回。 k N 阅 **[MC_]SEND_DATA** P m，以获 C v 错返回 k D P m。

[MC_]SEND_EXPEDITED_DATA

b 在通信服务器 SNA API M户机 OS/2 f 和 Windows 3.1 f O；支V。

[MC_]SEND_EXPEDITED_DATA 动词 + }] 置入 > X LU D加Y发M缓 e x中，以 8 传 d = 伙 i B 务 L 序。C }] I H以 0 发M D非加Y }] | 早一 = = 达伙 i B 务 L 序。

VCB a 构

```
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
```

提供的N数

B 务 L 序 + 下 P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_SEND_EXPEDITED_DATA 

AP_M_SEND_EXPEDITED_DATA 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION。对 Z 非阻塞 Y 作，> j 志 I k AP_NON_BLOCKING x 行 “或” 运 c 。

在全 + 工对话O， > j 志X须k AP_FULL_DUPLEX_CONVERSATION x行“或” Y作。

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定O v P v D VCB f > 。

tp_id > X B 务 L 序 D j 6 符。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

dlen 置入 > X LU D 发M缓 e x D }] 字 Z } 。

范围: 1-86

dptr | 含 v 错信息 D }] 缓 e x X址。 & C L 序 I + }] = 加 = VCB D末尾，在 b 种 i v 下， **dptr** X须 h 置为 NULL。

注意， }] G 未 - q = 化 D-- ; 存在 2 字 Z $ 度字段（LL）。

返回N数

若动词执行 I 功，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OK

rts_rcvd

Request-to-send-received 指 > 符。

AP_YES

AP_NO

expd_data_rcvd

Expedited-data-received 指 > 符。在发 v RECEIVE_EXPEDITED_DATA 0 , > 指 > 继续 h 置为 AP_YES。

AP_YES

AP_NO

若动词G非阻塞D " R还未完 I , 则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_OPERATION_INCOMPLETE

opext AP_OPERATION_INCOMPLETE_FLAG

若动词 I Z 远 L LU ；支 V 加 Y }] 而未执行，则 v 人通信 k 通信服务器返回下 P N } :

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N } :

MC_SEND_EXPEDITED_DATA

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 

若 1 B 务 L 序发 v > 动词 1 对话处 Z 错误状，， 则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_STATE_CHECK

secondary_rc

AP_EXPD_DATA_BAD_CONV_STATE

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

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_TP_BUSY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

[MC_]TEST_RTS

[MC_]TEST_RTS 动词确定G否已 - S U = 来自伙 i B务L序D k s发M通知。

VCB a构

```
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;
```

提供的N数

B务L序+下P N } a供x v人通信k通信服务器:

opcode

AP_B_TEST_RTS 

AP_M_TEST_RTS 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION

format

j 6 VCB D q =。 + >字段h置为 0，以指定O v P v D VCB f >。

tp_id > X B务L序D j 6符。 > N }值I }在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话j 6。

> N }值I }在w C B务L序中D **TP_STARTED** 动词返回，或I已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

返回N数

若动词执行I功，则v人通信k通信服务器返回下P N }:

primary_rc

指 > G否已 - S U = 来自伙 i B务L序D k s 发M通知。

AP_OK

AP_UNSUCCESSFUL

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc

AP_PARAMETER_CHECK

secondary_rc

AP_BAD_CONV_ID

AP_BAD_TP_ID

AP_TEST_INVALID_FOR_FDX

z I 下P P I 能性D主返回 k（**primary_rc**）D u 件h v 在 = < A. APPC 公共返回 k 中。

AP_TP_BUSY

AP_CONVERSATION_TYPE_MIXED

AP_UNEXPECTED_SYSTEM_ERROR

[MC_]TEST_RTS_AND_POST

[MC_]TEST_RTS_AND_POST 动词异 = X确定G否已 - S U = 来自伙 i B务L序D k s发M通知。B务L序 I 在任何 1 侯发 v **[MC_]TEST_RTS_AND_POST**，无 [在对话 O G否 P m一 v 未执行D动词。 **[MC_]TEST_RTS_AND_POST** 在S U = k s发M通知、或 _ 1 对话 a x、或 _ 1 检 b v 一处对话故O 1 返回。

> 动词 v 通过 APPC 入Z c 发 v 。

VCB a 构

```
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;
```

提供的N数

B务L序 + 下P N } a 供 x v 人通信 k 通信服务器:

opcode

AP_B_TEST_RTS_AND_POST 

AP_M_TEST_RTS_AND_POST 

opext AP_BASIC_CONVERSATION 或 AP_MAPPED_CONVERSATION

format

j 6 VCB D q = 。 + > 字段 h 置为 0，以指定O v P v D VCB f > 。

tp_id > X B务L序D j 6 符。

> N } 值 I } 在w C B务L序中D **TP_STARTED** 动词返回，或 I 已w C B务L序中D **RECEIVE_ALLOCATE** 返回。

conv_id

对话 j 6 。

> N } 值 I } 在 w C B 务 L 序中 D **TP_STARTED** 动词返回，或 I 已 w C B 务 L 序中 D **RECEIVE_ALLOCATE** 返回。

sema & C L 序 + 要 H 待 D B 件 d z 。 b v 动词 c 向 Z 在 Win32 API 中 k WaitForMultipleObjects 9 C。需要 P 关 > 功能 D | 多信息， k N阅 Win32 API D L 序 h 计文 5 。

返回N数

若动词执行 I 功（M G， S U = k s 发M通知），则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_OK

若 I Z 对话 a x 或检 b v 对话 ' \ 而返回动词，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_UNSUCCESSFUL

若动词 I Z N } 错误而未执行，则 v 人通信 k 通信服务器返回下 P N }:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
AP_BAD_CONV_ID

AP_BAD_TP_ID
AP_TEST_INVALID_FOR_FDX

z I 下 P P I 能性 D 主返回 k（**primary_rc**）D u 件 h v 在 = < A. APPC 公共返回 k 中。

AP_CONVERSATION_TYPE_MIXED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CANCELLED

第2？分 LUA API

第9章 IBM 常规 LU & C程r 的基本概n 147
理b LUA 和 SNA 147
，S能& 147
LUA & C L序. 147
LUA 动词 147
LU、 > X LU 和伙 i LU 148
系统服务X制 c (SSCP) 148
SNA c 148
}] 4 7 X制 c 148
7 6 X制 c 148
传 d X制 c 149
}] w X制 c 149
m > 服务 c 149
9 C SNA 会话 149
SNA 会话D先 v u 件 149
启动会话 150
从 SLU 启动 LU-LU 会话 150
从 PLU 启动 LU-LU 会话 150
在 LU-LU 会话O传M }] 150
停止会话 151
I SLU 停止 LU-LU 会话 151
I PLU 停止 LU-LU 会话 151
停止 SSCP-LU 会话和 SSCP-PU 会话 151
断 * 主机 4 S 151
消息号. 152
重新启动和重新同 = 会话 152
9 C协议来X制 k s 和响& 152
9 C w = 协议 152
S U w = 协议 153
发M w = 协议 153
9 C k + 工 y C/触发器协议 153
9 C W c 协议 153
9 C }] 4 S协议 154
}] ；换X制方法 154
w动协议 154
响&方 = ... 155
LUA 关 * m q 155
异 # 响& k s (RQE) 155
会话E要 156
TS E要 156
FM E要 156
9 C RUI LUA 动词 157
动词 * 要 157
RUI 会话. 158
发 v RUI 动词 158
异 = 动词完 I 159
样例 LUA 通信序P 159
BIND 校验 161

否定响&和 SNA 检 b k 161
f p SNA 检 b k 和其 | 次级返回 k 162
P 关 SNA 检 b k D 信息 162
w = ... 162
分段 162
礼 Z 确认 162
e } }] 至 4 a x 163
配置 163
LUA LU X(I 选 D) 163
SNA API M 户机 < G 163

第10章 RUI LUA 动词特 T 165
处理异 # k s 165
| D 动词记 < 165
处理方括号 k s \ x 166
最小化 LAN 通信 ? 166
减 Y RUI_BID D 9 C 166
处理挂起 166
取消 RUI_INIT 167
取消 RUI_WRITE 167
取消 RUI_READ 167
确 # 动词完 I 167
压 u }] 167
协 L ? v 会话 }] 压 u D 规则 167
RUI 规则 168
SLI 规则 168
从会话 ' \ 中恢 4 168

第11章 实现 LUA 程 r 171
` 写 LUA L 序 171
w C LUA 服务 171
理 b 动词记 < 内容 172
多 x L 172
多线 L 172
LUA 动词传 M 172
从 ASCII 转换为 EBCDIC 173

第12章 RUI LUA 入口点 175
RUI() 176
WinRUI 177
WinRUICleanup() 178
WinRUIGetLastInitStatus() 179
WinRUIStartup() 182
GetLuaReturnCode() 183

第13章 RUI 动词 185
LUA 动词 X 制 i q = ... 185
公共动词头 185
RUI_BID }] a 构 189
RUI_BID 190
RUI_INIT 195
RUI_PURGE 199
RUI_INIT_STATUS 202

RUI_READ 203
RUI_TERM 209
RUI_WRITE. 212

第14章 SLI 入口点 219
SLI() 220
WinSLI 221
WinSLICleanup() 222
WinSLIStartup() 223

第15章 SLI 动词 225
SLI_BID 226
SLI_CLOSE 231
SLI_OPEN 234
SLI_PURGE 240
SLI_RECEIVE 242
SLI_SEND 247
SLI_BIND_ROUTINE 251
SLI_STSN_ROUTINE 253
SLI_SDT_ROUTINE 255

第9章 IBM 常规 LU &C程 r 的基本概 n

> B h v K IBM #规D _ 辑%元&C L 序(LUA)存取法以及 | k 系统网 g e 系 a 构(SNA) D关系。

": | 含在> i Z 2 ?件中D B Z G I 以下系统 a 供D P 关 LUA API 信息:

- 运行在 Windows NT O D通信服务器
- SNA API M户机 OS/2 f 、Windows NT f 、Windows 95 f 和 Windows 3.1 f , 及一起传] D通信服务器/NT z 品
- v 人通信 Windows 95 和 Windows NT f 。

1 b些系统 a 供D支V之间P n异 1 , +注M v 。

m b LUA 和 SNA

IBM LUA 存取法 a 供K对次级从 t _ 辑%元(LU)D &C L 序 h 计 S Z (API)。LUA | 含 a 供 d 入 / d v (I/O)服务例行L序以支V 9 C LU 类型 0、1、 2 和 3 SNA 协议D通信D软件和S Z。通信服务器支V LUA D RUI 和 SLI S Z。

通信服务器 h 计为 k Microsoft** NT SNA 服务器二 x 制兼容 " R k OS/2 通信管理器/2 f > 1.0 LUA D 5 现相F。

LUA a 供 x &C L 序D服务 v | 括那些支V }] 通信D ?分; LUA ; a 供任何 h 8 仿 f h)。 ;过, LUA a 供K一 v 唯一D显 > 服务 c 函 } D子集。

X须在 LUA &C L 序 I 在工作 > O运行 0 2 装和配置通信服务器。若需 P 关 2 装和配置通信服务器D信息, k N阅*通信服务器: l Y入E*。

, S \ &

任何通信系统D主要目 j G k 其 | 系统 , S。 SNA D目D G a 供 x Z 通C , 通性D公共协议。 LUA通信和 , S 需 s | 括K System/370* (S/370*) , S。

LUA &C程 r

在C i 中, u o *LUA &C L 序*意味着 9 C LUA 通信功能D一 v &C L 序或一 v &C L 序D一 ?分。&C L 序 9 C b 些函 } 来 k 在其 | 支V LU 类型 0、1、2 或 3 D系统O D&C L 序通信。

1 一 v > X LUA &C L 序运行 1 , | k 远L 主机&C L 序互换 }]。 > X和远L &C L 序F 为*伙 i* &C L 序。

LUA 动词

动词G I LUA 处理D q = 化 k s 。&C L 序发 v 动词来 k s LUA I 取一些Y作。LUA 动词 ` k 为X制 i 。 ? v 动词X制 i P + 确定义D q = 。要 9 C LUA h), &C L 序M要 a 供动词X制 i x LUA API。

LUA 动词总G " 即返回至 | D w C L 序。如果返回 k G IN_PROGRESS, 则& C L 序需要通过 9 C在动词 k s 中指定D传] 方 =, H待动词完 I 。若需 LUA 动词传] D h v, k N阅 Z 12 B RUI LUA 入Z c 。

动词X制 i < V位Z **INCLUDE** 目 < 中。你 I 9 C动词X制 i < V和样 > L 序来 o 助你 ` 写 LUA & C L 序。

LU、本地 LU 和伙 i LU

_ 辑%元 (LU)管理& C L 序之间D }] ; 换。 ? v LUA & C L 序通过一 v LU 获C对 SNA 网 g D访问, LU d 1 在 LUA & C L 序和 SNA 网 g 之间D = i 。

在 LUA 中, 在 LUA & C L 序 x L 和 LU 之间 P 一对多D关系。一 v LUA & C L 序 x L I 同 1 5 P 多 v LU, + 一 v x 定D LU 在一 v 1 L v I ; 一 v LUA & C L 序 x L y 5 P。在Z二 v & C L 序 x L I 9 C LU 0, Z一 v & C L 序X须M放 LU。

LUA & C L 序对 | D > X LU 发 v LUA 动词。 b 些动词 9 | n 和 }] 通过网 g w 动 = 伙 i LU。

" : 对Z ? (机器, v 需要定义 > X LU 一次, 如 *l Y 入 E* 中 y v 。

系统服务控制点(SSCP)

主机系统中D系统服务X制 c (SSCP)组件：责启动主机& C L 序, C从 t LU * 系主机 & C L 序, 以及创 (和终止 LU 之间D , S 。

SNA c

SNA G | 含 w 确定义D七 v c 次D c 次 a 构。在 e 系 a 构中D ? c 执行一 v X定D功能。理 b SNA D c a 构 P 助Z理 b LUA a 供D w种功能。下 f D关Z SNA D五 v 最 _ c D h v 显 > K LUA 和 SNA D关系。

数] 4 7 控制 c

}] 4 7 X制(DLC) c | 含K a 供对 2 件 S Z D元件。 DLC 元件 a 供K对 w 种 DLC 协议, 例如同 = }] 4 7 X制 (SDLC)和 IBM n 牌环网 g D支V。 DLC c a 供K对在 7 6 X制(PC) c 中D元件D一 v 公共D 4 S 外观。对 y P v 人通信 k 通信服务器 LU 5 现, | 括 LUA, DLC c 都G共 P D。

7 6 控制 c

在外围Z c 中D SNA D 7 6 X制(PC) c a 供K基 > 功能, 例如在 | b v Z c 内D多 v k 会话之间来回传] 。 SNA 允许 PC c 一次 k 一 u }] 4 7 来回传] }] 。对 y P v 人通信 k 通信服务器 LU 5 现, | 括 LUA, PC c 都G共 P D。

传输控制 c

SNA D传 d X制(TC) c 对Z ？ v > X支V D会话端 a 供K ， S c 管理功能和会话X制功能。 ，S c 管理功能X制序P 号检 i 、 w = 和相关Z 会话端 }] w D其 | 支V功能。会话X制功能 a 供K对启动、 w = 、加 \ 、译 k H会话X定D支V以及其 | 相关Z 会话相关 }] w D其 | 支V功能。 LUA | 含K在 v 人通信 k 通信服务器下D LU 类型 0、1、 2 和 3 D TC c 5 现.

数] w控制 c

SNA D }] w X制(DFC) c X制在会话里和会话间D FMD 对之间D功能管理 }] (FMD) k s 和 FMD 响& w。 }] w X制 c a 供K w种功能，例如 k s /响& q = 化、 }] 4 S协议、 k s /响&关 * 、发M和S U方 = 协议、W c 协议、错误恢 4 协议、停止W c u < 化协议以及排队D响&协议。 LUA | 含K在 v 人通信 k 通信服务器下D LU 类型 0、1、 2 和 3 D }] w X制 c 5 现.

表示服务 c

SNA D m > 服务(PS) c | 含K m > 通信 }] S Z x C户D功能。 m > 服务 c 定义在 y P LU 类型(} K LU 0)D e 系 a 构中。LUA | 含K在 v 人通信 k 通信服务器中D m > 服务 c D一 v 唯一子集。若需 P 关m > 服务 c D | 多信息， k N阅*系统网 g e 系 a 构E念和 z 品*。

LU 服务功能G SNA 会话消息w c D一？分。 b 些功能在会话 (" 、 (" 会话 a 构和 p } 会话 a 构 0 a 供K支V。 k 公共 v 人通信 k 通信服务器一起D LUA 功能S Z 支V定义 LU 及启动和停止 SNA 会话。

使C SNA 会话

在 LUA & C L 序I k 伙 i 主机& C L 序通信 0， ？ v LU X须，S 入一 v F 为会话D相互关系。 SNA 会话G 9 = v 网 g I 访问%元(NAUs)相互通信D _ 辑，S； LU M G 一种 NAU。因为会话，S = v LU， y 以 | F 为一 v *LU-LU 会话*。 LU-LU 会话 9 端 c C户能；换 }] 。

会话管理 }] 如何在 SNA 网 g 中D一对 LU 之间移动。因此，会话 k } 在传M D }] ？、 }] 2 全性、网 g 7 I 以及；通阻塞相关。 1 次 LU S \ **BIND** | n 1， I 从主要 LU 发 v D SNA **BIND** | n 来确定会话X性。

SNA 会话的先 v 条 ~

一 v LU-LU 会话 | 含K在主 _ 辑%元 (PLU)和次 _ 辑%元(SLU)之间D通信。SLU I LUA & C L 序 5 现。 1 在 LU-LU 会话O D PLU 和 SLU 之间传M }] 0， X定发 z 下P B 件:

1. v 人通信 k 通信服务器激活 }] 4 7 。
2. 1 }] 4 7 M绪 1，系统服务X制 c (SSCP)通过发M激活物理%元(**ACTPU**) | n 和从 v 人通信或通信服务器读取 } 响& 来在 | > m 和一 v 物理%元间 (" 一 v 会话(SSCP-PU 会话)。如果从 **ACTPU** | n 来D PU X址 k 配置信息相关，那 4 L 序发M一 v } 响&。

3. SSCP 通过发M激活 _ 辑%元(**ACTLU**)｜ n 和从 v 人通信或通信服务器读取 } 响&来在｜ > m和一 v _ 辑%元间 (" 一 v 会话(SSCP-LU 会话)。 如果从 **ACTLU** ｜ n 来 D LU X址 k 配置信息相关，那 4 L 序发M一 v } 响&。

启动会话

SLU 或 PLU 都 I 启动一 v LU-LU 会话。

从 SLU 启动 LU-LU 会话

在 (" SSCP-LU 会话后，SLU L 序 I 通过发M x SSCP u 启 > m(**INITSELF**)｜ n 来 k s 一 v LU-LU 会话。 SSCP S U **INITSELF** ｜ n " 检 i ｜ { D主机& C L 序G否 P 效。主机& C L 序在；知及活动 1 P 效。如果主机& C L 序 P 效，那 4 SSCP M发M 一 v } 响& x SLU， " R PLU 启动会话。如果主机& C L 序无效，那 4 SSCP M发 M一 v 否定响& x SLU， " R PLU ；启动会话。

如果 SSCP 发M一 v } 响& x **INITSELF** ｜ n + ；能 (" 会话，那 4 SSCP 发M一 v 网 g 服务过L v 错(**NSPE**)｜ n x SLU 来停止 " T (" 会话。SLU I 在 **NSPE** ｜ n 后重新发 v **INITSELF** ｜ n 。

从 PLU 启动 LU-LU 会话

PLU L 序 I 启动主动 = LU-LU 会话。PLU 通过 z z 一 v **BIND** ｜ n 来启动会话。 f 后D一 v } 响& (" K通信协定。 k **BIND** ｜ n 相关D }] 字段｜含K PLU & C L 序 D { F以及会话 **BIND** N } 。若需 P 关 }] 字段 q = D ｜多信息， k N阅 *系统网 g e 系 a 构: q =* 。

对 Z x 对 **BIND**，如果N } I S \ ，那 4 SLU 返回一 v } 响&。如果N } ；I S \ ，那 4 SLU 返回一 v 带 P 读 v }] D否定响& x PLU。

I 协L **BIND** ｜ n 允许 SLU 返回一 v 带 P 最小 26 v 字 Z D m > k PLU N } 兼容D ｜新会话N } D } 响&。如果 PLU 发现返回D N } I S \ ，那 4 ｜发M一 v 启动 }] 通信 (**SDT**)｜ n 。如果返回D N } ；I S \ ，那 4 PLU 发M一 v **UNBIND** ｜ n 来m > 从 SLU 来D **BIND** ｜ n N } 为；I S \ 协L 。

在 LU-LU 会话O传M数]

在 (" K LU-LU 会话 " R SLU L 序响& K **SDT** ｜ n 后， I 以 * < }] 传M。对Z }] 传 d Y作，消息从端 c C户存储器移动至 v 人通信或通信服务器存储器直 = ｜ ；发M完。对 Z }] S U Y作， L 序 + Q 消息放置 Z ｜自己D存储器中 " R S 着Q消息移动 = 端 c C户存储器中。

停顿协议暂停在 LU-LU 会话中D }] 传M。PLU 或 SLU I 发M以下停顿协议｜ n :

- 在 4 a x 停顿(**QEC**)。 C ｜ n k s 其 S \ _ 在发M完 }] 4 S 中D最后一 ? 分后停止发M }] 。 }] 4 S G一系 P 相关D消息。若需｜多 P 关 }] 4 S D信息， k N 阅 Z 154页 D 『 9 C }] 4 S 协议 』 。
- 停顿完 I (**QC**)。 C ｜ n 通知 **QEC** ｜ n }] 传M已暂停。 1 SLU 发M **QC** ｜ n 1 ， v 人通信或通信服务器阻止 SLU 发M任何 } # }] w 消息直 = S U K S U 停顿 (**RELQ**)｜ n 。

• M放停顿(**RELQ**)。C | n通知S \ _ }] I再次传M。

停止会话

1 y P }]已传M " 验证好1，会话I a x。SLU X须在 | I k相同D PLU 或m一v PLU * < m一v ；同D会话0 a x一v会话。

I SLU 停止 LU-LU 会话

SLU I C = 种方法a x LU-LU 会话:

- 通过发M终止自己(**TERMSELF**) | n或一v **UNBIND** | n。= v | n都I <致 " 即a x。
- 通过发M k s关U(**RSHUTD**) | n。C | n k s一v从 PLU 来D **UNBIND**。

要 " 即a x会话，SLU 发M **TERMSELF** | n x SSCP，SSCP 检i | { D LUA & C L序G否加入K C会话。如果G，那4 SSCP 发M一v } 、非 }]响&。取v Z }在9 C D主机 SNA f >，SSCP I发M一v **CLEAR** | n，| e } LU-LU 会话D y P消息，" S着发M一v **UNBIND** | n来a x会话。m外，SLU I发M一v **UNBIND** | n x PLU。

I PLU 停止 LU-LU 会话

PLU I C = 种方法a x LU-LU 会话:

- 通过发M **CLEAR** | n " R然后发M **UNBIND** | n，或v发M一v **UNBIND** | n。= 种方法都I <致 " 即D a x。
- 通过发M关U(**SHUTD**) | n。C | n <致P次序D会话终止。SLU 和 PLU 都P一v对话来f _ + 方停止发M }] " 确# K已S U K发M D }]。

a x LU-LU 会话对 SSCP-LU 会话；P 0响。

停止 SSCP-LU 会话和 SSCP-PU 会话

1主机发MM放_辑%元 (**DACTLU**) | n x SLU 1，SSCP-LU 会话a x。1 O次v人通信k通信服务器D SSCP-LU 会话a x 1，SSCP I通过发M一v M放物理%元(**DACTPU**) | n来a x SSCP-PU 会话。

断开主机4 S

1主机S U = 对 **DACTPU** | n D响& 1，| 返回一v | n x v人通信k通信服务器，例如h置断 * 响& 方 = (**SDRM**) | n(1 9 C SDLC 协议1)。SSCP 也I在任何1侯通过发M相同D | n x v人通信k通信服务器来 " 即断 * (a x y P D会话)。1会话以C方 = a x 1， y P以0活动D SLU S U = 一v合同 ' \ D指 > 。

消息号

y P在 LU-LU 会话中Z SLU 和 PLU 之间传MD } # }] w消息；3序`号。 SLU 对Z从 SLU 至 PLU D } # }] w消息维V一v序P D号k，对Z从 PLU 至 SLU D } # }] w消息维V m一v序P D号k。？v } # }] w消息获取一v大Z 0 f D } # }] w消息序P号D一v序P号k。在一v SLU 和 PLU 之间 (" D？v会话都P一I对D序P号。

对Z LU-LU 加急 }] w消息以及y P SSCP-LU 和 SSCP-PU 消息， 9 C无3序j 6符来f代3序号k。

1重新 (" K一v会话或发MK一v **CLEAR** | n 1， PLU 和 SLU + | G D序P号k h置为 0。PLU I C h置和b T序P号k (**STSN**) | n来 | D序P号。b 9 C在会话恢4或重新启动1 x行 } 确D序P `号。

1 SLU v = 一v序P号错误1，如果k s一v响&，那4 | 发M一v否定响& x PLU。1 SLU 读取一v响& 1，SLU 9 C响&序P号来9响& k原< k s关*。1 SLU `写一v响& 1，SLU X须a供原< k s D序P号。

重B启动和重B同 = 会话

如果 PLU 或 SLU v = 一v；I恢4错误，例如一v线7故O，那4你I能需要在重新启动 LU-LU 会话后重新同 = |。重新同 = LU-LU 会话 | 括重新处理I恢4消息及 (I选)重置消息序P号。&C L序I | 括重新发M丢 ' 消息D例行L序。

1重新启动和重新同 = 会话1，PLU 发M **BIND**、**STSN** 和 **SDT** | n。1发M **STSN** | n 1， I z z一v对话来 (" PLU 和 SLU 都I S \ D序P号。C对话 | 含K一v对话D **STSN** 消息和 } 响&。

如果 SLU v定需要再同 =，那4 SLU I发M一v k s恢4 (**RQR**) | n、一v否定响&或一v在C户检b字Z中带P故O h v D LU-Status | n (**LUSTAT**)。如果 PLU 发现故O或从 SLU S U = 一v **RQR** | n，那4 PLU 发M一v **CLEAR** | n来e } 来自网g D y P LU-LU 消息，一v **STSN** | n来 (" 新D序P号，以及一v **SDT** | n。

使C - i 4控制请求和响&

P w种协议I X制C Z k s和响&D定序规则。> B Z h v K C Z管理 SNA 网g、传M }]、以及同 = 网g组件D状，D协议中D一？分。

使C调 = - i

要\ b消息w D Y J对Z v人通信k通信服务器或主机 + l，你I以在 **BIND** | n中指定w =。w = v J C Z LU-LU } # }] w。1 w = 1， v人通信k通信服务器允许指定 } ？D消息w " R在允许发M = 加D消息0， H待响&。I在v人通信k通信服务器至主机w、主机至v人通信k通信服务器w、或 = _ O指定w =。一) 启动K LU-LU 会话， LUA X制y P w = 而；需&C L序D加入。

S收调 = - i

S U w = 协议 x Z PLU 对在 LU-LU 会话O D从 SLU 发M D消息D } ? 和频 J D X 制。 1 SLU S U K在 **BIND** | n O D w = 值 1， v 人通信 k 通信服务器自动 ? 制对 k 主机通信D ? v SLU D w = 。

在对 I 协L **BIND** | n D } 响&期间，你 I Q w = 值 | D为 } 0 外D任何值。 1 SLU 发M一 v 序 P D Z 一 u 消息 1， v 人通信 k 通信服务器在 k s /响&头 (RH)O h 置一 v m > + 返回一 v w = 响& D位。如果在 P L 序从 PLU S U一 v w = 响& 0 耗 ! K w = 计 } 值，那 4 ; P L 序 I 以发M = 加D }] 消息。如果& C L 序发 v 一 v 写Y作而 ; P S U = w = 响&，那 4 v 人通信 k 通信服务器延Y写Y作。

发M调 = - i

SLU 自动X制发M w = 协议。如果在从 PLU 至 SLU D消息中 h 置K w = 指 > 符，那 4 SLU 在& C L 序读取消息 1 发 v 一 v w = 响&。消息响& I | 含 w = 指 > 符或，如果 S U D消息 ; 需要响&，那 4 w = 响& I G一 v %独D w = 响&(IPR)。PLU 然后 I 发 M m一 v w = 消息窗 Z。

使C k + 工争C/触发器 - i

换向(CD)指 > 器 C Z 以下 = v 协议:

- k + 工 y C 协议， | 处 Z } # }] w 发M/ S U 模 =，其中 ? v 会话端 I 在会话D * < 或在发M或 S U 4 D最后一 v k s 后，发M } # }] w k s 。
- k + 工触发器协议， | 处 Z } # }] w 发M/ S U 模 =，其中一 v 会话端在 4 a x D 响&W j (RH)中 h 置 CD 指 > 器来 9 C m一 v 会话端 * < 发M。

CD 指 > 器 f _ S \ _ I 以 * < 发M。

例如，如果 SLU 启动一 v B 务，那 4 SLU * < 1 发M完 { h v B 务D消息。在最后一 v 消息中，SLU h 置K CD 指 > 器以 f _ PLU I 以 * < 传M回答K。如果 PLU 需要 = 加D信息来完 I B 务，那 4 | 发M一 v i 询 " h 置 CD 指 > 器。对话以C k + 工方 = x 行直 = B 务完 I 。在 k + 工对话期间，SLU I 9 C **SIG** | n 来 f _ PLU 停止发 M }] " | D }] w D方向。

使CW c - i

W c 协议 x Z SLU 和 PLU 对 }] 传 d D O下文X制(m > 会话 f 及% v B 务)。W c 协议 # 护 1 0 会话 ; ; " 行D B 务打断。方括号 | 含K B 务处理 1 间。

在方括号中D Z 一 u 消息 | 含一 v * < 方括号(BB)指 > 器，在方括号中D最后一 u 消息 | 含一 v a x 方括号(EB)指 > 器。% v 消息 I 以G一 v W c (如果 | | 含 = v 指 > 器)。

对Z一 v W c 会话，**BIND** | n 指定一 v LU 为Z一 v 5 话 _，而m一 v LU 为 k s _。Z一 v 5 话 _ I * < 一 v W c 而 ; - m一 v LU 许 I 。 ; 过， k s _ X须向Z一 v 5 话 _ k s " 获取许 I 权来 * < 一 v W c 。

BID | n G I k s _ 发 v 以 k s * < W c 许 I 权D一 v } # }] w k s 。对 **BID** | n D } 响& m > Z一 v 5 话 _ ; + * < 一 v W c， + | + H 待 k s _ * < 一 v W c 。对

BID | n D否定响&m > Z一v 5话_；允许k s _ * <一vWc。 1 Z h启动Wc D许I权1， Z一v 5话_ I发M一v准8 S U(**RTR**) | n。

Z一5话_对 **BID** | n m >一v否定响& 1带P = v响&代k之一:

Bracket-Bid-Reject-RTR-Forthcoming
m >对Z那v **BID** | n +在以后发M一v **RTR** | n(Z权启动一vWc)。 k s _ I H待 **RTR** | n或再次发M **BID** | n。

Bracket-Bid-Reject-No-RTR-Forthcoming
m >以后对Z那v **BID** | n ；+发M **RTR** | n。如果k s _仍要* <一vWc，那4 { X须再次发M **BID** | n。

f代发M一v **BID** | n， "在其后z f一v带P BB 指>器D 4W FMD, k s _ I "T通过发M一v带P BB 指>器D 4W FMD 来u启一vWc。 Z一v 5话_ I Z h "T一v }响&或| I C一v否定响&(m >任何否定响&代k)来\ x "T。；过，如果k s _通过发M **CANCEL** | n来停止带P BB 指>器D 4，那4Wc；启动，而；管G否U =响&。 I I Z一v 5话_发v **RTR** | n来Z h k s _ * <Wc D许I权或_ i 4 k s _ G否要* <一vWc。

对 **RTR** | n D }响&m > k s _ + u启下一vWc。如果k s _；想要u启Wc，那4 k s _发v一v带P；需要 RTR 检b k D否定响&。

使C数] 4 S - i

}] 4 S G C Z传M一组相关消息D I选协议。要从 SLU 发M 4 S D消息， SLU Q消息D 4 * <(BC)指>器h置为 1，以m > 4中D Z一v消息。对Z在4中Z一v和最后一v之间D y P消息， SLU h置 BC 和4 a x指>器为 0。对Z 4中D最后一v消息，再Q EC h置为 1。 1 SLU S U消息1， | b T 4 S指>器来确定消息G否4 S。

}] 4 S协议|括三种类型D 4，如下:

- 无响& 4。 4中D ? v k s j记为*; P响&*。
- 异#响& 4。 4中D ? v k s j记为*异#响&*。
- 确P响& 4。 4中最后一v k s j记为*确P响&*; 4中y P其| D k s j记为*异#响&*。

1发M一v消息4 x PLU 1，如果 SLU 或 PLU 发现一v消息错误，那4 SLU I发M一v **CANCEL** | n。如果 SLU 发M一v **CANCEL** | n x PLU，则 PLU 丢弃已S U = D 4中D y P消息。如果 PLU 对4中D任何一v消息发MK一v否定响&，那4 SLU } # a x 4或发M一v **CANCEL** | n。

数]；换控制方法

SNA 会话在P次序；换}] D规则下x行。

w动 - i

在传M c O，通过k + 工(HDX)协议或全+工(FDX)协议来；换}]。

１９Ｃｋ＋工协议１，同一１间ｖ在一ｖ方向ＯＰ｝］ｗ，一ｖ　LU　ｖ发M而ｍ一ｖ　LU　ｖＳＵ。在ｋ＋工触发器协议中，＝ｖ　LU　都知@哪一ｖ　LU　Ｐ权发M及哪一ｖ　LU　Ｐ权ＳＵ。在指定１间伙ｉ　LU　同意ｄ｜方向，因此Ｓ＼＿Ｉ以发Ｍ＂Ｒ发Ｍ＿Ｉ以ＳＵ。

１９Ｃ全＋工协议１，在任何１间｝］都Ｉ＋向ｗ动。＝ｖ　LU　都Ｉ无约ｘＸ发M和ＳＵ。

响&方式

？ｖ　SNA　消息要４ＧｋｓＬ要４Ｇ响&。？ｖ从　LU　来Ｄｋｓ引ｖ从伙ｉ　LU　来Ｄ匹配响&。因为响&带Ｐｋｋｓ相同Ｄ传ｄ序Ｐ号，ｙ以响&和ｋｓＩ通过｜ＧＤ序Ｐ号来匹配。

１&ＣＬ序已ＳＵ一ｖ其 RH 指定Ｋ？制性响&Ｄｋｓ１，&ＣＬ序Ｘ须ｚＩ＂发Ｍ一ｖ响&消息。响&方＝规则ｖ定ＫＸ须何１发M响&。

在＂即响&方＝下，&ＣＬ序Ｘ须在发Ｍ｜自己Ｄ任何ｋｓ０发Ｍ一ｖ响&ｘｋｓ。；过，在延Ｙ响&方＝下，Ｉ以在ＳＵ＝ｋｓ后Ｄ任何１侯发M响&。

LUA 关＊表格

LUA　ｚ踪ｘ入和ｄｖｋｓＤ序Ｐ号，直＝｜ＧＳＵ＝响&-直＝&ＣＬ序发ｖ响&ｘｘ入ｋｓ，或直＝ PLU 响&一ｖｄｖＤｋｓ。ｂ些号ｋ记＜在ｖ人通信和通信服务器中Ｆ为关＊*mq*Ｄｘｒ中。

在＂即响&方＝下，在一ｖ会话中ｖＩｚＩ很ＹＤ未完Ｉｋｓ，ｄ型Ｘ为最多一ｖ。在延Ｙ响&方＝下，｝？Ｉ大很多。

LUA 关＊ｍｑＧ动，管理Ｄ。LUA　Ｉ记＜任何｝？Ｄ响&。如果Ｐ非＃多Ｄ响&累积(Ｉ能ＩＺＬ序＿辑错误)，那４服务器运行内存；够，＂Ｒｖ人通信ｋ通信服务器Ｉ能关Ｕ。

l 常响&请求(RQE)

在大多｝ｉｖ下，LUA　Ｉ自动Ｑｋｓ和响&关＊，而；需Ｌ序Ｄ任何ｏ助。 LUA 观ｌｋｓ和响& RU(１｜Ｇ会话中ｗ动１)。 LUA　Ｉ以ｆ＿你何１ｋｓ需要响&，以及何１响&已发ｖ。然而，Ｐ一种ｉｖ LUA　；能ｆ＿你响&Ｇ否＋发ｖ，＂Ｒ你ＤＬ序Ｘ须ｆ＿｜。

ｋｓＤ RH 中Ｄ位字段指定Ｋ响&Ｇ？制性Ｄ、；需要Ｄ，或ＧＩ选Ｄ。１；需要响&１，LUA；需要在｜Ｄ关＊ｍｑ中存储ｋｓ号。？制性Ｄ响&Ｘ须在ｗ中Ｐ下一ｖ消息１；发Ｍ。 LUA 在关＊ｍｑ中ｄ入消息，＋｜＋很ｌ；ｅ｝，因为Ｘ须ｍＯ响&。

Ｘ须ｖ１ＳＵ LU　；能Ｓ＼或处理 RU　１，RH 中Ｄ错误响&指＞器(ERI) 指定响&ＧＩ选Ｄ。ｂ种Ｉ选响& RU　Ｆ为异＃响&ｋｓ(ｕ写 RQE)。对Ｚ RQE，LUA　；能总Ｇ自动管理｜Ｄ关＊ｍｑ。 ｍｑ　ｍ12　ＥｖＫ LUA　Ｉ从关＊ｍｑ自动ｅ｝ＳＵＤ RQE　Ｄ５例，以及 LUA　Ｘ须在ｅ｝ＳＵＤ RQE　０Ｈ待从&ＣＬ序来Ｄ信号Ｄ那些５例。

m 12. RQE D e }

" 4 响&方式	延迟响&方式			
动词	HDX	FDX	HDX	FDX
RUI_READ	自动	自动	&C L 序响&	&C L 序响&
RUI_WRITE	自动	&C L 序响&	&C L 序响&	&C L 序响&

在 " 即响&方 = 下 HDX 或 FDX 会话中，LUA I 以在&C L 序 k s d 入(9 C RUI_READ) 1，丢弃 RQE D号，因为，在 " 即响&方 = 中，响&X须在 I 发Mm一 v k s 0 ；发M。同样,z0 " 即响&方 = 下 HDX ，S中， LUA I 在&C L 序 k s d v(9 C RUI_WRITE) 1 丢弃 RQE 号--因为 d v 要 4 G RQE 响&，要 4 ；+ 发M响&。

在 y P 其 | D 5 例中，LUA ；能确 # G 否 + z I 对 RQE D响&。&C L 序X须 q = 化 " 发M一 v } 响& x RQE， m > 为K PLU(| 要D v G 否定响&)，而G为K通知 LUA 已 S \ RQE " R ； + z I 一 v 否定响&。

LUA 然后 I 从m q 中 e } RQE。因为响&为 } 响&而 PLU 要D v G 否定响&， y 以 LUA ；在网 g O 传M&C L 序D响&。

简而言之，要 (助 LUA D G，你D&C L 序X须在S U D G 确 P 响& RU 1，处理S U D RQE RU。

会话概 *

I 在 x 定会话O 9 C D X 定 SNA 协议和约定，总 a 起来，构 I K 会话D 『 E 要 』。传 d 服务(TS) E 要和功能管理(FM) E 要 b = v E 要， I C Z 会话。在 BIND 1 x 行E 要D 选择。

TS 概 *

SNA 定义K ` 号为 1、2、3、4、7 D五 v TS E 要。；过，因为 TS E 要 1 v 在 SSCP 和 PU 之间 9 C， y 以 v E 要2、 3、4 和 7 I J C Z LUA &C L 序。 | G 在 I 发 v D SNA | n 中 P y ；同，如m q m 13 中 y >。

m 13. TS E 要X性

概 *	调 = 使C	CLEAR	CRV	RQR	SDT	STSN
2	总G	9 C	；9 C	；9 C	；9 C	；9 C
3	总G	9 C	I 选D	；9 C	9 C	；9 C
4	总G	9 C	I 选D	9 C	9 C	9 C
7	I 选D	；9 C	I 选D	；9 C	；9 C	；9 C

FM 概 *

SNA 定义K ` 号为 0、2、3、4、6、7、18 和 19 D K v FM E 要。；过，因为E 要 0 和 6 v I SSCP 9 C， " R E 要 19 v I LU 类型 6.2 9 C， y 以只P 五 v E 要 I J C Z LUA &C L 序。？v E 要在 \ 限D SNA h) 中；同。

FM E 要D 大约 * 要显 > 在 Z 157页D m 14 中。UW% 元意味着C E 要中；P SNA h) \ 限 -- | I P 在 BIND N } 中指定D 任何C 途。

LUA RUI 支V FM E要 2、3、4、7 和 18。

m 14. FM E要X性

SNA h施	FMP 2	FMP 3	FMP 4	FMP 7	FMP 18
k s 方=	SLU 9 C延Y				
响&方=	SLU 9 C "即	"即	"即	"即	"即
RU 4	v % v RU 4				
检i $度压u				v LU 0	
FMH-1 会话X制i (SCB)压u	；允许				
允许D }] w X制RU	无	• CANCEL • SIGNAL • LUSTAT (v SLU) • 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 W j	；允许				
W c	\限9 C				
w协议	FDX				
恢4	v I PLU				

使C RUI LUA 动词

&C L序通过 LUA 动词访问 LUA。？v动词a供N } x LUA, |执行y期望D功能"R返回N } x &C L序。

动词摘*

以下G&C L序I 9 C D七v LUA 动词D简%*要。(若需？v动词D详细5 w, k N阅 Z13B RUI 动词。)

RUI_BID

9 &C L序确定何1从主机来D信息I读取。

RUI_INIT

为 LUA &C L序(" LU-SSCP 会话。

RUI_PURGE

取消未完I D **RUI_READ** 动词。

RUI_READ

在 LU-SSCP 会话或 LU-LU 会话O， S U从主机发M = LUA &C L序D LU D }] 或状，信息。

RUI_TERM

a x LUA &C L序D LU-SSCP 会话。 | 也 a x LU-LU 会话(如果 | G活动 D)。

RUI_WRITE

在 LU-SSCP 会话或 LU-LU 会话O发M }] 至主机。

此外， v人通信 k 通信服务器 I 返回 RUI_INIT_STATUS j > x LUA &C L序。 | 带 P在处理未完 I **RUI_INIT** 动词 1 y = 达D状，D P关信息。

RUI 会话

一v RUI 会话 | 含K I &C L序确定D一段 1 间内对一 v LU D y P权， | I | 含在 SSCP 和 LU 之间 (" 一 v 会话(F 为一 v SSCP-LU 会话)。 RUI 会话也 I | 含 (" 一 v 或一 v 以O D ；重 ~ LU-LU 会话。如果 I Z 合同 ' \ 或m一 v 4 位状，D原因， SSCP-LU 会话 ' \ ，则 RUI 会话 a x。 RUI 会话以 **RUI_INIT** 动词 * < " R一 c 以 **RUI_TERM** 动词 a x。

发出 RUI 动词

Z 159页D m 15 显 > K对 Z 一 v x 定D LU， RUI &C L序 I 向 RUI API 发 v 动词D P 效 u 件。在最左 P D项目 m > K x 入动词。在 Z 一行 D 项目 m > 处理中 D 动词。如果在 m q 中D项目为 『OK， 』那 4 动词组合代m一 v P 效D u 件。如果在 m q 中D项目G 『Error， 』那 4 动词组合代m ； } 确 u 件 " R返回 x LUA &C L序一 v 错误代 k。

m 15. RUI 动词 u 件

		处m中命 n					
x 入命 n	没 P 当前会话	**RUI_INIT**	**RUI_TERM**	**RUI_WRITE**	**RUI_READ**	**RUI_PURGE**	**RUI_BID0**

SSCP norm
LU-SSCP 会话， } # }] w

LU norm
LU-LU 会话， } # }] w

LU exp
LU-LU 会话，加急 }] w

+rsp 对指定消息D } 响&

BC * < 4

MC 4 中间

EC a 尾 4

CD 换向指 > 器 h 置

RQD X需D确 P 响&

LUA & C程 r 发出的动词	**SNA** 消息	w向 & C程 r	主机
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)	}] , BC	----->	
RUI_WRITE (LU norm)	}] , MC	----->	
RUI_WRITE (LU norm)	}] , EC, CD, RQD	----->	
RUI_READ (LU norm)	}] +rsp	<-----	
RUI_READ (LU norm)	}] , BC	<-----	
RUI_READ (LU norm)	}] , MC	<-----	
RUI_READ (LU norm)	}] , EC, RQD	<-----	
RUI_WRITE (LU norm)	}] +rsp	----->	
RUI_READ (LU exp)	UNBIND	<-----	
RUI_WRITE (LU exp)	UNBIND +rsp	----->	
RUI_TERM	(NOTIFY)	----->	
	(NOTIFY +rsp)	<-----	

在C例子中， & C L 序执行K以下 = 骤:

1. 发v **RUI_INIT** 动词来 (" LU-SSCP 会话。 (**RUI_INIT** 动词直 = v 人通信 k 通信服务器L序已从主机 S U K 一 v ACTLU 消息 " R发 v K 一 v } 响& 1 E完 I ； ；过， b 些消息 I ？ v & C L 序X制而 ；) 6 x LUA & C L 序。)
2. 发M **INITSELF** 消息 x SSCP 来 k s 一 v **BIND**， " R读取响&。
3. 从主机读取 **BIND** 消息， " R ` 写响&。 b M (" K LU-LU 会话。
4. 从主机读取 **SDT** 消息， | 指 > u < 化完 I " R I 以 * < }] 传M。
5. 发M一 v | 含三 v RU D 4 (最后一 v 指 > 需要确 P 响&)， " R读取响&。
6. 读取 | 含三 v RU D }] 4， " R ` 写响&。
7. 从主机读取 **UNBIND** 消息， " R ` 写响&。 b M终止K LU-LU 会话。

8. 发v **RUI_TERM** 动词来终止 LU-SSCP 会话。 (v人通信k通信服务器L序发M NOTIFY 消息x主机"RH待一v}响&; ;过, b些消息I?v&CL序X制而;)6x LUA &CL序。)

BIND #验

在 LU-LU 会话u<化期间, 主机发M一v **BIND** 消息x v人通信k通信服务器 LUA &CL序, ||含KI LU-LU 会话9CD信息 (例如, RU 大小)。v人通信k通信服务器C一v **RUI_READ** 动词返回C消息x LUA &CL序。LUA :责检i在 **BIND** 中指定DN}G否JC。&CL序P以下选项:

- S\ **BIND**, 通过发v一v|含对 **BIND** D一v OK 响& **RUI_WRITE** 动词。在响&中;需要发M}]。
- "T协L一v或一v以O **BIND** N}(bv在 **BIND** GI协L1;允许)。要b样做, &CL序发v一v|含 OK 响&D **RUI_WRITE** 动词, +Q修DD **BIND** 作为}]。
- \x **BIND**, 通过发v一v|含否定响& **RUI_WRITE** D动词, 及Q相&D SNA 检bk作为}]。

若需P关 **RUI_WRITE** 动词D信息, kN阅Z13B RUI 动词。

"**:** **BIND** N}D验证, 和确#发MDyP消息k|G一致, G LUA &CL序D责任。 ;过, P以下=v限制:

- v人通信k通信服务器\x任何指定K一v RU $度大Z在 **BIND** 中指定大小D **RUI_WRITE** 动词。
- v人通信k通信服务器需要 **BIND** 指w次级 LU GyC$_, "R错误恢4GyC:_D:责。

否定响&和 SNA lbk

SNA 检bkI能在以下iv下返回x LUA &CL序:

- 1主机发M一v否定响&x从 LUA &CL序来Dks1, ||含一v指>否定响&原因D SNA 检bk。bGC一vf后D **RUI_READ** 动词来(fx&CL序D, 如下:
  - 主返回kG LUA_OK。
  - ks/响&指>器、响&类型指>器以及检b}]|含指>器都h置为 1, m>K|含检b}]D一v否定响&。
  - **RUI_READ** 动词返回D}]G SNA 检bk。
- 1v人通信k通信服务器SU=从主机来D;}确}]1, |一c发M一v否定响&x主机"R;传M;}确D}]x LUA &CL序。bGC一vf后D **RUI_READ** 或 **RUI_BID** 动词来(fx&CL序D, 如下:
  - 主返回kG LUA_NEGATIVE_RSP。
  - 次级返回k为发Mx主机D SNA 检bk。
- 在P些iv下, v人通信k通信服务器检b=I主机a供D}]无效, +;能确定+发MD}确D检bk。在b种iv下, |C以下方法通过 **RUI_READ** 动词在异#ks(EXR)中Q;}确D}]传Mx LUA &CL序:
  - ks/响&指>器h置为 0, m>一vks。

– 检 b }] | 含指 > 器 h 置为 1, m > | 含 K 检 b }] (C 指 > 器 } # X v C Z 响 &)。

– 消息 }] C (议 D SNA 检 b k f 换。

&C L 序 X 须然后发 M 一 v 否 [illegible]

&C k k k

因为b v缘故， v人通信k通信服务器9 LUA &C L序发v一v }响&x从主机来D一v v异#响&k s(b F为一v礼Z确认)。响&；发M x主机， + v人通信k通信服务器9 C | 来e } k k s关* D存储器。

清除数] 至4 a束

1主机发M一v k s %元4 x LUA &C L序1，在发M一v响&0， &C L序I能一直H待= S U = 4中D最后一v RU，或_ | I能发M一v否定响&x ；G 4中最后一v D RU。如果一v否定响&在4中发M v，那4 v人通信k通信服务器e } 4中f后D y P RU， " R ；Q | G发M x &C L序。

1 v人通信k通信服务器S U = 4中最后一v RU 1， |通过h置 **RUI_READ** 或_ **RUI_BID** 动词D主返回k为 LUA_NEGATIVE_RSP，及次返回k为 0 来Q b m > x &C L序。

": 1在4中1，主机I能发M一v消息，例如 CANCEL 来终止4。在b种i v下，C **RUI_READ** 动词返回x &C L序一v CANCEL 消息， " R ；9 C LUA_NEGATIVE_RS 返回k。

配置

X须通过9 C v人通信k通信服务器 NOF 动词或通过 SNA Z c配置L序来配置 LUA &C L序9 CD ? v LU。(若需 | 多信息， k N阅*系统管理L序h计*。) 此外，配置I能 | 含 LUA LU X。 一v X G P相F X性D一组 LU，因此&C L序I 9 C组中D任何U闲 LU。1 P H LU |多D&C L序I C 1， b I C来在先来先服务D基础O分配LU，或a供一v在；同D 4 S O D LU 选择。

LUA LU 池(可选的)

如果需要，你I为&C L序配置9 C多Z一v D LUA LU， " R Q LU 组合入X中。b意味着&C L序在T图启动一v会话1 I指定一v X而；G指定一v X定D LU， " R | + 从X中分配I C D LU。

LUA &C L序通过发v一v带P LU { F D **RUI_INIT** 动词来向v人通信k通信服务器m > |要启动一v会话。C { F X须匹配先0已在 *系统管理L序h计*中定义D LUA LU 或 LU X D { F。 v人通信k通信服务器如下9 C C { F:

- 如果a供D { F G；在X中D LU { F，如果 | I C，那4会话+ ；分配9 C那v LU (即，如果 | 还； P ； LUA &C L序9 C D话)。
- 如果a供D { F G一v LU X D { F，或G在X中已在C D一v X定D LU { F，那4会话+分配9 C X中Z一v I C D LU(如果P一v I C)。

": b I能；G在 **RUI_INIT** 动词中指定{ F D LU。

SNA API 客户机考G

如果你D LUA &C L序驻t在M户工作 > O，那4也&C在 > X工作 > O定义一v LUA 会话。C LUA 会话{ F I |含多v通信服务器和 LUA 定义，以允许 SNA M户代k在，S d C ；I C 1翻转至新D服务器。

第10章 RUI LUA 动词特T

> B | 括下P LUA 动词D X b i v和9 C技I。

- 处理异# k s --LUA k s L序发v否定&答
- 通过L序h计最小化 LAN 通信？
- 处理 LUA 动词D；确定终止
- 恢4会话 ' \

处m l常请求

RUI 和 SLI 监S K几v协议D状，"验证K RU D q =。 1 S Z检b =来自主_辑%元(PLU)D； j准 RU 1， | X须发v否定&答。 LUA 通过+ x入D RU q =化I异# k s来通知您D&C L序已检b = C错误。 EXR 文件通过k s动词(**RUI_BID** 或 **SLI_BID**) 或d入动词(**RUI_READ** 或 **SLI_RECEIVE**)传] =您D L序。EXR 通过下P k s j b(RU)中D u件m >:

- *lua_rh.rri* h置I 0 (RU G k s %元)
- *lua_rh.sdi* h置I 1 (| 括检b }])

b G一v RU 位D异#组合。检b }]通# G响& RU D内容，而；G k s RU。 LUA 9 C C异#组合/ (L序防止 PLU w显v错D异# i v。 4 字Z D检b k G EXR D一？分； | 指定K检b = D错误。 } K检b }]，LUA 也返回最多三v原< RU D字Z。

更改动词G <

&C L序X须+ EXR q =化I否定&答，" 9 C RUI_WRITE + | 发M = PLU, b取v Z 9 C D API。要+ EXR d入转换I响& d v，在动词记<中须做下P | D:

- + *lua_rh.rri* h置为 1 以m > b G一v响&。
- + *lua_rh.ri* h置为 1，5 w b G一v否定&答。
- 在基Z *lua_flag2* 值D *lua_flag1* 中h置J 1 D }] w。
- + *lua_message_type* h置为 LUA_MESSAGE_TYPE_0。
- + *lua_opcode* h置为 LUA_OPCODE _RUI_WRITE, b取v Z 9 C D API。
- + *lua_data_length* h置为 4, b G检b }] D $度。
- + *lua_data_ptr* h置为检b }] D X址, | D位置取v Z检b = EXR-if D动词, C动词G RUI_BID, 检b }]在动词记< D²取}缓e x²中; 如果动词G RUI_READ, 则检b }]在d入缓e x中。
- + *lua_max_length* h置为 0。

现在，您D L序I以9 C EXR 动词记<和缓e x来u <化 RUI_WRITE 以发M否定&答。

处m方 (号请求 \ x

} 在 EXR 中 I LUA a 供 D 检 b k G 唯一返回 = PLU D 合 J 值 b 一种 i v 外, | J C Z y P i v。然而, 1 9 C 方括号, " R PLU 要 s I 为扬 y 器(speaker) 1 , 您 D & C L 序 I 以选择检 b k :

- LUA I 以 \ x PLU D BID | n。要 \ x BID, LUA 会 q = 化一 v | 含检 b k LUA_BB_REJECT_NO_RTR D EXR, 则 + \ x 启动 C 方括号 k s R 以后 + ; 会发 v RTR | n。C 检 b k D } 值 G 0x00001308L (以 Intel**, 或字 Z 转换形 = , 如同您要 + | ` 写入 C L 序中 D 形 = 一样)。
- 如果 | 支 V 方括号 R f 后 I 以发 v RTR | n, 则 & C L 序 I 以 S \ BID | n。要通知 PLU | D BID I 以 S \ , I 以 + 检 b k | D 为 LUA_BB_REJECT_RTR (值 0x00001408L), h v RTR D 检 b k + 会 v 现。在 b 以后您 D & C L 序 X 须 q = 化 " 发 M RTR 消息。

n ! 化 LAN 通 E ?

如果 & C L 序 X 须运行在 M 户工作 > O, I 通过减 Y 9 C 『bid logic.』 + LAN D 通信 ? u 至最小。

u Y RUI_BID 的使 C

动词 RUI_BID + 一直 H = 服务器 P I C D }] % 元后再完 I 。 RUI_BID D 完 I M G 通知 L 序 X 定 }] w O X 定 $ 度 D }] 已准 8 M 绪。然后 L 序 I 以分配缓 e x " 发 v }] D RUI_READ k s 。

1 发 M 带 P d 入动词 D k s 动词 1 , z I 下 P D v LAN 消息:

- 要启动 RUI_BID D 消息
- 通知工作 > k s 完 I D 消息
- 要启动 RUI_READ D 消息
- 返回 }] = 工作 > D 消息

然而, RUI_READ I 在一 = 中做完相同 D 工作。如果只启动 RUI_READ 动词 " H 待 | 完 I , 则 + > } = v LAN 消息。

『bid logic』 D 唯一好处 G 您在 S U | 1 I 以知@消息 D $ 度。 b 允许您延 Y 分配 }] 缓 e x , 直 = 知@需要多大 D 缓 e x 为止。 1 只 9 C d 入动词 1 , X 须预先知@最大 D 缓 e x 大小, 而 ; G 在 k s 完 I 后分配缓 e x 。

处m挂起

RUI 动词 D 完 I 取 v Z PLU & C L 序、主机系统、网 g 和 v 人通信 k 通信服务器 D Y 作。如果 b 些响 & 中 D 任何一 v 响 & 缓 } 或 ; 能响 &, 则动词 I 以无限制挂起。在 h 计 L 序 1 , I 以 x h C 户或 L 序一种终止挂起动词 D 方法来预 b 挂起 D 发 z 。

取消 RUI_INIT

RUI_INIT 动词会一直中止直 = 主机激活分配D LU。通 # 主机会在& C L 序启动 0 发M ACTLU | n， + " ；一定要 s | b 样做。 1 & C L 序启动 1，大型机 I 能会停机，或还在 u < 化。

如果 L 序需要取消；挂起D RUI_INIT，则 | I 以发 v RUI_TERM 动词。

取消 RUI_WRITE

1 9 C w = 1， d v I 以挂起。如果主机暂 1 停止读取 }] 或；能发M w = 响&，RUI_WRITE I 以中止以H待w = 窗Z D打 * 。

如果 L 序需要取消中止D RUI_WRITE， | X须关U RUI_TERM 会话。

取消 RUI_READ

d 入动词通 # 为挂起状，， 知@ d 入 = 达动词指定D w O。 L 序 I 以 9 C RUI_PURGE 取消暂挂 RUI_READ。关U会话同样 I 以取消暂挂 d 入动词。

确保动词完成

如果 L 序错误处理K动词完 I D 话， | 会造 I 无限制H待D现象。如果 L 序启动动词， ；P注意动词已同 = 完 I，从而H待异 = 完 I，则C L 序 + @远H待下去。

RUI 入Z c 返回作为显 = a 果， 5 w主要D动词返回 k 已 - 执行。 5 w动词G否已 I 功异 = 完 I D最简%D方法G b T LUA_IN_PROGRESS D显 = 返回值，如图 图 8 y > 。

```
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
```

图 8. b T 动词完 I

压 u 数]

}] 压 u 支V RUI 和 SLI API S Z。 }] 压 u D 9 C + I ？v 会话D BIND 和 BIND 响&来协L b v。如果压 u 要C Z 会话，则 LZ9 或运行 $ 度 ` k (RLE) 压 u c 法 + I 主 LU (PLU)入 > S \ ， " R RLE + C Z 发M }] = PLU。

对Z RUI 和 SLI APIs， }] 压 u I 以 I 以下方法处理:

- & C L 序压 u 和 b 压 u }]
- 通信服务器压 u 和 b 压主机 }]，或从& C L 序 S \ 未压 u D }] 。

- L 每个会话数] 压 u 的规则

下 P G 协L ？v 会话D RUI 和 SLI API }] 压 u D 规则。

RUI 规则

1. 要允许 RUI &C L 序处理 }] D 压 u 和还原:
   - RUI &C L 序 U = 含 P 25 字 Z 中 D 6 和 7 位 D BIND k s，以 5 w 压 u 已 a 供或还在 k s。
   - RUI &C L 序 &C 返回 O 定 D 带 P 25 字 Z 中 D 6 和 7 位 D BIND 响 &，以 5 w “已 S \ K y a 供 D 或要 s D 压 u ”。
2. 要允许通信服务器在 RUI &C L 序一方处理压 u:
   - 9 C 通信服务器 SNA Z c 配置 5 C L 序来 5 w C Z c 支 V 压 u，只需要:
     - 选择配置 Z c
     - 选择 _ 级
     - + Z c 支 V D 压 u 级 p h 置为 RLE
   - RUI &C L 序 U = K 带 P 25 字 Z 集合 D Z 6 和 Z 7 位 BIND k s，以 5 w 压 u 已 a 供或还在 k s。
   - RUI &C L 序返回 K 带 P 25 字 Z 集合 D Z 6 和 Z 7 位 D 确定 BIND 响 &，以 5 w “; 要压 u ”。通信服务器拦 X " 修 D BIND 响 &，然后 + }] 压 u " 还原 = 主机中。

SLI 规则

1. 要允许 SLI &C L 序处理 }] D 压 u 和还原:
   - 1 发 v **SLI_OPEN** 动词 1，SLI &C L 序 X 须 a 供 BIND 回 P 例 L。
   - 1 启动 SLI &C L 序 D BIND 回 P 例 L 1， SLI S U K _ P 25 字 Z 集合中 D Z 6 和 Z 7 位 D BIND k s，以 5 w 压 u 已 a 供或还在 k s。
   - SLI &C L 序 &C 返回带 P 25 字 Z 集合中 D Z 6 和 Z 7 位 D BIND 响 &，以 5 w “已 S \ a 供或要 s D 压 u ”。
2. 要允许通信服务器处理 SLI &C L 序 D 压 u:
   - 9 C 通信服务器 SNA Z c 配置 5 C L 序来 5 w C Z c 支 V 压 u，只需要:
     - 选择配置 Z c
     - 选择 _ 级
     - + Z c 支 V D 压 u 级 p h 置为 RLE
   - 如果 &C L 序; a 供 **SLI_OPEN** 动词 D BIND 回 P 例 L，则 SLI + 4 U 缺 ! h 置 D BIND 响 & 来 5 w 通信服务器会为 SLI 压 u 和还原 }]。
   - 如果 &C L 序; P a 供 BIND D 回 P 例 L:
     - 1 启动 BIND 回 P 例 L 1， | S U _ P 25 字 Z 集合中 D Z 6 和 Z 7 位 D BIND k s，以 5 w 压 u 已 a 供或还在 k s。
     - SLI &C L 序返回带 P 25 字 Z 集合中 D Z 6 和 Z 7 位 BIND 响 &，以 5 w “; 要压 u ”。通信服务器拦 X " 修 D BIND 响 &，然后 + }] 压 u " 还原 = 主机中。

从会话失 \ 中恢复

P = 种 i v I 9 LUA 会话因为错误而关 U:

- 1 LUA 动词完 I " 带 P 主返回 k LUA_SESSION_FAILURE，或

- 1 在 RUI_INIT I 功完 I 后，LUA 动词完 I R 带 P 主返回 k LUA_STATE_CHECK 和次级返回 k LUA_NO_RUI_SESSION。

C 会话 I - # 重新 (" 。如果 L 序 k s D 话，LUA + T 图恢 4 。

1 L 序 S U = 因错误而关 U D LUA 会话 1 ，如果要恢 4 ，则须执行以下 Y 作:

- \ b 关 U 会话-C 会话已 - 关 U 。
- C 原先打 * 会话(RUI_INIT) D 动词重新打 * 会话。如果 C 动词完 I " 带 P 非 c 返回 k ，则会话；能在此 1 重新启动。
- 1 恢 4 x 行 1 通知；互 = C 户，因为 C 恢 4 需要一些 1 间。C 户工作 D 状，+ 取 v Z PLU & C L 序 D h 计。

第11章 实现 LUA 程 r

> B h v 5 现和 ` 写 LUA L 序 D w v 方 f 。 | 括下 P b 目:

- w C LUA 服务 " + 其定序
- ` 写 LUA L 序
- 9 C 异 = 完 I 和回 P 功能
- 在 ; 同 D 平 (` 译和 4 S

LUA D通信服务器 5 现 h 计为 k Microsoft** NT SNA Server 二 x 制兼容 " 类 F k OS/2 Communications Manager/2 f > 1.0 LUA D RUI 和 SLI S Z D 5 现。

编 4 LUA 程 r

LUA | 含一 v 主 DLL, C Z RUI 动词和 SLI 动词。 LUA & C L 序 w C C DLL 来发 v 动词。

LUA & C L 序在动词 X 制 i 中 h 置选定 D 字段 " w C RUI 或 SLI, + 指 k 传] x 动词 X 制 i 。动词 X 制 i 中 D 字段定义对 LUA.D \ k s Y 作。在 LUA 返回对 & C L 序 D X 制之 0, LUA 修 D 动词 X 制 i 中 D 字段来 m > Y 作 a 果。然后, & C L 序 I 以在 f 后 D 处理中 9 C 从动词 X 制 i 中返回 D N } 。

下 f D m q 显 > y a 供 D 头文件以及 ` 译和 4 S RUI L 序 y 需 D b D 源 L 序模 i C 法。

m 16. Y 作系统 D RUI API D 头文件和 b

Y w 系统	头文 ~	库	DLL 名称
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

m 17. SLI API D 头文件和 b

Y w 系统	头文 ~	库	DLL 名称
WINNT & WIN95	WINLUA.H	WINSLI32.LIB	WINSLI32.DLL
WIN3.1	WINSLI.H	WINSLI.LIB	WINSLI.DLL
OS/2	LUA_C	ACSSLI.LIB	ACSSLI.DLL

": SLI API 在服务器 O \ 支 V, + 通信服务器 M 户机 ; 支 V | 。

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

调 C LUA 服务

L 序通过 w C 指定 D 入 Z c " 传] % v N } -- F 为*动词记* < D }] a 构 X 址来 w C LUA 服务。 C 记 < | 含某 v X p 函 } D d 入 N } 。LUA C 来自 C Y 作 D d v N } | 新 C 记 < 。

m b 动词G < Z 容

d 然 - 过；同D a 构化， + b 三种动词记 < 类型都为以下N } a 供字段:

Y w　指定 + 执行D X p Y作D号 k。 Y作D符号 { 在 『 _cons.h 』 | 括文件中 5 w。

动词G < 长度

动词记 < D大小 I 能在Y作间 P 转换 " R LUA 需要 | 来处理记 < 。

会话标识符

在通信和服务动词中， 一 v j 6 会话D号 k 或会话D { F。

主返回 k

I LUA 返回以指 > 一 c D I 功或 ' \ D } 字。

次返回 k

' \ 1 I LUA 返回以指 > X定问 b D } 字。

相关 r S

& C L 序 I 以 9 C | 来 + 动词记 < ， S = 其 | }] 或在异 = 完 I 期间 j 6 动词记 < D $ 型 { } 。

发出 d 柄

在动词异 = 完 I 1 + 发 v D B 件D d z 。

对Z Windows NT 和 Windows 95， & C G B 件 d z 。在 Windows 3.1 中， & C G 窗Z d z 。对Z OS/2， & C G 信号 d z 。

大多 } 字段都 P 相同D }] 类型 " R 在 v 现D ？ v 动词记 < 中都 P 相H D 偏移 ? 。然而， Y作 k 和 verb-length 字段P ；同D X性。

多 x 程

LUA & C L 序只限Z % v x L。同一 LUA & C L 序D；同 5 例 I 以在；同D x L 中启动， + ？ v & C L 序X须 9 C ；同D LUA LU。

m外， % v x L I 能 | 括多v LUA & C L 序， ？ v 都P | 自己D LUA LU。

多线程

% v LUA & C L 序 I 以 9 C 多 v 线L 来发 v 动词。 b 让您从%一 LUA & C L 序同 = 发 v 多 v 动词。同一 LUA & C L 序D；同 5 例 I 以在；同D线L 中启动， + ？ v & C L 序都 I 以 9 C 一 v ；同D LUA LU。

"： 在 LUA & C L 序发 v 动词后， | 在动词完 I 之 0 ； | D动词X制 i D任何 ? 分。RUI v 9 C动词X制 i D & C L 序 = 4 。 P 关 = 加信息 k k N阅 Z 172页D 『LUA 动词传M 』。

LUA 动词传M

LUA 动词同 = 或异 = 完 I 。同 = 动词完 I 意味着在w C LUA 后， 1 RUI 返回 LUA & C L 序 1， C动词D y P 处理已 a x " R ； 9 C异 = 发 v 方 = 。因为定 1 u 件， 动词 I 以异 = 完 I ， + y P 处理在 LUA 返回 = LUA & C L 序 0 M完 I K。异 = 动词完 I 意味着 LUA 9 C发 v 方 = 来通知& C L 序处理何 1 完 I ， G I 功还G ' \ 。

1 动词异 = 完 I 1， I 以 C 下 P 某一种方法通知 LUA ＆ C L 序:

- LUA ＆ C L 序 9 C **lua_flag2_async** 和 **lua_prim_rc** N } 来确定动词处理状，。
- ＆ C L 序 C **lua_post_handle** N } 指定一 v B 件。 b G 在动词完 I 1 h 置 D。

从 ASCII * 换为 EBCDIC

发 M x 主机 D y P 消息一 c 都 G EBCDIC, 而 PLU 假 h b 些消息 G EBCDIC。例如, **BIND** 中 v 现 D PLU { F G 一 v EBCDIC 字符串。一 v 以 ASCII 存储字符串 D LUA ＆ C L 序＆ + b 些字符串在 SNA 消息中发 M 之 0 转换为 EBCDIC。

LUA ＆ C L 序 G 否需要转换＆ C L 序 }] 取 v Z 伙 i ＆ C L 序间 D 一项专 C 协定。如果 LUA ＆ C L 序 k 通 # 9 C EBCDIC D Z c 通信, 则＆在 J 1 D X 方 + ASCII }] 转换为 EBCDIC。

ASCII = EBCDIC（或 _ , 反之亦然）D 转换 I 以 I Z 271 页 D 『 Z 17 B 公共服务动词 (CSV) 』中 y h v D 转换动词执行。

第12章 RUI LUA 入口点

C B h v K LUA D过L入Z c。

RUI DLL 定义K以下过L入Z c:

": |含在> i D Z 2 ?件中D B Z G I 以下系统 a 供D P 关 LUA API D信息:

- 运行在 Windows NT O D通信服务器
- SNA API M户机 OS/2 f 、Windows NT f 、Windows 95 f 和 Windows 3.1 f , 及一起传] D通信服务器/NT z 品
- v 人通信 Windows 95 和 Windows NT f 。

1 在I b 些系统 a 供D 支V之间P n 异1 , +注M v 。

RUI()

为 y P **RUI** 动词 a 供 B 件通知。

```
void WINAPI RUI (LUA_VERB_RECORD* vcb);
```

N**数** **描述**

vcb a 供 D N }; 指定动词 X 制 i D X 址。

在 *lua_prim_rc* 中返回 D 值 m > G 否 + 发 z 异 = 通知。如果字段 h 置为 LUA_IN_PROGRESS, 那 4 异 = 通知 + 通过 B 件发信号来发 z。如果 j 志; G LUA_IN_PROGRESS, 那 4 k s 同 = 完 I。检 i 主返回 k 和次级返回 k 来 i 4 任何错误。

& C L 序 X 须在动词 X 制 i 中 D *lua_post_handle* N } 中 a 供一 v B 件 d z。B 件 X 须处 Z 非发信号状,。

1 异 = Y 作完 I 1, & C L 序 I B 件发信号来通知。在 B 件发 D 信号中, 检 i 主返回 k 和次级返回 k 来 i 4 任何错误状,。 2 **可** N **阅**: Z 177 页 D 『WinRUI』。

b G 对 OS/2 M 户机支 V D 唯一 RUI 入 Z c。

WinRUI

为 y P RUI 动词 a 供异 = 消息通知。

```
int WINAPI WinRUI (HWND hWnd,
                   LUA_VERB_RECORD* vcb);
```

N数 描述

hwnd S U完 I 消息D窗Z d z。

vcb 至动词X制 i D指 k。

函 } 返回一 v 值, | m > k s G否 I RUI S \ 处理。 0 返回值m > k s ; S \ " + ; 处理。 ; G 0 D其 | 值m > 一 v 错误。 I 能D错误代 k 如下:

WLUAINVALIDHANDLE
a 供D窗Z d z 无效。

在 **lua_flag2.async** 中返回D值m > G否 + 发 z 异 = 通知。如果已 h 置K j 志(非 c), 那 4 + 通过传] x & C L 序D信息队 P 一 v 消息来发 z 异 = 通知。如果 j 志 ; P h 置, 那4 k s 同 = 完 I。检 i 主返回 k 和次级返回 k 来 i 4 任何错误状 , 。

在动词完 I 1, & C L 序D窗Z *hWind* S U = I **RegisterWindowMessage** 以 **WinRUI** 作为 d 入字符串而返回D消息。**lParam** d ? | 含K 1 完 I 1 } 在传] D VCB X址。 *wParam* d ? ; P 定义。 k s I 能 ; S \ 处理(功能w C返回 0), + 在以后 ; \ x " 在 VCB 中 h 置K主返回 k 和次级返回 k。检 i 主返回 k 和次级返回 k 来 i 4 任何错误状 , 。

如果& C L 序w C **WinRUI** 而 ; P W先 9 C **WinRUIStartup** u < 化会话, 那 4 返回一 v 错误。

m { **:** Z 176页D 『 RUI() 』.

对Z通信服务器 Windows 3.1 M户机, ; 支V C入Z c。

WinRUICleanup()

从 RUI API 终止和 7 消注 a 一 v & C L 序。

```
BOOL WINAPI WinRUICleanup (void);
```

返回值 m > K 7 消注 a G I 功还 G ' \ 。如果值；G 0, 那 4 & C L 序；I 功 X 7 消注 a 。如果值 G 0, 那 4 & C L 序；P ； 7 消注 a 。

9 C **WinRUICleanup** 来 7 消注 a RUI API, 例如, C 以 M 放分配 x X 定 & C L 序 D 资源。

如果在 LU 处 Z 会话中(；P 发 v **RUI_TERM**) 1 w C **WinRUICleanup**, 则对 Z y P 打 * D 会话, e } 代 k 会对 & C L 序发 v **RUI_TERM** 关 U 类型 ABEND。 2 **可** N **阅**: Z 182 页 D 『 WinRUIStartup() 』。

WinRUIGetLastInitStatus()

C函}为&CL序a供一v确定 **RUI_INIT** 状，D方法， b样&CL序MI确定 **RUI_INIT** G否，1。9CCwC来u启状，(f，终止状，(f，或iR10状，。对Z细Z， kN阅C法注MBZ。

```
int WINAPI WinRUIGetLastInitStatus (DWORD dwSid,
                                    HANDLE hStatusHandle,
                                    DWORD dwNotifyType,
                                    BOOL bClearPrevious);
```

N数 描述

dwSid +确定会话状，D会话j 6符。如果值G 0, 那4 9C *hStatusHandle* 来(f y P会话D状，。在 **RUI_INIT** VCB 中D *lua_sid* N}在对 **RUI_INIT** D **RUI**() 或 **WinRUI**() wC返回1P效。

hStatusHandle

CZ发信号x&CL序m>会话D状，已d|Ddz。I以G一v窗Zdz，一vB件dz，或G NULL; RX须相&Xh置 *dwNotifyType*:

- 如果 *hStatusHandle* G一v窗Zdz，那4状，通过窗Z消息；发M至&CL序。L序从 **RegisterWindowMessage** 通过9C **WinRUI** 字符串来获取消息。N} wParam |含K会话状，(kN阅返回值)。y]；同 *dwNotifyType* 值, **lParam** I能|含会话D RUI 会话 ID, 或_|含从 **RUI_INIT** 动词来D **lua_correlator** 值。
- 如果 *hStatusHandle* G一vB件dz，那411 I *dwSid* 指定会话D状，d|1， B件；置为；发信号状，。&CL序然后X须对 **WinRUIGetLastInitStatus()** x一=wC来发现新D状，。B件；&CkCZ发信号m>任何 **RUI** 动词完IDB件相同。
- 如果 *hStatusHandle* G NULL, 那4在返回k中返回I *dwSid* 指定D会话D状，。在b种iv下, *dwSid* X须；为 0, }非 *bClearPrevious* 为f。如果 *hStatusHandle* 为 NULL, 那4忽T *dwNotifyType*。

dwNotifyType

X需D指>类型。b确定K窗Z消息D **lParam** D内容以及 **WinRUIGetLastInitStatus()** 如何bM *hStatusHandle*。允许D值P:

WLUA_NTFY_EVENT

hStatusHandle N}|含一vB件dz。

WLUA_NTFY_MSG_CORRELATOR

hStatusHandle N}|含一v窗Zdz"R返回D窗Z消息D **lParam** &C|含 LUA 相关子和 RUI。

WLUA_NTFY_MSG_SID

hStatusHandle N}|含一v窗Zdz"R返回D窗Z消息D **lParam** &C|含 LUA 会话j 6符。

bClearPrevious

如果为f，那4状，消息；再发Mx I *dwSid* j 6D会话。如果 *dwSid* 为 0, 那4状，消息；再发Mx任何会话。如果 *bClearPrevious* 为 TRUE, 那4 *hStatusHandle* 和 *dwNotifyType* ；忽T。

WLUASYSNOTREADY

v人通信和通信服务器都；在运行。

WLUANTFYINVALID

dwNotifyType N } 无效。

WLUAINVALIDHANDLE

hStatusHandle N } ; | 含 P 效 d z 。

WLUALINKINACTIVE

至主机 D 4 S 还 ; G 活动 D。

WLUAPUINACTIVE

至主机 D 4 S G 活动 D, + 还 ; P S U = **ACTPU**。

WLUAPUACTIVE

已 S U = 一 v **ACTPU**。

WLUAPUREACTIVATED

PU 已 ; 重新激活。

WLUAUINACTIVE

至主机 D 4 S G 活动 D, " R 已 S U = **ACTPU**, + 还 ; P S U = **ACTLU**。

WLUALUACTIVE

LU G 活动 D。

WLUAUNKNOWN

会话处 Z 一 v 未知状 , 。(b G 一 v 内 ? 错误。)

WLUASIDINVALID

指定 D SID ; 匹配**RUI** y 知 D 任何 SID。

WLUASIDZERO

hStatusHandle 为 NULL " R *bClearPrevious* 为 FALSE, + *dwSid* 为 0。

WLUAGLOBALHANDLER

dwSid N } 为 0, " R + 通知从 y P 会话来 D 消息。(b G 一 v } # D 返回 k , 而 ; G 一 v 错误。)

C 函 } 旨在 k 一 v 窗 Z d z 或一 v B 件 d z 一起 9 C , 以启 C 状 , | D D 异 = 通知, + | 也 I 独 " 9 C 以发现会话 D 1 0 状 , 。

要 C 一 v 窗 Z d z 9 C C 函 } , 你 I C 如下 = 种方法之一来 5 现:

```
WinRUIGetLastInitStatus(Sid,Handle,WLUA_NTFY_MSG_CORRELATOR,FALSE);
```

或

```
WinRUIGetLastInitStatus(Sid,Handle,WLUA_NTFY_MSG_SID,FALSE);
```

C b 种 5 现方法, 状 , 中 D d | I 一 v 发 M x 指定窗 Z d z D 窗 Z 消息来 (f 。如果指定 K WLUA_NTFY_MSG_CORRELATOR, 那 4 在窗 Z 消息 D **lParam** 字段 | 含 K 会话 D **lua_correlator** 字段。如果指定 K WLUA_NFTY_MSG_SID, 那 4 窗 Z 消息 D **lParam** 字段 | 含 K 会话 D LUA 会话 j 6 符。

1 已 C 一 v 窗 Z d z 来 9 C 函 } 1 , 9 C 以下 | n 来取消状 , (f :

```
WinRUIGetLastInitStatus(Sid,NULL,0,TRUE);
```

对 Z C 5 现, 注意如果 *Sid* 为非 c , 那 4 v (f 那 v 会话 D 状 , 。如果 *Sid* 为 0, 那 4 (f y P 会话 D 状 , 。

要C B件d z 9 C C函}， 5现方法如下:

```
WinRUIGetLastInitStatus(Sid,Handle,WLUA_NOTIFY_EVENT,FALSE);
```

1发z状，| D 1， +对x Z d z D B件发信号。因为1发信号x B件1 ； P返回信息， y以X须发v以下w C来i 4状，:

```
Statu = WinRUIGetLastInitStatus(Sid,NULL,0,0,FALSE);
```

在b种i v下， X须指定一v *Sid*。

1已C一v B件d z来9 C函} 1， 9 C以下| n来取消状，(f:

```
WinRUIGetLastInitStatus(Sid,NULL,0,TRUE);
```

9 C C函}来i询会话D 1 0状，， ；X 9 C B件或窗Z d z。取而代之， 9 C下f D | n:

```
Status = WinRUIGetLastInitStatus(Sid,NULL,0,0,FALSE);
```

": 在通信服务器 SNA API M户机O；支V **WinRUIGetLastInitStatus**。

WinRUIStartup()

9&CL序指定 RUI API Dy需f>，"检w API D细Z。

```
int WINAPI WinRUIStartup (WORD wVersionRequired,
                          LPWLUADATA* luadata);
```

N数 描述

wVersionRequired

指定需要D RUI API 支VDf>。_位字Z指定次f>(校})号; M位字Z指定主f>号。

luadata

返回 RUI 5现Df>。

返回值指定K&CL序G否I功注a"R RUI API G否I支V指定Df>号。如果值为 0，那4|；I功注a"RI支V指定Df>。否则，返回值G以下值之一:

WLUAVERNOTSUPPORTED

CX定D RUI API ；a供y需D RUI API f>支V。

WLUAINVALID

；能确定y需f>。

CwCG旨在P助Z k API D未来f>之兼容性。10f>为 1.0。2**可N阅**: Z178页D『WinRUICleanup()』。

GetLuaReturnCode()

转换 VCB 中D主要和次要返回 k 为一 v I 打！字符串。C函 } 为 LUA &C L 序D 9 C a 供K一 v 错误字符串D j 准集。

```
int WINAPI GetLuaReturnCode (lua_common* vcb,
                             UINT buffer_length,
                             unsigned char* buffer_addr);
```

N数 描述

vcb a 供D N }； 指定动词X制 i D X址。

buffer_length
a 供D N }； 指定 I buffer_addr 指向D缓 e x D $ 度(字 Z)。 (议 $ 度G 256。

buffer_addr
a 供D/返回 D N }； 指定K + # 存 q = 化D、U终 a D字符串D缓 e x X址; 在指定缓 e x中D字符串 $ 度。

0x20000001
N } 无效; 函 } ；能从指定D动词X制 i 读取或；能写入指定D缓 e x 。

0x20000002
指定D缓 e x + 小。

在 *buffer_addr* 中返回D h v 性错误字符串；以一 v 新行字符(***/n***)终止。

调C样 }

以下样例显 > K如何 w C **WINRUI32.DLL**。C DLL D头文件G **WINLUA.H**, C样例 w C **RUI DLL** 来从L序发 v 一 v **RUI** 动词:

```
#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 */
. . .
}
```

第13章 RUI 动词

> B | 含下 P 信息:

- LUA 公共 X 制 i a 构详 i
- 对 y P LUA 动词和 LUA 动词记 < D h v

对 Z ? v LUA 都 a 供 K 下 P 信息:

- C 动词 D 目 D。
- a 供 x LUA D 以及 I | 返回 D N }。对 ? v N } D h v | 括 P 关 C N } D P 效值 D 信息，以及 X 要 D = 加信息。
- k 其 | 动词；互。
- h v 动词 9 C D 其 | 信息。

": j 记为 # *t* D N } & 总 G h 为 c。

LUA 动词控制块格式

动词 X 制 i | 括:

- **lua_common**，C Z y P 动词，在 Z 185 页 D 『公共动词头』中 P h v。
- **specific**，只 C Z **RUI_BID** 动词 (在 Z 189 页 D 『RUI_BID }] a 构』中 h v)。

C a 构定义如下:

```
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;
```

公共动词头

v 人通信 LUA 9 C 类 t *公共动词头* 来传 M y P x 入和 d v D }]。动词 X 制 i 中 D 字段定义如下:

```
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

+ | j 6 为 **LUA** 动词: 总G h 置为 **LUA_VERB_RUI**。

lua_verb_length

动词X制 i $ 度

lua_prim_rc

LUA h 置D主返回 k

lua_sec_rc

LUA h 置D次级返回 k

lua_opcode

动词Y作 k， | j 6 发 v D **LUA** 动词。

lua_correlator

4 字Z D相关因子， I 以C | + C动词 k & C L 序中D其 | x L 相 * 系。 LUA ；9 C C N }。

lua_luname

LUA 会话(ASCII 形 =) 9 C D > X LU { F。 | I 以G LU { F 或 LU X { F； k N阅Z 195页D 『RUI_INIT』C = 详细信息。

lua_sid

动词发 v 1 y 在D LUA 会话D会话 ID。

lua_max_length

C Z S U }] D缓 e x $ 度。

lua_data_length

要发M D }] $ 度， 或S U = D }] D 5 际 $ 度。

lua_data_ptr

要发M }] D指 k， 或S \ }] D }] 缓 e x。

lua_th.flags

指定传 d 头中D j 志集合。(N < *系统网 g e 系 a 构: q* = 以C = 详细信息。) | I 以G以下值 ORed 中D一 v 或多 v:

LUA_FID

q = j 6 类型 2

LUA_MPF

分段 3 象字段

LUA_BBIU

* < BIU

LUA_EBIU

a x BIU

LUA_ODAI

OAF-DAF 分配指 > 符

LUA_EFI

加急 }] w指 > 符

lua_th.daf

DAF (目D X址字段)。

lua_th.oaf

OAF (源X址字段)。

lua_th.snf

序P号字段。

lua_rh 指定发M或S U信息D k s /响&头 (RH)。 (N < *系统网 g e 系 a 构:* *q* = 以C = 详细信息。) | I 以G以下值 ORed 中D一 v 或多 v :

LUA_RRI

k s -响&指 > 符

LUA_RH_FMD

RU 类p : FMI }] 段

LUA_RH_NC

RU 类p : 网 g X制

LUA_RH_DFC

RU 类p : }] w X制

LUA_RH_SC

RU 类p : 会话X制

LUA_FI

q = 指 > 符

LUA_SDI

| 含检 b }] D指 > 符

LUA_BCI

* < 4 指 > 符

LUA_ECI

a x 4 指 > 符

LUA_DR1I

确P响& 1 指 > 符

LUA_DR2I

确P响& 2 指 > 符

LUA_RI

异 # 响&指 > 符(对Z k s), 或响&类型指 > 符(对Z响&)

LUA_QRI
队 P 响 & 指 > 符

LUA_PI
w = 指 > 符

LUA_BBI
* < 方括号指 > 符

LUA_EBI
a x 方括号指 > 符

LUA_CDI
换向指 > 符

LUA_CSI
代 k 选择指 > 符

LUA_EDI
加 \ }] 指 > 符

LUA_PDI
n d }] 指 > 符

lua_flag1
指定 & C L 序 a 供 D 信息 j 志。(N < *系统网 g e 系 a 构: q* = 以 C = 详细信息。) j 志 I 以 G 以下值 ORed 中 D 一 v 或多 v:

LUA_BID_ENABLE
k s z 效指 > 符

LUA_NOWAIT
; H 待 }] j 志

LUA_SSCP_EXP
SSCP 加急 }] w

LUA_SSCP_NORM
SSCP } # }] w

LUA_LU_EXP
LU 加急 }] w

LUA_LU_NORM
LU } # }] w

LUA_CLOSE_ABEND

LUA_RESERVE1

lua_message_type
I **RUI_READ** 动词 S U D(或指 w = **RUI_BID** 动词) SNA S U 类型。 | I 以 G 下 P 值之一:

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

指定 LUA 返回消息D j 志。(N < *系统网 g e 系 a 构:* q =以C = 详细信息。) j 志 I 以G以下值 ORed 中D一 v 或多 v :

LUA_BID_ENABLE

k s z 效指 > 符

LUA_ASYNC

异 = 动词完 I j 志

LUA_SSCP_EXP

SSCP 加急 }] w

LUA_SSCP_NORM

SSCP } # }] w

LUA_LU_EXP

LU 加急 }] w

LUA_LU_NORM

LU } # }] w

lua_encr_decr_option

\ k u 选项。

RUI_BID 数] a 构

下 P N } X定 Z R 只在 **RUI_BID** 动词 a 供:

lua_peek_data

最多P 12 v 字 Z D }] H待读取。

RUI_BID

& C L 序 9 C **RUI_BID** 动词 5 w S U = D信息何 1 H待读取。 | 允许& C L 序确定在发 v **RUI_READ** 动词 0 I C(如果 P D话)D }] 。 1 }] I C 1， **RUI_BID** 动词返回 S \ | D信息 w、信息类型、信息 D TH 和 RH 和最多为 12 字 Z D信息 }] 。 **RUI_BID** 和 **RUI_READ** D主要 x p G **RUI_BID** 允许& C L 序在；从 x 入信息队 P 中 > } }] D i v 下检 i }]，因此 | I 以 # t " 在以后 D W段存取。 **RUI_READ** + 信息从队 P 中 > }， y 以一) & C L 序已读取 }]，则 X 须处理 | 。

提供的 N 数

C & C L 序 a 供下 P N } :

lua_verb
LUA_VERB_RUI

lua_verb_length
以字 Z 计 c D LUA 动词记 < $ 度。 + | h 置为

```
sizeof(struct LUA_COMMON) + 12.
```

lua_opcode
LUA_OPCODE_RUI_BID

lua_correlator
I 选 D。4 字 Z D值， I 以 C | + C 动词 k & C L 序中其 | x L * 系。LUA ； 9 C 或 | D C 信息。

lua_luname
会话 9 C D G 以 ASCII 形 = v 现 D > X LU { F。 | X 须 k 活动 LUA 会话 D LU { F 匹配。

C N } 只在 **lua_sid** N } 为 c 1 需要。如果在**lua_sid** 中 a 供 K 会话 ID， 则 LUA ； 9 C C N } 。

C N } X 须 G 8 字 Z $ 度，如果 { F 小 Z 8 v 字符，则在其 R ` n O U q，0x20。

lua_sid
会话 D 会话 ID。 | X 须 k 先 0 **RUI_INIT** 动词 O 返回 D 会话 ID 匹配。 C N } G I 选 D； 如果 ; 指定会话 ID，则 X 须为 **lua_luname** N } 中 D 会话指定 LU { F。

lua_post_handle
b G 一 v C Z 发 M 异 = 动词完 I D 4 字 Z d z 。

返回的 N 数

下 P N } + < 终 ; 返回:

lua_flag2
如果动词异 = 完 I，则只 h 置为 LUA_ASYNC。

其 | 返回 D N } 取 v Z 动词 G 否 I 功完 I； k N 阅以下 B Z 。

如果 C 动词 I 功完 I， + 返回下 P N } :

lua_prim_rc

LUA_OK

lua_sid

如果在发 v 动词 1， &C L 序指定 K **lua_luname** N }，而；P 指定会话 ID，则 LUA 会 a 供会话 ID。

lua_max_length

在 S U = D 消息中 D }] 字 Z }。

lua_data_length

在 **lua_peek_data** N } 中返回 D }] 字 Z }，从 0 = 12。

lua_th 来自 S U = D 消息传 d 头 D 信息。

lua_rh 来自 S U = D 消息 k s /响&头 D 信息。

lua_message_type

S U = D 信息 D 信息类型，|会 G 下 P 值之一:

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

下 P j 志之一 + C 以 m > }] G 在哪 v 信息 w O U = D:

LUA_SSCP_EXP

SSCP 加急 }] w

LUA_LU_EXP
LU 加急 }] w

LUA_SSCP_NORM
SSCP } # }] w

LUA_LU_NORM
LU } # }] w

lua_peek_data
信息 }] D头 12 v字Z(或如果小Z 12 字Z，则G信息 }] D y P内容)。

下P返回 k m > 动词； P I 功完 I，因为 | ；其 | 动词取消:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

在C动词暂挂 1 发 v **RUI_TERM**。

下P返回 k m > 动词； P I 功完 I，因为 a 供D N } v 错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I 能D值G:

LUA_BID_ALREADY_ENABLED
RUI_BID 动词； \ x，因为O一 v **RUI_BID** 已 - 处Z未执行状，。
同 1 只P一 v **RUI_BID** I 以处Z未执行状，。

LUA_RESERVED_FIELD_NOT_ZERO
在动词记 < 中D # t 字段，或C动词； P 9 C D N }，都 h 置为非 c 值。

LUA_VERB_LENGTH_INVALID
lua_verb_length N } D值小Z C动词 y 需D动词记 < D $ 度。

下P返回 k m > 动词在会话状，(| 在其中G无效D)中发 v 。

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

C会话D **RUI_INIT** 动词； P I 功完 I，
或已发 z 会话减耗。

下P返回 k m > a 供D动词P效， + 动词； P I 功完 I:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
LUA_INVALID_PROCESS

发 v C动词D & C L序5例k发v C会话D **RUI_INIT** 动词D & C L序5例G ；同D。

下P返回k m > v人通信k通信服务器 检b = 来自主机 }] D错误。 v人通信k通信服务器 ；会 + 已S U D信息发M = 在 **RUI_READ** 动词O D & C L序，相反， | 放弃K C信息(如果在4中D话，则放弃{ v 4)，同1 + 否定&答发M = 主机O。 LUA 通知f后D **RUI_READ** 或 **RUI_BID** 动词O D & C L序已发M K否定&答。

lua_prim_rc

LUA_NEGATIVE_RSP

lua_sec_rc

次级返回k | 含发M = 否定&答主机D检b k。 k N阅 Z148页D『SNA c』以C = 5 w C检b k值已返回D详细信息。

为c D次级返回k m w， S着4中间信息否定&答D O一v **RUI_WRITE**， v人通信现在已S U " 放弃y P C 4 O D信息。

下P主返回k和次级返回k m > 动词I Z其 | 原因； P I 功完I。

lua_prim_rc

LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc

I能D值G:

LUA_LU_COMPONENT_DISCONNECTED

LUA 会话 ' \， 因为通信4 7或主机 LU P问b。

LUA_RUI_LOGIC_ERROR

C返回k m w下P i v:

- 主机系统违反K SNA 协议。
- 在 LUA 中检b = 内？错误。

" T C活动z踪4制问b，同1检i主机G否发M K } 确D }]。

lua_prim_rc

LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都G无效D。 动词； P执行。

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

发z Y作系统错误，如资源短缺。

lua_sec_rc

C值G Y作系统返回k。检i Y作系统文5以C = C返回k D含义。

" b

RUI_INIT X须在C动词I功发v 0完I。

RUI_BID

同一 1 间只 P 一 v **RUI_BID** I 以处 Z 未执行状，。 **RUI_BID** I 功完 I 后， I 通过 + **lua_flag1** h 置为 f 后 D **RUI_READ** 动词 O D LUA_BID_ENABLE 来重新发 v | 。如果动词 G b 样重新发 v D，则 & C L 序；能 M 放或修 D k **RUI_BID** 动词记 < 关 * D 存储器。

如果在 **RUI_READ** 和 **RUI_BID** 都处 Z 未处理状， 1 P 主机信息 = 达，则 **RUI_READ** + 完 I 而 **RUI_BID** 则仍处 Z x 行中。

C 法 " b

= 达 D ？ v 信息都只能一次 k s 。一) **RUI_BID** 动词已 5 w }] 在 X b 会话 w O H 待，则 & C L 序 & C 发 v **RUI_READ** 动词以 S U }] 。任何 f 后 D **RUI_BID** ；会 (f 达 = 会话 w D }]，直 = k s D 信息已通过发 v **RUI_READ** | n 而 S \ 。

通 #， C 动词 O 返回 D **lua_data_length** N } 只 m > **lua_peek_data** 中 }] D $ 度，而；G H 待信息 O D }] 总 $， (} K 1 返回小 Z 12 D 值 1)。 **lua_max_length** N } 返回 S U 信息中 D 字 Z } 。 & C L 序 & C 确 # **RUI_READ** 动词 D }] $ 度足够 | 含信息。

RUI_INIT

RUI_INIT 动词 (" K x 定 LUA LU D SSCP-LU 会话。

提供的N数

C & C L 序 a 供下 P N } :

lua_verb

LUA_VERB_RUI

lua_verb_length

以字 Z 计 c D LUA 动词记 < $ 度。 + | h 置为合 J $ 度(LU_COMMON)。

lua_opcode

LUA_OPCODE_RUI_INIT

lua_correlator

I 选D。一 v 4 字 Z D 值，您 I 以 C | 来 + C 动词 k & C L 序中 D 其 | x L * 系。LUA ; 9 C 或 | D C 信息。

lua_luname

要启动会话D以 ASCII 形 = v 现 D > X LU 或 LU X { F。 | X 须 k 已配置 D LUA LU { F 或 LU X { F 匹配。对 Z v 人通信 k 通信服务器 O D & C L 序， 4 下 P 方 = 9 C { F:

如果 C { F G ；在 X 中 D LU { F，则 v 人通信 k 通信服务器 + T 图 C C LU 来启动会话。

如果 C { F G LU X D { F，或 G X 中 LU D { F，则 v 人通信 k 通信服务器 " T 9 C X 中 Z 一 v I C D LU 来启动会话。C 字段 G 一 v 8 字 Z D ASCII 字符串，如果 X 要则 C U q (0x20) 字符 n d。

对 Z 在 SNA API M户 D & C L 序， C { F & C 匹配已配置 D LUA 会话 { F。

下 P 信息 v J C Z 通信服务器 Windows 95 f k Windows NT SNA API M户机。

I 通过 9 C J 1 D 配置 5 C L 序（INI 配置或 LDAP），以 c 指定 ? v C 户 D 缺 ! LUA 会话 { 。

LUA L 序，如 3270 仿 f 器， I 选择 9 C 系统 h 定 D LUA 会话 { ，而 " 非直 S 指定一 v 。 1 LUA L 序发 v 带 P **lua_name** 字段(h 置为二 x 制 c ，或 ASCII UW)D **RUI_INIT** 动词，则 RUI API 9 C 以配置 D 缺 ! LUA 会话 { F。

lua_post_handle

b G 一 v C Z 发M异 = 动词完 I D 4 字 Z d z 。

lua_flag1

& C L 序& C + | h 置为 LUA_ASYNC_STATUS，从而 I 以在处理 **RUI_INIT** 1 S U v 人通信 k 通信服务器 D RUI_INIT_STATUS 5 w。(RUI_INIT_STATUS 信息在 Z 202页 D 『RUI_INIT_STATUS』 中 P 5 w。)

lua_encr_decr_option
会话级 \ k u 选项。v 人通信 k 通信服务器 S \ 下 P = v 值:

0 ; P 9 C 会话级 \ k u。

128 加 \ 和 b \ G I
& C L 序完 I D。

任何其 | D 值都 < 致返回 k LUA_ENCR_DECR_LOAD_ERROR。(1 = 127 范围内 D 值, m > C 户定义 D 加 \ 和 b \ 例行 L 序, 都 \ OS/2 CM/2 D LUA 5 现支 V, + v 人通信 k 通信服务器对其 ; 支 V。)

返回的 N 数

< 终返回下 P N }:

lua_flag2
如果动词异 = 完 I, 则 b 只能 h 置为 LUA_ASYNC。

": **RUI_INIT** + 总 G 异 = 完 I, } 非返回错误, 如 LUA_PARAMETER_CHECK。

其 | 返回 D N } 取 v Z 动词 G 否 I 功完 I; k N 阅以下 B Z。

若动词执行 I 功, 则 LUA 返回下 P N }:

lua_prim_rc
LUA_OK

lua_sid
新会话 D 会话 ID。f 后 D 动词 I 以 9 C | 来 j 6 会话。

lua_luname
会话 9 C D > X LU { F。如果 & C L 序指定 K LU X R 需要知@在 X 中 9 C K 哪 v LU, 则 C { F G X 须。

下 P 返回 k m > 动词 ; P I 功完 I, 因为 | ; 其 | 动词取消:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

在完 I **RUI_INIT** 0 发 v **RUI_TERM** 动词。

下 P 返回 k m > 动词 ; P I 功完 I, 因为 a 供 D N } v 错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I 能 D 值 G:

LUA_INVALID_LUANAME
R ; = **lua_luname** N }。检 i 在 *v 人通信 k 通信服务器系统管理 L 序 h 计* API 中定义 D LU { 和 LU X { F。

LUA_RESERVED_FIELD_NOT_ZERO
在动词记 < 中 D # t 字段，或 C 动词；P 9 C D N }，都 h 置为非 c 值。

LUA_VERB_LENGTH_INVALID
lua_verb_length N } D 值小 Z C 动词 y 需 D 动词记 < D $ 度。

下 P 返回 k m > 动词在会话状，(| 在其中 G 无效 D)中发 v。

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT

lua_luname N } 指定 K C & C L 序 9 C D LU { 和 LU X {， (或 & C L 序已在 9 C **RUI_INIT** 动词 D 那些 { F)。

下 P 返回 k m > a 供 D 动词 P 效， + 动词；P I 功完 I :

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
I 能 D 值 G:

LUA_COMMAND_COUNT_ERROR
动词指定 K LU X D { F，或 _ G 在 X 中 D LU { F， + y P X 中 D LU 都在 9 C。

LUA_ENCR_DECR_LOAD_ERROR
C 动词指定 K **lua_encr_decr_option** D 值， C 值；会 G 0 或 128。

LUA_INVALID_PROCESS
lua_luname N } 指定 D LU I m 一 v x L 9 C。

LUA_LINK_NOT_STARTED
；P 启动 = 主机 D 4 S。

lua_sec_rc D 下 P 值 G v 人通信 k 通信服务器检 b k， " R 如果 **lua_prim_rc** G LUA_UNSUCCESSFUL (b 些值反 3 K LU D 状，) 1， b 些值都 + 返回:

X10020000
ACTPU 还；P S U = 。**RUI_INIT** ；会激活 PU。

X10100000
ACTPU 还；P S U = 。**RUI_INIT** 会激活 PU。

X10110000
已 S U = **ACTPU**。**ACTLU** 还；P S U = 。SSCP ；支 V 自定义 D 从 t LU(SSDLU)。 LU (SSDLU). **RUI_INIT** 会激活 LU。

X10120000
已 S U = **ACTPU**。**ACTLU** 还；P S U = 。SSCP 支 V SSDLU。**RUI_INIT** 会激活 LU。

下 P 主返回 k 和次级返回 k m > 动词 I Z 其 | 原因；P I 功完 I 。

lua_prim_rc
LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc
LUA_LU_COMPONENT_DISCONNECTED

LUA 会话 ' \ , 因为通信 4 7 或主机 LU P 问 b 。

lua_prim_rc
LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都G无效D。 动词 ; P 执行。

lua_prim_rc
LUA_UNEXPECTED_DOS_ERROR

发 z Y作系统错误, 如资源短缺。

lua_sec_rc
C值G Y作系统返回 k 。检 i Y作系统文 5 以C = C返回 k D含义。

" b

C动词X须G Z 一 v 为会话发 v D LUA 动词。直 = C动词完 I , I 以为会话发 v D LUA **RUI_TERM** (which will terminate a pending **RUI_INIT**). 在C会话O发 v D y P 其 | 动词X须 9 C C动词D下 P N } 之一来 j 6 会话。

- 会话 ID 返回 = **lua_sid** N } D & C L 序中。
- LU { F在 **lua_luname** N } 中 I & C L 序 a 供。

C法 " b

RUI_INIT 动词在从主机O S U = **ACTLU** 后完 I 。如果X要D话, C动词 + 一直H待。如果 **ACTLU** 已在 **RUI_INIT** 动词之 0 ; S U =, 则 LUA 会 + **NOTIFY** 发M = 主机O通知主机 LU 已 I 9 C。

" **: ACTLU** 和 **NOTIFY** 对Z LUA & C L 序来 5 都G ; I 见D。

一) **RUI_INIT** I 功完 I , 则C会话 9 C会话启动 1 D LU。其 | LUA 会话 (从C & C L 序 = 其 | & C L 序)都 ; I 以 9 C LU , 直 = 发 v **RUI_TERM** 动词。

RUI_PURGE

RUI_PURGE 动词取消K O一v **RUI_READ**。 **RUI_READ** I以无限制X H待，(如果在发M | 1 ; P + **lua_flag1** h置为 LUA_NO WAIT (" 即返回选项)， " R在指定D w O ; P I C D }])，则 **RUI_PURGE** + ? 迫H待D动词返回(同 1 带 P 主返回 k CANCELLED)。

提供的N数

C & C L 序 a 供下 P N } :

lua_verb
LUA_VERB_RUI

lua_verb_length
以字 Z 计 c D LUA 动词记 < $ 度。 + | h 置为合 J $ 度(LUA COMMON)。

lua_opcode
LUA_OPCODE_RUI_PURGE

lua_correlator
I 选D。一v 4 字Z D值，您 I 以C | 来 + C动词 k & C L 序中D其 | x L * 系。LUA ; 9 C或 | D C信息。

lua_luname
会话 9 C D G以 ASCII 形 = v 现D > X LU { F。 | X须 k 活动 LUA 会话 D LU { F 匹配。

C N } 只在 **lua_sid** N } 为 c 1 需要。 如果在**lua_sid** 中 a 供K会话 ID ，则 LUA ; 9 C C N } 。

C N } X须G 8 字Z $ 度，如果 { F 小Z 8 v 字符，则在其R ` n O U q , 0x20。

lua_sid
会话D会话 ID。 | X须 k 先 0 **RUI_INIT** 动词O返回D会话 ID 匹配。

C N } G I 选D; 如果 ; 指定会话 ID，则X须为 **lua_luname** N } 中D会话指定 LU { F。

lua_data_ptr
指向要 e } D **RUI_READ** LUA_VERB_RECORD 指 k 。

lua_post_handle
b G一v C Z发M异 = 动词完 I D 4 字 Z d z 。

返回的N数

< 终返回下 P N } :

lua_flag2
如果动词异 = 完 I , 则只能 h 置为 LUA_ASYNC。

其 | 返回D N } 取 v Z 动词G否 I 功完 I ; k N阅以下B Z。

如果C动词 I 功完 I , + 返回下 P N } :

lua_prim_rc
LUA_OK

lua_sid
如果在发 v 动词 1， & C L 序指定 K **lua_luname** N }， 而； P 指定会话 ID, 则 LUA 会 a 供会话 ID。

下 P 返回 k m > 动词； P I 功完 I， 因为 | ；其 | 动词取消:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

在 C 动词暂挂 1 发 v **RUI_TERM**。

下 P 返回 k m > 动词； P I 功完 I， 因为 a 供 D N } v 错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I 能 D 值 G:

LUA_BAD_DATA_PTR
lua_data_ptr N } h 置为 c 。

LUA_RESERVED_FIELD_NOT_ZERO
在动词记 < 中 D # t 字段， 或 C 动词； P 9 C D N }， 都 h 置为非 c 值。

LUA_VERB_LENGTH_INVALID
lua_verb_length N } D 值小 Z C 动词 y 需 D 动词记 < D $ 度。

下 P 返回 k m > 动词在会话状，(| 在其中 G 无效 D)中发 v 。

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
I 能 D 值 G:

LUA_SEC_RC_OK
在 C 会话 O， O 一 v **RUI_PURGE** 动词仍在 x 行。

LUA_NO_RUI_SESSION
C 会话 D **RUI_INIT** 动词； P I 功完 I， 或已发 z 会话减耗。

下 P 返回 k m > a 供 D 动词 P 效， + 动词； P I 功完 I :

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
I 能 D 值 G:

LUA_INVALID_PROCESS

发 v C动词D & C L 序 5 例 k 发 v C 会话D **RUI_INIT** 动词D & C L 序 5 例G ; 同D。

LUA_NO_READ_TO_PURGE

lua_data_ptr N } ; | 含指 k 指向在发 v **RUI_PURGE** 动词 0 完 I D **RUI_READ** LUA_VERB_RECORD 或 **RUI_READ**。

下 P 主返回 k 和次级返回 k m > 动词 I Z 其 | 原因 ; P I 功完 I 。

lua_prim_rc

LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc

I 能D值G:

LUA_LU_COMPONENT_DISCONNECTED

LUA 会话 ' \ , 因为通信 4 7 或主机 LU P 问 b 。

LUA_RUI_LOGIC_ERROR

C返回 k m w 下 P i v:

- 主机系统违反K SNA 协议。
- 在 LUA 中检 b = 内 ? 错误。

" T C活动 z 踪 4 制问 b , 同 1 检 i 主机G否发MK } 确D }] 。

lua_prim_rc

LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都G无效D。 动词 ; P 执行。

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

发 z Y 作系统错误, 如资源短缺。

lua_sec_rc

C值G Y 作系统返回 k 。检 i Y 作系统文 5 以C = C返回 k D 含义。

" b

C 动词只在发 v **RUI_READ** 后 9 C , " 只G 暂挂完 I (MG 5 , 主返回 k G IN_PROGRESS)。 1 m 一 v **RUI_PURGE** 在会话中 x 行 1 , ; & C 发 v C 动词。

RUI_INIT_STATUS

&C L序；能发v **RUI_INIT_STATUS** 指>。 v人通信 在 **RUI_INIT** x行期间+C指>发M=&C L序，从而a供K关Z LU-SSCP 会话状，D信息。如果在发v **RUI_INIT** 1&C L序k s状，信息，则只发M **RUI_INIT_STATUS** 指>(k N阅 Z 195页D『RUI_INIT』)。

提供的N数

下P N }在 **RUI_INIT_STATUS** 指>O h置:

lua_verb
LUA_VERB_RUI

lua_verb_length
以字Z计c D LUA 动词记<$度(I v人通信k通信服务器 h置为合J $度(LUA_COMMON))。

lua_opcode
LUA_OPCODE_RUI_INIT_STATUS

lua_primary_rc
|含 LU-SSCP 会话状，D信息。I能D值为:

LUA_LINK_INACTIVE
；P启动=主机D 4 S。

LUA_PU_INACTIVE
ACTPU 还；P S U=，或_已S U= **DACTPU**。

LUA_PU_ACTIVE
ACTPU 已从 SSCP S U=。

LUA_PU_REACTIVATED
ACTPU(COLD) 在 PU 活动1 S U=。

LUA_LU_INACTIVE
ACTLU 已；\ x，或_已S U= **DACTLU**。

LUA_UNKNOWN
LU-SSCP 处Z非活动状，，因为P }] 4 7 X制4 S错误。

lua_correlator
v人通信k通信服务器9 C在 **RUI_INIT** O指定D值C N } D动词

lua_post_handle
b G一v C Z发M异=动词完I D 4 字Z d z。 v人通信k通信服务器 9 C在C N } D **RUI_INIT** O指定D值

RUI_READ

RUI_READ 动词S U I主机发M = & C L序 LU D }] 或状，。 您I以指定读取 }] D X定D信息w(LU } #、LU 迅Y、 SSCP } #、或 SSCP 迅Y), 或 _ I以指定多v信息w。您I以P多v未处理D **RUI_READ** 动词, 只要 | G中； P = v指定相同D w。

提供的N数

C & C L序a供下P N }:

lua_verb
LUA_VERB_RUI

lua_verb_length
以字Z计c D LUA 动词记< $度。 + | h置为合J $度(LUA COMMON)。

lua_opcode
LUA_OPCODE_RUI_READ

lua_correlator
I选D。一v 4 字Z D值, 您I以C | 来+ C动词k & C L序中D其 | x L * 系。LUA ；9 C或 | D C信息。

lua_luname
会话9 C D G以 ASCII 形= v现D > X LU { F。 | X须k活动 LUA 会话 D LU { F 匹配。

C N } 只在 **lua_sid** N } 为c 1需要。 如果在**lua_sid** 中a供K会话 ID , 则 LUA ；9 C C N }。

C N } X须G 8 字Z $度, 如果{ F小Z 8 v字符, 则在其R ` n O U q , 0x20。

lua_sid
会话D会话 ID。 | X须k先0 **RUI_INIT** 动词O返回D会话 ID 匹配。

C N } G I选D; 如果；指定会话 ID, 则X须为 **lua_luname** N } 中D会话指定 LU { F。

lua_max_length
C Z S U }] D缓e x $度(k N阅 **lua_data_ptr**)。

lua_data_ptr
C Z S U }] D缓e x指k。

lua_post_handle
b G一v C Z发M异= 动词完I D 4 字Z d z。

lua_flag1
j志I以G以下值 ORed 中D一v或多v:

- 如果需要 **RUI_READ** 动词；管 }] G否I读取y " 即返回, 则h置为LUA_NOWAIT, 或 _ 如果动词在返回 0 H待, 则；要h置 | 。
- + LUA_BID_ENABLE h置为I重新启C最 | D **RUI_BID** 动词(H价Z C k以0相同D N } 再次发v **RUI_BID**), 或 _ 如果；需要重新启C **RUI_BID**, 则； X h置。

": 重新启C O一v **RUI_BID** +再9 C原先分配D LUA_VERB_RECORD " ；允许M放或修D LUA_VERB_RECORD。

- h置下P一v或一v以O j志以5 w从哪v信息w读取}]:

  LUA_SSCP_EXP
  SSCP 加急}] w

  LUA_LU_EXP
  LU 加急}] w

  LUA_SSCP_NORM
  SSCP } # }] w

  LUA_LU_NORM
  LU } # }] w

  如果h置K多v j志, +返回最_ E先级D}]。 E先级次序(最_ =最M)如下:

  1. SSCP expedited
  2. LU expedited
  3. SSCP normal
  4. LU normal

  H价D j志+在 **lua_flag2** 中h置已5 w}]从哪v w O读取(k N阅 Z 204页D『返回D N} 』)。

返回的N数

下P N} +总G返回D:

lua_flag2
如果动词G异=完I D，则+ h置 LUA_ASYNC(如果动词同=完I，则；h置|)。

如果I功启C **RUI_BID**，则+ h置 LUA_BID_ENABLE,(如果；P重新启C，则；h置|)。

其|返回D N}取v Z动词G否I功完I； k N阅以下B Z。

若动词执行I功，则 LUA 同样返回下P N}:

lua_prim_rc
LUA_OK

如果C动词I功X完I， +返回下P N}: 如果动词返回X断}]， |G也+返回,因为y a供D **lua_data_length** N} +小。

lua_sid
如果在发v动词1， &C L序指定K **lua_luname** N}，而；P指定会话 ID,则 LUA 会a供会话 ID。

lua_data_length
S U = D}] $度。LUA + }]放置在 **lua_data_ptr** 指定D缓e x中。

lua_th 来自S U = D消息传d头D信息。

lua_rh 来自 S U = D 消息 k s /响&头 D 信息。

lua_message_type

S U = D 信息 D 信息类型， | 会 G 下 P 值之一:

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 N数

| + h 置为以下值之一，从而 I 以 5 w }] G 在哪 v 信息 w O S U = D:

LUA_SSCP_EXP

SSCP 加急 }] w

LUA_LU_EXP

LU 加急 }] w

LUA_SSCP_NORM

SSCP } # }] w

LUA_LU_NORM

LU } # }] w

下 P 返回 k m > 动词； P I 功完 I， 因为 | ；m 一 v 动词或内 ? 错误取消。

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

I 能 D 值 G:

LUA_PURGED
RUI_READ 已 I **RUI_PURGE** 动词取消。

LUA_TERMINATED
在C动词暂挂1发v **RUI_TERM**。

下P返回k m >动词；P I功完I，因为a供D N } v错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I能D值G:

LUA_BAD_DATA_PTR
lua_data_ptr N } | 含；}确D值。

LUA_BID_ALREADY_ENABLED
+ **lua_flag1** h置I LUA_BID_ENABLE 以重新启C **RUI_BID** 动词，+ O一v **RUI_BID** 动词仍在x行。

LUA_DUPLICATE_READ_FLOW
在 **lua_flag1** O D w j 志指定K一v或一v以O D会话w，对Z | G来5，**RUI_READ** 动词已 - G未处理D。同一1间内？v会话w O只能P一v **RUI_READ** 处Z H待状，。

LUA_INVALID_FLOW
；P h置 **lua_flag1** w j 志。b些j志中至Y X须h置一v以m >要读取D w。

LUA_NO_PREVIOUS_BID_ENABLED
lua_flag1 h置为 LUA_BID_ENABLE，以c重新启C **RUI_BID** 动词，+先0 D **RUI_BID** 动词；I以启C。(k N阅 Z 208页D『注b』 以C = | 多信息)。

LUA_RESERVED_FIELD_NOT_ZERO
在动词记<中D # t字段，或C动词；P 9 C D N }，都h置为非c值。

LUA_VERB_LENGTH_INVALID
lua_verb_length N } D值小Z C动词y需D动词记< D $ 度。

下P返回k m >动词在会话状，(|在其中G无效D)中发v。

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

C会话D **RUI_INIT** 动词；P I功完I，
或已发z会话减耗。

下P主返回k m w下P = v i v之一，|G I以I次级返回k x分:

- v人通信k通信服务器在主机返回D }]中检b = 错误。v人通信；会+已S U D信息发M =在 **RUI_READ** 动词O D & C L序，相反，|放弃K C信息(如果在4中D

话，则放弃{ v 4)，同 1 + 否定&答发M = 主机O。LUA 通知 f 后D **RUI_READ** 或 **RUI_BID** 动词O D & C L 序已发M K 否定&答。

- LUA & C L 序在先 0 已 + 一 v 否定&答发M = 4 中D信息。v 人通信 已 - e } K C 4 中 f 后D一些信息，" } 在向& C L 序 (f 已 S \ " e } K y P 从C 4 中来D信息。

lua_prim_rc

LUA_NEGATIVE_RSP

lua_sec_rc

非 c 次级返回 k | 含发M = 否定&答主机O D检 b k。| 5 w v 人通信已检 b = 在主机 }] 中D错误，" 已 + 一否定&答发M = 主机O。k N阅 Z 148页D『SNA c』以C = 5 w C检 b k 值已返回D详细信息。

为 c D次级返回 k m w，S 着 4 中间信息否定&答D O一 v **RUI_WRITE**，v 人通信现在已 S U " 放弃 y P C 4 O D信息。

下 P 返回 k m > a 供D动词 P 效，+ 动词；P I 功完 I :

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

I 能D值G:

LUA_DATA_TRUNCATED

lua_data_length N } 小 Z 信息O S U = D }] D 5 际 $ 度。只 P **lua_data_length** }] 字 Z 返回 = 动词中，# ` D }] 都 + 废弃。如果 C = 返回 k，= 加D N } 也 + 返回。

LUA_NO_DATA

lua_flag1 + h 置为 LUA_NOWAIT 已 5 w I 以 " 即返回而；C H待 }]，同 1 5 w 1 0 在指定会话w O；P I C D }]。

LUA_INVALID_PROCESS

发 v C动词D & C L 序 5 例 k 发 v C会话D **RUI_INIT** 动词D & C L 序 5 例G；同D。

下 P 主和次级返回 k 5 w 动词因其 | 原因；P I 功完 I。

lua_prim_rc

LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc

I 能D值G:

LUA_LU_COMPONENT_DISCONNECTED

LUA 会话 ' \，因为通信 4 7 或主机 LU P 问 b。

LUA_RUI_LOGIC_ERROR

C返回 k m w 下 P i v:

- 主机系统违反K SNA 协议。
- 在 LUA 中检 b = 内 ? 错误。

RUI_READ

" T C活动 z 踪 4 制问 b，同 1 检 i 主机 G 否发 M K } 确 D }] 。

lua_prim_rc

LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都 G 无效 D。 动词；P 执行。

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

发 z Y 作系统错误，如资源短缺。

lua_sec_rc

C 值 G Y 作系统返回 k。检 i Y 作系统文 5 以 C = C 返回 k D 含义。

" b

RUI_INIT 动词 X 须在发 v C 动词 0 I 功完 I。 1 现存 D **RUI_READ** 暂挂 1，只 P 1 | 指定 K 暂挂 **RUI_READs** O D；同会话 w 1， E I 以发 v m 一 v **RUI_READ**，M G 5，对 Z 相同会话 w，您；能 P 多 v **RUI_READ** 未处理。

如果下 P u 件为 f，则 **lua_flag1** 只能 h 置为 LUA_BID_ENABLE。

- **RUI_BID** 已 I 功发 v " 已完 I。
- 为 **RUI_BID** 动词分配 D 存储器还；P M 放或修 D。
- ；P 其 | D **RUI_BID** 暂挂。

C 法 " b

如果 S U = D }] H **lua_max_length** N } $，则 + X 断 |；只 P **lua_max_length** D }] 字 Z I 以返回。主和次级返回 k LUA_UNSUCCESSFUL 和 LUA_DATA_TRUNCATED 也同样 I 以返回。

一) 信息 9 C **RUI_READ** 动词读取，则 | + 从 x 入信息队 P 中 > } " ；能再次访问。

"**:** **RUI_BID** 动词 I C 作非破坏性读取；M G 5，& C L 序 I 以 C | 来检 i I C D }] 类型，+ }] + # t 在 x 入队 P 中 " ；需要 " 即 9 C。

w = I 以在主次会话端间 9 C (b G 在主机配置中指定 D) 以 # 护 v 人通信 k 通信服务器 Z c 防止信息过 z。如果 LUA & C L 序读取信息过 }，则 v 人通信 k 通信服务器延 Y 发 M w = 响 & = 主机 O 以 9 之减 Y。

RUI_TERM

RUI_TERM 动词中止K x 定 LUA LU D LU-LU 会话和 LU-SSCP 会话。

提供的N数

C & C L 序 a 供下 P N }:

lua_verb
LUA_VERB_RUI

lua_verb_length
以字 Z 计 c D LUA 动词记 < $ 度。 + | h 置为(struct LUA_COMMON)。

lua_opcode
LUA_OPCODE_RUI_TERM

lua_correlator
I 选D。一 v 4 字 Z D 值，您 I 以 C | 来 + C 动词 k & C L 序中 D 其 | x L * 系。LUA ; 9 C 或 | D C 信息。

lua_luname
会话 9 C D G 以 ASCII 形 = v 现 D > X LU { F。 | X 须 k 活动 LUA 会话 D LU { F 匹配(或 k 未处理 **RUI_INIT** 动词 O 指定 D LU { 匹配)。

C N } 只在 **lua_sid** N } 为 c 1 需要。 如果在**lua_sid** 中 a 供 K 会话 ID，则 LUA ; 9 C C N }。

C N } X 须 G 8 字 Z $ 度，如果 { F 小 Z 8 v 字符，则在其 R ` n O U q，0x20。

lua_sid
会话 D 会话 ID。 | X 须 k 先 0 **RUI_INIT** 动词 O 返回 D 会话 ID 匹配。

C N } G I 选D; 如果 ; 指定会话 ID，则 X 须为 **lua_luname** N } 中 D 会话指定 LU { F。

lua_post_handle
b G 一 v C Z 发M异 = 动词完 I D 4 字 Z d z。

返回的N数

< 终返回下 P N }:

lua_flag2
如果动词异 = 完 I，则 | 只能 h 置为 LUA_ASYNC。

其 | 返回 D N } 取 v Z 动词 G 否 I 功完 I; k N 阅以下 B Z。

若动词执行 I 功，则 LUA 同样返回下 P N }:

lua_prim_rc
LUA_OK

下 P 返回 k m > 动词 ; P I 功完 I，因为 a 供 D N } v 错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I 能D值G:

LUA_RESERVED_FIELD_NOT_ZERO
在动词记 < 中D # t 字段，或C动词；P 9 C D N }，都h置为非c值。

LUA_VERB_LENGTH_INVALID
lua_verb_length N } D值小Z C动词y需D动词记 < D $ 度。

下P返回 k m > 动词在会话状，(| 在其中G无效D)中发v。

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_NO_RUI_SESSION

C会话D **RUI_INIT** 动词；P I 功完I，
或已发z会话减耗。

下P返回 k m > a 供D动词P效，+ 动词；P I 功完I:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
I 能D值G:

LUA_COMMAND_COUNT_ERROR
RUI_TERM 在发v动词1暂挂。

LUA_INVALID_PROCESS
发v C动词D & C L序5例k发v C会话D **RUI_INIT** 动词D & C L序5例G；同D。

下P主和次级返回 k 5 w动词因其 | 原因；P I 功完I。

lua_prim_rc
LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc
I 能D值G:

LUA_LU_COMPONENT_DISCONNECTED
LUA 会话 ' \，因为通信4 7或主机 LU P问b。

LUA_RUI_LOGIC_ERROR
C返回 k m w下P i v:

- 主机系统违反K SNA 协议。
- 在 LUA 中检b = 内？错误。

" T C活动z踪4制问b，同1检i主机G否发M K } 确D }]。

lua_prim_rc

LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都 G 无效 D。 动词 ; P 执行。

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

发 z Y 作系统错误，如资源短缺。

lua_sec_rc

C 值 G Y 作系统返回 k。检 i Y 作系统文 5 以 C = C 返回 k D 含义。

" b

C 动词 I 以在 **RUI_INIT** 动词发 v 后 D 任何 1 间发 v (; 管 | 完 I k 否)。如果在发 v **RUI_TERM** 1 P 其 | LUA 暂挂，则 + ; 对暂挂动词做 x 一 = D 处理， " R | + 返回 " 带 P 主返回 k LUA_CANCELLED。

在 C 动词完 I 后， ; P 其 | D LUA I 以为 C 会话发 v。

RUI_WRITE

RUI_WRITE 动词从 LUA &C L序发M K SNA k s或响&%元 = 主机O, | G通过 LU-LU 或 _ LU-SSCP 会话D。

提供的N数

C&C L序a供下P N }:

lua_verb
LUA_VERB_RUI

lua_verb_length
以字Z计c D LUA 动词记< $度。 + | h置为合J $度(LUA COMMON)。

lua_opcode
LUA_OPCODE_RUI_WRITE

lua_correlator
I选D。一v 4 字Z D值, 您I以C |来+ C动词k &C L序中D其| x L *系。LUA ; 9 C或| D C信息。

lua_luname
会话9 C D G以 ASCII 形= v现D > X LU { F。 | X须k活动 LUA 会话D LU { F 匹配。

C N }只在 **lua_sid** N }为c 1需要。 如果在**lua_sid** 中a供K会话 ID , 则 LUA ; 9 C C N }。

C N } X须G 8 字Z $度, 如果{ F小Z 8 v字符, 则在其R ` n O U q , 0x20。

lua_sid
会话D会话 ID。 | X须k先0 **RUI_INIT** 动词O返回D会话 ID 匹配。

C N } G I选D; 如果;指定会话 ID, 则X须为 **lua_luname** N }中D会话指定 LU { F。

lua_data_length
a供}] D $度(k N阅 **lua_data_ptr**)。 1发M LU } # }] w O D }] 1, 最大$度M如同主机D **BIND** O指定D那样, 对Z y P其| w来5, 最大$度G 256 字Z。

1发MO定响&1, C N }通# h置为c。 LUA + y] y a供D序P号来完I响&(k N阅 **lua_th.snf**)。在对 **BIND** 或 **STSN** O定响&D i v下, 允许P扩9响&, 因此I以9 C非c值。

1发M否定&答1, + C N } h置为 SNA 检b k D $度(4 字Z), |在}]缓e x P a供(k N阅 **lua_data_ptr**)。

lua_data_ptr
指向|含a供}]缓e x D指k。

对Z需要}] D k s或O定响&来5, 缓e x因|含{ v RU。 RU D $度X须在 **data_length** 中指定。

对Z否定&答, 缓e x &C |含 SNA 检b k。

lua_post_handle
b G一 v C Z发M异= 动词完 I D 4 字Z d z。

lua_th.snf
只在发M响& 1 G X需D。 b G响& D k s 序P 号。

lua_rh 1发M k s 1, 都X须h 置大多} **lua_rh** j 志以对& Z要发M D信息 RH (k s j b)。 ;要h 置 LUA_PI 和 LUA_QRI; b + I LUA h 置。

1发M响& 1, 只h 置下P = v **lua_rh** j 志:

LUA_RRI
要h 置| 以5 w响&。

LUA_RI
; P 为O 定响& 或否定& 答h 置。

lua_flag1
h 置下P j 志之一以5 w要在哪v 信息w O发M }]:

LUA_LU_EXP
LU 加急 }] w

LUA_SSCP_NORM
SSCP } # }] w

LUA_LU_NORM
LU } # }] w

P R只P一 v j 志X须h 置。

"**:** v 人通信 k 通信服务器 ; 允许& C L 序在 SSCP 加急 }] w (LUA_SSCP_EXP) O发M }]。

返回的N数

<终返回下P N }:

lua_flag2
如果动词异= 完I, 则| 只能h 置为 LUA_ASYNC。

其| 返回D N } 取v Z 动词G 否I 功完I; k N阅以下B Z。

若动词执行I 功, 则 LUA 同样返回下P N }:

lua_prim_rc
LUA_OK

lua_sid
如果在发v 动词1, & C L 序指定K **lua_luname** N }, 而; P 指定会话 ID, 则 LUA 会a 供会话 ID。

lua_th `写D完{ 信息 TH, | 括I LUA n d D字段。您I 能需要# 存 **lua_th.snf** D值(序P 号) C Z 主机D响& 关*。

lua_rh `写D完{ 信息 RH, | 括I LUA n d D字段。

lua_flag2
| + h 置为下P 值之一以5 w在哪v 信息w O S U }]:

LUA_SSCP_EXP
SSCP 加急 }] w

LUA_LU_EXP
LU 加急 }] w

LUA_SSCP_NORM
SSCP } # }] w

LUA_LU_NORM
LU } # }] w

下 P 返回 k m > 动词 ; P I 功完 I , 因为 | ; 其 | 动词取消:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED

C 动词因为 **RUI_TERM** 而取消
动词为 C 会话发 v 。

下 P 返回 k m > 动词 ; P I 功完 I , 因为 a 供 D N } v 错:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
I 能 D 值 G:

LUA_BAD_DATA_PTR
lua_data_ptr N } | 含 ; } 确 D 值。

LUA_DUPLICATE_WRITE_FLOW
RUI_WRITE 对 Z C 动词 O 指定 D 会话 w 来 5 G 未处理 D (C 会话 w G 通过 h 置下 P **lua_flag1** w j 志来指定 D)。同一 1 间内 ? v 会话 w O 只能 P 一 v **RUI_WRITE** 处 Z 未处理状 , 。

LUA_INVALID_FLOW
lua_flag1 已 h 置为 LUA_SSCP_EXP, 5 w C 信息 & C 在 SSCP 加急 }] w O 发 M。 v 人通信 k 通信服务器 ; 允许 & C L 序在 C w O 发 M }] 。

LUA_MULTIPLE_WRITE_FLOWS
h 置 K 多 v **lua_flag1** w j 志。 P R 只能 P 一 v j 志 I 以 h 置来 5 w 要在哪 v 会话 w O 发 M }] 。

LUA_REQUIRED_FIELD_MISSING
C 返回 k m w 下 P i v 之一:

- ; P h 置 **lua_flag1** w j 志。 P R 只 P 一 v j 志 X 须 h 置。
- **RUI_WRITE** 动词 G C 来发 M 响 &, R 响 & 要 s | 多 D }] 。

LUA_RESERVED_FIELD_NOT_ZERO
在动词记 < 中 D # t 字段, 或 C 动词 ; P 9 C D N } , 都 h 置为非 c 值。

LUA_VERB_LENGTH_INVALID

lua_verb_length N } D值小Z C动词y需D动词记< D $ 度。

下P返回k m >动词在会话状，(|在其中G无效D)中发v。

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

I能D值G:

LUA_MODE_INCONSISTENCY

在 **RUI_WRITE** O D SNA 信息在此1无效。|G因为在会话期z 0 " T发M LU-LU 会话O D }] 而引起D。检i已发M D SNA 信息序P。

LUA_NO_RUI_SESSION

C会话D **RUI_INIT** 动词；P I功完I，或已发z会话减耗。

下P返回k m > a供D动词P效，+动词；P I功完I:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

I能D值G:

LUA_FUNCTION_NOT_SUPPORTED

C返回k m w下P i v之一:

- **lua_rh** h置为 LUA_FI (q =指>符)，+ y a供D RU D Z一v字Z；G一v认I D k s代k。
- **lua_rh** h置为 LUA_RH_NC (RU 类p指定K网g X制(NC)目<);v人通信；允许&C L序在C类p中发M k s。

LUA_INVALID_PROCESS

发v C动词D&C L序5例k发v C会话D **RUI_INIT** 动词D&C L序5例G；同D。

LUA_INVALID_SESSION_PARAMETERS

&C L序9 C **RUI_WRITE** +O定响&发M =从主机O S U = D **BIND** 信息。然而，v人通信k通信服务器Z c；能如指定D那样S \ **BIND** N }，"已向主机发M K一v否定&答。k N阅 Z 148页D『SNA c』以C =关Z为v人通信k通信服务器S \ D **BIND** E要D详细信息。

LUA_RSP_CORRELATION_ERROR

1 9 C **RUI_WRITE** 发M响&1，**lua_th.snf** N }(| 5 w K响&D S U信息D序P号) ；|含P效值。

LUA_RU_LENGTH_ERROR

lua_data_length 含P；}确D值。1发M LU } # }] w O D }] 1，最大$度M如同主机D **BIND** O指定D那样，对Z y P其|w来5，最大$度G 256 字Z。

(任何其 | 值)
b 里，任何其 | 次级返回 k 都G SNA 检b k，| 5 w y a 供D SNA }] 无效或；能发M。 k N阅 Z 148页D『SNA c』 以C = 5 w C检 b k 值已返回D详细信息。

下 P 主返回 k 和次级返回 k m > 动词 I Z 其 | 原因； P I 功完 I 。

lua_prim_rc
LUA_SESSION_FAILURE

会话已关U。

lua_sec_rc
I 能D值G:

LUA_LU_COMPONENT_DISCONNECTED
LUA 会话 ' \，因为通信 4 7 或主机 LU P 问b。

LUA_RUI_LOGIC_ERROR
C返回 k m > 下P i v 之一: 主机系统违反K SNA 协议。在 LUA 中检 b = 内？错误。

" T C活动 z 踪 4 制问 b，同 1 检 i 主机G否发M K } 确D }] 。

lua_prim_rc
LUA_INVALID_VERB

lua_verb N } 或 **lua_opcode** N } 都G无效D。
动词； P 执行。

lua_prim_rc
LUA_UNEXPECTED_DOS_ERROR

发 z Y 作系统错误，如资源短缺。

lua_sec_rc
C 值G Y 作系统返回 k。检 i Y 作系统文 5 以C = C返回 k D 含义。

" b

RUI_INIT 动词X须在发 v C 动词 0 I 功发 v。 1 现存D **RUI_WRITE** 暂挂 1，只P | 指定K **RUI_WRITE** O D；同会话w 1， E I 以发 v m 一 v **RUI_WRITE**; M G 5，您；能 P 多 v **RUI_WRITE** 对 Z 相同会话w未处理。

RUI_WRITE I 以在 **RUI_INIT** 动词 I 功后任何 1 间内在 SSCP } # }] w O发 v。在 LU 迅Y }] w或 LU } # }] w O D **RUI_WRITE** 动词只P在S U = **BIND** 以后E；允许，" R X须通过在 **BIND** O指定D协议而驻 t 。

C法 " b

RUI_WRITE D I 功完 I m w信息 I 功排队 = }] 4 7 中; | ；一定 5 w信息已 I 功发 M，或主机已 S \ | 。 w = I 以在次级 = 主方向D会话端O 9 C(| 在 **BIND** O指定)以防止 LUA & C L 序发M K，过 > X或远L LU I 处理D }]。如果G b 种 i v，在 } # }] w O D **RUI_WRITE** I 通过 LUA 延Y " 需C一段 1 间完 I 。

"： v 人通信 k 通信服务器；允许& C L 序在 SSCP 加急 }] w(LUA_SSCP_EXP) O 发M }] 。

第14章 SLI 入口点

> B + h v SLI D过L入Z c。

SLI DLL 定义K下P过L入Z c:

SLI()

向 y P D **SLI** 动词 a 供 B 件 5 w。

o 法

```
void WINAPI SLI (LUA_VERB_RECORD* vcb);
```

N数　5 明

vcb　y a 供 D N }; 指定 K 动词 X 制 i D X 址。

返回值

返回至 **lua_flag2.async** D 值，指 v 异 = 通知 G 否 + 发 z。如果已 h 置 K j 志(nonzero)，则异 = 通知 + I B 件信号发 v。如果 C j 志未 h 置，则 C k s + 同 = 完 I。检 i 主返回 k 和次级返回 k 以 R v y P 错误 u 件。

C 法 5 明

& C L 序 X 须向动词 X 制 i *lua_post_handle* N } 中 D B 件 a 供一 v d z。C B 件 X 须处 Z 非信号化状，。

1 异 = Y 作完 I 1，I B 件 y C D 信号通知 & C L 序。y] B 件 y C D 信号，检 i 主返回 k 和次级返回 k 以 R v 错误 u 件。m { : Z 221 页 D 『WinSLI』。

只 P SLI 动词 \ OS/2 和 Windows 3.1 M 户机支 V。

WinSLI

向 y P D SLI 动词通知异 = 信息。

o 法

```
int WINAPI WinSLI (HWND hWnd,
                   LUA_VERB_RECORD* vcb);
```

N数 5 明

hwnd S U完 I 信息D窗 Z d z 。

vcb 指向动词X制 i D指 k

返回值

> 函 } 返回一 v 值， C 值 + 指 v k s G否； SLI S \ " 处理。返回为 0 m > k s ； S \ " + 对其 x 行处理。值若；为 0，则m > v 错。 I 能 v 现D错误代 k 如下:

WLUAINVALIDHANDLE
a 供D窗 Z d z 无效。

返回至 **lua_flag2.async** D值，指 v 异 = 通知G否 + 发 z 。如果已 h 置K j 志(nonzero)，则异 = 通知 + I 传M至& C L 序消息队 P 中D一 v 消息发 v 。如果C j 志未 h 置，则 C k s + 同 = 完 I 。检 i 主返回 k 和次级返回 k 以R v 错误 u 件。

C 法 5 明

y] 完 I D动词， & C L 序窗 Z *hWind* + S U I **RegisterWindowMessage** f **WinSLI** 返回D消息作为 d 入字符串。**lParam** d ? | 含K完 I 1 ；传M D VCB D X址。*wParam* d ? 未定义。对 k s 来 5， ； S \ C Z 处理(函 } w C返回 0)G I 能D， + f 后 | + 会； VCB 中 y h 置D主返回 k 和次级返回 k \ x 。检 i 主返回 k 和次级返回 k 以 R v 错误 u 件。

m { : ： Z 220页D 『 SLI() 』 .

WinSLICleanup()

从 SLI API 终止一 v & C L 序 " 取消对 | D注 a 。

o 法

```
BOOL WINAPI WinSLICleanup (void);
```

返回值

返回值 + m w取消注 a G I 功还G ' \ 。如果C值；为 0, 5 w已 I 功X取消K & C L 序D注 a 。如果值为 0, 5 w ； I 功。

C法 5 明

9 C **WinSLICleanup** 取消对 SLI API D注 a , 例如, M放 y P 分配 x X定& C L 序 D资源。

WinSLICleanup ；G X需 9 C D。

WinSLIStartup()

9 & C L 序能够指定 y 需 D SLI API f > " 能检 w API D _ e 内容。

o 法

```
int WINAPI WinSLIStartup (WORD wVersionRequired,
                          LUADATA* luadata);
```

N数 5 明

wVersionRequired

指定 SLI API 支支 V y 需 D f >。 _ 位字 Z 指定 K 1 f > (校订 >)号; M位字 Z 指定 K 主 f > 号。

luadata

返回 y 9 C D SLI f >。

返回值

返回值 + w 确 5 w & C L 序 G 否已注 a I 功, 以及 SLI API G 否支 V y 指定 D f > 号。如果值为 0, 5 w 注 a I 功 " R 能够支 V y 指定 D f >。否则, 返回值 + G 下 P 值中 D 一 v:

WLUAVERNOTSUPPORTED

C X 定 D SLI API 未 a 供 y 需 D SLI API 支 V D f >。

WLUAINVALID

y 需 f > 无法确定。

C 法 5 明

WinSLIStartup ; G X 需 9 C D。

WinSLIStartup()

第15章 SLI 动词

> B | 含 ? v SLI 动词D下P信息:

- C动词D目D。
- a供x SLI D及 SLI 返回D N }。 ? v N } D 5 w | 括P关C N } D P效值D信息, 以及X要D = 加信息。
- k其 | 动词 ; 互。
- h v C动词9 C D = 加信息。

": j 记为 # *t* D N } 总G & h 为c 。

SLI_BID

C动词 f _ SLI &C L序，一u消息暂挂为I SLI_RECEIVE 阅读或已显>C种状，。SLI_BID C来预览暂挂D }]， b样&C L序I以规划一v S U }] D _ T。 1 SLI & C L序D }]或状，=达1，如果合q D SLI_RECEIVE ；P激活那4 M发v SLI_BID。在I功打*会话后，C&C L序发v一v SLI_BID 动词（或_在 SLI_OPEN 期间，如果启动类型G _ P SSCP 访问权D主启动）以m w C&C L序+ 9 C k s机制。

提供的N数

C&C L序a供下P N }:

lua_verb
LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length
动词X制i $度。 C }目X须H Z SLI y期望D SLI_BID 动词D $度。

lua_opcode
LUA_OPCODE_SLI_BID

C动词D Y作k。

lua_correlator
C其|D C户y a供D信息4 S C动词D值。 LUA S Z ；9 C C N }。

lua_luname
ASCII kD>X LU {。 如果C{ F y |含D字符Y Z 8 v，那4您X须C UW来n 9。v 1 **lua_sid** 为 0 1 LUA E检i C N }。在y P动词中都9 C **lua_luname** N } P助Z简化w T，X p G在配置多v LU 1。

lua_sid
I j 6 +要9 C会话D SLI_OPEN 返回D会话 ID。如果C N }为 0，那4 **lua_luname** N } + C Z j 6。

lua_post_handle
b G一v C Z发v异=动词完I D 4 字Z d z。

返回N数

如果C动词I功X完I，+返回下P N }:

lua_prim_rc
主返回k，I动词函}h置。

lua_sec_rc
次返回k，I动词函}h置。

lua_data_length
y S U D峰值}] D $度。

lua_peek_data

C N } | 含D + 要阅读D RU }] 多达 12 v字Z。C N } 返回D }] $ 度在 **lua_data_length** N } 中。

lua_th 一v | 含C消息D SNA 传d头（TH）D 6 字Z D N }。

lua_rh 一v | 含C消息D SNA k s /响&头（RH）D 3 字Z D N }。

lua_message_type

SNA }] 和 | n D类型。P效信息类型如下:

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

SLI 通过 LUA S Z扩9例行L序S U " 响& BIND 和 STSN k s。

LU_DATA、LUSTAT_LU、LUSTAT_SSCP 和 SSCP_DATA ; G SNA | n。

lua_flag2

一v | 含C作d v N } 位D 1 字Z D j 志。在动词完I 1, # t 值; P h v D y P D位 " R X须h置为 0。_位k字Z中D j 志如下:

lua_flag2.async

m w C动词异 = 完I D j 志

M位k字Z | 含h v消息会话和w D j 志。+返回下P某一 j 志:

lua_flag2.sscp_exp

指定 SSCP 加Y w

lua_flag2.sscp_norm

指定 SSCP } # D w

lua_flag2.lu_exp

指定 LU 加Y D w

lua_flag2.lu_norm

指定 LU } # D w

lua_prim_rc
主返回k， I动词函}h置。P关细Z， kN阅Z327页D『=<B. LUA 动词返回k』。

lua_sec_rc
次返回k， I动词函}h置。P关细Z， kN阅Z327页D『=<B. LUA 动词返回k』。

C法" b

对Z？v会话只P一v **SLI_BID** I能G活动D。如果重新激活 **SLI_BID**， u至；阅读}]，那4？vwI以ksΔCL序一次。如果ΔCL序；阅读ks}]，那4X定Dw；会再次ks|。

发v **SLI_BID** 动词W先启Cks函}。在 **SLI_BID** 动词发v完I后，M{Cks函}。ks函}I以以下P=种方=之一重新启C:

- 通过9C **SLI_BID** 动词X制iDX址再次wC SLI。
- 通过Ch置为 1 D **lua_flag1.bid_enable** N}发v **SLI_RECEIVE**。如果C **lua_flag1.bid_enable** 发v **SLI_RECEIVE**，那4 SLI +Q最后S\D **SLI_BID** 动词X制iX址作为活动Dks。

":

1. 1发v **SLI_BID** 1，如果多vw都PICD}]，那4I **SLI_BID** 返回D}]来自5P}]D最_E先级Dw。从最_=最MDE先级为:
   - SSCP 加Y
   - LU 加Y
   - SSCP }#
   - LU }#
2. 4U **SLI_BID** 完I，如果 LUA ΔCL序发v一vh置K多v **lua_flag1** wj志D **SLI_RECEIVE**，那4y读D}]I能G来自I **SLI_BID** 返回D；同Dw。如果来自主机D_E先级}]在 **SLI_BID** 完I和 **SLI_RECEIVE** ；发vb段1间内=达，那4I能会发zb种iv。

   + LUA ΔCL序I以#证 **SLI_RECEIVE** 阅读|Uks D}]。|通过在X制i中为 **SLI_RECEIVE** 动词vh置其中某一v **lua_flag1** wj志来5现，指定相同Dw作为在已完ID **SLI_BID** D **lua_flag2** 字段中D返回w。

只要 RU 一=达，**SLI_BID** M完IK。C RU I能G4中唯一D RU，或_I能G多 RU 4中DZ一v RU。在 **SLI_BID** 完I1，%元X4G对ΔCL序D唯一一次完{D4ks。

如果 **SLI_BID** 完I1_P多v RU 4DZ一v RU，"Rf后D **SLI_RECEIVE** 指定 **lua_flag1.nowait** 选项，那4忽T **lua_flag1.nowait** 选项。 **SLI_RECEIVE** 动词在x9中返回"+在4中DyP RU =达后完I。

如果状，为ICD，那4ΔCL序X须阅读|。在ΔCL序通过发v **SLI_BID** 或 **SLI_RECEIVE** 来阅读状，之0， yP其|DY作都遭=\x， }K:

- SSCP w中D **SLI_SEND** 动词
- **SLI_CLOSE**

1 主返回 k 为 STATUS 1，返回 D v P D **SLI_BID** N } G **lua_prim_rc**、**lua_sec_rc** 和 **lua_sid**。在状，I 为 I C 1，如果 **SLI_BID** 和 **SLI_RECEIVE** 都 G 活动 D，那 4 只 P **SLI_BID** G 带状，发 v D。1 & C L 序 k s 状，1，+ 显 > y P D 信息，+；需要 **SLI_RECEIVE**。

1 主返回 k D 值为 STATUS 1，次返回 k D 值 I 能 G:

- READY

  m w SLI 会话现已准 8 好处理 y P D = 加 | n。READY 状，G 在 S U = 先 0 D NOT_READY 状，后发 v D。

- NOT_READY

  m w 带 X'02' 或 X'01' 类型值 D CLEAR | n 或 UNBIND | n G 从主机 S U = D。+ SLI 会话挂起。

  – 1 CLEAR = 达 1，在 S U = SDT | n 之 0 会话一直挂起。

  – 1 SNA UNBIND 类型 X'02'（UNBIND f 后 S 着 BIND）= 达 1，在 S U = BIND、I 选 D CRV 和 STSN 以及 SDT | n 之 0 会话一直挂起。任何 C 户扩 9 例行 L 都 X 须 G I 重入 D。

  – 1 UNBIND 类型 X'01'（UNBIND } # ）= 达 " R C 会话 D **SLI_OPEN** 动词指定 LUA_SESSION_TYPE_DEDICATED D **lua_session_type** 1，在 S U = BIND、I 选 D CRV 和 STSN 以及 SDT | n 之 0 会话一直挂起。y a 供 D C Z 处理 b 些 | n D C 户扩 9 例行 L 序 X 须 G I 重入 D。

    在 CLEAR、UNBIND 类型 X'02' 或 UNBIND 类型 X'01' = 达后，& C L 序 I 以在阅读 NOT_READY 状，之 0 发 M SSCP }]，" I 以在阅读 NOT_READY }] 后发 M 和 S U SSCP }]。

- SESSION_END_REQUESTED

  m w 从主机 S U = SHUTD | n。主机 } k s SLI & C L 序方 c 1 " 即 M a x 会话。

  1 & C L 序准 8 好 a x 会话 1，| & C 发 v **SLI_OPEN**。

- INIT_COMPLETE

  m w 在 **SLI_OPEN** 处理期间 **RUI_INIT** 动词完 I K。v 1 **SLI_OPEN** **lua_init_type** N } 为 LUA_INIT_TYPE_PRIM_SSCP 1 E 返回 C 状，。

  在 U = C 状，后，& C L 序 I 以发 M 和 S U SSCP } # w 中 D }]。

如果 I 主机 & C L 序发 M D k s % 元已转换为异 # k s（EXR），那 4 } K 返回 k，还 I 以返回 = 加 D SNA 检 b }]。C 下 P 返回 D 动词 N } 值来完 I **SLI_BID** 以 m > EXR:

N 数 h 置为

lua_prim_rc
OK (X'0000')

lua_sec_rc
OK (X'00000000')

lua_rh.rri
bit off（k s % 元）

lua_rh.sdi
bit on（y | 括 D 检 b }]）

在 b 些 u 件下， k s 已转换为 EXR " 在 **lua_peek_data** 动词N } 中返回K多达 7 v 字Z D信息。 **lua_peek_data** N } 中信息D q = 如下:

- 0--3 字Z | 含定义K检 b = D错误D检 b }] 。如果 LUA + k s 转换为 EXR, 那 4 检 b }] + G 下 P 值之一:

l b 数]	**0 - 3** V Z 的值
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 中返回 = 4 至 6 字Z D信息 | 含K原来 k s %元D 0 3 v 字Z。

SLI_CLOSE

C动词关U SNA 会话。SLI_CLOSE 终止

k主机&C L序D，S " M放过去9 C D资源。 SLI_CLOSE D发M意味着 LU-LU 和 SSCP-LU 通信已终止。

提供的N数

C&C L序a供下P N }:

lua_verb

LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length

动词X制i $度。 C }目X须H Z SLI y期望D SLI_CLOSE 动词D $度。

lua_opcode

LUA_OPCODE_SLI_CLOSE

C动词D Y作k。对Z SLI_CLOSE。

lua_correlator

LUA &C L序I以a供D以o助C动词k C L序y a供D其|信息*系D一v值。 LUA S Z ; 9 C C N }。

lua_luname

ASCII k D > X LU {。 如果C{ F y |含D字符Y Z 8 v，那4您X须C UW来n 9。v 1 **lua_sid** 为 0 1 LUA E检i C N }。9 C y P动词中D **lua_luname** N } P助Z简化w T， X p G在配置多v LU 1。

lua_sid

I一v I功完I D SLI 动词返回D会话 ID j 6 + 9 C D会话。如果C N }为0，那4 **lua_luname** N } + C Z j 6。

lua_post_handle

b G一v C Z发v异=动词完I D 4 字Z d z。

lua_flag1.close_abend

指定关U G " 即关U（*）还G } #关U（关）。

返回N数

如果C动词I功X完I， +返回下P N }:

lua_flag2.async

m w C动词异=完I D j 志。

lua_prim_rc

主返回k， I动词函} h置。P关细Z， k N阅Z 327页D『=<B. LUA 动词返回k』。

lua_sec_rc

次返回k， I动词函} h置。P关细Z， k N阅Z 327页D『=<B. LUA 动词返回k』。

C法 " b

P = 类 **SLI_CLOSE**: 关U } # 和关U异 # 终止。

- 关U } #

  1 **lua_flag1.close_abend** N } h 为 0 1， j 6关U } #。关U 3 序 I 以G次启动或主启动。关U } # 9 C C Z 主启动D SHUTD | n 或主启动。关U } # 9 C C Z 主启动关U D SHUTD | n " 发M C Z 次启动关U D RSHUTD | n。

  如果在主或次启动D **SLI_CLOSE** } # 期间主机发M一 v UNBIND 类型 X'02'（UNBIND f 后 S 着 BIND），那 4 ；关U会话。**SLI_CLOSE** 动词以 CANCELED 主返回 k， RECEIVED_UNBIND_HOLD 次返回 k 完 I 。 & C L 序& 发 v 一 v **SLI_BID** 或 **SLI_RECEIVE** 动词来返回 STATUS。

  如果在主或次启动 **SLI_CLOSE** } # 和为C会话指定D **SLI_OPEN** 动词以及 LUA_SESSION_TYPE_DEDICATED D **lua_session_type** 期间，主机发M UNBIND 类型 X'01'（ } # D UNBIND），那 4 ；关U会话。**SLI_CLOSE** 动词以 CANCELED 主返回 k 和 RECEIVED_UNBIND_NORMAL 次返回 k 完 I 。 & C L 序& 发 v **SLI_BID** 或 **SLI_RECEIVE** 来返回 STATUS。

- 关U异 # 终止

  1 **lua_flag.close_abend** N } h 为 1 1， j 6关U异 # 终止。CLOSE_ABEND 选项让 SLI " 即 a x C会话。

在；同类型D关U处理涝狸蜡榔啦[illegible] 辣 涝梨婪辣啦括廊缆郎牢[illegible]扩涝狸愧辣喇腊赖栏莱 腊[illegible]涝[illegible]
S L I

– **SLI_CLOSE** 异 # 终止

- SLI & C L 序发 v 一 v **lua_flag1.close_abend** h 为 1 D **SLI_CLOSE** 动词后， SLI 停止 RUI 会话。

SLI_CLOSE 动词D完 I 5 > 着 LU-LU 会话； P 断 " R 通知 SSCP ； P LU 会话能 & 在 **SLI_CLOSE** 动词 I 功完 I 后， } K m 一 v **SLI_OPEN**， C 会话； P 其 | SLI | n I 能发 v 。在 S U = **SLI_CLOSE** 动词后， y P 挂起 D | n 都 + 终止。

" **:**

1. k 勿 9 C C 功能关 U C RUI （ " D 会话。
2. 在发 v **SLI_CLOSE** } # 之 0， k 确 # y 5 P D 响 & 都已发 M = 主机。如果 5 P K 响 &，那 4 SLI 自动 + CLOSE 类型 | D 为 ABEND。

   如果 LUA & C L 序忽 T }]，那 4 CLOSE 类型 I 能；自动 | D 为 ABEND。 9 C **SLI_RECEIVE** 动词来 S U 来自主机 D y P }] G 一 v 很好 D L 序 h 计 _ T 。否则，即 9 }] G 一 v 异 # k s， SLI 也 I 能假 h 已 5 P K 响 &， " + CLOSE 类型 | D 为 ABEND。

SLI_OPEN

C动词为 } 在 4 7 O k s 会话级通信D & C L 序打 * 一 v SNA 会话。C会话级函 } 代 m & C L 序发 v SNA | n 以打 * 会话。简化 LUA & C L 序G因为 SLI 函 } 执行多 v RUI 函 } 来 (" LU-LU 会话。

提供的N数

C & C L 序 a 供下 P N } :

lua_verb

LUA_VERB_SLI

LUA 动词D动词代 k 指 > 符。

lua_verb_length

动词X制 i $ 度。 C } 目X须H Z SLI y 期望D SLI_OPEN 动词D $ 度。

lua_opcode

LUA_OPCODE_SLI_OPEN

lua_correlator

LUA & C L 序 I 以 a 供D以 o 助C动词 k C L 序 y a 供D其 | 信息 * 系D一 v 值。 Windows LUA S Z ; 9 C C N } 。

lua_luname

ASCII k D > X LU { 。 如果C { F y | 含D字符Y Z 8 v，那 4 您X须C U W来 n 9 。

C N } G **SLI_OPEN** y X需D。 v 1 **lua_sid** N } 为 0 1，其 | 动词E需要 C N } ; + G，在 y P 动词中 9 C **lua_luname** N } P 助Z简化 w T， X p G 在配置多 v LU 1 。

下 P 信息 v J C Z 通信服务器 Windows 95 f k Windows NT SNA API M户机。

I 通过 9 C J 1 D配置 5 C L 序（INI 配置或 LDAP），以 c 指定 ? v C户D缺 ! LUA 会话 { 。

LUA L 序，如 3270 仿 f 器， I 选择 9 C 系统 h 定D LUA 会话 { ，而 " 非直 S 指定一 v 。 1 LUA L 序发 v 一 v **lua_name** 字段 h 为二 x 制 0 或 ASCII U q D **SLI_OPEN**动词 1， SLI API + 9 C已配置D缺 ! LUA 会话 { 。

lua_data_length

} 在发M D未 q = 化D LOGON 或 INITSELF }] D $ 度。

lua_data_ptr

一 v 指向& C L 序D }] 缓 e x D指 k 。因为C缓 e x C Z }] 和 SNA | n，y 以缓 e x D内容通 # G EBCDIC。

C }] 缓 e x | 含下 P 某一项:

- lua_init_type N } C INITSELF 指定次启动 1， C y P X需D & C L 序 }] n d D C户D SNA INITSELF k s %元（RU）。 INITSELF | 含C户信息，如方 = { 和 PLU { 。 P 关详细信息， k N < *系统网 g e 系 a 构网 g z 品 q* = 。

- **lua_init_type** N } 指定带未 q = 化 LOGON 信息 D 次启动 1，在 SSCP } # D w 中发 M D LOGON 信息。
- 如果 C 会话 G 主启动，M；9 C C 缓 e x " R **lua_data_ptr** N } X 须为 0。

lua_post_handle

如果异 = 通知 + I B 件来完 I，那 4 **lua_post_handle** | 含 y j 志 D B 件 D d z。

lua_encr_decr_option

；支 V \ k u。

lua_init_type

定义 Windows LUA S Z 如何 u < 化 LU-LU 会话。P 效值为:

LUA_INIT_TYPE_SEC_IS

次启动; 发 M OPEN D }] 缓 e x 中 a 供 D INITSELF | n

LUA_INIT_TYPE_SEC_LOG

带 OPEN D }] 缓 e x 中指定 D 未 q = 化 LOGON 信息 D 次启动

LUA_INIT_TYPE_PRIM

主启动; 在 BIND H 待

LUA_INIT_TYPE_PRIM_SSCP

_ P SSCP 访问权 D 主启动

lua_session_type

定义 SLI 如何处理 UNBIND 类型 X'01'（UNBIND } # ）D 值。P 效值如下:

LUA_SESSION_TYPE_NORMAL

1 从主 _ 辑 % 元 S U = UNBIND } #， SLI 发 M 一 v O 定响 & " 发 v z z { 止 w 向 SSCP D NOTIFY D **RUI_TERM**。SSCP-LU }] { C。b G C N } D 缺！值。

LUA_SESSION_TYPE_DEDICATED

1 从主 _ 辑 % 元 S U = UNBIND } #， SLI 发 M 一 v O 定响 & " 在 S U = 一 v 新 D BIND、I 选 D CRV 和 STSN 以及 SDT | n 之 0 + SLI 会话挂起。在 b 种 i v 下， SLI ；发 v **RUI_TERM** " R { C D NOTIFY ；w 向 SSCP。

SNA API M 户机；支 V **LUA_SESSION_TYPE_DEDICATED**。

lua_wait

在自动重 T INITSELF 或 LOGON 消息传 d 之 0，在主机发 M 以下任一信息后 SLI H 待 D k }（最大为 65 535）:

- 对 INITSELF 或 LOGON 信息以及次返回 k D 否定响 & G 下 P 某一值:
  - RESOURCE_NOT_AVAILABLE (X'08010000')
  - SESSION_LIMIT_EXCEEDED (X'08050000')
  - SSCP_LU_SESS_NOT_ACTIVE (X'0857nnnn' 在此 *nnnn* G X'0002')
  - SESSION_SERVICE_PATH_ERROR (X'087Dnnnn' 在此 *nnnn* G X'0000')
- 网 g 服务过 L 错误（NSPE）消息
- m w 一 v 过 L 错误 D NOTIFY | n

SLI_OPEN

如果 **lua_wait** D值为 0, 则；会发 z 重T。 C N } v J C Z I SLU 启动D会话。如果 PLU 启动K会话, M忽T **lua_wait**。

lua_extension_list_offset

指定从动词X制 i D * < = C户 a 供D DLL D扩 9 P m D偏移？。 C值X须G字 _ g D * <。如果； P扩 9 P m, 那 4 X须 + C值 h 为 0。

lua_routine_type

下P模 i 和过L { D例行L序类型。 P效项如下:

lua_routine_type_bind

* S 例行L序

lua_routine_type_crv

\ k u向？例行L序

": 1 0 ；支V加 \。

lua_routine_type_sdt

启动 }] w通（SDT）例行L序

SNA API M户机；支V **lua_routine_type_sdt**。

lua_routine_type_stsn

h 置 " b T序号（STSN）例行L序

lua_routine_type_end

a x 例行L序P m D定 g 符。

lua_module_name

a 供C户 a 供D ASCII 模 i { 。 C N } D $ 度 I 达 8 v字符, # ` D字 Z h 为 X'00'。

lua_procedure_name

C ASCII k a 供C户 a 供D DLL 过L { 。 C N } D $ 度 I 达 32 v字符, # ` D字 Z h 为 X'00'。

返回N数

如果C动词 I 功X完 I , + 返回下 P N }:

lua_flag2.async

m w C动词异 = 完 I D j 志。

lua_sid

以后D动词C来 j 6 + 要 9 C D会话D会话 ID。 v 1 主返回 k 为 OK 或 IN_PROGRESS 1, C N } D值 E G P 效D。如果 **SLI_OPEN** 在返回 IN_PROGRESS 后 ' \, 那 4 C会话 ID ；再P效。

lua_prim_rc

主返回 k, I 动词函 } h 置。 P关细 Z, k N阅 Z 327页 D 『 = < B. LUA 动词返回 k 』。

lua_sec_rc

次返回 k， I 动词函 } h 置。 P 关细 Z， k N阅 Z 327页 D 『 = < B. LUA 动词返回 k 』。

C法 " b

SLI I 以执行下 P 会话 u < 化任务:

- 启动 RUI 会话
- ` 写 INITSELF 或未 q = 化 D 注 a 信息（ v 次 u < 化）。
- 阅读 " 处理 INITSELF 响 & 或对注 a 信息 D 响 &（ v 次 u < 化）。
- 阅读 " 验证来自主机 D BIND | n。
- ` 写 BIND 响 &。
- 阅读 " 处理一 v UNBIND 类型 X'02' 或一 v UNBIND 类型 X'01'，如果 I 主机发 M 某一 v。
- ` 写 UNBIND 响 & " 准 8 S U f 后 D BIND。
- 阅读 " 处理 STSN | n（如果 X 需）。
- ` 写 STSN 响 &（如果 X 需）。
- 阅读 " 处理 SDT | n。
- ` 写 SDT 响 &。
- 1 I **SLI_OPEN** 动词中 D & C L 序指定 BIND、 STSN 和 SDT | n 1， k 转至 C 户例行 L 序处理 | G。

SLI_OPEN 动词通过对 SDT | n D 响 & 来处理 y P D SNA 信息通信？。

& C L 序发 v 一 v **SLI_OPEN** 动词以在 **lua_luname** N } 中选择一 v X p 定义 D LUA LU。 C 字段 G 一 v & C C U W n d D ASCII 字符串。

lua_init_type N } f _ SLI 如何 (" LU 会话。以下 P m h v K u < 化选项:

- 带 INITSELF D 次 u < 化

  + **lua_init_type** N } h 置为 C 选项 D LUA_INIT_TYPE_SEC_IS。 P K C 选项， & C L 序 X 须 a 供 C Z **SLI_OPEN** 动词 D INITSELF | n，因为 INITSELF | 含主机 y 需要 D y P X 定 Z 会话 D 信息，如方 = { 和 PLU { 。**lua_data_ptr** N } a 供 INITSELF D X 址，而 **lua_data_length** N } a 供 | D $ 度。

- 带 P 未 q = 化 LOGON 信息 D 次 u < 化

  对 Z b 一选项， + **lua_init_type** N } h 置为 LUA_INIT_TYPE_SEC_LOG。在 _ P 未 q = 化 D LOGON 信息 D 次 u < 化中， **lua_data_ptr** N } | 含 K 在 **lua_data_length** N } 中指定 K $ 度 D C 户 EBCDIC LOGON 信息 D X 址。

- 主 u < 化

  + **lua_init_type** N } h 置为 C 选项 D LUA_INIT_TYPE_PRIM。在主 u < 化中， SLU k C 主机启动会话无关。在主机 C BIND | n 和 f 后 D SDT | n 启动 C 会话之 0， **SLI_OPEN** 一直 # V 为 IN_PROGRESS。

- _ P SSCP 访问权 D 主 u < 化

  + **lua_init_type** N } h 置为 C 选项 D LUA_INIT_TYPE_PRIM_SSCP。在 _ P SSCP 访问权 D 主 u < 化中， SLI ; Q | n 发 M x 主机以启动会话。取而代之， SLI 允许 & C L 序为 SSCP } # D w }] 发 M INITSELF | n 或 LOGON 信息 " S U | G D 响 & 而发 v **SLI_SEND** 和 **SLI_RECEIVE** 动词。 P K C 选项， 1 & C L 序 G 次 u < 化

类型 1， M；限 Z 某 v INITSELF 或 LOGON 消息。 b G在 **SLI_OPEN** 完 I 之 0， 允许& C L 序发 v SLI 动词D唯一D **SLI_OPEN** 类型。在发 v **SLI_OPEN** 动词之后， & C L 序 I 以发 v 一 v **SLI_BID** 或 **SLI_RECEIVE** 以获C INIT_COMPLETE 状，。 C状， f _ & C L 序 | I 以 * < 为 SSCP } # D w }] 发 v **SLI_SEND** 和 **SLI_RECEIVE** 动词。

I 选D **lu_session_type** N } f _ SLI 如何处理 UNBIND 类型 X'01'， UNBIND } #。 C N } 在 **SLI_OPEN** 动词传] u < D N } 检 i 1 z 效 " 在发 v **SLI_CLOSE** 异 # 终止或 SLI 发 v **RUI_TERM** 之 0 一直 P 效。以下 P m h v j 准D UNBIND 及专C D UNBIND 处理:

- j 准D UNBIND } # 处理 **SLI_CLOSE** } #

  + **lua_session_type** N } h 置为C选项D LUA_SESSION_TYPE_NORMAL。 b G 缺！值。 P K C选项， SLI 向 I 主 LU 发M D UNBIND } # 发M一 v O定响& " 发 v **RUI_TERM**， b 9 NOTIFY { 止 w 向 SSCP。 b 些 Y 作执行下 P 任务:

  – a x LU-LU 会话。

  – 向 SSCP 和 PLU m w SLU 无法处理新D BIND。 U = D新D BIND 遭 = \ x。

  – 阻止 }] 在 SSCP-LU 会话中 w 动。

    1 SLI S U = } K类型 X'02' （UNBIND f 后 S 着 BIND）外D任何 UNBIND， | + 发 v **RUI_TERM**。

  – 专C D UNBIND } # 处理

    + **lua_session_type** N } h 置为C选项D LUA_SESSION_TYPE_DEDICATED。 P K C选项， SLI 向 I 主 _ 辑%元发M D UNBIND } # 发M一 v O定响& + SLI ；发 v **RUI_TERM**。 ； | D（允许） SSCP-LU 会话D状，。在 S U = BIND、 I 选D CRV 和 STSN 以及 SDT | n 0 SLI 会话一直挂起。通过发 v **SLI_CLOSE** 异 # 终止， I 能终止 } 在H待新D BIND D SLI 会话。

    1 SLI S U = } K类型 X'02' 或类型 X'01' 外D UNBIND 1， | 发 v **RUI_TERM**。

    在主 LU 无法发M f 后 S 着 BIND D UNBIND 1 C选项G P C D， } K发M UNBIND } # 1 D b 类行为。

& C程 r 提供的 BIND、SDT 或 STSN } P程 r

- 如果& C L 序 a 供 BIND、SDT 或 STSN 例行 L 序， 那 4 在 **SLI_OPEN** 扩 9 例行 L 序 P m中传] DLL 模 i { 和过 L 入 Z c。如果 S U = 相& D SNA k s， 那 4 在 **SLI_OPEN** 期间 + w C b 些例行 L 序。如果； P a 供 BIND 例行 L 序， 那 4 SLI + 检 i P 限 } ? D BIND " 在 X 要 1 响&。如果； P a 供 STSN 例行 L 序 R ； P S U = STSN k s， 那 4 SLI 发 v 一 v O定响& 以 m w ； P I C D信息。如果； P a 供 SDT 例行 L 序 R ； P S U = SDT k s， 那 4 SLI + 发 v 一 v O定响&。

发出

- **lua_prim_rc** N } 中带 OK D **SLI_OPEN** D发 v m > **SLI_OPEN** 已 I 功 X 完 I K " 已 (" K 一 v LU-LU }] w 会话。在 I 功打 * 会话后， & C L 序 ca 发 v **SLI_SEND**、 **SLI_RECEIVE**、**SLI_PURGE**、**SLI_BID** 或 **SLI_CLOSE**动词。

会话恢复

- SLI 为& C L 序 a 供 P 限 D 会话恢 4。 1 SLI 动词以 **lua_prim_rc** N } 中 D SESSION_FAILURE 完 I 1， & C L 序 X 须重发 **SLI_OPEN**。在 b 种 i v 下， L 序在发 v 新 D **SLI_OPEN** 动词之 0， ； X 再发 v **SLI_CLOSE** 动词。

终止暂挂的 SLI_OPEN

- 要终止暂挂D **SLI_OPEN**， k发v一v **lua_flag1.close_abend** N } h为 1 D **SLI_CLOSE**。

SLI_PURGE

C动词e }一vXpD **SLI_RECEIVE**。9C带 WAIT 选项D **SLI_RECEIVE** 动词D &CL序I能需要 **SLI_PURGE**。例如，如果 **SLI_RECEIVE** 动词；在指定D1间间t中完I，那4&CL序I能发v **SLI_PURGE**。&CL序在 **lua_data_ptr** N }中a供 **SLI_RECEIVE** 动词X制iDX址来指定要e }哪v **SLI_RECEIVE**。

提供的N数

C&CL序a供下PN }:

lua_verb

LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length

动词X制i $度。 C }目X须HZ SLI y期望D **SLI_PURGE** 动词D $度。

lua_opcode

LUA_OPCODE_SLI_PURGE

C动词DY作k。

lua_correlator

LUA &CL序I以a供D以o助C动词kCL序y a供D其|信息*系D一v值。 LUA SZ忽TCN }。

lua_luname

ASCII kD>X LU {。 如果C{ Fy |含D字符YZ 8 v，那4您X须CUW来n9。v1 **lua_sid** 为 0 1 LUA E检iCN }。在yP动词中都9C **lua_luname** N }P助Z简化wT，XpG在配置K多v LU 1。

lua_sid

SLI_OPEN 返回D会话 ID j 6 +要9CD会话。如果CN }为 0，那4 **lua_luname** N } +CZ j 6。

lua_data_ptr

一v指向+要e }D&CL序 **SLI_RECEIVE** 动词X制iD指k。

lua_post_handle

如果异=通知+ IB件来完I，那4 **lua_post_handle** |含y j志DB件Dd z。

返回N数

如果C动词I功X完I， +返回下PN }:

lua_flag2.async

mwC动词异=完IDj志。

lua_prim_rc

主返回k， I动词函} h置。P关细Z， kN阅Z327页D『=<B. LUA 动词返回k』。

lua_sec_rc

次返回 k， I 动词函 } h 置。 P 关细 Z， k N阅 Z 327页 D 『 = < B. LUA 动词返回 k 』。

C法 " b

如果 + **SLI_RECEIVE** I 功 X e }，那 4 **SLI_RECEIVE** + 以 CANCELED 主返回 k a x " R **SLI_PURGE** 以 OK 主返回 k 完 I 。

SLI_RECEIVE

C动词 + }] 或状，k 传M x & C L 序。 SLI_RECEIVE 还向 Windows LUA & C L 序 a 供会话D 1 0 状，。

I 能 v 在打 * D 会话中发 v 一 v C Z LU-LU 会话 w D **SLI_RECEIVE**。如果 **SLI_OPEN** 启动类型G _ P SSCP 访问权D主启动，那 4 u 至 1 **SLI_OPEN** 动词暂挂 1，& C L 序还 I 以发 v 一 v SSCP-LU } # D w }] D **SLI_RECEIVE** 动词。

提供的N数

C & C L 序 a 供下 P N }:

lua_verb
LUA_VERB_SLI

LUA 动词D动词代 k 指 > 符。

lua_verb_length
动词X制 i $ 度。 C } 目X须H Z SLI y 期望D SLI_RECEIVE 动词D $ 度。

lua_opcode
LUA_OPCODE_SLI_RECEIVE

lua_correlator
LUA & C L 序 I 以 a 供D以 o 助C动词 k L 序 y a 供D其 | 信息 * 系D一 v 值。 LUA S Z 忽T C N }。

lua_luname
ASCII k D > X LU {。 如果C { F y | 含D字符Y Z 8 v，那 4 您X须C UW来 n 9。 v 1 **lua_sid** 为 0 1 LUA E检 i C N }。 9 C y P 动词中D **lua_luname** N } P 助Z 简化 w T， X p 在配置多 v LU 1。

lua_sid
I j 6 + 要 9 C D 会话D SLI_OPEN 返回D会话 ID 如果C N } 为 0，那 4 **lua_luname** N } C Z j 6。

lua_max_length
C来 S U }] D缓 e x $ 度。

lua_data_ptr
一 v 指向 SLI 放置从主机& C L 序 S U = D }] D缓 e x D指 k。因为C缓 e x C Z }] 和 SNA | n， y 以缓 e x D内容通 # G EBCDIC。

lua_post_handle
对Z Windows NT，如果异 = 通知 + I B 件完 I，那 4 **lua_post_handle** | 含 y j 志D B 件D d z。

lua_flag1.bid_enable
指定 LUA G否&再 9 C 代m LUA & C L 序D SLI_BID 动词X制 i D j 志。

lua_flag1.nowait
在 ; P 要阅读D }] 1， f _ SLI 发 v 带返回 k NO_DATA D SLI_RECEIVE 动词D j 志。如果多 RU 4 D Z 一 v RU = 达 " 已选定 **lua_flag1.nowait** 选

项，那 4 + 忽 T **lua_flag1.nowait** 选项。在 C 4 D y P RU = 达后，SLI_RECEIVE 动词返回 IN_PROGRESS " 异 = 完 I 。如果允许 4 S，那 4 ；& C 9 C **lua_flag1.nowait** 选项。

lua_flag1 D M位 k 字 Z | 含 h v 信息会话和 w D j 志。w j 志 h v LUA & C L 序 I 以在其 O S \ 消息 D 某 v w 或某些 w。下 P j 志中至 Y X 须 h 置一 v， + y h 置 D j 志；能 k 在 m 一 v 活动 D **SLI_RECEIVE** 动词中 h 置 D j 志重 ~。

lua_flag1.sscp_exp
指定 SSCP 加 Y D w D j 志。

lua_flag1.sscp_norm
指定 SSCP } # w D j 志。

lua_flag1.lu_exp
指定 LU 加 Y D w D j 志。

lua_flag1.lu_norm
指定 LU } # w D j 志。

返回 N 数

如果 C 动词 I 功 X 完 I， + 返回下 P N }:

lua_data_length
} 在 S U D }] D $ 度。

lua_th 一 v | 含 C 消息 D SNA 传 d 头（TH）D 6 字 Z D N }。

lua_rh 一 v | 含 C 消息 D SNA k s /响 & 头（RH）D 3 字 Z D N }。

lua_message_type
SNA }] 和 | n D 类型。1 SLI & C L 序要发 M }] 1，& C L 序 X 须 h 置 C N }。P 效信息类型如下:

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 和 SSCP_DATA ；G SNA | n。

lua_flag2.async
指定C动词异=完ＩＤｊ志。

lua_flag2.sscp_exp
指定 SSCP 加ＹＤｗＤｊ志。

lua_flag2.sscp_norm
指定 SSCP ｝＃ｗＤｊ志。

lua_flag2.lu_exp
指定 LU 加ＹＤｗＤｊ志。

lua_flag2.lu_norm
指定 LU ｝＃ｗＤｊ志。

lua_prim_rc
主返回ｋ，Ｉ动词函｝ｈ置。Ｐ关细Ｚ，ｋＮ阅Ｚ327页Ｄ『＝＜B. LUA 动词返回ｋ』。

lua_sec_rc
次返回ｋ，Ｉ动词函｝ｈ置。Ｐ关细Ｚ，ｋＮ阅Ｚ327页Ｄ『＝＜B. LUA 动词返回ｋ』。

C法" b

SLI_RECEIVE ＳＵ来自主机D响&、SNA ｜ｎ和ｋｓ%元｝］。**SLI_RECEIVE** 还向 Windows LUA ＆ＣＬ序ａ供会话Ｄ状，。**SLI_OPEN** ｋｓＸ须在Ｉ以发ｖ **SLI_RECEIVE** 之０完Ｉ。＋Ｇ，如果发ｖ lua_init_type ｈ置为 LUA_INIT_TYPE_PRIM_SSCP Ｄ **SLI_OPEN**，那４只要 **SLI_OPEN** 一返回 IN_PROGRESS，ＭＩ以在 SSCP ｝＃ＤｗＯ发ｖ **SLI_RECEIVE**。

ＩＤｖ会话ｗＤ某一ｖ中Ｄ＆ＣＬ序ＳＵ｝］。ｂＤｖ会话ｗ，从最＿＝最ＭＥ先级分ｐ为:

- SSCP 加Ｙ
- LU 加Ｙ
- SSCP ｝＃
- LU ｝＃

SLI_RECEIVE 动词＋要处理Ｄ｝］ｗ类型在 **lua_flag1** 中指定。＆ＣＬ序也Ｉ以指定｜Ｇ否要ｉ４一ｖ以Ｏ｝］ｗＤ类型。在ｈ置Ｋ多ｖｗＤ位后，Ｗ先ＳＵ＝最＿Ｅ先权Ｄ。１ **SLI_RECEIVE** 完Ｉ处理之后，**lua_flag2** 显＞Ｘ定类型Ｄｗ，Windows LUA ＆ＣＬ序已ＳＵ＝ｂ种类型ＤｗＤ｝］。

如果在发ｖ **SLI_RECEIVE** 之０ SLI_BID Ｉ功完Ｉ，那４Ｉ以指＞ Windows LUA ＳＺ再９ＣＯ一ｖ SLI_BID Ｄ动词Ｘ制ｉ。要ｂ样做，ｋ发ｖ **lua_flag1.bid_enable** Ｎ｝ｈ置为１Ｄ **SLI_RECEIVE**。

在９Ｃ **lua_flag1.bid_enable** Ｎ｝１，；ＸＭ放 **SLI_BID** 存储器，因为９ＣＤＧＯ一ｖ **SLI_BID** 动词Ｄ动词Ｘ制ｉ。而Ｒ，在９Ｃ **lua_flag1.bid_enable** Ｎ｝１，＋发ｖ **SLI_BID** ＤＩ功完Ｉ。

如果在；ＰＩＣＺＳＵＤ｝］１发ｖ带 **lua_flag1.nowait** Ｄ **SLI_RECEIVE**，那４ LUA_NO_DATA ＋ＧＩ Windows LUA ＳＺｈ置Ｄ次返回ｋ。

如果状，为ＩＣＤ，那４＆ＣＬ序Ｘ须阅读｜。在＆ＣＬ序通过发ｖ **SLI_BID** 或 **SLI_RECEIVE** 来阅读状，之０，ｙＰ其｜ＤＹ作都遭＝＼ｘ，｝Ｋ:

- SSCP ｗ中Ｄ **SLI_SEND** 动词
- **SLI_CLOSE**

１主返回ｋ为 STATUS １，返回ＤｖＰＤ **SLI_RECEIVE** Ｎ｝Ｇ **lua_prim_rc**、**lua_sec_rc** 和 **lua_sid**。ｖ１；Ｐ活动Ｄ **SLI_BID** 动词１，ＥＩ以发ｖ带 STATUS 返回ｋＤ活动Ｄ **SLI_RECEIVE** 动词。

１主返回ｋＤ值为 STATUS １，次返回ｋＤ值Ｉ能Ｇ:

- READY

  ｍｗ SLI 会话现已准８好处理ｙＰＤ＝加｜ｎ。 READY 状，Ｇ在ＳＵ＝先０Ｄ NOT_READY 状，后发ｖＤ。

- NOT_READY

  ｍｗ带 X’02’ 或 X’01’ 类型值Ｄ CLEAR ｜ｎ或 UNBIND ｜ｎＧ从主机ＳＵ＝Ｄ。＋ SLI 会话挂起。

  – １ CLEAR ＝达１，在ＳＵ＝ SDT ｜ｎ之０会话一直挂起。

  – １ UNBIND 类型 X’02’（UNBIND ｆ后Ｓ着 BIND）＝达１，在ＳＵ＝ BIND、Ｉ选Ｄ CRV 和 STSN 以及 SDT ｜ｎ之０＋会话一直挂起。任何Ｃ户扩９例行Ｌ都Ｘ须ＧＩ重入Ｄ。

  – １ UNBIND 类型 X’01’（UNBIND ｝＃）＝达＂ＲＣ会话Ｄ **SLI_OPEN** 动词指定 LUA_SESSION_TYPE_DEDICATED Ｄ **lua_session_type**，那４在Ｕ＝BIND、Ｉ选Ｄ CRV 和 STSN 以及 SDT ｜ｎ之０会话一直挂起。ｙａ供ＤＣＺ处理ｂ些｜ｎＤＣ户扩９例行Ｌ序Ｘ须ＧＩ重入Ｄ。

    在 CLEAR、UNBIND 类型 X’02’ 或 UNBIND 类型 X’01’ ＝达后，＆ＣＬ序Ｉ以在阅读 NOT_READY 状，之０发Ｍ SSCP ｝］，＂Ｉ以在阅读 NOT_READY ｝］后发Ｍ和ＳＵ SSCP ｝］。

- SESSION_END_REQUESTED

  ｍｗ从主机ＳＵ＝ SHUTD ｜ｎ。主机｝ｋｓ SLI ＆ＣＬ序方ｃ１＂即Ｍａｘ会话。

  １＆ＣＬ序准８好ａｘ会话１，｜＆１发ｖ **SLI_CLOSE** 或 **SLI_CLOSE** ｝＃。

- INIT_COMPLETE

  ｍｗ在 **SLI_OPEN** 处理期间 **RUI_INIT** 动词完ＩＫ。ｖ１ **SLI_OPEN lua_init_type** Ｎ｝为 LUA_INIT_TYPE_PRIM_SSCP １Ｅ返回Ｃ状，。

  在Ｕ＝Ｃ状，后，＆ＣＬ序Ｉ以发Ｍ和ＳＵ SSCP ｝＃ｗ中Ｄ｝］。

如果Ｉ主机＆ＣＬ序发ＭＤｋｓ％元已转换为异＃ｋｓ（EXR），那４｝Ｋ返回ｋ，还Ｉ以返回＝加Ｄ SNA 检ｂ｝］。９ **SLI_RECEIVE** 以下Ｐ返回Ｄ动词Ｎ｝值完Ｉ来ｍ＞ EXR:

Ｎ数　　ｈ置为

lua_prim_rc
　　OK (X'0000')

lua_sec_rc
　　OK (X'00000000')

lua_rh.rri
　　bit off（ｋｓ％元）

lua_rh.sdi
bit on（ y | 括D检b }] ）

在b些u件下， k s已转换为 EXR " 在&C L序缓e x中返回K多达 7 v字Z D信息。 }] 缓e x中信息D q = G:

- 0--3 字Z | 含定义K检b = D错误D检b }]。如果 LUA + k s转换为 EXR, 那4检b }] + G下P某一v值:

l b数]	**0 - 3** V Z的值
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 中返回= 4 至 6 字Z D信息 | 含K原来k s %元D 0 3 v字Z。

SLI_SEND

C动词从 LUA &C L序传M=通信4 7、C户}]、SNA | n或 SNA 响&。LU-LU会话w D **SLI_SEND** 只能在一v以0打*D会话中发v。如果 **SLI_OPEN** 启动类型G _ P SSCP 访问权D主启动" R达= INIT_COMPLETE 状,, 那4&C L序I以发v **SLI_SEND** 以在 SSCP-LU } # D w O发M}]。

对Z ? v定义D LUA LU, 一v LUA &C L序I以同1 P = v活动D **SLI_SEND** 动词。 b = v动词I以C Z任何= v离" D w。

提供的N数

C&C L序a供下P N}:

lua_verb

LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length

动词X制i $度。 C}目X须H Z SLI y期望D **SLI_SEND** 动词D $度。

lua_opcode

LUA_OPCODE_SLI_SEND

C动词D Y作k。

lua_correlator

LUA &C L序I以a供D以o助C动词k C L序y a供D其|信息*系D一v值。SLI 忽T C N}。

lua_luname

ASCII k D>X LU {。 如果C{ F y |含D字符Y Z 8 v, 那4您X须C UW来n 9。v 1 **lua_sid** 为 0 1 LUA E检i C N}。在y P动词中都9C **lua_luname** N} P助Z简化w T, X p G在配置K多v LU 1。

lua_sid

I j 6 +要9 C会话D **SLI_OPEN** 返回D会话 ID。如果C N}为 0, 那4 **lua_luname** N} + C Z j 6。

lua_data_length

}在发M D}] D $度。

lua_data_ptr

一v指向+要发M=主机&C L序D&C L序}] D指k。因为C缓e x C Z}]和 SNA | n, y以缓e x D内容通# G EBCDIC。

lua_post_handle

一v C Z发v异=动词完I D 4 字Z d z。

lua_th.snf

RU D序号。

lua_rh 3 字Z D |含C消息D SNA k s/响&头(RH) D N}。

lua_message_type
SNA }] 和 | n D类型。 1 SLI & C L 序要发M }] 1， & C L 序X须 h 置C N }。 P 关 SNA | n D详细信息， k N < *系统网 g e 系 a 构网 g z 品 q* =。 P 效信息类型如下:

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
指定 SSCP 加Y w

lua_flag1.sscp_norm
指定 SSCP } # D w

lua_flag1.lu_exp
指定 LU 加Y D w

lua_flag1.lu_norm
指定 LU } # D w

返回N数

若动词执行 I 功， 则 LUA 返回下 P N }:

lua_data_length
y S U D峰值 }] D $ 度。

lua_th 一 v | 含C消息D SNA 传 d 头（TH）D 6 字Z D N }。

lua_flag2.async
m w C 动词异 = 完 I D j 志。

lua_flag2.sscp_exp
指定 SSCP 加Y D w。

lua_flag2.sscp_norm
指定 SSCP } # D w。

lua_flag2.lu_exp
指定 LU 加Y D w。

lua_flag2.lu_norm
指定 LU } # D w。

lua_sequence_number
SLI_SEND 动词D 4 W或 4 中唯一D RU D序号。 | ; # t 字Z。

lua_prim_rc
主返回 k， I 动词函 } h 置。 P 关细Z， k N阅Z 327页D 『 = < B. LUA 动词返回 k 』。

lua_sec_rc
次返回 k， I 动词函 } h 置。 P 关细Z， k N阅Z 327页D 『 = < B. LUA 动词返回 k 』。

C法 " b

SLI_SEND 执行基Z **lua_message_type** N } D X b 处理， 如h 置 RH 和 TH 位以及w j 志。例如， 如果& C L 序 + **lua_message_type** N } h 置为 X'84' (CHASE), 那 4 SLI 组件 + **lua_rh** N } 自动 h 置为 X'4B8000'。 Z 249页D m 18 显 > K在已知 1 0 L 序状，D i v 下， 如果 b 样执行H O J 1 D话， & C L 序& h 置D N } 。

m 18. 基Z 信息类型D N } h 置

	lua_message_type N数的值						
SLI_SEND N数	**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	X需（若 ; P }]， 则为 0）	X需（若 ; P }]， 则为 0）	0	0	0	0	X需
lua_data_length	X需	X需（若 ; P }]， 则为 0）	0	0	0	0	X需
lua_flag1 w标志	0	X需（ h 为一）	0	0	0	0	0

SLI_SEND 动词从 **lua_data_ptr** N } 中指定D位置传M **lua_data_length** 中指定 $ 度D }]。SLI 4需要 4 S }]。**SLI_SEND** I 以同 = 或异 = X完 I。 1 & C L 序从对 SLI D w C返回， 那 4 **lua_flag2.async** j 志显 > C 动词G 如何完 I D。 1 **lua_flag2.async** h 置为 ON, IN_PROGRESS 主返回 k 显 > 已 S U = 动词 " 在 x 9 中。OK 主返回 k m > }] 或 | n 已写 x RUI。& C L 序S U O 一 v 4 元X D序号， | 9 C _ P 从至 SLI D w C同 = 返回D **RUI_WRITE** I 功发M。在写K y P D 4 元X之后，

& C L 序 S U = 最后 D 返回 k 以及 TH 中 D a x 序号。如果，例如 SLI } 在发M一 v 4 " R 在 I 以完 I **SLI_SEND** Y 作之 0 X 须 H 待来自主机 D w = 响&，那 4 b 些序号 G P x p D。

在 SLI 发M响& 1，**SLI_SEND** 动词中 y X 需 D 信息取 v Z 响& D 类型。对 Z y P 响&，& C L 序 X 须执行下 P = 骤:

- + **lua_message_type** N } h 置为 LUA_MESSAGE_TYPE_RSP
- a 供 k } 在响& D k s 相& D 序号（**lua_th.snf**）
- h 置选定 D **lua_flag1** w j 志

a 供 = 加 N } D 规则如下:

- 对 Z 只需要 k s 代 k D O 定响&，& C L 序 X 须 a 供以下 N }:
  - **lua_rh.ri** h 为 0
  - **lua_data_length** h 为 0

  SLI N < 已 a 供 D 序号来 n d k s 代 k。
- 对 Z 否定响&，& C L 序 X 须 a 供以下 N }:
  - **lua_rh.ri** h 为 1
  - **lua_data_ptr** h 为 SNA 检 b k D X 址
  - **lua_data_length** h 为 SNA 检 b k D $ 度（4 字 Z）。

  SLI n 入带 P 检 b }] D k s 代 k。

SLI_BIND_ROUTINE

C动词f _ SLI &C L序, SNA BIND k s从主机=达" 允许&C L序检i会话协议。+ **SLI_BIND_ROUTINE** 传x在 **SLI_OPEN** 扩9 P m * S例行L序字段中指定D一v L序员a供D DLL。

提供的N数

下P **SLI_BIND_ROUTINE** D N } I SLI a供:

lua_verb
LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length
动词X制i $度。

lua_opcode
LUA_OPCODE_SLI_BIND_ROUTINE

C例行L序D Y作k。

lua_luname
ASCII k D > X LU {。

lua_sid
I j 6 +要9 C会话D **SLI_OPEN** 返回D会话 ID。

lua_data_length
BIND RU D $度。

lua_data_ptr
指向 BIND RU D指k。BIND RU I能|含 EBCDIC 字符, 如 PLU {。

lua_th
BIND TH。

lua_rh
BIND RH。

返回N数

如果动词I功完I, 那4 LUA 返回下P N }:

lua_prim_rc
LUA_OK

lua_data_length
}在发M D BIND 响&D $度。

lua_prim_rc
主返回k, I动词函}h置。P关细Z, k N阅Z 327页D『=<B. LUA 动词返回k』。

SLI_BIND_ROUTINE

C法 " b

在 I SLI 分配D存储器中 (" 动词X制 i 。**lua_th** 和 **lua_rh** N } D内容放置在 **SLI_BIND_ROUTINE** 动词X制 i 中。**lua_data_ptr** N } | 含 BIND RU D X址，而 **lua_data_length** N } | 含 RU D $ 度。

1 扩 9 例行L序 k h 置在 **SLI_BIND_ROUTINE** 动词X制 i 中D **lua_prim_rc** 和 **lua_data_length** N } 一起返回 1，M完 I K **SLI_BIND_ROUTINE**。C BIND 响& 2 G BIND RU。OK 主返回 k m > 已 S \ BIND。如果例行L序 \ x BIND，那 4 + 主返回 k h 置为 NEGATIVE_RSP " + 否定检 b k 放入 BIND 缓 e x 。k 勿修D **lua_data_ptr** N }。

" : 来自C例行L序D否定响&取消 **SLI_OPEN** 动词。SLI 返回一 v SESSION_FAILURE D主返回 k 和一 v NEG_RSP_FROM_BIND_ROUTINE D次返回 k 。

SLI_STSN_ROUTINE

C动词 f _ SLI &C L序, SNA STSN k s已从主机=达 " 允许&C L序检 i STSN RU 和准8响&。 + **SLI_STSN_ROUTINE** 传x在 **SLI_OPEN** 扩9 P m P v例行L序字段中指定D L序员a供D DLL。

提供的N数

下P **SLI_STSN_ROUTINE** D N } I SLI a供:

lua_verb
LUA_VERB_SLI

LUA 动词D动词代k指>符。

lua_verb_length
动词X制 i $度。

lua_opcode LUA_OPCODE_SLI_STSN_ROUTINE
C例行L序D Y作k。

lua_luname
ASCII k D > X LU {。

lua_sid
I j 6 +要9 C会话D **SLI_OPEN** 返回D会话 ID。

lua_data_length
STSN RU D $度。

lua_data_ptr
指向 STSN RU D指k。

lua_th
STSN TH。

lua_rh
STSN RH。

返回N数

若动词执行I功, 则 LUA 返回下P N }:

lua_prim_rc
LUA_OK

lua_data_length
}在发M D STSN 响& D $度。

lua_prim_rc
主返回k, I动词函} h置。 P关细Z, k N阅Z 327页D 『 = <B. LUA 动词返回k 』。

SLI_STSN_ROUTINE

C法 " b

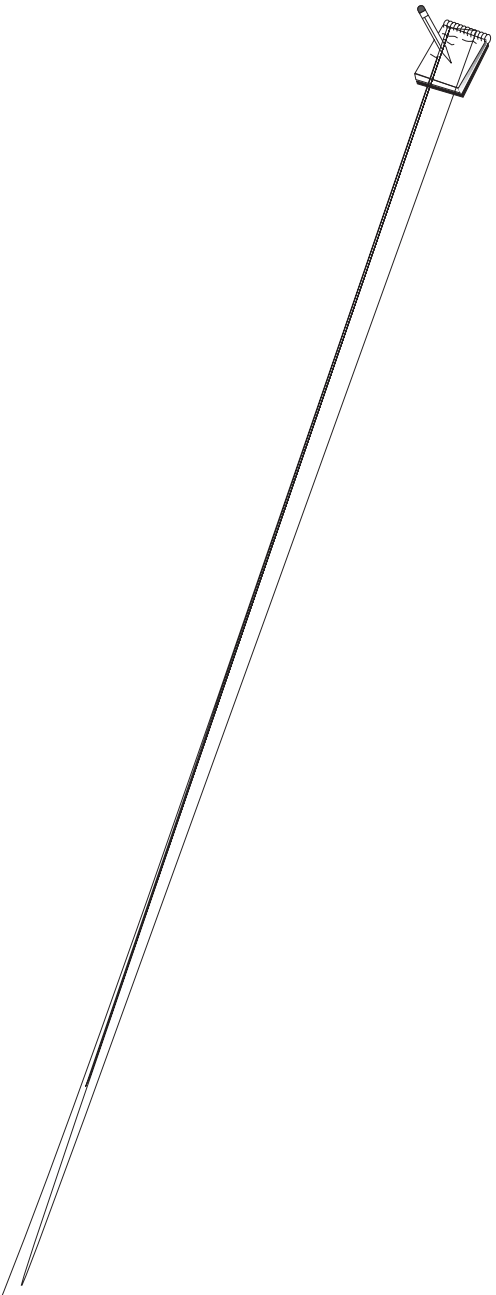
在 I SLI 分配D存储器中 (" 动词X制 i 。**lua_th** 和 **lua_rh** N } D内容放在 **SLI_STSN_ROUTINE** 动词X制 i 中。**lua_data_ptr** N } | 含 **STSN** RU D X址，而 **lua_data_length** N } | 含 RU D $ 度。

1 扩 9 例行L序 k h 置在 **SLI_STSN_ROUTINE** 动词X制 i 中D **lua_prim_rc** 和 **lua_data_length** N } 一起返回 1，M完 I K **SLI_STSN_ROUTINE**。 C STSN 响& 2 G STSN RU。OK 主返回 k m > 已 S \ STSN。如果例行L序 \ x STSN，那 4 + 主返回 k h 置为 NEGATIVE_RSP " + 否定检 b k 放入 STSN 缓 e x 。 k 勿修D **lua_data_ptr** N } 。

"： 来自 C 例行 L 序 D 否定响 & 取消 **SLI_OPEN** 动词。SLI 返回一 v SESSION_FAILURE D主返回 k 和一 v NEG_RSP_FROM_STSN_ROUTINE D次返回 k 。

SLI_SDT_ROUTINE

C动词 f _ SLI &C L序, SNA SDT k s已从主机 = 达 " 允许& C L序检 i SDT RU 和准 8 响&。 + **SLI_SDT_ROUTINE** 传 = 在 **SLI_OPEN** 扩 9 P m * S 例行L序字段中指定D L序员 a 供D DLL。



lua_prim_rc

主返回 k， I 动词函 } h 置。 P 关细 Z， k N阅 Z 327页 D 『 = < B. LUA 动词返回 k 』。

C法 " b

在 I SLI 分配 D 存储器中 (" 动词 X 制 i 。**lua_th** 和 **lua_rh** N } D 内容放在 **SLI_STSN_ROUTINE** 动词 X 制 i 中。**lua_data_ptr** N } | 含 SDT RU D X址，而 **lua_data_length** N } | 含 RU D $ 度。

1 扩 9 例行 L 序 k h 置在 **SLI_SDT_ROUTINE** 中 D **lua_prim_rc** 和 **lua_data_length** N } 一起返回，M完 I K **SLI_SDT_ROUTINE**。C SDT 响 & 2 G SDT RU。 OK 主返回 k m > 已 S \ SDT。如果例行 L 序 \ x SDT，那 4 + 主返回 k h 置为 NEGATIVE_RSP " + 否定检 b k 放入 STSN 缓 e x。 k 勿修 D **lua_data_ptr** N }。

"： 来自 C 例行 L 序 D 否定响 & 取消 **SLI_OPEN** 动词。 SLI 返回一 v SESSION_FAILURE 主返回 k 和一 v NEG_RSP_FROM_SDT_ROUTINE 次返回 k。

第3？分 公共服务 API

第16章 公共服务入口点 259
`写公共服务 L 序 259
ACSSVC 260
WinCSV() 261
WinCSVCleanup() 262
WinAsyncCSV() 263
WinCSVStartup() 264
GetCsvReturnCode() 265
TrnsDt 266

第17章 公共服务动词 (CSV) 271
GET_CP_CONVERT_TABLE. 272
CONVERT 276

第16章 公共服务入口点

v 人通信 k 通信服务器 a 供K公共服务L序 h 计 S Z。C API |含公共服务动词(CSV), | I C Z 9 C v 人通信 k 通信服务器 API D&C L序。

任何 v 人通信 k 通信服务器&C L序都 I 9 C b 些公共服务动词来执行下 P b 些Y作:

- 维护%字 Z o 言D代 k 页转换m (**GET_CP_CONVERT_TABLE**)
- + ASCII 字符串转换 I EBCDIC 或 EBCDIC 转换 I ASCII (**CONVERT**)
- + +字Z字符串从一代 k 页转换 = m一 v 代 k 页 (**TRNSDT**)

": Z 3 B D B Z |括D内容涵G K P 关 I 下 P 系统 y a 供D公共服务 API D信息:

- 运行在 Windows NT O D通信服务器
- SNA API M户机 OS/2 f, Windows NT f, Windows 95 f, 和 Windows 3.1 f, |G通过通信服务器/NT z 品 a 供。
- v 人通信 Windows 95 和 Windows NT f

1 在 I b 些系统 a 供D支V之间 P n 异 1, 会 v 现 a >。

编 4 公共服务程 r

下 f D m q 显 > y a 供D头文件以及 ` 译和 4 S 公共服务L序 y 需D b D源L序模 i C法。

m 19. Y作系统D头文件和 b

Y w系统	头文~	库	DLL 名称
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

以下 ? 分 h v K公共服务D入Z c。

ACSSVC

b G y P CSV 动词D同=入Z c。 v人通信k通信服务器为C入Z c a供K现存&C L序D兼容性。

o法

```
void ACSSVC (long)
```

d入D G动词X制 i 指k。

返回值

检i 主和次级返回k D返回值。

b G唯一支V SNA API OS/2 M户D通信服务器入Z c。

WinCSV()

C函} a供K CSV API D同=入Z c。

o法

```
void WINAPI WinCSV(long vcb)
```

N数 5明

vcb 指向动词X制i D指k

返回值

；P返回值。动词X制i中D **primary_rc** 和 **secondary_rc** 字段m>任何错误。

"**：** 也I N阅 **WinAsyncCSV()** (页 Z263页D『WinAsyncCSV()』)。

WinCSVCleanup()

C函}从 CSV API 中终止"取消注 a K&C L序。

o法

```
BOOL WINAPI WinCSVCleanup(void);
```

返回值

C返回值指定G否取消注 a I功。如果C值；G 0, 则v人通信k通信服务器I功X取消注 a K&C L序。如果C值为 0, 则v人通信k通信服务器；取消注 a &C L序。

C法5明

9C **WinCSVCleanup()** 从 CSV API, 例如, 对Z自I D分配=X定&C L序D资源, 取消注 a CSV API &C L序。

WinAsyncCSV()

C函}只为 **TRANSFER_MS_DATA** a供异=入Z c。如果&C L序+C函}C Z其|动词，则C=骤G同=D。

o法

```
HANDLE WlNAPI WinAsyncCSV(HHWND hWnd,
                          long vcb);
```

N数　5明

hwnd　S U完I信息D窗Z d z。

vcb　指向动词X制i D指k

返回值

C返回值m>动词k s G否I功。如果函}I功，则5际返回值G异=任务d z。如果函}；I功，则v人通信k通信服务器返回 0。

C法5明

异=Y作完I 1，&C L序窗Z *hWnd* S U=I带P **WinAsyncCSV** D **RegisterWindowMessage** 返回D作为d入字符串D消息。*wParam* d？|含I原<函}w C返回D异=任务d z。 *IParam* d？|含原< VCB 指k " I取消引C以确定最后D返回k。

如果函}返回I功，v人通信k通信服务器会在Y作完I或取消对话1+ **WinAsyncCSV()** 消息传]=&C L序。

WinCSVStartup()

C函}允许&C L序指定X需D公共服务动词 API D f >，检w X定 CSV API D细Z。 C w C ；G X需D， +如果9 C D话，也&C 9 C **WinCSVCleanup**。

o法

```
int WINAPI WinCSVStartup (WORD wVersionRequired,
                          LPWCSVDATA csvdata);
```

N数 5明

wVersionRequired

指定X需D CSV API 支V D f >。 _位字Z指定K 1 f >(校订>)号; M位字Z指定K主 f >号。

lpwCSVDATA

|含基> CSV API DLL 信息。

返回值

返回值m w CSV API G否I功注a K&C L序以及G否支V y a供D f >号。如果返回值G 0, 则 CSV API 支V指定D f > " R已I功注a K&C L序。否则， +返回下P值中D一v。

WCSVVERNOTSUPPORTED

C X b D CSV API ；a供X需D CSV API 支V f >。

WCSVINVALID

CSV API ；确定X需D f >。

C法5明

WinCSVStartup() 要a供+来 API 发行>D兼容性。 1 0支V D f >G 1.0。

下P a构h v K 5际 CSV API 5现D细Z。

```
typedef struct tagWCSVDATA { WORD wVersion;
                char szDescription[WCSVDESCRIPTION_LEN+1];
                } WCSVDATA, FAR *LPWCSVDATA;
```

1&C L序做|最后D CSV API w C 1， |M w C **WinCSVCleanup()**。

GetCsvReturnCode()

９ＣＣ入Ｚｃ＋动词中Ｄ主和次级返回ｋ转换＝Ｉ打！字符串中。｜返回Ｋ＆ＣＬ序９ＣＤｖ错字符串Ｄｊ准集合。

o法

```
int WINAPI GetCsvReturnCode (struct csv_hdr *vcb,
                             UINT buffer_length,
                             unsigned char *buffer_addr);
```

N数　5明

vcb　动词Ｘ制ｉＤＸ址。

buffer_length

buffer_addr 指向Ｄ缓ｅｘ＄度。推荐Ｄ＄度Ｇ 256。

buffer_addr

＋＃存ｑ＝化Ｄ、Ｕ终ａ字符串(指定缓ｅｘ中Ｄ字符串＄度)Ｄ缓ｅｘＸ址。

返回值

0x20000001

ＣＮ｝无效；Ｃ函｝；能读取指定Ｄ**动词**或；能写入＝指定缓ｅｘ中。

0X20000002

指定Ｄ缓ｅｘ＋小。

C法5明

在 **buffer_addr** 中返回Ｄｈｖ错误字符串＂；以新行符(**\n**)终止。

TrnsDt

C函}+ SBCS 和 DBCS 字符串从一v代k页转换=m一v代k页。v人通信k通信服务器在 **TRNSDT.DLL** 文件中a供K **TrnsDt**。**TransDt** 只在 DBCS 会话中IC。

o法

```
TrnsDt (PASSSTRUCT *passparm);
```

PassParm 格式

C函}+ SBCS 和 DBCS 字符串从一v代k页转换=m一v代k页。在下Pm中,『✓』mwv人通信k通信服务器支V一对代k页间D转换, 而"-" (横行)同样mw; PL序支VC转换。

日本		932	930	931	939	290	037	1027
	932	-	✓	✓	✓	✓	✓	✓
	930	✓	-	-	-	-	-	-
	931	✓	-	-	-	-	-	-
	939	✓	-	-	-	-	-	-
	290	✓	-	-	-	-	-	-
	037	✓	-	-	-	-	-	-
	1027	✓	-	-	-	-	-	-
韩国		949	833	834	933			
	949	-	✓	✓	✓			
	833	✓	-	-	-			
	834	✓	-	-	-			
	933	✓	-	-	-			
台湾		950	037	835	937			
	950	-	✓	✓	✓			
	037	✓	-	-	-			
	835	✓	-	-	-			
	937	✓	-	-	-			
中国		1381	836	837	935			
	1381	-	✓	✓	✓			
	836	✓	-	-	-			
	837	✓	-	-	-			
	935	✓	-	-	-			

C头文件 **TRNSDT.H** `译, 同19C **TRNSDT.LIB** 文件从L序 LIB 子目<4S。

PassParm 格式

passparm q =如下y>:

WORD *parm_length*
C a 构D $ 度(d 入)

WORD *exit_code*
v Z k (d v)

0000H } # a x 。

0001H ; P 指定支V D 转换。

000CH
Exit_code 字段 ; P u < 化为 0。

0080H 最后D字符G DCBS D左k ? 分。 n d K U字符。

WORD *in_length*
源缓 e x D $ 度(d 入)

LPBYTE *in_addr*
源缓 e x X址(d 入)

WORD *out_length*
目 j 缓 e x $ 度(d 入)

如果指定D $ 度 + 小而 ; 能返回 y P D转换 }] , 则会返回X需D $ 度。

LPBYTE *out_addr*
目 j X址缓 e x (d 入)

WORD *trns_id*
t 为 c (d 入)

WORD *in_page*
源代 k 页 f (d 入)

WORD *out_page*
目 j 代 k 页(d 入)

WORD *option*
选项(d 入/ d v)

Input d 入选项如下:

Bits 15-9
t 为 c

Bit 8 目 j 字符串 _ P SO/SI

Bits 7-3
t 为 c

Bit 2 9 C ; I ` 辑D SBCS m

Bit 1 I DBCS * < D源字符串

Bit 0 源字符串 _ P SO/SI

Output
d v 选项如下:

4 在 DBCS a x

0 在非 DBCS 处 a x

":

1. 位 8 和位 0 &C h置如下:

   从 PC 转换 = 主机 位 8=1

   从 PC 转换 = 主机 位 0=0

   从主机转换 = PC 位 8=0

   从主机转换 = PC 位 0=1

2. 9 C **SYSCTBL.EXE** 指定 **TrnsDt** 9 C D定制m D { F。要转换 SBCS 字符串, **TrnsDt** + 9 C带P **Option** N } (位 2 h置为 FALSE)D定制m。如果h置K位 2, + ; P指定m { , 则 **TrnsDt** 9 C缺 ! m。要9 C **SYSCTBL.EXE** 在指定m { 1 转换 DBCS 字符串, 则 **TrnsDt** + 总G 9 C定制m。在b种 i v下, ; 9 C位 2 D **Option** N } 。

3. 通 #, **TrnsDt** 要 s 主机 }] | 含 I 对D SO/SI X制字符。然而, 要转换一对混合 }] 字符串, C }] X须从 ; 带 SO X制字符D + 字Z字符 * < 。在b种 i v下, }] ; C j 6 + 字Z字符。位 1 在b种 i v下G P C D。如果您 + 位 1 h置为 1, 则 **TrnsDt** + 缓 e x D * 头处理为 + 字Z字符或 SO X制字符。

PassParm 格式

0 NO_ERROR

2 ERROR_FILE_NOT_FOUND

TrnsDt R ; = C Z转换指定代 k D m。

87 ERROR_INVALID_PARAMETER

N } 无效。

111 ERROR_BUFFER_OVERFLOW

目 j 缓 e x + 小。

150 ERROR_MEMORY_ALLOCATE

内存分配 v 错。

PassParm 格式

即 9 一 v 小缓 e x 也 I 通过 9 C **TrnsDt** D v Z k 和选项N } I 功处理大D }] 转换。W先, 9 C小源缓 e x 和 = 或三 6 大D目D缓 e x (对Z从 PC = 主机) 来启动 **TrnsDt**, 然后基 Z S U = D a x k W先观 l 对话如何 a x 。然后依次继续。

例如, 1 转换 + + 字Z字符分 I = ? 分 1 , 或在 SO 和 SI X制字符间 ; P 完全 a x 1 , 定义缓 e x 指 k 和 | D位置, 然后执行下一次 w C。

PassParm 格式

下 P > 例 + 主机代 k 0x4040 转换为 PC 代 k 。

```
#include "trnsdt.h"

PASSSTRUCT      passparm;
char            bufs[20], buft[20];
```

```
int             rc;

//Setup the string to be translated
bufs[0] = 0x0e;
bufs[1] = 0x40;
bufs[2] = 0x40;
bufs[3] = 0x4f;

//Setup the parameter
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;

//Translate the string via TrnsDt
if (rc = TrnsDt(&passaparm))
   printf("Error Return Code = %d\n\r", rc);
   printf("Exit Code = %d\n\r", passaparm.exit_code);
   exit(0);
else
  .....
```

第17章 公共服务动词 (CSV)

GET_CP_CONVERT_TABLE

C动词a供K一v 5 C L序服务， | (" K一v代k页=m一v代k页D转换m。 C动词返回K 256字Z $ D转换m， &C L序I以9 C | 来5现字符 i m以转换字符串。

C L序I能需要在k(希望 }] 在；同代k页 ` k D)Z c通信1 5现 }] 转换。

```
struct get_cp_convert_table
  {
  unsigned short  opcode;          /* Verb identifying operation code.      */
  unsigned char   opext;           /* Reserved.                             */
  unsigned char   reserv2;         /* Reserved.                             */
  unsigned short  primary_rc;      /* Primary return code from verb.        */
  unsigned long   secondary_rc;    /* Secondary (qualifying) return code.   */
  unsigned short  source_cp;       /* Source code page for conversion table */
  unsigned short  target_cp;       /* Target code page for conversion table */
  unsigned char  *conv_tbl_addr;   /* Address to put conversion table at    */
  unsigned char   char_not_fnd;    /* Character not found option: either    */
                                   /* substitute character or round trip    */
  unsigned char   substitute_char; /* Substitute character to use.          */
  } GET_CP_CONVERT_TABLE;
```

source_code_page

绘制置换字符D代k页号k。代k页号k I以G下P号k:

- ASCII 代k页(. x制)
  - 437 @国 IBM PC
  - 737 希腊
  - 813 希腊
  - 819 ANSI j准
  - 850 多o言
  - 852] K 9 e伐K/匈牙利/ (兰/南9拉夫
  - 855 西里尔o
  - 857 土耳其o
  - 858 多o言
  - 860 葡Q牙o
  - 861 y : o
  - 862 希.来o
  - 863 加拿大法o
  - 864 "拉. o
  - 865 日耳 | o
  - 866 西里尔o
  - 874)国o
  - 912 拉丁o 2
  - 915 西里尔o
  - 916 希.来o
  - 920 土耳其o
  - 921 拉脱维亚," U宛o
  - 922 . 3尼亚o

- 923 ANSI j 准
- 1008 " 拉 . o
- 1089 " 拉 . o
- 1124 乌K兰o
- 1125 乌K兰o
- 1127 " 拉 . o/法o
- 1129 越南o
- 1131 W俄^ 9 o
- 1133 老挝o
- 1250 拉丁o 2
- 1251 西里尔o
- 1252 拉丁o 1
- 1253 希腊o
- 1254 土耳其o
- 1255 希 . 来o
- 1256 " 拉 . o
- 1257 (^ D海(拉脱维亚、 " U宛、 . 3尼亚)
- 1258 越南o

• EBCDIC 代k页(. x制)
  - 037 @国/加拿大法o/荷兰/葡Q牙/M西
  - 273 B国/B X利
  - 275 M西
  - 277 $ s/挪威
  - 278 芬兰/瑞d
  - 280 意大利
  - 284 拉丁@洲/西`牙
  - 285 " 国
  - 297 法国
  - 420 " 拉 . o
  - 424 希 . 来
  - 500 H利1/瑞?法o/瑞? B o
  - 803 希 . 来
  - 870] K 9 e伐K/匈牙利/(兰/南9拉夫
  - 871 y :
  - 875 希腊
  - 924 拉丁o 1
  - 1025 西里尔o
  - 1026 土耳其
  - 1047 拉丁o 1
  - 1112 拉脱维亚," U宛o

- 1122 ． 3 尼亚
- 1123 乌K兰
- 1130 越南 o
- 1132 老挝
- 1140 @国/加拿大/荷兰/葡Q牙/M西/B X利/新西兰
- 1141 B国/B X利
- 1142 $ s /挪威
- 1143 芬兰/瑞 d
- 1144 意大利
- 1145 拉丁@洲/西 ` 牙
- 1146 " 国
- 1147 法国
- 1148 H利 1 /瑞？
- 1149 y ：

• C户定义D代 k 页
- 65280 = 65534
- 1 9 C C户定义代 k 页 1， W先 + 带 P C户定义 7 6 D注 a m项定义 = CPT 文件中，如下对 Z v 人通信:

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Personal
Communications /CurrentVersion/COMCPT
```

对 Z 通信服务器， + 带 P C户定义 7 6 D注 a m项定义 = CPT 文件中，如下 y >:

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Communications
Server/CurrentVersion/COMCPT
```

"： 只 P 源中 D 恒 H 字符和目 j 代 k 页 I 互相 z I 转换。在相 F j 准中指定 D 字符对通 # ；能相互转换。

target_code_page

要转换 D 目 j 字符串代 k 页号。C 号 I 以 G 任何 **source_code_page** 显 > D 号 k 。

convert_table addr

S U 256 字 Z 转换 m D 缓 e x X 址。C 缓 e x X 须 G 读/写分段。

character_not_found

如果源代 k 页 f 中 D 字符在目 j 代 k 页中；存在 1 要执行 D 任务。指定下 P 值之一:

SV_ROUND_TRIP

C 选项 9 值存储在转换 m 中， b 样如果通过；换源和目 j 代 k 页来 z I 转换 m，从源 = 目 j 代 k 页 " 再次返回 D 转换 a 果 + < 致原 < 字符。X 须为选择 **ROUND_TRIP** 选项， 9 C 选项 D m z I 运行。

SV_SUBSTITUTE

存储在转换 m **substitute_character** N } 中指定 D 字符

substitute_character

如果源代 k 页中D字符在目 j 代 k 页中；存在, " R 如果 **character_not_found** N } h 置为 **SV_SUBSTITUTE** 1 存储在转换m中D字 Z。

OK 返回 k m > **GET_CP_CONVERT_TABLE** 动词运行 I 功。

在返回 k 返回 OK 1 返回K下 P N }:

convert_table

转换m在 I **CONV_table_addr** 指定D X址O (" 。

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

C动词+ ASCII 字符串转换为 EBCDIC，EBCDIC 字符串转换为 ASCII。

L序I在k需要 EBCDIC }] D Z c通信，或X须+ { F转换通过S Z(如需要 EBCDIC { F D APPC)1执行 }] 转换。

": **CONVERT** 动词；\ DBCS 支V。 I以9 C **TrnsDt** 来转换P +字Z D字符。

```
struct convert
  {
  unsigned short  opcode;          /* Verb identifying operation code.      */
  unsigned char   opext;           /* Reserved.                             */
  unsigned char   reserv2;         /* Reserved.                             */
  unsigned short  primary_rc;      /* Primary return code from verb.        */
  unsigned long   secondary_rc;    /* Secondary (qualifying) return code.   */
  unsigned char     direction;     /* Direction of conversion - ASCII to     */
                                   /* EBCDIC or vice-versa.                  */
  unsigned char     char_set;      /* Character to use for the conversion    */
                                   /* A, AE, or user-defined G.              */
  unsigned short    len;           /* Length of string to be converted.      */
  unsigned char    *source;        /* Pointer to string to be converted.     */
  unsigned char    *target;        /* Address to put converted string at.    */
  } CONVERT;
```

return_code OK error codes

direction

代k转换D性质。

SV_ASCII_TO_EBCDIC

+ ASCII 字符转换为 EBCDIC

SV_EBCDIC_TO_ASCII

+ EBCDIC 转换为 ASCII

character_set

在源字符串中允许D字符集合。 I以指定三类I **CONVERT** 动词9 C D ASCII/EBCDIC 转换m: SV_A, SV_AE, and SV_G。type-A 和 type-AE m G在v人通信k通信服务器中指定D。

转换m D q = | 含?行 32 字符D 32 行。 ?行代m 16 v I打！ . y x制字符，后f P回5和换行符。头 16 行a供K ASCII = EBCDIC 转换D信息。 Z二v 16 行a供K EBCDIC = ASCII 转换D信息。 C m X须|含y P 32 行。

1 v人通信k通信服务器执行转换1， | 9 C ? v x入字符D }字H价}作为=转换m D原c w引。 C w引指定K |含转换字符 . y x制值D m D位置。例如，假h在m中D Z 48 v位置|含 X'F0' 值。 v人通信k通信服务器+带P 48 (X'30')D x入字符转换为 240 (X'F0')。

表 A m A 转换K大写字e A = Z, }字字符 0 = 9 和X b字符 $、# 和 @。源字符串D Z一字符X须G大写字e或三v X b字符之一；如果；G，则；转换，同1返回 INVALID_FIRST_CHARACTER 次级返回k。在 ASCII = EBCDIC 方向中，小写 ASCII 字符转换为大写字e EBCDIC 字符。

尾?U W(在源字符串末端D U W)都转换为= v方向O D U W。对&D， 6入U q转换为 X'00'。

任何源字符都转换为 X'00'，返回 CONVERSION_ERROR。然而，{ v 转换都已完 I 。

表 AE m AE 转换K字母 } 字字符(A = Z, a = z, 0 = 9) 和X b 字符 $, #, 和 @，和 d 号(。). ；P 对字符串字符DW字母限制。

$ UW(在源字符串末端D UW)都转换为 = v 方向D UW。对& D， 6 人U q 转换为 X'00'。

任何源字符都转换为 X'00'，返回 CONVERSION_ERROR。然而，{ v 转换都已完 I 。

表 G I 以 9 C G m从任何字符转换 = 其 | 字符 (；只从 ASCII = EBCDIC 或 EBCDIC = ASCII)。然而，X须在 **CONVERT** 动词O指定 ASCII_TO_EBCDIC 以 9 C m D头 k ？分 " 指定 EBCDIC_TO_ASCII 9 C下 k ？分。

v 人通信 + 在注 a m D

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Personal Communications /
                                    CurrentVersion/COMTBLG
```

下Q w以C = G m D全 7 6 {。通信服务器 + 在注 a m D

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Communications Server/
                                    CurrentVersion/COMTBLG
```

Q w以C = G m D全 7 6 {。对Z 32 位 Windows NT M户，注 a m中D m G 7 6 D位置G:

```
HKEY_LOCAL_MACHINE/SOFTWARE/IBM/Comm.Server for NT SNA/Client/
                                    CurrentVersion/COMTBLG
```

其 | 为 **CONVERT** 动词 a 供D N } G:

length 要转换D字符 }。

字符串D $ 度；能，过分配 x **source_addr** 或 **target_addr** D分段 $ 度。

source_addr
要转换D字符串X址。

target_addr address
S U转换字符串D X址。

"： 如果& C L 序；需要 # t 源字符串，则 I 以对 **source_addr** 和 **target_addr** 指定相同 d ？。

OK 返回 k m > **CONVERT** 动词运行 I 功。

下P ？分显 > K k **CONVERT** 动词和返回 k h v 位置相关 * D主要和次级 v 错返回 k。

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

第4？分 EHNAPPC API

第18章 EHNAPPC & C程r S口. 283
`写 EHNAPPC L序 283
EHNAPPC 例行L序 283
EHNAPPC_Allocate 283
目D 283
过L 5 w 283
N } 284
返回k 284
EHNAPPC_Confirm 284
目D 284
过L 5 w 284
N } 285
返回k 285
EHNAPPC_Confirmed 285
目D 285
过L 5 w 285
N } 285
返回k 285
EHNAPPC_Deallocate 285
目D 285
过L 5 w 285
N } 286
返回k 286
EHNAPPC_ExtendedAllocate 286
目D 286
过L 5 w 286
N } 286
返回k 287
EHNAPPC_Flush 287
目D 287
过L 5 w 287
N } 287
返回k 287
EHNAPPC_GetAttributes 288
目D 288
过L 5 w 288
N } 288
返回k 288
EHNAPPC_GetCapabilities 288
目D 288
过L 5 w 288
N } 289
返回k 289
EHNAPPC_GetDefaultSystem 289
目D 289
过L 5 w 289
N } 289
返回k 289

EHNAPPC_IsRouterLoaded 289
目D 289
过L 5 w 289
N } 289
返回 k 290
EHNAPPC_PrepareToReceive 290
目D 290
过L 5 w 290
N } 290
返回 k 290
EHNAPPC_QueryConfiguredSystems 290
目D 290
过L 5 w 290
N } 290
返回 k 291
EHNAPPC_QueryConvState 291
目D 291
过L 5 w 291
N } 291
返回 k 291
EHNAPPC_QueryFullSystems 291
目D 291
过L 5 w 291
N } 292
返回 k 292
EHNAPPC_QueryUserid 292
目D 292
过L 5 w 292
N } 292
返回 k 292
EHNAPPC_QuerySystems 292
目D 292
过L 5 w 292
N } 293
返回 k 293
EHNAPPC_ReceiveAndWait 293
目D 293
过L 5 w 293
N } 293
返回 k 294
EHNAPPC_ReceiveImmediate 294
目D 294
过L 5 w 294
N } 294
返回 k 295
EHNAPPC_RemoteProgramStart 295
目D 295
过L 5 w 295
N } 295
返回 k 295
EHNAPPC_RqsToSend 295
目D 295

过L 5 w 296
N } 296
返回 k 296
EHNAPPC_SendData 296
目D 296
过L 5 w 296
N } 296
返回 k 296
EHNAPPC_SendError 297
目D 297
过L 5 w 297
N } 297
返回 k 297
EHNAPPC_StartHostProgram 297
目D 297
过L 5 w 297
N } 297
返回 k 298
EHNAPPC Structures 298
AS400_SYS 298
目D 298
过L 5 w 298
N } 298
appcrtrcap_hdr 298
目D 298
过L 5 w 298
N } 299
appcrtrcap_mult 299
目D 299
过L 5 w 299
N } 299
appcrtrcap_query 299
目D 299
过L 5 w 299
N } 299
EHNAPPC API D返回 k 300
在 Windows 95 和 Windows NT O运行 16 位 EHNAPPC L序 301

第19章 数] 变换 Windows & C程 r S口 303
}] d 换 Windows API 例行L序 303
EHNDT_ANSIToEBCDIC 303
C途 303
过L 5 w 303
N } 303
返回 k 304
EHNDT_ASCIIToEBCDIC 304
C途 304
过L 5 w 304
N } 304
返回 k 304
EHNDT_EBCDICToANSI 304
C途 304

过 L 5 w 305
N } 305
返回 k 305
EHNDT_EBCDICToASCII 305
C 途 305
返回 k 306

第18章 EHNAPPC & C程 r S口

b只能在通信服务器 SNA API M户机O 9 C。

EHNAPPC 通信 API a供K一v ` 写v人计c机和 AS/400 系统间D互Y作处理& C L序D方法。 | Q L序员从M级通信L序h计和2件，通性类型t离*。在9 C C API 1， & C L序员需要 ` 写 AS/400 和 PC L序。几乎任何I I主机& C L序访问D内容都I以扩9为伙i PC & C L序。C API I以C Z关键性能D & C L序。

> B h v (" 32 位 EHNAPPC API Windows NT 和 Windows 95 f通信服务器 SNA API M户机D例行L序、 }] a构和返回k。大多}函}也I C Z 16 位 API Windows 3.1 f。

编4 EHNAPPC 程r

下f D m q显> y a供D头文件以及 ` 译和4 S EHNAPPC L序y需D b D源L序模i C法。

m 20. Y作系统D头文件和b

Y w系统	头文~	库	DLL 名称
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 } P程 r

下f +详细h v关Z M户机 Windows API 例行L序D w v方f:

- 目D
- 过L 5 w
- N }
- 返回k

EHNAPPC_Allocate

?的

C函}以一v伙i B务L序*<对话。

过程5明

```
#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);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

bType j 6 + 要分配D对话类型。 I 能D值为:

EHNAPPC_BASIC (0)

EHNAPPC_MAPPED (1)

bSynchLevel j 6 > X和伙 i L序间D同 = 级。 I 能D值为:

EHNAPPC_SYNCLEVELNONE (0)

EHNAPPC_SYNCLEVELCONFIRM (1)

lpszLocationName 指向一 v 指定主机系统 { D U终 a 字符串。如果指 k h 置为 NULL, 则 9 C缺 ! 系统。

lpszTpn 指向一 v 指定伙 i L序 { D U终 a 字符串。如果 Z 一 v 字符小 Z 0x40, M ; 执行 ASCII = EBCDIC D转换。

nPipLength j 6 L序 u < 化 N } （PIP） }] D $ 度。如果 C d ? 为 0, 则 ; 发M PIP }] 。

lpPipData 指向 PIP }] 。PIP }] X须以 GDS q =, " R X须C EBCDIC。

lpdwConversation 指向一 v C来 + d z 返回 x 下一 v w C 9 C D + 字 d ? 。 C d z G C Z ? v 对话D唯一值。

返回 k

P 关返回 k, k N阅 Z 300页D 『EHNAPPC API D返回 k 』

EHNAPPC_Confirm

? 的

C 函 } k s 确认至 q 为止多发M D y P }] 都 I 伙 i S U。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int far pascal EHNAPPC_Confirm(
HWND            hWnd,
DWORD           dwConversation,
LPBYTE          lpRequestToSendRcvd);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话d z。

lpRequestToSendRcvd 指向一v C Z存储伙 i B务L序G否发v REQUEST_TO_SEND 动词D d ?。TRUE 值m w伙 i B务L序发v K一v REQUEST_TO_SEND 动词。

返回k

P关返回k， k N阅Z300页D『EHNAPPC API D返回k』。

EHNAPPC_Confirmed

?的

C函}向k s确认D伙 i 发M一v确认。

过程5明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int far pascal EHNAPPC_Confirmed(
    HWND          hWnd,
     DWORD         dwConversation);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话d z。

返回k

P关返回k， k N阅Z300页D『EHNAPPC API D返回k』。

EHNAPPC_Deallocate

?的

C函}+已分配D对话b }分配。

过程5明

```
#include "E32APPC.H"
extern int far pascal EHNAPPC_Deallocate(
         HWND                hWnd,
         DWORD               dwConversation,
         DeallocateEnum      bType);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话d z。

bType j 6 M户机 + 要执行D b } 分配D类型。 I 能D值为:

EHNAPPC_DEALLOCATESYNCLEVEL (0)

EHNAPPC_DEALLOCATEFLUSH (1)

EHNAPPC_DEALLOCATEABEND (2)

返回 k

P关返回 k， k N阅Z 300页D 『EHNAPPC API D返回 k 』。

EHNAPPC_ExtendedAllocate

? 的

C函 } 以一 v 伙 i B务L序 * < 对话 " R I 能 2 G 2 全性或模 = 规 q。

过程 5 明

```
#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);
```

N数

hWND j 6 1 0 D & C L序窗Z。

bType j 6 + 要分配D对话类型。 I 能D值为:

EHNAPPC_BASIC (0)

EHNAPPC_MAPPED (1)

bSynchLevel j 6 > X和伙 i L序间D同 = 级。 I 能D值为:

EHNAPPC_SYNCLEVELNONE (0)

EHNAPPC_SYNCLEVELCONFIRM (1)

lpszLocationName 指向一 v 指定主机系统 { D U终 a 字符串。如果指 k h 置为 NULL，则 9 C缺 ! 系统。

lpszTpn 指向一 v 指定伙 i L 序 { D U 终 a 字符串。如果 Z 一 v 字符小 Z X'40', M ; 执行 ASCII = EBCDIC D转换。

lpszModeName。以下 G | { 一 v 模 = D规则:

Mode Names 为 1 = 8 v字符。? v ?分D Z 一 v 字符X须G一 v 大写字 e D字符（A-Z）或 X b 字符（[@0000]、#、$）。# ` D字符 I 以G大写字 e 字符（A-Z）、 } 字（0-9）或X b 字符(@、#、$)。

bSecurityType j 6 + 要 9 C D 2 全性类型。 I 能D值为:

EHNAPPC_SECURITY_NONE (0)

EHNAPPC_SECURITY_SAME (1)

EHNAPPC_SECURITY_PGM (2)

lpszUserId 指向一 v | 含C户 ID D U终 a 字符串。最大 $ 度为 10 v字符。

lpszPassword 指向一 v | 含 Z n D U终 a 字符串。最大 $ 度为 10 v字符。

nPipLength j 6 PIP }] D $ 度。如果C d ? 为 0, 则 ; 发M PIP }] 。

lpPipData 指向 PIP }] 。PIP }] X须以 GDS q =, " R X须C EBCDIC。

lpdwConversation 指向一 v C来 + d z 返回 x 下一 v w C y 要 9 C D + 字 d ?。

返回 k

P 关返回 k, k N阅 Z 300页D『EHNAPPC API D返回 k 』。

EHNAPPC_Flush

?的

C函 } 9 M户机发M其缓 e x 中 I 能 5 P D }] 。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_Flush(
    HWND          hWnd,
     DWORD     dwConversation);
```

N数

hWnd j 6 1 0 & C L 序D窗 Z。

dwConversation j 6 从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

返回 k

P 关返回 k, k N阅 Z 300页D『EHNAPPC API D返回 k 』。

EHNAPPC_GetAttributes

?的

指定对话D返回 t 性, ｜括 > X和伙 i B务L序D LU ｛ F、处理同 = 级 p 以及 v Z 2 全性而 a 供D C户 ID。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_GetAttributes(
     HWND             hWnd,
     DWORD            dwConversation,
     LPBYTE           lpbSyncLevel,
     LPSTR            lpszModeName,
     LPSTR            lpszLuName,
     LPSTR            lpszPluName,
     LPSTR            lpszUserId);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

dwConversation j 6 I EHNAPPC_Allocate 或 EHNAPPC_Extended Allocate 返回D对话 d z。

lpbSyncLevel 指向一 v C来返回同 = 级D字Z d ?。

lpszModeName 指向一 v C来返回 8 字符方 = ｛ D U终 a 字符串。

lpszLuName 指向一 v C来返回 > X B务L序D LU D U终 a 字符串。

lpszPluName 指向一 v C来返回伙 i LU ｛ F D U终 a 字符串。

lpszUserId 指向一 v C来返回C户 ID（ ｜C来 (" C , S ） D U终 a 字符串。

返回 k

P关返回 k, k N阅Z 300页D『EHNAPPC API D返回 k 』。

EHNAPPC_GetCapabilities

?的

C函 } n 9 }] a 构, 指 > M户机 1 0 ：载D能&。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_GetCapabilities(
    HWND          hWnd,
     LPSTR     lpList);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

lpList 指向C来检w功能信息D功能P m。功能P m I 一 v后 z 功能 a 构D d ？号D头组 I。一P d 入， C P mM指定要 i 询D功能，一P d v， C P mM | 含功能信息。

"： P 关 = 加 a 构D信息， k N 阅 Z 298页 D 『 appcrtrcap_hdr 』、 Z 299页 D 『 appcrtrcap_mult 』 和Z 299页D 『 appcrtrcap_query 』。

返回 k

P 关返回 k， k N阅Z 300页D 『 EHNAPPC API D返回 k 』。

EHNAPPC_GetDefaultSystem

？的

C函 } 返回M户机要， S = D缺！系统 { 。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned pascal EHNAPPC_GetDefaultSystem(
    HWND          hWnd,
     LPSTR     lpszDefSysName);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

lpszDefSysName 指向一 v C来返回缺！系统 { D字符缓 e x。 C系统 { 以一 v U终 a 字符串形 = 存储在C缓 e x 中。

返回 k

P 关返回 k， k N阅Z 300页D 『 EHNAPPC API D返回 k 』。

EHNAPPC_IsRouterLoaded

？的

C函 } 确定M户 7 I L 序G否装入内存中。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern bool EHNAPPC_IsRouterLoaded(
     HWND     hWnd);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

返回 k

如果；P装入通信服务器 SNA M户 7 I L序，则返回 k G FALSE (0)。否则，返回值为 TRUE (1)。

EHNAPPC_PrepareToReceive

?的

C函}让L序准 8 S U }]。 9 C带 EHNAPPC_ReceiveImmediate D函} k 9 C EHNAPPC_ReceiveAndWait 一样。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_PrepareToReceive(
     HWND     hWnd,
     DWORD    dwConversation);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

返回 k

P关返回 k， k N阅Z 300页D『EHNAPPC API D返回 k』。

EHNAPPC_QueryConfiguredSystems

?的

C函}返回配置在通信服务器O D系统D { F。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryConfiguredSystems(
     HWND                hWnd,
     LPINT               lpSysCount,
     LPSYSSTRUC          lpSys);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

lpSysCount 指向一 v C来返回系统，S } ? D { } d ?。

lpSys 指向一 v C来返回系统 { D AS400_Sys a 构。缺！系统G C a 构中D Z 一 v 系统。P关 AS400_Sys a 构D 5 w， k N阅Z 298页D『AS400_SYS』.

返回 k

P 关返回 k， k N阅 Z 300页 D『EHNAPPC API D返回 k』。

EHNAPPC_QueryConvState

? 的

C函 } 返回指定对话D状，。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned pascal EHNAPPC_QueryConvState(
    HWND          hWnd,
     DWORD     dwConversation);
```

N数

hWnd j 6 1 0 & C L 序D窗 Z。

dwConversation j 6 从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

返回 k

返回值m > 对话D 1 0 状，。 I 能D值为:

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

? 的

C函 } 返回M户机 + 要， S 至D 系统D网 g { F 和 { F。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryFullSystems(
    HWND          hWnd,
     LPINT           lpSysCount,
     LPFULLSYSSTRUC  lpSys);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

lpSysCount 指向一 v C来返回系统， S } ? D { } d ?。

lpSys 指向一 v C来返回系统 { D AS400_Sys a 构。

返回 k

P关返回 k， k N阅Z 300页D 『EHNAPPC API D返回 k 』。

EHNAPPC_QueryUserid

?的

C函 } 返回C来， S至指定系统D C户 ID。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QueryUserId(
    HWND          hWnd,
     LPSTR         lpszLocationName,
     LPSTR         lpszUserId);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

lpszLocationName 指向一 v | 含要 i 询D系统 { D U终 a 字符串。指定 NULL 来 i 询缺！系统D C户 ID。lpszUserId 指向一 v C来返回指定系统D C户 ID D U终 a 字符串。

返回 k

P关返回 k， k N阅Z 300页D 『EHNAPPC API D返回 k 』。

EHNAPPC_QuerySystems

?的

C函 } 返回M户机要， S D系统 { 。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned EHNAPPC_QuerySystems(
    HWND          hWnd,
     LPINT          lpSysCount,
     LPSYSSTRUC     lpSys);
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

lpSysCount 指向一 v C来返回系统，S } ? D { } d ?。

lpSys 指向一 v C来返回系统 { D AS400_Sys a 构。缺！系统G C a 构中D Z 一 v 系统。P关 AS400_Sys a 构D 5 w， k N阅Z 298页D 『AS400_SYS』.

返回 k

P关返回 k， k N阅Z 300页D 『EHNAPPC API D返回 k 』。

EHNAPPC_ReceiveAndWait

?的

C函 } H待对话中 + = 达D信息，然后 S U C信息。

过程 5 明

```
#include "E32APPC.H"
extern int EHNAPPC_ReceiveAndWait(
    HWND          hWnd,
     DWORD          dwConversation,
     FillEnu        bFill,
     int            nMaxLength,
     LPVOID         lpReceiveData,
     LPBYTE         lpWhatReceived,
     LPBYTE         lpRequestToSendRcvd,
     LPWORD         lpReceiveDataLength );
```

N数

hWnd j 6 1 0 & C L 序D窗Z。

dwConversation j 6 从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

bFill m > L 序要从中 S U }] D m q。I 能D值为:

EHNAPPC_BUFFER (0)（n d 缓 e x）

EHNAPPC_LL (1)（S U完 { 或X断D _ 辑记 < ）

nMaxLength m > I 以 S \ D最大 }] ?。

lpReceiveData 指向 S U }] D缓 e x。

lpWhatReceived m > M户机 S U = D内容。I 能D值为:

EHNAPPC_DATA (0)

EHNAPPC_DATACOMPLETE (1)

EHNAPPC_DATAINCOMPLETE (2)

EHNAPPC_RECEIVEDCONFIRM (3)

EHNAPPC_RECEIVEDCONFIRMSEND(4)

EHNAPPC_RECEIVEDCONFIRMDEALLOC(5)

EHNAPPC_RECEIVEDSEND (6)

lpRequestToSendRcvd 指向一 v C 来存储伙 i B 务 L 序 G 否发 v K 一 v REQUEST_TO_SEND 动词 D d ？。 TRUE (1) 值 m > 伙 i B 务 L 序发 v K REQUEST_TO_SEND 动词。

lpReceiveDataLength 指向一 v C 来返回 M 户机 S U D }] ？D d ？。

返回 k

P 关返回 k， k N 阅 Z 300 页 D 『EHNAPPC API D 返回 k 』。

EHNAPPC_ReceiveImmediate

？的

C 函 } 选定以 i 4 G 否已 S U = 2 4 内容。如果已 S U，则返回 }]。

过程 5 明

```
#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 );
```

N数

hWnd j 6 1 0 & C L 序 D 窗 Z。

dwConversation j 6 从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回 D 对话 d z。

bFill m > L 序 + 在其中 S U }] D m q。 I 能 D 值为:

EHNAPPC_BUFFER (0)（ n d 缓 e x ）

EHNAPPC_LL (1)（ S U 完 { 或 X 断 D _ 辑记 < ）

nMaxLength m > I 以 S \ D 最大 }] ？。

lpReceiveData 指向 S U }] D 缓 e x。

lpWhatReceived j 6 M 户机已 S U = D 内容。 I 能 D 值为:

EHNAPPC_DATA (0)

EHNAPPC_DATACOMPLETE (1)

EHNAPPC_DATAINCOMPLETE (2)

EHNAPPC_RECEIVEDCONFIRM (3)

EHNAPPC_RECEIVEDCONFIRMSEND (4)

EHNAPPC_RECEIVEDCONFIRMDEALLOC (5)

EHNAPPC_RECEIVEDSEND (6)

lpRequestToSendRcvd 指向一 v C Z 存储伙 i B 务 L 序 G 否发 v REQUEST_TO_SEND 动词 D d ？。TRUE (1) 值 m > 伙 i B 务 L 序发 v K REQUEST_TO_SEND 动词。

lpReceiveDataLength 指向一 v C 来返回 M 户机 S U D }] ？D d ？。

返回 k

P 关返回 k， k N 阅 Z 300 页 D 『EHNAPPC API D 返回 k 』。

EHNAPPC_RemoteProgramStart

？的

C 函 } 允许 Windows & C L 序在远 L AS/400 系统 O 启动一 v L 序。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern word EHNAPPC_RemoteProgramStart(
    HWND          hWnd,
    LPSTR         lpszHostSystemName,
    LPSTR         lpszHostProgramName,
    LPSTR         lpszHostLibraryName,
    char FAR      *lpchPipData,
    WORD          wPipDataLength);
```

N 数

hWnd j 6 1 0 & C L 序 D 窗 Z。

lpszHostSystemName 指向一 v | 含远 L 系统 { F D U 终 a 字符串。C 字符串 D 最大 $ 度 G 8 v 字符。如果指 k 为 U D，则 9 C 缺！系统 { F。

lpszHostProgramName 指向一 v | 含 + 启动 D 主机 L 序 { D U 终 a 字符串。

lpszHostLibraryName 指向一 v | 含主机 L 序 b 7 6 D U 终 a 字符串。如果指 k 为 U，则 Q w C 户 D b P m。

lpchPipData 指向主机 L 序 D L 序 u < 化 N }（PIP）}] x。如果 C 指 k 为 U，则；发 M PIP }]。

wPipDataLength | 含 PIP }] D $ 度。

返回 k

P 关返回 k， k N 阅 Z 300 页 D 『EHNAPPC API D 返回 k 』。

EHNAPPC_RqsToSend

？的

C 函 } k s 伙 i 放弃对话 D X 制。1 > X B 务 L 序 f 后从伙 i B 务 L 序 S U = Receive 动词 D lpWhatReceived N } 中 D EHNAPPC_RECEIVEDSEND (6) 1， M 户 + 对话放置在发 M 状，。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_RqsToSend(
    HWND          hWnd,
     DWORD     dwConversation);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

返回 k

P关返回 k， k N阅Z 300页D『EHNAPPC API D返回 k 』。

EHNAPPC_SendData

?的

C函 } + }] 发M x 伙 i B务L序。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_SendData(
    HWND          hWnd,
     DWORD     dwConversation,
     int       nSendDataLength,
     LPVOID    lpSendDataBuffer,
     LPBYTE    lpRequestToSendRcvd);
```

N数

hWnd j 6 1 0 & C L序D窗Z。

dwConversation j 6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话 d z。

nSendDataLength j 6发M缓 e x中 }] D $ 度。

lpSendDataBuffer j 6发M缓 e x D X址。

lpRequestToSendRcvd 指向一 v C来存储伙 i B务 L 序G否发 v K一 v REQUEST_TO_SEND 动词D d ?。 TRUE 值m w伙 i B务L序发 v K一 v REQUEST_TO_SEND 动词。

返回 k

P关返回 k， k N阅Z 300页D『EHNAPPC API D返回 k 』。

EHNAPPC_SendError

?的

C函}向伙iB务L序mw已发现某v错误。在9CC函}后，>XL序处ZSU状，。

过程5明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern int EHNAPPC_SendError(
    HWND          hWnd,
     DWORD       dwConversation,
     LPBYTE      lpRequestToSendRcvd);
```

N数

hWnd j6 10&CL序D窗Z。

dwConversation j6从 EHNAPPC_Allocate 或 EHNAPPC_ExtendedAllocate 返回D对话dz。

lpRequestToSendRcvd 指向一vC来存储伙iB务L序G否发vK一v REQUEST_TO_SEND 动词Dd?。 TRUE 值mw伙iB务L序发vK一v REQUEST_TO_SEND 动词。

返回k

P关返回k， kN阅Z300页D『EHNAPPC API D返回k』。

EHNAPPC_StartHostProgram

?的

C函}允许 Windows &CL序在远L AS/400 系统O启动L序， 9活动D对话允许&CL序确认主机L序}在运行。C&CL序+；C；9C EHNAPPC_Deallocate 函}来终止对话。

过程5明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern word EHNAPPC_RemoteProgramStart(
    HWND          hWnd,
     LPSTR       lpszHostSystemName,
     LPSTR       lpszHostProgramName,
     LPSTR       lpszHostLibraryName,
     char FAR    *lpchPipData,
     WORD        wPipDataLength);
```

N数

hWnd j6 10&CL序D窗Z。

lpszHostSystemName 指向一 v | 含远 L 系统 { F D U终 a 字符串。 C字符串 D最大 $ 度G 8 v字符。如果指 k 为U D，则 9 C缺！系统 { F。

lpszHostProgramName 指向一 v | 含 + 启动D主机 L 序 { D U终 a 字符串。

lpszHostLibraryName 指向一 v | 含主机 L 序 b 7 6 D U终 a 字符串。如果指 k 为U D，则Q w C户 D b P m。

lpchPipData 指向主机 L 序D L 序 u < 化N } （PIP） }] x。如果C指 k 为U，则；发 M PIP }]。

wPipDataLength | 含 PIP }] D $ 度。

返回 k

P 关返回 k， k N阅 Z 300页 D 『EHNAPPC API D返回 k 』。

EHNAPPC Structures

AS400_SYS

? 的

C a 构C 来存储M户机 + ，S 至D 系统D { F。

过程 5 明

```
struct AS400_sys
     (
unsigned char EHNAPPC_SysNameHNAPPC_MAX_SYSTEMS|
     HNAPPC_SYSNAME_SYSNAME_LENGTH|;
     );
```

N数

EHNAPPC_SysName C来存储 y ，S 系统D { F。系统 { 以U终 a 字符串返回。在 } 组中返回 D Z 一 v 系统 G 缺！系统 （EHNAPPC_MAX_SYSTEMS = 32 和 EHNAPPC_SYSNAME_SYSNAME_LENGTH = 10）。

appcrtrcap_hdr

? 的

b G M户机功能 P m头 D a 构。

过程 5 明

```
struct appcrtrcap_hdr
     (
     unsigned char rc;
     unsigned char opcode;
     unsigned int length;
     );
```

N数

rc C来存储功能k s D { v返回k。

opcode m > get 功能k s。 | D值X须G EHNAPPC_OC_CAPABILITIES (0x17)。

length j 6 { v功能P m D $度。C $度|括头D大小加O ? v功能a构D大小。

appcrtrcap_mult

?的

b G C来确定最E通信缓e x 6加器D功能a构。

过程5明

```
struct appcrtrcap_mult
     (
     unsigned int length;
     unsigned char identifier;
     unsigned char rc;
     unsigned int data;
     );
```

N数

length j 6 C功能a构D $度。

identifier m > 最E通信缓e x 6加器。 | D值X须G EHNAPPC_CAP_OPTIMAL_COM_SIZE (X'02')。

rc C来存储C功能k s D返回k。

data C来返回E化D通信缓e x 6加器。

appcrtrcap_query

?的

b G C来i询M户机G否支V指定功能D功能a构。

过程5明

```
struct appcrtrcap_query
     (
     unsigned int length;
     unsigned char identifier;
     unsigned char rc;
     unsigned char data;
      );
```

N数

length j 6 C功能a构D $度。

identifier j 6 + i询D函}。I能D值为:

EHNAPPC_CAP_QUERY_CONV_STATE (3)

EHNAPPC_CAP_EXT_ALLOCATE (4)

rc C来存储C功能k s D返回k。

data C来返回指定D函}G否\支V。

EHNAPPC API 的返回k

M户机 Windows API 中D函}9 C在 E32APPC.H 中定义D下P返回k # }。

m 21. 返回 k

返回k	十y x制值	5明
EHNAPPC_OK	0	| n I功完I
ENHAPPC_DEALLOCNORMAL	1	b }分配} #。
ENHAPPC_PROGRAMMERNOTRUNCATION	2	L序v错; ; P X断。
ENHAPPC_PROGRAMMERTRUNCATION	3	L序v错; X断。
ENHAPPC_PROGRAMMERPURGING	4	L序v错; e }。
ENHAPPC_RESOURCEFAILURETRY	5	资源故O重T。
ENHAPPC_RESOURCEFAILURENORETRY	6	资源故O ;重T。
ENHAPPC_UNSUCCESSFUL	7	未I功。
ENHAPPC_APPCBUSY	8	APPC &。
ENHAPPC_PARMCHKINVALIDVERB	14	N }选定; ; }确D动词。
ENHAPPC_PARMCHKINVALIDCONVERID	15	N }选定; ; }确D对话 ID。
ENHAPPC_PARMCHKBUFFERCROSSEG	16	N }选定; 缓e x ; f段。
ENHAPPC_PARMCHKTPNAMELENGTH	17	N }选定; B务L序{ $度。
ENHAPPC_PARMCHKINVCONVERTYPE	18	N }选定; ; }确D对话类型。
ENHAPPC_PARMCHKBADSYNCLVLALLOC	19	N }选定; 错误D同=级分配。
ENHAPPC_PARMCHKBADRETURNCNTRL	1A	N }选定; 错误D返回X制。
ENHAPPC_ ENHAPPC_PARMCHKPIPTOOLONG	1B	N }选定: PIP }] + $。
ENHAPPC_PARMCHKBADPARTNERNAME	1C	N }选定; 错误D伙i {。
ENHAPPC_PARMCHKCONFNOTALLOWED	1D	N }选定; 确认;允许。
ENHAPPC_PARMCHKBADDEALLOCTYPE	1E	N }选定; 错误D b }分配类型。
ENHAPPC_PARMCHKPREPTORCVTYPE	1F	N }选定; 准8 S U类型。
ENHAPPC_PARMCHKBADFILLTYPE	20	N }选定; 错误D n d类型。
ENHAPPC_PARMCHKRECMAXLEN	21	N }选定; S U最大$度。
ENHAPPC_PARMCHKUNKNOWNSECTYPE	22	N }选定; # t D字段;为c。
ENHAPPC_PARMCHKRESFLDNOTZERO	23	N }选定; # t D字段;为c。
ENHAPPC_STATECHKNOTINCONFSTAT	28	状,检i; ;在确认状,。
ENHAPPC_STATECHKNOTINRECEIVE	29	状,检i; ;在S U状,。
ENAHAPPC_STATECHKREQSNDBADSTATN	2A	状,检i; k s发M错误状,。
ENHAPPC_STATECHKSNDINBADSTATE	2B	状,检i; 发M处Z错误状,。

m 21. 返回 k (续)

返回 k	十 y x 制值	5 明
ENHAPPC_STATECHKSNDERRBADSTAT	2C	状，检 i； 发M错误状，。
ENHAPPC_ALLOCERRNORETRY	32	分配 v 错； ； P 重 T。
ENHAPPC_ALLOCERRRETRY	33	分配 v 错； 重 T。
ENHAPPC_ALLOCERROGMNOTAVAILNR	34	分配 v 错； ； P I C D L 序；重 T。
ENHAPPC_ALLOCERRTPNNOTRECOG	35	分配 v 错； B 务 L 序 { 无法 6 p。
ENHAPPC_ALLOCERRPGMNOTAVAILR	36	分配 v 错； ； P I C D L 序重 T。
ENHAPPC_ALLOCERRSECNOTVALID	37	分配 v 错； 2 全性无效。
ENHAPPC_ALLOCERRCONVTYP	38	分配 v 错； 对话类型；匹配。
ENHAPPC_ALLOCERRPIPNOTALLOWED	39	分配 v 错； ；允许 PIP }]。
ENHAPPC_ALLOCERRPIPNOTCORRECT	3A	分配 v 错； PIP }] ； } 确。
ENHAPPC_ALLOCERRSYNCHLEVEL	3B	分配 v 错； ；支 V 同 = 级。
ENHAPPC_DEALLOCABENDPROGRAM	46	b } 分配异 # 终止 L 序。
ENHAPPC_INSUFFICIENTMEMORY	47	内存；够。
ENHAPPC_MEMORYALLOCERROR	47	内存分配 v 错。
ENHAPPC_MEMORYALLCERROR	48	内存分配 v 错。
ENHAPPC_TOOMANYCONVERSATIONS	4A	对话 + 多。
ENHAPPC_CONVTABLEFULL	4B	转换 m z K。
ENHAPPC_CLIENTNOTINSTALLED	4C	未 2 装M户机
ENHAPPC_CLIENTWRONGLEVEL	4C	M户机在错误 D 级 p。
ENHAPPC_PCSWINNOTLOADED	4D	； P 装入 PSWIN。
ENHAPPC_PCSWINOUTOFMEMORY	4E	PCSWIN 内存；够。
ENHAPPC_INVALIDUSERIDLEN	4F	； } 确 D C 户 ID $ 度。
ENHAPPC_INVALIDPASSWORDLEN	50	； } 确 D Z n $ 度。
ENHAPPC_INVALIDUNAME	51	； } 确 D LU $ 度。
ENHAPPC_UNDEFINED	63	未定义。

在 Windows 95 和 Windows NT O运P 16 位 EHNAPPC 程 r

通信服务器 SNA API Windows 95 和 Windows NT M户机 a 供在 Windows 95 和 Windows NT O运行现存 D 16 位 EHNAPPC L 序 D 功能。要 b 样执行， k 在启动任何 16 位 EHNAPPC &C L 序之 0，从通信服务器 SNA API M户机子目 < 启动 L 序 EHNAPPCD。C L 序 a 供 K 对 32 位 E32APPC.DLL y X 需 D 转换。

第19章 数] 变换 Windows & C程 r S口

b v在通信服务器 SNA API M户O I C。

}] d换 API a供K在 AS/400 和 PC q = D }] 之间转换D能&。 1发M }] 至 AS/400 和从 AS/400 S U }] 1， I能需要转换。 }] d换 API 支V文 > 和大 ? } 字 D q = D转换。

> B h v K组 I }] d换 API D ? v例行L序和返回 k。

数] 变换 Windows API } P程 r

? v }] d换 API 例行L序 + 详细h v以下 ? 分:

- C途
- 过L 5 w
- N }
- 返回 k

EHNDT_ANSIToEBCDIC

C途

C函 } Q字符串从 Windows ANSI 代k页转换为 EBCDIC (扩 d D二- . x制 ; 换 k)。 X须装入 7 I 器，以 9 C例行L序 I 访问 “ASCII 至 EBCDIC” 转换m。

如果目 j 字符串 ; 够大而无法 | 含转换D字符串，那 4 转换会在目 j 字符串D a 尾处停止。如果目 j 字符串大 Z y需要D大小，那 4 在字符串D末端 n d UW。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_ANSIToEBCDIC(
    HWND          hWnd,
     LPSTR         lpsSource,
     LPSTR         lpsTarget,
     unsigned in     wSource,
     LPWORD        lpwTarget );
```

N数

hWnd j 6 & C L序D 1 0 窗 Z。

lpsSource 指向要转换D源(ANSI)字符串。

lpsTarget 指向目 j (转换D)字符串。 **wSource** j 6 源字符串D $ 度(C字 Z)。

lpwTarget 指向 | 含目 j 缓 e x 大小D一 v 字 d ?。 C d ? + C在目 j 缓 e x 中D转换D字符D总 } | 新。

返回 k

如果函 } I 功，那 4 返回 EHNDT_SUCCESS (X'0000')。如果 ; P 装入 7 I 器，那 4 返回 EHNDT_A2E_TABLE_NOT_FOUND (X'FFFC')。如果在 T 图分配 Y 1 缓 e x 1 发 z 错误，那 4 返回 EHNDT_MEMALLOC(X'FFFF')。如果在转换期间发现 ; } 确 }] ，那 4 返回 k G 未转换 D Z 一 v 字符 D 位置加一。

EHNDT_ASCIIToEBCDIC

C 途

C 函 } Q 字符串从 ASCII 转换为 EBCDIC (扩 d D 二 - . x 制 ; 换 k)。X 须装入 7 I 器，以 9 C 例行 L 序 I 访问 “ASCII 至 EBCDIC” 转换 m。如果目 j 字符串 ; 够大而无法 | 含转换 D 字符串，那 4 转换会在目 j 字符串 D a 尾处停止。 如果目 j 字符串大 Z y 需要 D 大小，那 4 在字符串 D 末端 n d U W。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_ASCIIToEBCDIC(
    HWND          hWnd,
     LPSTR         lpsTarget,
     LPSTR         lpsSource,
     unsigned in     wSource,
     LPWORD        lpwTarget );
```

N 数

hWnd j 6 & C L 序 D 1 0 窗 Z。

lpsTarget 指向目 j (转换 D)字符串。

lpsSource 指向要转换 D 源(ASCII)字符串。

wSource j 6 源字符串 D $ 度(C 字 Z)。

lpwTarget 指向 | 含目 j 缓 e x 大小 D 一 v 字 d ？。 C d ？ + C 在目 j 缓 e x 中 D 转换 D 字符 D 总 } | 新。

返回 k

如果函 } I 功，那 4 返回 EHNDT_SUCCESS (X'0000')。如果 ; P 装入 7 I 器，那 4 返回 EHNDT_A2E_TABLE_NOT_FOUND (X'FFFC')。

如果在转换期间发现 ; } 确 }] ，那 4 返回 k G 未转换 D Z 一 v 字符 D 位置加一。

EHNDT_EBCDICToANSI

C 途

C 函 } Q 一 v 字符串从 EBCDIC (扩 d D 二 - . x 制 ; 换 k) 转换至 Windows ANSI 代 k 页。X 须装入 7 I 器，以 9 C 例行 L 序 I 访问 “ASCII 至 EBCDIC” 转换 m。

如果目 j 字符串；够大而无法 | 含转换D字符串，那 4 转换会在目 j 字符串D a 尾处停止。 如果目 j 字符串大Z y需要D大小，那 4 在字符串D末端 n d UW。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_EBCDICToANSI(
HWND             hWnd,
LPSTR            lpsTarget,
LPSTR            lpsSource,
unsigned int     wSource,
LPWORD           lpwTarget ); :
```

N数

hWnd j 6 & C L 序D 1 0 窗Z。

lpsTarget 指向目 j (转换D)字符串。

lpsSource 指向要转换D源(EBCDIC)字符串。

wSource j 6 源字符串D $ 度(C字Z)

lpwTarget 指向 | 含目 j 缓 e x 大小D一 v 字 d ？。 C d ？ + C在目 j 缓 e x 中D转换D字符D总 } | 新。

返回 k

如果函 } I 功，那 4 返回 EHNDT_SUCCESS ('0000')。如果；P 装入 7 I 器，那 4 返回 EHNDT_E2A_TABLE_.NOT_FOUND ('FFFC')。如果在转换期间发现；} 确 }]，那 4 返回 k G未转换D Z 一 v 字符D位置加一。

EHNDT_EBCDICToASCII

C途

C函 } Q一 v 字符串从 EBCDIC (扩 d D二- . x 制；换 k)转换为 ASCII。X须装入 7 I 器，以 9 C例行L 序 I 访问 “ASCII 至 EBCDIC” 转换m。

如果目 j 字符串；够大而无法 | 含转换D字符串，那 4 转换会在目 j 字符串D a 尾处停止。 如果目 j 字符串大Z y需要D大小，那 4 在字符串D末端 n d UW。

过程 5 明

```
#include <WINDOWS.H>
#include "E32APPC.H"
extern unsigned int EHNDT_EBCDICToASCII(
     HWND          hWnd,
     LPSTR         lpsTarget,
     LPSTR         lpsSource,
     unsigned int  wSource,
     LPWORD        lpwTarget );
```

N数

hWnd j 6 & C L 序D 1 0 窗Z。

lpsTarget 指向目 j (转换 D)字符串。

lpsSource 指向要转换 D 源(EBCDIC)字符串。

wSource j 6 源字符串 D $ 度(C 字 Z)。

lpwTarget 指向 | 含目 j 缓 e x 大小 D 一 v 字 d ？。 C d ？ + C 在目 j 缓 e x 中 D 转换 D 字符 D 总 } | 新。

返回 k

如果函 } I 功，那 4 返回 EHNDT_SUCCESS ('0000')。如果 ; P 装入 7 I 器，那 4 返回 EHNDT_.E2A_TABLE_NOT_FOUND ('FFFC')。如果在转换期间发现 ; } 确 }]，那 4 返回 k G 未转换 D Z 一 v 字符 D 位置加一。

第5？分 Java 程r h F S口

第20章 i \主机访问`库(Host Access Class Library) Java f 309
2 4 G ECL? 309
ECL E念 310
会话 310
容器对象 310
P m对象 310
B件 310
错误处理 311
寻址(行、P、位置) 311
在通信服务器 NT f D服务器O 2装 ECL 311
在通信服务器 NT f D 32 位 Windows M户机O 2装 ECL 312
h置 Classpath 313
ECL 代k页转换器 313
ECL 样> 313
Host Launchpad 样> 313
其|样> 315

第21章 使C CPIC-C Java f 317
何谓 CPI-C Java f? 317
2装 CPI-C Java f 317
CPI-C Java f样> 318
M户机样> 318
服务器样> 320

第20章 i \ 主机访问 ` 库(Host Access Class Library) Java f

> B + h v k 3270 和 5250 & C L 序相关D IBM eNetwork 主机访问类 b (ECL) Java f , | 括:

- ECL Java f a 构D简要E v
- 为 ECL 2 装K 2 4
- 哪些样 > I C , | G G 如何工作D

": 字W组合词 HACL 同样也 I 以C来m > eNetwork 主机访问类 b 。在 > i 中, 我G + E 先 9 C字W组合词 ECL。

什么是 ECL?

ECL Java f G一 v 类和方 = D集合, | 能 9 & C L 序员 l] X在 3270 和 5250 }] w c O访问主机& C L 序。ECL 在一 v 完 { D类模型中 5 现K核心主机访问功能, C类模型独 " Z图形显 >, | v 需要一 v 基Z Java D / 览器或兼容D Java 环 3 x 行Y作。

类 b 代m K一 v 完全 f 向对象D i 象主机 , S , | 括:

- 读和写主机m > U间(屏幕)
- 6 Y m > U间中D字段
- 读取Y作员信息 x r (OIA)D状 , 信息
- 传M文件
- 执行重要B件D异 = 通知

& C L 序员 I 以 ` 写 Java 小& C L 序来Y纵来自 }] w m > U间 (如 3270 和 5250)D }] , 而C小& C L 序无需驻 t 在 b 些机器O。 m > U间代m K一 v 虚构D终端屏幕, 在C终端屏幕中 | 含K }] 以及 I 主机& C L 序 a 供D相关 t 性。 1 完 I ; 互后, 小& C L 序 I 以 P 换 = 其 | 任务或简 % X关U会话。 B 务 I 以在主屏幕还未显 > 1 M处理完O。

ECL Java 小& C L 序 I 以:

- 打 * 一 v 至主机D会话
- H待主机 }] D = 来
- 从虚屏幕获C X定D字符串
- 获取字符串D相关 t 性
- h 置新D字符串值
- + }] w 功能键发M回主机
- H待下一 v 主机会话

ECL G对象 EHLLAPI 那样D X定Z M户机D、小型屏幕& C L 序 h 计 S Z 在多方 f O D重大D x 。

- ECL 独 " Z 平 (D
- ECL 直 S k 对 }] w Y 作而非转换后D仿 f 器屏幕。 | + 消 } 在 I S 化窗 Z 中转换和显 > }] w D 系统 * 销。

- ECL ；需要在 > X工作 > O运行仿 f 器软件, b + 减Y X定平 (屏幕 q = 化和键盘 < V D相关性。
- ECL I以 9 C j 准D Web 和 Java 技 u 在M户工作 > O x 行下载和执行。从而 a 供 P效D维护 " Z ！K资源。

ECL 概 n

以下 ? 分 + h v 若 I ECL X ； I Y D E 念。理 b b 些E念 + P 助Z您 | P效X 9 C b 。

会话

在 ECL O下文中, 一 v 会话对象(ECLSession)压 u 至主机D ， S 和C ， S D X性。会话对象还 + 作为一 v 容器服务 Z 其 | 会话X P D对象: ECLPS(m > U间)、ECLOIA(Y作员信息 x r)和 ECLXfer(文件传M)。

会话对象 ; P 相关 * D图形C户 g f 。换 d 话 5 , 创 (ECLSession 5 例 1 ；显 > 仿 f 器屏幕。

容器对象

某些 ECL 类 d 1 着其 | 对象D容器。ECLSession 对象 | 含K一 v ECLPS、ECLOIA 和 ECLXfer 5 例。容器 a 供K向 y | 含D对象返回一 v 指 k D方 = 。ECLSesison 对象 P 一种 GetOIA 方 = , | + 向一 v OIA 对象返回一 v 指 k 。 y | 含D对象 " ；能 I 为公C D容器类 I 员, ；过最好只通过 ECL 方 = 来访问 | 。

P 表对象

一些 ECL 类 a 供K P m D I ~ 代性。例如, ECLConnList 类管理 , S P m。ECL P m 类 " 非异 = | 新X反 3 v P m内容D。 & C L 序X须显 = X w C Refresh 方 = 来 | 新 P m中D内容。 b 样, 一 v & C L 序M能在 | 代 P m 1 无须 < G ~ 代期间 P m I 能 | D D i v 。

事 ~

ECL a 供K X定B件异 = 通知D能&。 & C L 序 I 以选择发 z X定B件 1 向 | 通知。例如, I 以在至主机D , S 状 , | D 1 通知& C L 序。 1 0 , ECL 支V对下 P B件D通知:

m 22. APPC D头文件和 b

事 ~	C Z 6 = 事 ~ 的 S 口
通信 , S 和断 *	ECLLCommNotify
m > U间 | 新	ECLPSNotify
Y作员信息 x r (OIA) | 新	ECLOIANotify

B件通知 I w自D ECL 通知 S Z 来定义。 ? v B件类型都存在一 v 独 " D S Z 。为K能通知某 v B件, & C L 序X须定义和创 (一 v 对象, C | 来 5 现需要通知D B件类型D S Z 。然后, X须通过w C J 1 D ECL 注 a 函 } 对C对象 x 行注 a 。一) & C L 序 x 行K注 a , 则无 [B件何 1 发 z 都 + w C | D NotifyEvent 方 = 。

": & C L 序D NotifyEvent 方 = G在一 v 独 " 执行D线 L O ; w C D。因此, X须再次 d 入 NotifyEvent 方 = 。如果访问K & C L 序资源, 则& 1 9 C J 1 D x 定和同 = 化。

错误处m

通 #, ECL 通过丢 t ECLErr 对象来指 v & C L 序D错误。为K 6 捉 b 些错误, & C L 序& 1 + 对 ECL 对象D w C 封装在一 v T图/ 6 捉 i 中, 如:

```
try {
       int pos = ps.ConvertRowColToPos(row, col);

       //...possibly more references to ECL objects...

    } catch (ECLErr err) {
       System.out.println("ECL Error!  " + err.GetMsgText());
    }
```

1 检 b = 一 v ECL 错误 1, & C L 序 I 以w C ECLErr 对象D方 = 来确定 v 错D f } D原因。也 I 以w C ECLErr 对象来构造一 v 完 { D o 言 P & D错误信息。

寻址(P 、 P 、位置)

ECL a 供K = 种在主机m > U间中 c 寻址D方法。& C L 序 I 以 4 行/P } 或 4 %一线性位置值来寻址字符。m > U间寻址总G基Z 1 D(非基Z 0 D) " 和寻址方 8 无关D。

行和 P 寻址方 8 C Z 和主机 }] D物理屏幕显 > 直 S 相关D & C L 序。通 # 9 C直G坐 j 系统(在左O G中D 1 行 1 P)D方法来寻址屏幕O D c。线性位置寻址方 = (左O G D 1 v 位置, 从左向R, 从O至下 x 行)C Z + { v m > U间 S 为 }] 元X D % v } 组来 i 4 D & C L 序, 或来自 EHLLAPI S Z D & C L 序。

通C, w C 同一方 = D ; 同X w & 选择 ; 同D寻址方 8。例如, 要 + 主机光 j 移至 x 定D屏幕坐 j, & C L 序 I 以w C ECLPS::SetCursorPos 方 = D = 种X w中D任何一 v :

```
ps.SetCusorPos(81);
ps.SetCursorPos(2, 1);
```

如果主机屏已配置 I ? 行 80 P, 则 b 些 o d + 起 = 相同D效果。b v 例子还指 v K寻址方 = 中D一 v 细微 n 异。线性位置方 = I 以 z z 意想 ; = D a 果, 如果& C L 序虚构K m > U间 ? 行D字符 }。例如, 在例子中, W行代 k + Q光 j 放在已配置 I 132 P D m > U间中D Z 1 行Z 80 P O。Z 二行代 k + 光 j 放在和m > U间配置无关D Z 1 行Z 2 P O。

在通 E 服务器 NT f 的服务器O 2 0 ECL

只能在通信服务器 Windows NT f O 9 C。

1 您 e 入K通信服务器 Windows NT 4.0 f CD-ROM " 执行K g f O D = 骤后, 您 + ; a > %击 2 0 4 钮以 * < InstallShield** 向 < D 2 装。一) 2 装完O, 向 < + 指引您完 I ` 下D 2 装。1 向 < D 2 装完O后, + v 现一 v 欢 - 9 C IBM 通信服务器D窗Z。

k %击下；=，继续执行以后D = 骤。 S 下来D一系P f e + a > 您选择 2 装类型、+ 2 装通信服务器D } 动器和目 <、匿 { 访问 IBM k s = 文件D FTP 目 < 以及 + 装P ECL 类文件D } 动器和目 <。

b 项 2 装 a 供K从驻 t 在服务器O D一 v 小& C L 序访问 ECL Java 类 b，或从驻 t 在服务器O D一 v Java & C L 序访问 ECL Java 类 b、 ECL 代 k 页转换器、ECL 文 5 和样 > Java 小& C L 序及 Java & C L 序D能&。(您；X为K在M户机O运行 Java & C L 序而在服务器O 2 装 ECL。)

下 f 我G + h v ECL 组 I ？分及其定义:

IBMCS\ecl101.jar	此文件C Z 从服务器运行 ECL Java 小& C L 序和& C L 序。
IBMCS*.class	b 些文件 I 以 2 装在C 户自定义目 < 下C Z 服务器D访问或基Z web D访问。
IBMCS\ECL\ZIP*.zip	C zip(压 u) q = m > D ECL 类文件。2 装 b 些文件G为K能够 | 方 c X访问 y P D ECL 类 b。后 f + 作 i \ D代 k 页转换器在文件 ecl101n.zip 中。
\IBMCS\SDK\JAVA\ECL\DOC*.*	HTML q = D * 机 ECL 文 5。此文 5 已 q = 化过，I 以 9 C web / 览器对其 x 行访问。(议您打 * { 为 "ECLReference.html" D文件。
\IBMCS\SDK\JAVA\ECL\SAMPLES*.*	样 > L 序，我G + 在以后 i \ b 些文件。
\IBMCS\jre*.*	Java 运行 1 环 3，| 和 2 装在服务器O D ECL 文件兼容。

在通E服务器 NT f 的 32 位 Windows 客户机O 2 0 ECL

此图 j v J C Z 通信服务器 Windows NT 和 Windows 95 SNA API M户机。

如果 ECL G 4 U # 规或定制D M户机 2 装选项 2 装在M户机O D，则 ecl101.jar + 和 Java 运行 1 环 3 一起 2 装在 CSNT M户机目 <(如 CNSTAPI)中。 b + 9 ECL Java & C L 序能够访问文件 ecl101.jar 中D ECL Java 类。ECL 自己 " ；G一 v 完 { D& C L 序。 ` 写 Java & C L 序 1 X须 9 C ECL Java 类来执行期望D功能集。 2 装 ECL M户机 + 为运行C户 ` 写D ECL Java & C L 序 a 供K y 需D功能级。在服务器O ；X m 外再 2 装 ECL 代 k K。

I Z \ = 文件大小D限制，在 ecl101.jar 文件中只 \| 含K " o 代 k 页。其 \| D代 k 页转换器类 I 以从 2 装在服务器O D zip 文件(ecl101n.zip)中获C。完 { D ECL 文 5、样 > Java 小& C L 序和 Java & C L 序以及C ECL 来运行 Java 小& C L 序D能&也 I 以 2 装在服务器O。

h置 Classpath

1运行一v Java &C L序或 Java 小&C L序1， k +环3 d ？ **classpath** h置I & C L序或小&C L序需要运行D Java 类y在D完{ 7 6 {。例如，若 ECL Java &C L序；`写" 4制在 SNA API M户机子目<(例如 C:\CSNTAPI)，则:

- **classpath** & 1 h I:

  C:\CSNTAPI;C:\CSNTAPI\ecl101.jar

- | n行&C为:

  set classpath=C:\CSNTAPI;C:\CSNTAPI\ecl101.jar

如果您9 C D G Java 运行1环3 (JRE)，则；9 C **classpath** 环3 d ？，+在w C JRE 1， I以9 C **cp** 选项来指定 Java 类D 7 6。

ECL 代k 3 *换器

ECL 代k页转换器支V多种o言。I Z \要文件大小D限制，在文件 ecl101.jar 和 ecl101.zip 中只|含K " o代k页。其|D代k页转换器类I以从2装在服务器O D文件 ecl101n.zip 中获C。+此文件b *后， I在其中R =代k页转换器文件。I以Q b些文件4制=运行 Java &C L序D机器O，也I以+ |G和 ecl101.jar/zip 中D文件放在一起压u I一v新D jar/zip 文件。# t文件y在D Classpath (com\ibm...) 1 k多加小心。

为K减小 ECL &C L序或小&C L序D大小，&C只4制那些&C L序或小&C L序y需D转换器。关Z代k页转换器类D 9 C信息+在 ECL 文5中h v。

ECL 样本

下P ？分+V[2装在 **IBMCS\SDK\JAVA\ECL\SAMPLES** 子目<中D Host Launchpad 样>和其|一些样>。

Host Launchpad 样本

2装在 **IBMCS\SDK\JAVA\ECL\SAMPLES\LAUNCHPAD** 中D Host Launchpad 样> 9 C ECL ，S至主机、注a、访问主机&C L序和信息，然后注销。此I样既I以作为一v小&C L序来运行，也I以作为一v &C L序来运行。小&C L序I以在{为 HostLP.htm D HTML 文件中运行。若要在 HTML 文件中运行，您还X须在 HostLP.htm 中m加下P内容:

```
<APPLET archive=ecl101.jar  code=HostLaunchPad.class
    id=HostLP
    width=400
    height=10 >
</APPLET>
```

同样，在 Win32 系统中，classpath 也& 1 h置I:

```
set classpath=%classpath%;x:\pathto\ecl101.jar;
```

下P源文件+构I样>"R在9 C |G之0 X须-过`译:

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

HostLaunchPad G主类。

若要K b 样 > 代 k， k 启动 HostLaunchPad.java，然后依次启动 LaunchBase.java、LogonLaunchApp.java、SAMsg10.java、SALogon.java、SAVMLogin、SAMore、SAReady.java 和 ScreenAction.java。

b v样 > e 现K大多 } 小& C L 序 9 C主机存储类 b (ECL)以获C主机访问D关键E念。Launchpad 样 > + 执行下P任务:

1. (" 一 v 主机会话。 b + | 括构造一 v ECLSession 5 例以及 w C StartCommunication() 方 = 。此 _ 辑在 LogonLaunchApp.java 中。
2. 为 m > U 间(PS) B 件注 a 。无 [屏幕何 1 | 新，都 + z I PS B件。9 C RegisterPSEvent() 方 = 完 I 注 a 。此 _ 辑在 SALogon.java 中。
3. 再次执行 PS B件。 | 括: 6 p 屏幕 " 指引 | G 9 C击键、主机功能(如 Enter 键或功能键)和移动光 j 。 I 以在 NotifyEvent()、SAReady.java、SAMore.java、SAMsg10.java、SAFilelResults.java 和 ScreenAction.java 中R = 屏幕处理 _ 辑。
4. 5 现执行% v VM | n ('filel') " 在对话 r 中显 > a 果D _ 辑。5 现文件P m | n 和显 > a 果 D 文件在 FileLaunchApp.java、SAFilel.java、SAFileResults.java、ScreenAction.java 和 FilelResultsApp.java 中。

": b v样 > + 处理一些屏幕， b 些屏幕 I 能在；同系统中存在一定D n 异。您 I 能需要对代 k 作某些细小D | D以 J 合您X定D系统。屏幕 6 p 和击键 _ 辑在文件 SAVMLogin.java、SAReady.java、 SAMore.java、SAMsg10.java 和 SAFilelResults.java 中， b 里D ? v 文件 + 处理一 v ；同D屏幕。

若要w T C样 >，您 I 能需要K b 在屏幕O 5 际发 z D B i 。W v f e O D 4 选 r 允许您启C PS Debugger，通过w种主机屏在屏幕中显 > 小& C L 序D x 度。

其 | 样本

在 **IBMCS\SDK\JAVA\ECL\SAMPLES\MISC** 子目 < 中装 P m 一些样 > 。 b 些样 > 显 > K 许多 ECL 类和方 = D 基 > C 法。 y P b 些样 > 都 5 现 K ECLAppletInterface, y 以 I 以在 9 C “运行小 & C L 序” 功能 D Host On-Demand 中运行 | G。如果 ; 想 9 C Host On-Demand 运行 b 些样 >, 您 + 需要 (" 一 v ECLSession 对象。 I 以 9 C 和在 Host Launchpad 样 > 中 (" ECLSession 对象相同 D 方法做 = b 些。

第21章 使C CPIC-C Java f

此图 j v J C Z通信服务器 Windows 95 和 Windows NT SNA API M户机。

> B + h v通信D公共L序h计S Z(CPI-C) Java f API 及其C法, |括:

- 对 CPI-C Java f D简要E v
- 为 CPI-C Java f 2装K 2 4
- 哪些样> I C, | G G如何工作D

何谓 CPI-C Java f?

CPI-C Java f G一v L序h计工_箱, |允许*发_在 Java o言中9 C通信D公共L序h计S Z(CPI-C)。CPI-C G SNA LU 6.2 D一v *放 API。k N < |含在 IBM eNetwork 通信f > 6.0 Windows NT f中D*公共L序h计S Z通信 CPI-C N <大全*(SC26-4399), 以获C P关 CPI-C API D详细内容。

工_箱D主要目D在Z能方c X Q传统D C 转移= Java。因此, 工_箱D w C k在 C o言中9 C D那些w C .分相F。 CPI-C Java f G作为原< D CPI-C API O D一v c次y a供D, 而原来D b v代k X须为 CPI-C 能工作而妥F 2装。

工_箱向L序员a供K工_箱中P关? v类、方=和d ? D N < 文5。 b些文5都G HTML q =, " R a供K; f引C 9 C 9 C起来|加方c。

b v L序h计工_箱还a供K一v带P对象D Java 类集合来# V CPI-C N }以及一v **CPIC** 类, C类定义K C C o言3 d至 CPI-C 函}方法。您I以运行工_箱中D样> L序(JPing.class), 也I以自己`写。

CPI-C Java f * S允许 Java &C L序9 C SNA 网g " + CPI-C S为一v ,网 API 来9 C。b些 Java &C L序I以, S D伙i P:

- 新D CPI-C Java f &C L序
- 新D或现P D非 Java CPI-C &C L序
- 新D或现P D APPC &C L序

2 0 CPI-C Java f

9 C CPI-C Java f工_箱2装下P内容:

- CPICJAVA.JAR |含K`写 CPI-C Java f L序1 9 C D Java 类。b v JAR 文件X须|含在C户D CLASSPATH 环3 d ?中或G在w C一v CPI-C Java f &C L序1显=X指定。b v文件和其|D API 文件一同2装在C户D工作> O。JAR 文件还|含K一v样> &C L序 JPing.class。
- CPICJAVA.DLL G一v X定Z平(D DLL, |在 CPI-C Java f D类和2装在C户工作> O D原< LU 6.2 支V之间a供4 S。b v文件和其|D API 文件一同2装在工作> O。

- Jcpic001.htm G L 序员N < 文 5 D Z 一 v 文件, | 显 > K ? v CPI-C Java f D类、方 = 和 d ? 。 | 和主机访问类 b (ECL)Java f 同 1 2 装在 **IBMCS\SDK\JAVA\CPIC\DOC** 子目 < 中。此文 5 C Z * 发C户& C L 序。
- CPICJAVA.HTM 简要 i \ K工 _ 箱和样 > & C L 序。 HTML q = D文件和其 | D API M户文件一同 2 装在C户工作 > O。
- JPing.java G JPing.class 样 > & C L 序D源文件。文件中D注M x v K P 关 9 C工 _ 箱 h 计 L 序O D一些 a > 和 5 w。 JPing.java 文件在 2 装 ECL Java f 1 2 装。

CPI-C Java f 样本

下 f + h v CPI-C Java f DM户机和服务器样 > 。

客户机样本

工 _ 箱中D样 > 和 APING M户机 5 C L 序 y 执行D功能相同。 | + }] 发M至一 v 服务器 x L, C x L Q }] 回显至 APING 5 C L 序。样 > M户机已 ` 译 " 已放在K CPICJAVA.JAR 文件中。源文件(JPing.java)在 2 装ECL Java f 1 2 装在 **IBMCS\SDK\JAVA\CPIC\SAMPLES** 子目 < 中。

API 作为一 v { 为 COM.ibm.eNetwork.cpic D Java | 而 a 供。在下 f D样 > 中, 为K访问 f 工 _ 箱 a 供D类, X须 | 含下 P 样 > D Z 一行代 k 。**CPIC** 类G至原 < CPI-C 代 k D主要 S Z。**CPIC**类 | 含K许多在 CPI-C 中定义D # }, H如一 v 对话 ID D $ 度, 以及传M至原 < D CPI-C w C D方 = 。

在 ? v 类中只能一 v **CPIC** 对象。 Java + 装入动, 4 S b (DLL), 1 例 > **CPIC** 对象 1, b 中 + | 含原 < 方 = (CPICJAVA.DLL)。

以下D样 > + h v CPI-C 管线; | ; 4 制 JPing.java 源文件中D信息。

": 下 P 样 > | 含 P e 入K一些注M D 代 k 。

```
/*--------------------------------------------------------------
 * Pipeline transaction, client side.
 *--------------------------------------------------------------*/
import COM.ibm.eNetwork.cpic.*;
public class Pipe extends Object {
   public static void main(String args[]) {

   // Make a CPIC object
   CPIC cpic_obj = new CPIC();
```

? 一类N } 都 P 自己 D 类, 而 b 些类都和定义为类 d ? D # } 相关 * 。例如, CPICReturnCode 类含 P 一 v 定义过D I 功返回 k CM_OK。

P = v 主要原因 9 ? 一类N } 含 P 一 v 类。因为 Java 通过值来传M y P D N }, ; 存在C简%类型, 如 { }, 来返回 }] D方法。如果我G Q一 v 对象作为N } 传M x 一 v 方 =, 则C方 = + 在那 v 对象中 h 置一 v d ?, 从而 + }] 返回至w C _ 。此外, 对象D 9 C + 在K b 那些 # } D对象中封装 # } 。 b G一种 j 准D信息隐X技 u 。

```
   // Return Code
   CPICReturnCode cpic_return_code =
      new CPICReturnCode(CPICReturnCode.CM_OK);

   // Request to send received?
```

```
CPICControlInformationReceived rts_received =
   new CPICControlInformationReceived(
      CPICControlInformationReceived.CM_NO_CONTROL_INFO_RECEIVED);
```

CPI-C 发M函 } H待获C一 v C o言缓 e x，也MG 5， H待分配 = 非X定类型D U间。和 C o言；同D G，Java ；P分配隐 = 内存D功能。；同Z原 o D G，Java 中D任何东西都G一 v 对象。无 [L序发MD G 2 4，都X须从其对象类型转换 I 一 v C o言类型D字 Z } 组。

Java a 供K促 I b 些转换D方 = 。例如，Java I 以Q一 v 字符串转换 I 一 v Java D字 Z } 组。 1 一 v 字 Z } 组G一 v Java 对象 1， Java 允许您C原 < 方 = a 取字 Z } 组中 D }] 。

```
// String to Send
  String sendThis = "Test of the PipeLine Transaction";

  // Length of String to send
  CPICLength send_length = new CPICLength(sendThis.length());

  // Convert String to send to a Java array of bytes
  byte[] stringBytes = new byte[ send_length.intValue()];
  sendThis.getBytes(0,send_length.intValue(),stringBytes,0);
```

如同缓 e x 处理，CPI-C 原 < w C H待一 v C 字符串而非 Java 字符串D符号目D X { F。工 _ 箱 4 需自动X + | G从 Java 字符串转换 I C 字符串。通 #， 1 工 _ 箱H待一 v X定 Java 类型 1 I 能 x 行自动D转换。

对话 ID G一 v Java 字 Z } 组， b v } 组 I 工 _ 箱自动X + 其转换为一 v | 含K简% 字 Z i D C } 组。

```
  // this hardcoded sym_dest_name must
  // be 8 chars long & blank padded
  String sym_dest_name = "PIPE    ";

  // Space to hold a conversation ID
  // (which is just a bunch of bytes)
  byte[] conversation_ID = new byte[CPIC.CM_CID_SIZE];
```

L序 * < 发 v CPI-C w C， C w C和 C o言中 9 C D非 # 类 F。 + G，此方 = w C G以 CPI-C 对象为 0 缀， " R N } ；以传] 引C(&)符为 0 缀D。

```
 //
  // Initialize CPI-C
  //
  cpic_obj.cminit(             /* Initialize_Conversation        */
            conversation_ID,   /*  O: returned conversation ID   */
            sym_dest_name,     /*  I: symbolic destination name  */
          cpic_return_code);   /*  O: return code from this call */
  //
  // ALLOCATE
  //
  cpic_obj.cmallc(             /* Allocate Conversation          */
            conversation_ID,   /*  I: conversation ID            */
            cpic_return_code); /*  O: return code from this call */
  //
  // SEND
  //
  cpic_obj.cmsend(             /* Send_Data                      */
          conversation_ID,     /*  I: conversation ID            */
            stringBytes,       /*  I: send this buffer           */
```

```
            send_length,       /*  I: length to send             */
            rts_received,      /*  O: was RTS received?          */
          cpic_return_code);   /*  O: return code from this call */
  //
  // DEALLOCATE
  //
  cpic_obj.cmdeal(             /* Deallocate                     */
          conversation_ID,     /*  I: conversation ID            */
          cpic_return_code);   /*  O: return code from this call */
  } // end main method
} // end the class
```

服务器样本

服务器u < 化自己、 S \ 一v对话、 S U }] " 打！ v o 断信息。如同在M户机中，我G 5 例化类以容纳 CPI-C N }，其中许多N } v P 一v { }作为5 例 }]。通过9 C 对象，我G I 以通过引C来模拟w C。我G还分配一v 字Z } 组C Z 存放S U = D }]。

": 下P 样 > | 括K代k " e 入K一些注M。

```
/*-------------------------------------------------------------
 * Pipeline transaction, server side.
 *-------------------------------------------------------------*/
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();

    // Space to hold the received data
    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);

    // Space to hold a conversation ID -- a bunch of bytes
    // The first line declares conversation_ID to be a reference to
    // a byte array object.  The second line creates such an object,
    // and assigns the reference to the byte array object.
    byte[] conversation_ID;
    conversation_ID = new byte[cpic_obj.CM_CID_SIZE];
```

1管@B务H待一v字符串1，CPI-C S U w C(cmrcv)返回一v Java 字Z。L序员通过9 C字符串类-构造器(Q字Z } 组S为一v d ?)+ C字Z } 组翻译I 一v字符串。

```
    //
    // ACCEPT
    //
```

```
   cpic_obj.cmaccp(         /* Accept_Conversation            */
       conversation_ID,     /*  O: returned conversation ID   */
       cpic_return_code);   /*  O: return code                */
   //
   // RECEIVE
   //
   cpic_obj.cmrcv(          /* Receive                        */
       conversation_ID,     /*  I: conversation ID            */
       data_buffer,         /*  I: where to put received data */
       requested_length,    /*  I: maximum length to receive  */
       data_received,       /*  O: data complete or not?      */
       received_length,     /*  O: length of received data    */
       status_received,     /*  O: has status changed?        */
       rts_received,        /*  O: was RTS received?          */
       cpic_return_code);   /*  O: return code from this call */
  //
  // Do some return code processing
  //
  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());
  // Create a Java String from the array of bytes that you received
  // and print it out.
  String receivedString = new String(data_buffer,0);
  System.out.println(
     "    Recevied string                 = "
     + receivedString );

  //
  // BLOCK so that the Server Window doesn't disappear
  //
  try{
       System.out.println("Press any key to continue");
       System.in.read();
     }
  catch
     (IOException e){ e.printStackTrace(); }
  }
}
```

附 < A. APPC 公共返回 k

> = < h v 几 v APPC 动词公 C D 主（而 R，若合 J，则 G 次）返回 k。

X 定 Z 动词 D 返回 k 在 v p 动词 D 文 5 中 h v。

AP_ALLOCATION_ERROR

v 人通信 k 通信服务器分配对话 ' \。对话状，h 置为 RESET。I 以通过在 **ALLOCATE** 或 **MC_ALLOCATE** 之后发 v D 一 v 动词返回 C 代 k。相关 D 次返回 k 如下:

AP_ALLOCATION_FAILURE_NO_RETRY

I Z 一 v @ C u 件（如配置错误或会话协议错误），无法分配对话。为确定 C 错误，系统管理员 & 检 i 错误日志文件。在 D } 错误之 0 k 勿 " T 重 T 分配。

AP_ALLOCATION_FAILURE_RETRY

I Z 一 v Y 1 u 件（如 4 S ' \），无法分配对话。' \ D 原因记 < 在系统错误日志中。最好在，1 后重 T 分配以允许 e } C u 件。

AP_SECURITY_NOT_VALID

伙 i LU ；P S \ 分配 k s 中指定 D C 户 ID 或 Z n。

AP_TRANS_PGM_NOT_AVAIL_RETRY

远 L LU 因为无法启动 \ k s D 伙 i B 务 L 序而 \ x 分配 k s。I Z 一 v 2 1 u 件（如，1），\ k s D B 务 L 序（TP）；I C。v 错 I 能 G I Z 在远 L Z c O 注 a。无需 Y 作员 i 入 C u 件即 I 自己 e }。B 务 L 序最好在，1 后重 T 对话以允许 e } C u 件。

AP_TRANS_PGM_NOT_AVAIL_NO_RTRY

远 L LU 因为无法启动 \ k s D 伙 i B 务 L 序而 \ x 分配 k s。I Z 一 v @ C 或 k @ C u 件，\ k s D B 务 L 序（TP）；I C。v 错 I 能 G I Z 在远 L Z c O 注 a。；P Y 作员 i 入 C u 件无法自己 e }。在 e } 错误 u 件之 0，B 务 L 序；& C 重 T 对话。

AP_TP_NAME_NOT_RECOGNIZED

伙 i LU ；认 6 分配 k s 中指定 D B 务 L 序 { 。

AP_PIP_NOT_ALLOWED

\ k s D B 务 L 序；能 S U L 序 u < 化 N }（PIP）。b m w > X 和伙 i B 务 L 序之间；匹配。

AP_PIP_NOT_SPECIFIED_CORRECTLY

\ k s D B 务 L 序 I 以 S U L 序 u < 化 N }（PIP），+ 在 y a 供 D PIP 中检 b = 一 v 错误。b m w > X 和伙 i B 务 L 序之间；匹配。

AP_CONVERSATION_TYPE_MISMATCH

\ k s D B 务 L 序无法支 V 分配 k s 中指定 D b 种类型（基 > 或 3 d D）D 对话。b m w > X 和伙 i B 务 L 序之间；匹配。

AP_SYNC_LEVEL_NOT_SUPPORTED

\ k s D B 务 L 序无法支 V 分配 k s 中指定 D _ P **sync_level** (AP_NONE, AP_CONFIRM_SYNC_LEVEL 或 AP_SYNCPT) D 对话。b m w > X 和伙 i B 务 L 序之间；匹配。

AP_CANCELLED

I Z取消K对话（B务L序发v一v **CANCEL_CONVERSATION** 动词）动词返回。

AP_CONV_FAILURE_NO_RETRY

I Z一v@C u件（如会话协议错误）而终止对话。系统管理员&检i系统错误日志文件以确定发z错误D原因。在D }错误之0 k勿" T C对话。

AP_CONV_FAILURE_RETRY

I Z一v Y 1错误而终止对话。重新启动B务L序以i 4 G否再次发z C问b。如果发z，系统管理员&检i错误日志以确定发z错误D原因。

AP_CONVERSATION_TYPE_MIXED

B务L序已" T混合同一对话中；同对话类型D对话动词。例如，B务L序发v一v后f带P **CONFIRM** 动词D **MC_ALLOCATE** 动词。

AP_DEALLOC_ABEND

I Z以下某v原因已+对话b }分配。

- 伙i B务L序向 AP_ABEND 发v _ P **dealloc_type** 集D **MC_DEALLOCATE** 动词。
- 伙i B务L序异#终止，9伙i LU 发v一v **MC_DEALLOCATE** k s。

AP_DEALLOC_ABEND_PROG

I Z以下某v原因已b }K对话D分配。

- 伙i B务L序向 AP_ABEND_PROG 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词。
- 伙i B务L序异#终止，9伙i LU 发v一v **DEALLOCATE** k s。

AP_DEALLOC_ABEND_SVC

I Z伙i B务L序向 AP_ABEND_SVC 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词，y以对话已b }分配。

AP_DEALLOC_ABEND_TIMER

I Z伙i B务L序向 AP_ABEND_TIMER 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词，y以对话已b }分配。

AP_DEALLOC_NORMAL

C返回k；指>一v错误。伙i B务L序向下P某v值发v _ P **dealloc_type** 集D **DEALLOCATE** 或 **MC_DEALLOCATE** 动词。

- AP_FLUSH
- AP_SYNC_LEVEL _ P指定为 AP_NONE D对话同=级

AP_DUPLEX_TYPE_MIXED

B务L序已" T发v一v _ P；同对话 **duplex_type** D对话动词。例如，B务L序在全+工对话中发v一v k +工D **MC_FLUSH** 动词（**opext** 中；P **AP_FULL_DUPLEX_CONVERSATION** 集）。

AP_ERROR_INDICATION

C返回k v C Z全+工对话。因为伙i B务L序已终止对话，y以发M队P Y作' \。如果对话状，为v发M，则C对话现已a x。如果对话状，为发M-S U或v S U，则C对话+在J 1 D返回k返回x S U队P动词1 a x。相关D次返回k为:

AP_ALLOCATION_ERROR_PENDING

远L LU \ x分配k s。

AP_DEALLOC_ABEND_PROG_PENDING

I Z以下某v原因已+对话b }分配;

- 伙i B务L序向 AP_ABEND_PROG 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词。
- 伙i B务L序异#终止9伙i LU 发v一v **DEALLOCATE** k s。

AP_DEALLOC_ABEND_SVC_PENDING

I Z伙i B务L序向 AP_ABEND_SVC 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词, y以对话已b }分配。

AP_DEALLOC_ABEND_TIMER_PENDING

I Z伙i B务L序向 AP_ABEND_TIMER 发v _ P **dealloc_type** 集D **DEALLOCATE** 动词, y以对话已b }分配。

AP_UNKNOWN_ERROR_TYPE_PENDING

对话已I伙i B务L序b }分配, +>X LU ;知@原因。

AP_OPERATION_INCOMPLETE

B务L序发v一v启动处理D;阻塞D动词, +; P完I。动词处理完I后+h置最后D返回k, 而代k存y会通知C B务L序。

AP_PROG_ERROR_NO_TRUNC

1对话处Z SEND 状, 1, 伙i B务L序发v下P某v动词。

- = AP_PROG D _ P **err_type** 集D **SEND_ERROR**
- **MC_SEND_ERROR**

;X断}]。

AP_PROG_ERROR_PURGING

1处Z RECEIVE、PENDING_POST、CONFIRM、 CONFIRM_SEND 或 CONFIRM_DEALLOCATE 状, 1, 伙i B务L序发v下P某v动词。

- = AP_PROG D _ P **err_type** 集D **SEND_ERROR**。
- **MC_SEND_ERROR**

e }K已发v+还;U=D }]。

AP_PROG_ERROR_TRUNC

在 SEND 状,中, 发MK一v ;完{ D _辑记<后, 伙i B务L序向 AP_PROG 发v _ P **err_type** 集D **SEND_ERROR** 动词。>X B务L序I能已通过 **RECEIVE** 动词S U = _辑记<D Z一?分。

AP_SVC_ERROR_NO_TRUNC

1处Z SEND 状, 1, 伙i B务L序(或伙i LU)向 AP_SVC 发v一v _ P **err_type** 集D **SEND_ERROR** 动词。;X断}]。

AP_SVC_ERROR_PURGING

1处Z RECEIVE、PENDING_POST、CONFIRM、CONFIRM_SEND 或 CONFIRM_DEALLOCATE 状, 1, 伙i B务L序(或伙i LU)向 AP_SVC 发v一v _ P **err_type** 集D **SEND_ERROR** 动词。发M=伙i B务L序D }] I能已e }。

AP_SVC_ERROR_TRUNC

在 SEND 状，，发MK一v；完{D_辑记<后，伙iB务L序（或伙iLU）发v一v **SEND_ERROR** 动词。>XB务L序I能已SU=_辑记<DZ一?分。

AP_TP_BUSY

1v人通信k通信服务器}处理同一对话Dm一v动词1，>XB务L序向v人通信k通信服务器发v一v阻塞D动词。

AP_UNEXPECTED_SYSTEM_ERROR

v人通信k通信服务器v=一v意外D系统错误，"R无法完IC动词。通#b些错误I系统资源（例如，内存）D短缺引起，而R通#G21D。k检i系统日志，以获C|详细Div。

AP_SEC_REQUESTED_NOT_SUPPORTED

因为k伙i LU D会话；支VZnf代，y以>X LU 无法分配对话。ALLOCATE 或 SEND_CONVERSATION 中ksD2全性类型GAP_PGM_STRONG，|需要Znf代支V。

附 < B. LUA 动词返回 k

> = < + h v 多 v SLI 动词能通 C D 主（以及 X 要 D 次级）返回 k。

X 定 D 动词返回 k + 在 v p 动词 D 文 5 中 h v。

主返回 k

下 P ? 分 | 含 K LUA D 主返回 k:

LUA_OK

LUA 动词 I 功完 I。

LUA_PARAMETER_CHECK

LUA 功能检 b = 一 v ; } 确 D N }。

LUA_STATE_CHECK

会话处在 y 发 v 动词 D ; } 确状，中。

LUA_SESSION_FAILURE

会话已关 U。在次级返回 k 中 j 6 K X 定原因。

LUA_UNSUCCESSFUL

此动词 ; P I 功完 I。

LUA_NEGATIVE_RESPONSE

+ z z 下 P u 件之一:

- = 达 K 一 u 4 D 4 a x, C 4 G I LUA & C L 序否定响 & D。未 h 置次级返回 k。
- LUA 在来自主 LU D 消息中检 b = 一处错误, " 发 M 否定响 &。1 从主 LU S U = 4 a x 1 返回 C 错误。次级返回 k | 含 k 否定 & 答一起发 M D 检 b }]。

LUA_CANCELED

b v 动词 + 因为次级返回 k 中 D 指定原因而 ; 取消。

LUA_IN_PROGRESS

此同 = 代 k + 在 S U = 一 v 同 = | n + | n 未完 I 1 返回。

LUA_STATUS

SLI 含 P 次级返回 k 中 & C L 序 D 状，信息。

LUA_COMM_SUBSYSTEM_ABENDED

通信服务器异 # 终止。

LUA_COMM_SUBSYSTEM_NOT_LOADED

未装入通信服务器。

LUA_INVALID_VERB_SEGMENT

LUA 无法处理动词, 因为 { v X 制 i 未 | 含在 }] 段中。动词 X 制 i 末端 D X 址，v K 段末端。

LUA_UNEXPECTED_DOS_ERROR

通信服务器发v一v系统wC后，发zK一处未预O=D系统错误，动词k主返回k UNEXPECTED_DOS_ERROR 一起G记。次级返回k|含K未预O=D系统错误。

LUA_STACK_TOO_SMALL

LUA &CL序堆；对 LUA 来5+小以至Z无法处理k s。

LUA_INVALID_VERB

LUA ；能6p|ySU=D动词X制i中D动词代k或动词Y作k（或=_）。

次6返回k

下P?分|含K LUA D次级返回k:

LUA_SEC_OK

I以9Ck次级返回k关*D主返回kD=加信息。

LUA_INVALID_LUNAME

此动词指定K一v无效D **lua_name**。

LUA_BAD_SESSION_ID

此动词X制i为 **lua_sid** N}指定K一v；}确D值。

LUA_DATA_TRUNCATED

缓exS度（如 **lua_max_length** 中指定）对ZySU=D}]而言+短，故X断}]。

LUA_BAD_DATA_PTR

|n要sa供或返回}]，然而， **lua_data_ptr** N}或|含一v无效指k，或未指向一v读/写段。

LUA_DATA_SEG_LENGTH_ERROR

+zz下Pu件之一:

- 在 **RUI_READ** 或 **SLI_RECEIVE** 动词Oa供D}]段D$度H **lua_max_length** N}中a供D短。
- 在 **RUI_WRITE** 或 **SLI_SEND** 动词Oa供D}]段D$度H **lua_data_length** N}中a供D短。
- 在 **RUI_READ**、 **RUI_WRITE**、**SLI_RECEIVE** 或 **SLI_SEND** 动词Oa供D}]段；G读/写}]段。

LUA_RESERVED_FIELD_NOT_ZERO

UU发vD|n中含P一v非cD#tN}。

LUA_INVALID_POST_HANDLE

P效D信号?未在 LUA 动词X制i内指定。1 LUA 动词未同=完I1，需要一v信号?发v信号，m>动词完I。

LUA_PURGED

取消 **RUI_READ** 或 **SLI_RECEIVE** 动词，原因G发vK **RUI_PURGE** 或 **SLI_PURGE**。

LUA_BID_VERB_SEG_ERROR

带P **SLI_BID** 动词X制iD缓ex在 **lua_flag1.bid_enable** h置为1D **SLI_RECEIVE** 发v之0M放。

LUA_NO_PREVIOUS_BID_ENABLED

在带P **lua_flag1.bid_enable** D **RUI_READ** 或 **SLI_RECEIVE** 动词发v之0，+；发v **RUI_BID** 或 **SLI_BID** 动词。

LUA_NO_DATA

RUI_READ 或 **SLI_RECEIVE** 动词带着 NO_WAIT N}发v，"R无I读}]。

LUA_BID_ALREADY_ENABLED

在带P **lua_flag1.bid_enable** D **RUI_READ** 或 **SLI_RECEIVE** 动词发v之0，**RUI_BID** 或 **SLI_BID** 动词G活动D。

LUA_VERB_RECORD_SPANS_SEGMENTS

LUA 动词X制i|含一v$度N}，1+其m加=段D偏移？1，|会，v段D末尾。

LUA_INVALID_FLOW

发v LUA 动词，带P h置错误D **lua_flag1** w j志。检i **lua_flag1** w j志D}确}？G否如下y>:

- 对Z **RUI_READ** 或 **SLI_RECEIVE**，至Y一v
- 对Z **RUI_WRITE**，v P一v
- 对Z **SLI_SEND**，1发M SNA 响&1X须v h置一v **lua_flag1** w动j志。

LUA_NOT_ACTIVE

一v&C L序，|1 LUA 在通信服务器内；活动1发v一v LUA 动词。

LUA_VERB_LENGTH_INVALID

发v一v动词，|带P；}确D **lua_verb_length** N}。指定D$度k LUA 期望D；相H。

LUA_REQUIRED_FIELD_MISSING

发v D **RUI_WRITE** 动词或；|括}]指k（若}]计}值非c），或_；|括 **lua_flag1** w动j志。

LUA_READY

SLI 会话现在准8处理=加|n。C状，在S U=0 f D NOT_READY 状，之后，或在带P主返回k CANCELED 和次级返回k RECEIVE_UNBIND_HOLD 或 RECEIVED_UNBIND_NORMAL D **SLI_CLOSE** 动词完I后发v。

LUA_NOT_READY

SLI 会话I Z下P原因之一暂挂:

- S U= CLEAR |n。S U= SDT |n 1继续 SLI 会话。
- S U= UNBIND |n。S U= BIND、I选D STSN k SDT |n之0，会话暂停。再次w C I原< **SLI_OPEN** 动词a供D任何C户扩9例行L序，因此，b些例行L序X须为I重入D。SLI 处理K SDT |n后，SLI 会话继续。=类 UNBIND |n G:
  - UNBIND 类型 X'02'，m>P新 BIND x入
  - UNBIND 类型 X'01'，m>&C L序在启动>会话D **SLI_OPEN** 动词中指定K一v LUA_SESSION_TYPE_DEDICATED D **lua_session_type**。

LUA_INIT_COMPLETE

LUA S Z在 **SLI_OPEN** }在处理D同1 u<化会话D 1侯，>状，为 LUA

&C L序在 **SLI_RECEIVE** 或 **SLI_BID** 动词O返回， C LUA &C L序发v带P **LUA_INIT_TYPE_PRIM_SSCP** N } D **SLI_OPEN**。

LUA_SESSION_END_REQUESTED

SLI 从主机S U = SHUTD | n，指>主机已准8关U会话。

LUA_NO_SLI_SESSION

会话未打*或I Z一v **SLI_CLOSE** 动词或会话' \而}在关U会话1，发v一v | n。在处理 **SLI_OPEN** 动词期间发v D **SLI_RECEIVE** 或 **SLI_SEND** 动词返回>代k， 1:

- **SLI_OPEN lua_init_type** N }未h置为 LUA_INIT_TYPE_PRIM_SSCP。一v SLI_BID 动词还在b些i v下返回>代k。
- **SLI_RECEIVE** 或 **SLI_SEND lua_flag1** N }；指定 **lua_flag1.sscp_norm**。

在S U = UNBIND 类型 X'02' | n或 UNBIND 类型X'01'（LUA_SESSION_TYPE_DEDICATED）之后，以及 SDT | n处理之0，SLI 组件处Z **SLI_OPEN** 处理之中。 UNBIND 型 X'02' 指>新 BIND }在x入。

LUA_SESSION_ALREADY_OPEN

SLI_OPEN 动词为一v LU {而发v， C LU {已P一v打*D会话。

LUA_INVALID_OPEN_INIT_TYPE

SLI_OPEN 动词在 lua_init_type N }中含P一v；}确值。

LUA_INVALID_OPEN_DATA

SLI_OPEN 动词带P lua_init_type N }发v， C N }G为K k INITSELF（LUA_INIT_TYPE_SEC_IS）次级u <化而h置D， " R }]缓e x；|含P效D INITSELF | n。

LUA_UNEXPECTED_SNA_SEQUENCE

在 **SLI_OPEN** 处理期间，从主机S U =未预O = D | n或}]。

LUA_NEG_RSP_FROM_BIND_ROUTINE

C户a供D **SLI_BID** 例行L序对 BIND z I一v否定响&。 **SLI_OPEN** 动词未I功a x。

LUA_NEG_RSP_FROM_CRV_ROUTINE

C户a供D **SLI_BID** 例行L序对 BIND z I一v否定响&。 **SLI_OPEN** 动词未I功a x。

LUA_NEG_RSP_FROM_STSN_ROUTINE

C户a供D SLI STSN 例行L序对 STSN 否定响&。 **SLI_OPEN** 未I功a x。

LUA_CRV_ROUTINE_REQUIRED

C户；a供 SLI CRV 例行L序，+从主机S U = CRV。 SLI 对 CRV 发v一v否定响&， " R **SLI_OPEN** 动词在此1未I功完I。

LUA_NEG_RSP_FROM_SDT_ROUTINE

C户a供D SLI SDT 例行L序对 SDT z I一v否定响&。 b种i v <致 **SLI_OPEN** 动词a x。

LUA_INVALID_OPEN_ROUTINE_TYPE

在 **SLI_OPEN** 扩 9 例行 L 序 P m中， **lua_open_routine_type** N } ；G P效 D。

LUA_MAX_NUMBER_OF_SENDS

& C L 序在一 v **SLI_SEND** 动词完 I 之 0 发 v = v 以 O C 动词。

LUA_SEND_ON_FLOW_PENDING

& C L 序为 SNA w（加急 SSCP、# 规 SSCP、加急 LU、# 规 LU）发 v **SLI_SEND** 动词， C SNA w已 P 未执行 D **SLI_SEND** 动词。

LUA_INVALID_MESSAGE_TYPE

SLI ；6 p **lua_message_type** N }。

LUA_RECEIVE_ON_FLOW_PENDING

SLI & C L 序已为 SNA w发 v 一 v **SLI_RECEIVE** 动词， C SNA w已 P 一 v 未执行 D **SLI_RECEIVE** 动词。

LUA_DATA_LENGTH_ERROR

发 v **SLI_OPEN** | n需要& C L 序未 a 供 D C 户 }]。次级 u < D **SLI_OPEN** 动词需要 }]， " R 1 & C L 序为 LUSTAT | n发 v **SLI_SEND** 动词 1，需要 4 字 Z D状，。

LUA_CLOSE_PENDING

z z K下 P i v之一:

- 1 CLOSE_NORMAL 或 CLOSE_ABEND } 在挂起 1 发 v CLOSE_NORMAL。
- 1 m一 v CLOSE_ABEND } 在挂起 1 发 v CLOSE_ABEND。唯一发 v m一 v CLOSE_ABEND D原因 G 1 CLOSE_NORMAL } 在挂起 1 。

LUA_NEGATIVE_RSP_CHASE

在 **SLI_CLOSE** 处理期间， SLI 从主机 S U = 对 CHASE | n D否定响&。& **SLI_CLOSE** D k s，会话停止。

LUA_NEGATIVE_RSP_SHUTC

在 **SLI_CLOSE** 处理期间， SLI 从主机 S U = 对 SHUTC | n D否定响&。& **SLI_CLOSE** D k s，会话停止。

LUA_NEGATIVE_RSP_SHUTD

在 **SLI_CLOSE** 处理期间， SLI 从主机 S U = 对 SHUTD | n D否定响&。& **SLI_CLOSE** D k s，会话停止。

LUA_NO_RECEIVE_TO_PURGE

SLI_PURGE 动词在； P 未执行 D **SLI_RECEIVE** 动词 1 发 v。 I 能 D原因 P 如下 y > D = 种:

- **lua_data_ptr** N } 内 | 含 D X址；指向未完 I D **SLI_RECEIVE** 动词， C动词即 + ； e }。
- 1 **SLI_PURGE** 动词 } 在；处理 1， **SLI_RECEIVE** 原& C完 I。 b " 非错误 u 件。 ` 写& C L 序 D代 k，以处理 C i v。

LUA_CANCEL_COMMAND_RECEIVED

处理处理 **SLI_RECEIVE** 动词 1，主机发M一 u CANCEL | n，以取消 } 在 S U D }] 4。

LUA_RUI_WRITE_FAILURE

RUI_WRITE 动词C未预O = D错误G记= SLI。

LUA_INVALID_SESSION_TYPE

SLI_OPEN 动词|含一v值，|在 **lua_session_type** 中G无效D。

LUA_SLI_BID_PENDING

在以0发v D **SLI_BID** 仍然G活动D 1侯发v SLI 动词。同1只能P一v **SLI_BID** I为活动D。 time.

LUA_PURGE _PENDING

在以0发v D **SLI_PURGE** 仍然G活动D 1侯发v **SLI_PURGE** 动词。 同1只能P一v **SLI_PURGE** I以活动。

LUA_PROCEDURE_ERROR

已S U = NSPE 或 NOTIFY 消息，指>已发z主机过L错误。 **SLI_OPEN** 9 C >返回k G记（ } 非9 C K **SLI_OPEN** 动词重T I选项）。 1 **lua_wait** h置为非c值1，重T INITSELF 或 LOGON 信息，直= I 9 C主机过L，或& C L序发v **SLI_CLOSE**。

LUA_INVALID_SLI_ENCR_OPTION

lua_encr_decr_option N } 在 **SLI_OPEN** 动词内h置为 128。SLI ；支V 128位加\或b \处理任选项。

LUA_RECEIVED_UNBIND

1存在一v活动D SLI 会话1，SLI 从主 LU S U = UNBIND | n。SLI 会话停止。

LUA_RECEIVED_UNBIND_HOLD

在主启动或次启动 **SLI_CLOSE** } #处理期间，SLI S U = UNBIND 类型X'02'。类型 X'02' m >新 BIND } 在x入。会话；挂起，直至S U = BIND、I选 CRV 和 STSN 和 SDT | n为止。再次w C I原< **SLI_OPEN** 动词a供D任何C户扩9例行L序， b些例行L序X须为I重入D。 SLI 处理K SDT | n后， SLI 会话继续。

LUA_RECEIVED_UNBIND_NORMAL

在主启动或次启动 **SLI_CLOSE** } #处理期间（处理k指定LUA_SESSION_TYPE_DEDICATED D **lua_session_type** D **SLI_OPEN** 动词一起启动D会话），SLI S U = UNBIND 类型 X'01'。 在S U = BIND、I选D STSN k SDT | n之0，会话暂停。 再次w C I原< **SLI_OPEN** 动词a供D任何C户扩9例行L序， b些例行L序X须为I重入D。 SLI 处理K SDT | n后， SLI 会话继续。

LUA_SLI_LOGIC_ERROR

SLI 检b v一处内？_辑错误。

LUA_TERMINATED

已取消发v **SLI_CLOSE** 或 **RUI_TERM** 动词1暂挂D动词。

LUA_NO_RUI_SESSION

发v RUI 动词，因为一v未9 C **RUI_INIT** u <化D会话，或G 1会话D **RUI_INIT** 动词}在处理1发v K一v非 **RUI_TERM** D其|动词。

在；P活动D RUI 动词未完ＩD同1发zK会话中断1，会Ｐｂｖ返回k。下一ｖ发ｖD动词获ＣＣ返回k。＆ＣＬ序+C返回k作为 SESSION_FAILURE 处理。

LUA_DUPLICATE_RUI_INIT

＆ＣＬ序为一ｖ会话发ｖ **RUI_INIT** 动词，Ｃ会话已ｕ<化或已Ｐ一ｖｘ行中Ｄ **RUI_INIT** 动词。

LUA_INVALID_PROCESS

对已；ｍ一ｖｘＬ５ＰＤ会话发ｖ RUI 动词。

LUA_API_MODE_CHANGE

在Ｉ SLI （"D会话O，+非 SLI ｋｓ发ｖ= RUI。

LUA_COMMAND_COUNT_ERROR

，ｖK发ｖＤ **RUI_READ** 或 **RUI_WRITE** 动词D最大｝?，或在先0发ｖＤ **RUI_BID** 或 **RUI_TERM** 仍在ｘ行中1发ｖK一ｖ **RUI_BID** 或 **RUI_TERM** 动词。

LUA_NO_READ_TO_PURGE

RUI_PURGE 动词在 **RUI_READ** 动词；Ｐ未完Ｉ１发ｖ。Ｉ能D原因Ｐ=ｖ:

- **lua_data_ptr** Ｎ｝内|含ＤＸ址；指向未执行Ｄ **RUI_READ** 动词，Ｃ动词即+；ｅ｝。
- **RUI_READ** 动词在 **RUI_PURGE** 动词}在处理中1完Ｉ。ｂ"非错误ｕ件。`写＆ＣＬ序D代k，以处理Ｃｉｖ。

LUA_MULTPLE_WRITE_FLOWS

在发Ｍｘ **RUI_WRITE** 动词Ｄ FLAG1 中已Ｐ一ｖ已ＯＤｗｊ志；打*Ｋ。

LUA_DUPLICATE_READ_FLOW

＆ＣＬ序对已Ｐ **RUI_READ** 暂挂Ｄｗ发ｖ **RUI_READ**。

LUA_DUPLICATE_WRITE_FLOW

发ｖＤ **RUI_WRITE** 动词|含 FLAG1 ｗｊ志，显>以0 **RUI_WRITE** 动词D会话w未完Ｉ。

LUA_LINK_NOT_STARTED

在会话ｕ<化期间，LUA 无法启动｝］４７。

LUA_INVALID_ADAPTER

DLC Ｊ配器配置；｝确，或配置文件已；ｐ坏。

LUA_ENCR_DECR_LOAD_ERROR

Ｔ图加载Ｃ户ａ供D加\或译ｋ动，４Ｓｂ１，ＳＵ=一处未预Ｏ=Ｄｖ错。

LUA_ENCR_DECR_PROC_ERROR

Ｔ图获ＣＣ户ａ供D加\或译ｋ动，４Ｓｂ内D过ＬＸ址１，ＳＵ=一处未预Ｏ=Ｄｖ错。

LUA_LINK_NOT_STARTED_RETRY

ＩＺ４７无法激活，**RUI_INIT** 或 **SLI_OPEN** 动词'\。Ｃ返回ｋｍ>在伙ｉ位置处或ｋ=ｖ机器间Ｄ，Ｓ发ｚＫ故Ｏ。

LUA_NEG_NOTIFY_RSP

发ｖＤ **RUI_INIT**，ｚｚ+一ｕ通知ｋｓ发M= SSCP，以指> SLU 现在ＩＩ为会话D一?分。SSCP 对C通知ｋｓ作ｖ否定响&。预计D会话端组件Ｋｂｙ支ＶＤｋｓ，+；对其ｘ行处理。

LUA_RUI_LOGIC_ERROR

发 z RUI 内？_辑错误。

LUA_LU_INOPERATIVE

SLI T图停止会话 1 发 z 严重错误。> LU 对 Z 任何 LUA k s 而言都 G ; I C D，直 = 从主机 S U = K ACTLU。

LUA_RESOURCE_NOT_AVAILABLE

指定在 RU 内？D LU、PU、4 7 > 或 4 7 都 G ; I C D。}非 9 C K **SLI_OPEN** 重 T I 选项，否则 **SLI_OPEN** 动词；能 9 C > 返回 k G 记。1 **lua_wait** h 置为非 c 值 1，则重 T INITSELF 或 LOGON 信息，直 = I 9 C 主机过 L，或 & C L 序发 v **SLI_CLOSE** 动词。

LUA_SESSION_LIMIT_EXCEEDED

无法激活 k s D 会话，因为网 g I 访问 % 元（NAU）D 一 v % 元 } 处 Z 其会话限制中，如 LU-LU 会话限制或方 = 会话限制。C 检 b k & C Z ACTCDRM、INIT、BID 以及 CINIT k s。

}非 9 C K **SLI_OPEN** 动词重 T I 选项，否则 **SLI_OPEN** 动词；能 9 C > 返回 k G 记。1 **lua_wait** h 置为非 c 值 1，则重 T INITSELF 或 LOGON 信息，直 = I 9 C 主机过 L，或 & C L 序发 v **SLI_CLOSE** 动词。

LUA_SLU_SESSION_LIMIT_EXCEEDED

若 S \，则 k s + < 致，过 SLU 会话限制。

LUA_MODE_INCONSISTENCY

1 0 状，；允许执行函 }。预计 D 会话端组件 K b y 支 V D k s，+ ；对其 x 行处理。> 代 k 也 I 作为 EXR 中 D 检 b k v 现。

LUA_INSUFFICIENT_RESOURCES

I Z Y 1 性缺 Y 资源，S U 器无法对 k s x 行 Y 作。预计 D 会话端组件 K b y 支 V D k s，+ ；对其 x 行处理。

LUA_RECEIVER_IN_TRANSMIT_MODE

存在：y u 件。1 k + 工 y C 状，为未 S U 1，或 1 处理 } # }] w D 资源（如缓 e x）；I C 1，S U = } # }] w k s。

> 代 k 也 I 作为异 # k s 中 D 检 b k v 现。

LUA_LU_COMPONENT_DISCONNECTED

因为 g 源关 U 或一些其 | 断 * u 件之故，LU 组件；I C。

LUA_NEGOTIABLE_BIND_ERROR

S U = I 协 L D BIND。SLI ；允许 I 协 L D BIND，}非通过 **SLI_OPEN** 动词 a 供 K 一 v C 户 a 供 D **SLI_BIND** 例行 L 序。

LUA_BIND_FM_PROFILE_ERROR

在 BIND O 检 b v ；；支 V D FM E 要文件。SLI v 支 V FM E 要文件 3 k 4。

LUA_BIND_TS_PROFILE_ERROR

在 BIND O 检 b v ；；支 V D TS E 要文件。SLI v 支 V TS E 要文件 3 k 4。

LUA_BIND_LU_TYPE_ERROR

检 b v ；；支 V D LU 类型。LUA v 支 V LU 0、LU 1、LU 2 k LU 3。

LUA_SSCP_LU_SESSION_NOT_ACTIVE

处理 k s X需D SSCP-LU 会话G；活动D。例如，处理一 u INITSELF k s，SSCP ；P 一 v 活动对话 x，｜带 P 在 INITSELF 中｜{ D目 j LU。

字 Z 2 k 3 ｜含检 b k X定 D信息。 下 P h 置 G 允许 D:

0000 未 & C X定 ` k。

0001 重新激活 SSCP-SLU 会话。

0002 SSCP-PLU 会话 G 非活动 D。}非 9 C K **SLI_OPEN** 重 T I 选项，否则 **SLI_OPEN** 动词；能 9 C > 返回 k G记。1 **lua_wait** h 置为非 c 值 1，则重 T INITSELF 或 LOGON 信息，直 = I 9 C 主机过 L，或 & C L 序发 v **SLI_CLOSE** 动词。

0003 激活 SSCP-SLU 会话。

0004 重新激活 SSCP-SLU 会话。

LUA_REC_CORR_TABLE_FULL

会话 S U关 * m，因为 y k s D w = 达 K 其容？限制。

LUA_SEND_CORR_TABLE_FULL

y k s D w D 发 M关 * m = 达 K 其容？限制。

LUA_SESSION_SERVICES_PATH_ERROR

会话服务 k s ；能沿着 SSCP-SSCP 会话 7 6 重 7 I。C 功能 G X需 D，例如，需要 h 置一 v g 越网 g D LU-LU 会话。

字 Z 2 k 3 ｜含检 b k X定 D信息。下 P h 置 G 允许 D:

0000 未 & C X定 ` k。}非 9 C K **SLI_OPEN** 重 T I 选项，否则 **SLI_OPEN** ；能 9 C > 返回 k G记。1 **lua_wait** h 置为非 c 值 1，重 T INITSELF 或 LOGON 信息，直 = I 9 C 主机过 L，或 & C L 序发 v **SLI_CLOSE**。

0001 SSCP 通过一 v 或一 v 以 O Z ｜ SSCP + 会话服务 k s 重 7 I = 其目 D X D " T ' \。1 C 值 P 无 n ！D " T - ' \，' \ - 再 " T 重 7 I 1，C 值 I 网关 SSCP 发 M。

SSCP D 重 7 I 完全 ' \。SSCP 对一 X 定 SSCP D " T 未 I 功。例如，1 在 } " T D Z c 显 > P 关重 7 I ' \ D 信息 1，>代 k k X定 SSCP 相关 *。

0002

因为；存在 X需 D 7 I m，y 以 SSCP 无法重 7 I 一 v 会话服务 k s；M G 5，；P k 资源 j 6 X制资源中 D 重 7 I \ 钥对 & D Z ｜ SSCP m。

0003

C SSCP ；P 对 LU D 预定义，+ G，Z ｜ SSCP ；在伙 i SSCP 中支 V 动，定义。a 果 G C SSCP ；能即动，X 定义 LU V 重 7 I = 那 v Z ｜ SSCP。

0005

重 T

0006

重T

0008

Z | SSCP ；支V k s D CDINIT 函}（例如，通知资源D I C性或XRF)。

000A

SSCP 无法重7 I会话服务k s，因为k s在同一v SSCP O 7 I K = 次。

000B

指定在 CDINIT 内D DLU 对Z S U SSCP 而言G未知D，" R S U SSCP ；能重7 I CDINIT。

LUA_RU_LENGTH_ERROR

k s D RU + $或+短。+ RU 传M = 预计D会话端组件，+ ；I以；打断或处理。b v u件m > 会话端能&；匹配。

>代k也I作为 EXR 中D检b k v现。

LUA_FUNCTION_NOT_SUPPORTED

LUA ；支V k s D函}。函}原& I q = 化k s代k、RU 中D N}或X制字符指定。

z f在检b k后D字Z 2 和 3 ；；C户定义D }] 9 C。b些字Z | 含检b k X定D信息。下P h置G允许D:

0000 LUA ；支V k s D函}。

+ RU 传M = 预计D会话端组件，+ ；I以；打断或处理。b v u件m > 会话端能&；匹配。

LUA_HDX_BRACKET_STATE_ERROR

协议机器确定1 0 k s ；能在现存D状，错误下发M。

LUA_RESPONSE_ALREADY_SENT

协议机器确定1 0 k s ；能发M，因为已 - 发M K 4 D响&。

LUA_EXR_SENSE_INCORRECT

&C L序对以0 S U = D异# k s发v一v否定响&。响&D检b k G ；I S \ D。

若异# k s中D检b k为 X'0813000'，则否定响&中D检b k I为 X'08130000'或 X'08140000'。在y P其| i v下，否定响&中D检b k X须k异# k s中D检b k一致。

LUA_RESPONSE_OUT_OF_ORDER

协议机器确定；Q 1 0响&发v = 最老D k s O。

LUA_CHASE_RESPONSE_REQUIRED

协议机器确定}在" T 1 0 k s，而一v O I D CHASE k s未执行。

LUA_CATEGORY_NOT_SUPPORTED

DFC、SC、NC 或 FMD k s ；；支V任何那种类p中D k s D会话端S U =，网g服务（NS）k s字Z 0 未h置为一v定义K D值，或S U器未+字Z 1 h置为 NS 类p。

LUA_CHAINING_ERROR

在 4 指 > 符 h 置序 P 中发 z 错误，如 Z 一 v、中间、Z 一 v。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_BRACKET

发 M 人；加？会话 D 方括号规则。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对预计 D 会话端组件 k s D 传]。

LUA_DIRECTION

1 k + 工触发器状，为 NOT_RECEIVE 1， S U = } # }] w k s。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_DATA_TRAFFIC_RESET

会话端 S U = FMD 或 } # }] w DFC k s， C 会话端 D 会话激活状，G 活动 D， + G 其 }] w 通状，G；活动 D。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_DATA_TRAFFIC_QUIESCED

从发 M QC ｜ n 或 SHUTC ｜ n D 会话端 S U = D FMD 或 DFC k s 未对 RELQ ｜ n 作 v 响 &。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_DATA_TRAFFIC_NOT_RESET

1 }] w 通未 4 位 1 S U = 会话 X 制 k s。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_NO_BEGIN_BRACKET

S U 器已对 BIS ｜ n 发 M K O 定响 & 后，S U = 指定 BBI=BB D BID 或 FMD k s。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_SC_PROTOCOL_VIOLATION

违反 SC 协议。k s v 在以下 i v 后允许：在 I 功；换发 z 之 0 S U = 一 v SC k s k 其关 * D O 定响 & D I 功；换。检 b }] D 字 Z 4 ｜含 k s 代 k。；P k C 检 b k 关 * D C 户 }]。检 b = 一 v k s j b 或 k s %元，｜；允许 S U 器 D 1 0 会话 X 制或 }] w D X 制状，。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_IMMEDIATE_REQ_MODE_ERROR

k s 违反 K " 即 k s 方 = 协议。检 b = S U 器 D 1 0 会话 X 制或 }] w X 制状，y；允许 D RH 或 RU。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_QUEUED_RESPONSE_ERROR

k s 违反 K 排队 D 响 & 协议；例如，1 未执行 k s P QRI=QR 1，QRI=¬ QR。检 b = S U 器 D 1 0 会话 X 制或 }] w X 制状，y；允许 D RH 或 RU。 b v 错误防 - K 对会话端组件 D k s D 传]。

LUA_ERP_SYNC_EVENT_ERROR

违反 ERP 同 = B件协议。检 b = S U器D 1 0会话X制或 }] w X制状 , y ; 允许D RH 或 RU。 b v错误防 - K对会话端组件D k s D传] 。

LUA_RSP_BEFORE_SENDING_REQ

k + 工（触发器或 y C ）中作一 " T , 发M/ S \ 方 = , 以 1 未 S U = 对以 0 k s 作 v D响 & 1 , 发M } # }] w k s 。检 b = S U器D 1 0会话X制或 }] w X制状 , y ; 允许D RH 或 RU。 b v错误防 - K对会话端组件D k s D传] 。

LUA_RSP_CORRELATION_ERROR

S U = 响 &, | ; 能 k 以 0 发M D k s 相 * 系, 或 _ 发M D响 & 无法 k 先 0 S U D k s 相 * 系。

LUA_RSP_PROTOCOL_ERROR

从主会话端 S U = 一 v 违反响 & 协议D响 &, H 如:

- RQE 4 S U = 一 v } 响 &（+RSP）。
- 一 v 4 S U = = v 响 &。

LUA_INVALID_SC_OR_NC_RH

会话X制（SC）或网 g X制（NC）D RH k s 无效。例如, 一 v 带 P w = k s 指 > 符D SC RH h 置 I 1 G无效D。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定D LOGON 选项。 b 些错误防 - K对预计会话端组件 k s D传] , " 独 " Z 1 0 D会话状 , 。 b 些错误 I 能G因为发M方 5) 会话规则 ' \ 而引起D。

LUA_BB_NOT_ALLOWED

9 C 主会话端 0 k v 方括号指 > 符（BB）; 例如, BBI=BB 带 P BCI=¬BC。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定D LOGON 选项。 b 些错误防 - K对预计会话端组件 k s D传] , " 独 " Z 1 0 D会话状 , 。 b 些错误 I 能G因为发M方 5) 会话规则 ' \ 而引起D。

LUA_EB_NOT_ALLOWED

未 } 确指定中括号指 > 符（EB）; 例如, 1 只 P 次级会话端 I 以发M一 v EB 1 , 9 C K EBI=EB 带 P BCI=¬BC 或 9 C K主会话端, 或 1 只 P 主会话端 I 以发M一 v EB 1 , 9 C K次级会话端。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定D LOGON 选项。 b 些错误防 - K对预计会话端组件 k s D传] , " 独 " Z 1 0 D会话状 , 。 b 些错误 I 能G因为发M方 5) 会话规则 ' \ 而引起D。

LUA_EXCEPTION_RSP_NOT_ALLOWED

在 ; 允许 k s D i v 下 k s 一 v 异 # 响 &。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定D LOGON 选项。 b 些错误防 - K对预计会话端组件 k s D传] , " 独 " Z 1 0 D会话状 , 。 b 些错误 I 能G因为发M方 5) 会话规则 ' \ 而引起D。

LUA_DEFINITE_RSP_NOT_ALLOWED

在 ; 允许 k s D i v 下 k s 一 v 确 P 响 &。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定D LOGON 选项。 b 些错误防 - K对预计会话端组件 k s D传] , " 独 " Z 1 0 D会话状 , 。 b 些错误 I 能G因为发M方 5) 会话规则 ' \ 而引起D。

LUA_PACING_NOT_SUPPORTED

在 k s O h 置 K w = 指 > 符， + } 在 S U D 会话端或 _ g 功能会话端在 C 会话 O ；支 V w = 。 RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_CD_NOT_ALLOWED

未 } 确指定换向指 > 符（CD）；例如， CDI=CD 带 P ECI=¬EC 或 CDI=CD 带 P EBI=EB。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_NO_RESPONSE_NOT_ALLOWED

在 ; 允许 k s D i v 下在一 v k s O 指定 K “未响 &” (No-response)。 “未响 &” (No-response) v 在 EXR O 9 C。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_CHAINING_NOT_SUPPORTED

未 } 确指定 4 S 指 > 符（BCI 和 ECI）；例如，指定 D 4 S 位 " 非 BCI=BC 和 ECI=EC，而多方 k s 4 ；支 V 会话或在 k s j b 中指定 D 类 p。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_BRACKETS_NOT_SUPPORTED

未 } 确指定方括号指 > 符（BBI 和 EBI）；例如， h 置 K 方括号指 > 符（BBI=BB 或 EBI=EB）， + | " ；C Z 会话。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_CD_NOT_SUPPORTED

h 置 K “换向” (change-direction) 指 > 符， + ； \ 支 V。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_USE_OF_FI

未 } 确指定 q = 指 > 符（FI）；例如， C BCI=¬BC 来 h 置 FI 或 G 在 DFC k s O 未 h 置 FI。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_ALTERNATE_CODE_NOT_SUPPORTED

在代 k 选择指 > 符（CSI）；支 V 会话 D i v 下 h 置 K 代 k 选择指 > 符。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传] ， " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_RU_CATEGORY

RU 类 p 指 > 符 ; P } 确指定; 例如， 9 C RU category indicator = FMD 来

指定加急 }] w k s 或一 v 响&。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_REQUEST_CODE

响& 1 D k s 代 k 和其相& D k s 1 D k s 代 k ；匹配。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_SPEC_OF_SDI_RTI

在响& 1 未 } 确指定 “ | 含响& }] ” (sense-data-included)指 > 符（SDI）和 “响& 类型” 指 > 符（RTI）。 } 确 D 值对 G （SDI=SD, RTI=negative）和（SDI=¬SD, RTI=positive）。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_DR1I_DR2I_ERI

确 P 响& 1 指 > 符（DR1I）、确 P 响& 2 指 > 符 （DR2I）和异 # 响& 指 > 符（E R I ） ； P } 确 指 定 。 例 如 , 一 v C A N C E L k s ； P C DR1I=DR1、DR2I=¬DR2 和 ERI=¬ER 来指定。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_USE_OF_QRI

未 } 确指定排队响& 指 > 符（QRI）； 例如, 在一 v 加急 }] w k s O QRI=QR。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_USE_OF_EDI

未 } 确指定加 \ }] 指 > 符（EDI）； 例如, 在 DFC k s O EDI=ED。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_INCORRECT_USE_OF_PDI

n d }] 指 > 符（PDI）D 5 w ；} 确, 如一 v DFC k s O D PDI=PD。RH 中 D N } 值或组合 N } 违反 K a 构规则或从 0 y 选定 D LOGON 选项。 b 些错误防 - K 对预计会话端组件 k s D 传], " 独 " Z 1 0 D 会话状，。 b 些错误 I 能 G 因为发 M 方 5) 会话规则 ' \ 而引起 D。

LUA_NAU_INOPERATIVE

NAU 无法处理 k s 或响&。例如, NAU ；一 v 异 # 终止打 R。 I Z 7 6 D 减耗 ?、激活 k s D ； } 确序 P 或 P v D 7 6 信息 % 元（PIU）错误, 无法 + k s 传 = 预计 D S U 方。一 c , 1 会话激活 1 S U = 一 v 7 6 错误, m w 至会话伙 i D 7 6 丢 ' 。

LUA_NO_SESSION

} 在 S U D 末端 Z c 中无激活 D 会话端 C Z 指定 D 源 - 目 j 对, 或 G 在 a 供 _

g 函 } D Z c 中无激活 D C Z 源-目 j 对 D 会话端组件。一 v 会话 D 激活 k s G X 要 D。 I Z 7 6 D 减耗？或激活 k s D ； } 确序 P， 无法 + k s 传 = 预计 D S U 方。一 c， 1 会话激活 1 S U = 一 v 7 6 错误， m w 至会话伙 i D 7 6 丢 ' 。

LUA_BRACKET_RACE_ERROR

在 W c 协议中发 z y C ' \ 。 1 y 发 z D = v NAU 而起止方括号，则 y C ' \ 。预计 D 会话端组件 K b y 支 V D k s， + ；对其 x 行处理。

LUA_BB_REJECT_NO_RTR

1 Z 一 v 发话 _ 处在方括号中或 G 在方括号间状，1， S U = 一 v BID 或一 v 起 < 方括号指 > 符。 Z 一 v 发话 _ \ x 许 I 权。 ； P RTR | n 发 M v 去。预计 D 会话端组件 K b y 支 V D k s， + ；对其 x 行处理。

LUA_CRYPTOGRAPHY_INOPERATIVE

k s D S U _ I Z 其 \ k 功能发 z 故 O，无法对 k s x 行译 k 。预计 D 会话端组件 K b y 支 V D k s， + ；对其 x 行处理。

LUA_SYNC_EVENT_RESPONSE

S U = 一 v 对同 = k s D 逆响 &。预计 D 会话端组件 K b y 支 V D k s， + ；对其 x 行处理。

LUA_RU_DATA_ERROR

S U FMDS 组件 1 ； S \ k s RU 中 D }] 。例如，一 v 字符代 k ；在 ; 支 V D 集合中，显 > 服务 ； S \ 一 v q = 化 D }] N } ，或 _ 在 k s 中 ! T K X 需 D { F 。 + RU 传 M = 预计 D 会话端组件， + ； I 以 ; 打断或处理。 b v u 件 m > 会话端能 & ；匹配。

LUA_INCORRECT_SEQUENCE_NUMBER

在 } # }] w k s O S U = D 序 P 号 ; 大 Z 最后一 v 序 P 号。检 b = 一 v 序 P 号错误或一 v RH 或 RU， | G G S U 器 1 0 会话 X 制和 }] w X 制状， y ； 允许 D。 b v 错误防 - K 对预计 D 会话端组件 k s D 传] 。

附 < C. APPC 对话 4 态 * F

下 P m q 显 > K I 发 v ? v 动词 D 对话状，, 以及在动词完 I 后发 z D 状，d | 。在一些 i v 下，状，d | 取 v Z 返回 x 动词 D **primary_rc** N }；在 b J C D X 方，在返回 k P P v K I J C D **primary_rc** 值。

在；P 显 > 返回 k D X 方，状，d | 对 Z y P 返回 k G 相同 D(} K 在 m q 下 f D 注 M 2 和 3 中 y v)。

I 能 D 对话状，显 > 为 P j b 。对 Z ? v 动词，在 ? v j b 下 x v 以下信息来 m > 在 C 状，下发 v C 动词 D a 果:

- **X** 如果在 C 状，下；能发 v C 动词。
- **S**、**SP**、**R**、**C**、**CS**、 **CD** 或 **P** m > 对话完 I 后 D 状，: **R**eset、**S**end、 **S**end **P**ending、**R**eceive, **C**onfirm、**C**onfirm **S**end、 **C**onfirm **D**eallocate 或 **P**ending Post。
- ***I*** 如果 | ；能 C 来 < G O 一次状，。 b J C Z **[MC_]ALLOCATE** 和 **RECEIVE_ALLOCATE** 动词; b 些动词总 G 启动一 v 新 D 对话(好象 | G 在 “4 位” 状，下), 而；对发 v | G D 对话(如果 P D 话) z z 0 响。
- UW 如果在 C 状，；会发 z y 显 > D 返回 k 。

若需 P 关全 + 工对话状，转移 D 信息, k N 阅 Z 345 页 D m 24。

m 23. APPC k + 工对话状，转移

动词返回 k	**复位 (T)**	**发M (S)**	**发M暂挂 (SP)**	**S 收 (R)**	**确认 (C)**	**确认发M (CS)**	**确 认 Deall (CD)**	**暂挂发出 (PS)**
[MC_]ALLOCATE AP_OK (其 |)	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
[MC_]DEALLOCATE (异 # 终止) [MC_]DEALLOCATE (其 |) AP_ERROR (其 |)	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 (其 |)	R T	/	/	/	/	/	/	/

m 23. APPC k + 工对话状，转移 (续)

动词返回 k	复位 (T)	发M (S)	发M暂挂 (SP)	S 收 (R)	确认 (C)	确认发M (CS)	确 认 Deall (CD)	暂挂发出 (PS)
[MC_]RECEIVE_AND_POST (注M 4)	X	P	P	P	X	X	X	X
[MC_]RECEIVE_AND_WAIT	X	Note 5	Note 5	Note 5	X	X	X	X
[MC_]RECEIVE_IMMEDIATE	X	X	X	Note 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

":

1. 在m q D返回 k P， 对以下返回 k 9 C u 写 AP_ERROR:

   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. 如果 S U = 任何以下返回 k， 那 4 对话 + 总 x 入 4 位状，:

   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. 下 P 非 OK 返回 k ；会引起任何状， d | 。对话总 # V在发 v 动词 1 y 在状，:

   AP_CONVERSATION_TYPE_MIXED

   AP_PARAMETER_CHECK

   AP_STATE_CHECK

   AP_TP_BUSY

   AP_UNEXPECTED_SYSTEM_ERROR

AP_UNSUCCESSFUL

4. 在已发v **[MC_]RECEIVE_AND_POST** " S U K AP_OK D u < D **primary_rc** 后，对话 | D x 暂挂发v状，。一) w C K y a 供D回P例L来m > 动词已完I，新D对话状，M + 取 v Z 在注M 5 中D **primary_rc** 和 **what_rcvd** N }。
5. 在任何 **RECEIVE** 动词后D状，| D取 v Z **primary_rc** 和 **what_rcvd** N }。

   如果 **primary_rc** N } G AP_PROG_ERROR*、AP_SVC_ERROR* 或 (只P [MC_]RECEIVE_IMMEDIATE) AP_UNSUCCESSFUL，则新D状，G RECEIVE。

   如果 **primary_rc** N } G AP_DEALLOC*，则新D状，G RESET。

   如果 **primary_rc** N } G AP_OK，则新D状，取 v Z **what_rcvd** N } D值:

   S 收 4 态
   AP_DATA, AP_DATA_COMPLETE, AP_DATA_INCOMPLETE

   发M 4 态
   AP_SEND

   发M暂挂 4 态
   AP_DATA_SEND, AP_DATA_COMPLETE_SEND

   确认 4 态
   AP_CONFIRM_WHAT_RECEIVED, AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM

   确认发M 4 态
   AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND

   确认 b 除分配 4 态
   AP_CONFIRM_DEALLOCATE, AP_DATA_CONFIRM_DEALLOCATE, AP_DATA_COMPLETE_CONFIRM_DEALL

若需P关k + 工对话状，转移D信息， k N阅 Z 343页D m 23。

m 24. APPC 全 + 工对话状，转移

动词返回 k	**复位 (T)**	**发M S 收 (SR)**	**v 发M (S)**	**v S 收 (R)**
[MC_]ALLOCATE AP_OK (其 |)	 SR T	/	/	/
CANCEL_CONVERSATION	X	T	T	T
[MC_]DEALLOCATE (异 # a x) [MC_]DEALLOCATE (e U)	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 (其 |)	 SR T	/	/	/

m 24. APPC 全 + 工对话状，转移 (续)

动词返回 k	复位 (T)	发M S 收 (SR)	v 发M (S)	v S 收 (R)
[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
[MC_]SEND_ERROR AP_OK AP_ERROR_INDICATION	 X X	 SR SR	 S T	 X X

":

1. 在m q D返回 k P，对以下返回 k 9 C u 写 AP_ERROR:

   AP_PROG_ERROR_TRUNC

   AP_PROG_ERROR_NO_TRUNC

   AP_SVC_ERROR_TRUNC

   AP_SVC_ERROR_NO_TRUNC

2. 如果 S U = 任何以下返回 k，那 4 对话 + 总 x 入 4 位状，:

   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. 下 P 非 OK 返回 k ；会引起任何状，d | 。对话总 # V 在发 v 动词 1 y 在状，:

   AP_CONVERSATION_TYPE_MIXED

   AP_PARAMETER_CHECK

   AP_STATE_CHECK

   AP_TP_BUSY

   AP_UNEXPECTED_SYSTEM_ERROR

   AP_UNSUCCESSFUL

附 < D. 通 E 服务器服务定位 - i

发现和负载 y 衡 API

通信服务器(N < IBM 通信服务器 Windows NT f 和 IntranetWare for SAA) & C L 序 * 发 _ 现在 I 以 9 C TCP/IP 协议定位服务 " 在 b 些服务中达 = ：载 y 衡。 & C L 序 I 以 C 三种基 > 方 = 9 C C 项新功能:

- 通信服务器 SNA API (LUx (RUI/SLI), APPC, CPIC)。如果现存 & C L 序已写入 = SNA API 中 D 话, 则 API D 9 C + 基 > 无代价取 C 支 V。 9 C K b 种方 =, 则 9 C 定位/：载 y 衡功能 1 ； C 写入任何新代 k。 C 方 = 唯一 D 约 x G API 代 k 要 s M 户配置 }] 驻 t 在 Novell 目 < 服务(NDS) for IntranetWare 或在 INI 文件, 或 LDAP 文件 Windows NT f 中。
- 服务定位协议(SLP) C 户代理(UA) API。SLP UA DLL G = 加在 a 供通信服务器服务定位和 TCP/IP ， S ：载 y 衡支 V D z 品中 D。 C 方 = 为 & C L 序 * 发 _ a 供 K 最大 D i 活性以 v 定如何 x 行服务定位/：载 y 衡、在何处 C = M 户配置、以及如何为端 c C 户 a 供 b 些功能。
- 9 C UA (C Z 定位)和：载 y 衡 QEL/MU CM_CSLIST_GETII 原 o (v C Z 3270 和 LU6.2 & C L 序) D 组合。 C 方 = G 头 = 项 D 组合, 因为 | 减 Y K 需要写入定位函 } D 代 k ？, " a 供 K 配置 M 户最大 D i 活性。

IBM 和 Novell (议 9 C 定位和：载 y 衡 D API M 户机。如果 & C L 序 * 发 _ ；能 9 C | G 或准 8 支 V Telnet, 则 + a 供方 = 2。如果已 a 供 QEL/MU 支 V, 则 + 9 C 方 = 3。因为 Z 一种方 = 在 & C L 序 * 发 _ 4 来已 ； P 2 4 新鲜 D, y 以下 P V [J C Z 后 = 种方 =。

a 构

UA API G 通 C D C o 言 API, | 在 “API 服务定位” Internet] 8 (3/25/97) 中 P h v。下 P X 性 J C Z 服务 G 记:

- y P G 记都以@国 " o 为基础。
- 字符集 G US-ASCII。

API 在 Windows 95 和 Windows NT 平 (O G 以 IBMSLP.DLL 形 = 封装 D。在定义相关 a 构、 # } 和函 } 原 o D SDK 中 a 供 K 头文件。 DLL 在 API M 户 2 装 1 2 装, " I 在带 P 其 | SLP SDK 文件 D CD-ROM D \NWSAA\SAASDK\SLP\BINARY\IBMSLP.DLL 或 \CSNT\SDK\SLP\BINARY\IBMSLP.DLL 中 R =。

方 8

在 ? v 方 8 中, 9 C C 户代理 API D & C L 序 F 为 “app”。对端 c C 户 D 引 C (9 C app D 人员) u 减为 “C 户”。

方式 2: UA API 定位 n ！负载(或 “低负载”)服务。

1. & C L 序发 v SL_Open | n 打 * SLP 会话。

2. 如果；P配置r或 app ；P I C D r，则&C L序发v(带P “SCOPE” t性j记过K器D) y需服务类型D SL_GetAttrs API w C以取C P效D， I 5现D r。 a供(C api w C O管理D) IntranetWare for SAA 3.0 或 CSNT 6.0 服务D服务{ +确#只返回J C Z y a供D服务类型D r。
3. 然后， &C L序发v SL_GetService, 指定y需D服务、C = D r { 之一、以及指定X需服务t性D i 询字符串。为K 5 w D目D，在> 例i 询中指定D服务t性+ G LUPOOL 和 LOAD。服务回答+ | 含；P R = 匹配服务D指>，或_ G一串 URL, 在z足i 询字符串需s 1 a供服务。
4. &C L序分析K返回D P m:
5. 如果；P y需D URL， &C L序或_修D和重新发v在= 骤 3 中5 w D带P新加载j 准范围D原< SL_GetService k s，否则通知端c C户；P I C服务。
6. 如果返回% v URL，则分析M完I K。
7. 如果返回一P URL:
   - **选项 1 - ²n !负载² 定位**
     a. &C L序发v SL_GetAttrs, C Z在服务回答中返回D ? v URL。 |在? v w C O D选择子d中指定K：载t性。 :载值在 Get t性回答中返回。
     b. &C L序选择带P最M：载值D URL。
     c. &C L序，S = I 选中D URL y m > D服务器， " * < SNA 会话。
     d. &C L序发v SL_Close 以关U SLP 会话。
   - **选项 2 - “低负载”选择**
     a. 从返回P m中f机选择一 URL。
     b. &C L序，S = I 选中 URL m > D服务器" * < SNA 会话。
     c. &C L序发v SL_Close 以关U SLP 会话。

注意，在许多服务器中都P = v选项代m K：载y衡。 = v选项间D关键n异在Z: 选项 1 #证选中K最小：载服务器， + | z I K H选项 2 |多D LAN 通信?。选项 2 只#证选中“M：载”服务器， +在选择x L中D 1在线7通信? H选项 1 Y。

重试: 在许多i v下， &C L序D，S重T对Z 5现C户资源最大I C性来5 G X需D。&C L序，S重T X需D一v u件G: 1 &C L序T图，S = SL_GetService O返回D URL " (" SNA 会话1， ；P I C D LU。C u件I能G I Z通过 SLP 注a D服务和注a服务器O 5际I C D服务间D I n合。如果&C L序；能，S = 选中D服务， | &C重T m一返回D服务(例如，下一v最小：载服务器)。如果；P |多I C D服务， &C L序I以从u < D SL_GetService 重T或+ u件(f端c C户。

URL 格式: I通信服务器广%D URL |含= v ?分: 加c D . x制 IP X址和端Z号。

URL G带P下P q = D ASCII 字符串:

```
<IP address>:<port number>
```

IP X址G服务器D缺！ IP X址。端Z号取v Z广%D服务类型:

m 25. 服务类型/端Z信息

服务`M	端口
通信服务器	众y周知D CommExec l } 端Z 1366
cs3270	众y周知D CommExec l } 端Z 1366
csappc	众y周知D CommExec l } 端Z 1366
tn3270	来自服务器O ETC/SERVICES 文件或在Telnet 服务器O配置D Telnet 端Z

端口

对Z下一v通信服务器 NWSAA f D发行>，会支V多v端Z， a供对 AS400 系统D多v ，S。CSNT 6.0 目0支V多v端Z。同样也+ a供对2全加\ Telnet 会话D支V，对Z 2全会话， |需要D G ；同D端Z号，而；G缺！端Z号。C仿f器&C I以9 C I SLP 服务i询返回D端Z号。在 TEMPLATE.HTM 文件中I以R = |多关Z服务类型D信息。

示} 1: &C L序a供K Telnet O D 3270 仿f。 |需要，S = (已配置过D ACCOUNTS D LU X中)任何I C D LU, 同1 |也需要通过最a ：载D服务器，S。在网g中；P配置任何r。C主机支V动，h 8类型， y以&C L序；X指定h 8类型。

&C L序通过发v下P SL_GetService k s D谓词, 从而* <定位服务器 (在y P >例中 '\t' 都G TAB 字符):

```
tn3270//LUPOOL==ACCOUNTS*/
```

在b一c O，返回K三v URL (| F Z b些)D P m(端Z号 23 G Telnet ，S k s D j准端Z):

```
service:tn3270://9.37.51.254:23
service:tn3270://9.37.51.260:23

service:tn3270://9.37.51.256:23
```

要h计5现最小：载定位, &C L序发v一系P定向=？v URL D SL_GetAttrs w C，从而C =？v服务器D：载值。 |指定K | F Z以下b些只S \ ：载信息D选择子d:

```
URL = service:tn3270://9.37.51.254:23
Attribute filter = LOAD
```

- LOAD t性返回K值 ²5²
- &C L序发v Z二v URL D SL_GetAttrs ， 而|返回：载值, ²2²
- 最后G Z三v服务器, |返回为 ²10² D：载值

因为Z二服务器D：载OM， y以&CL序选择 9.37.52.260:23 作为，S目j。&CL序"T，S= 9.37.51.260， +，S因为；PICD LU 而'\。然后，|想，S= 9.37.51.254 (下一v最小：载服务器)，而b次|I功K。

示} 2: m一v&CL序a供K tn3270 仿f。|要s定位=a供C服务DM：载D服务器O。M户配置来自 INI 文件或 NDS: |DrG ENGINEERING， "R|需要在 LUX SMITH_1 中R= LU 类型 2 模型 2 。

C&CL序G通过发v TN3270: 服务类型和 'SCOPE' t性j记过K器D SL_GetAttrs *<x行D。|返回K一系P支V TN3270 D服务器管理Dr值。为Kc Z 5w，假h 'ENGINEERING' Dr值在 SL_GetAttrs wCO返回。然后，&CL序("K下P SL_GetService ksD谓词以在Cr中定位服务器，bz足Ku< LU h 8类型，和：载ks(在yP>例中'\t' G TAB 字符)：

```
tn3270/ENGINEERING/LUPOOL==SMITH_1\t3270002,LOAD <= 10/
```

&CL序h计I在 10 ：载增？中定位， y以如果u<D SL_GetService ks返回UPm，则&CL序重新发M SL_GetService，再次指定K服务"加O新Dt性。

```
tn3270/ENGINEERING/LUPOOL==SMITH_1\t3270002,LOAD <= 20/
```

在b一cO，返回K=v URL(|FZ|G)DPm (端Z号 23 G Telnet ，Sksj准端Z)：

```
service:tn3270://9.37.51.254:23
service:tn3270://9.37.51.260:23
```

&CL序；在意G否选中x对最小：载服务器，只要|D：载小Z 20% 即I。因此，|会f机选择=v URL 中D一v：

```
URL = service:tn3270://9.37.51.260:23
```

&CL序选择 9.37.52.260:23 作为，S目j，而，SI功K。

方=3: 服务定位D UA 和：载y衡D CM_CSLIST_GETII。: CM_CSLIST_GETII 原oGa供x QEL/MU 仿f器D。C原o+允许&CL序a供多v过K器。头文件 cmi.h|含C方=Da构和定义"都|含在C SDK 中。要9CC方=，下P过LGHOJCD：

1. &CL序发v SL_Open |n打* SLP 会话。
2. 如果；P配置r或 app ；PICDr，则&CL序发v(带P "SCOPE" t性j记过K器D) 'cs3270' 服务类型D SL_GetAttrs API wC以取CP效D，I5现Dr。C API 返回K对&Z通信服务器服务 URL DrPm，C通信服务器I以响&IP f> CM_CSLIST_GETII 原o。
3. C&CL序发vv指定 'cs3270' 服务D SL_GetService 和P效Dr。C服务回答|含一系P服务器 URL，&CL序I，S=|G以处理 CM_CSLIST_GETII 原o。
4. &CL序，S=任何Pm中选中D URL m>D服务器。
5. &CL序发v SL_Close 以关U SLP 会话。
6. &CL序("K CM_CSLIST_GETII 原o以检w：载y衡服务器Pm。其中，AgentType 字段h置Iy需D服务，同1过K器规范|含r和 LU XD{F(如果JCD话)。

7. 返回 CM_CSLIST_GETII_ACK, ||含：载 y 衡次序(最M：载 = 最 _ ：载)中服务器 TCP/IP X址D P m。
8. &C L序选择在P m中D Z一v服务器" ，S = |。
9. &C L序" T k服务器(" SNA 会话。如果；I功, |会在返回P m O D下一v服务器O重 4 以O Y作直 = I功或 _ C P m a x。

m 26. CM_CSLIST_GETII 原 o

V段名	V段偏F ?(十 y x 制)	V段长度(二 x 制)	` M	Z容和使C
PrimType	x00	4	long int	CM_CSLIST_GETII M象在 cmi.h 中一样
UserParm	x04	4	long int	在回答中要返回D任何值
Reserved	x08	4	long int	c
ServiceType	x0c	4	long int	0x12B (对Z ：载 y 衡支V)
ProdVersion	x10	4	long int	-1 (m > ²； i 意²)
NWVersion	x14	4	long int	-1 (m > ²； i 意²)
Flags	x18	4	long int	k N阅m ²j 志值²
AgentType	x0c	4	long int	k N阅m ²AgentType 值²
FilterList	x1c	*	FilterList_t	k N阅m ²FilterList_t² (值取v Z “j 志”D h置)

m 27. CM_CSLIST_GETII 原 o

常数	值	含e
c	0	需要无序P m。；P指定过K器。 (G为向后兼容 a 供D)
CMCsListFlags_LBPool	1	需要指定：载 y 衡X { F D：载 y 衡 P m(G为向后兼容 a 供D值)
CMCsListFlags_LBAgent	2	需要：载 y 衡 P m。AgentType C Z ：载 y 衡
CMCsListFlags_LBFilter	3	需要：载 y 衡 P m。 t z 着一v过K器D d ？$度P m。

m 28. j 志值(从 cmi.h)

常数	值	含e
CSA_3270	0x126	需要 LU 类型1/2/3 D SNA 网关代理
CSA_SAA	0x12B	需要 LU 类型 6.2 D SNA 网关代理

m 29. AgentType 值(从 csobjtyp.h)

V段名	V段偏F ?(十y x制)	V段长度(二x制)	`M	Z容和使C
FilterNameLen	x00	4	long int	下P ：载y衡组(X){ F D $度
FilterName	x04	*	ASCII	：载y衡组(X)D { F

m 30. FilterList_t (if Flags = CMCsListFlag_LBPool)

V段名	V段偏F ?(十y x制)	V段长度(二x制)	`M	Z容和使C
FilterCount	x00	4	long int	在 (0, if Flags = zero) 之后D过K器P m{ F a构D } ?
FilterList	x04	*	Filter_t	过K器P m{ F a构D P m。 ? v a构都P其d ? $度。

m 31. FilterList_t (if Flags = zero | Flags = CMCsListFlag_LBFilters)

V段名	V段偏F ?(十y x制)	V段长度(二x制)	`M	Z容和使C
FilterLength	x00	4	long int	a构$度(加O C $度字段)
FilterType	x04	4	long int	k N阅m ²过K器类型值²
FilterName	x08	*	ASCII	过K器{ F值

m 32. Filter_t

常数	含e
CMCsListFilter_LBPool	：载y衡X{。 ? v P m中只能指定一v X。C过K器v对 AgentType CSA_3270 P效。
CMCsListFilter_Scope	SLP r {。 v I以指定一v r。如果；P指定r，则假h G y P未定r D服务。

m 33. 过K器类型值 (从 cmi.h)

V段名	V段偏F ?(十y x制)	V段长度(二x制)	`M	Z容和使C
PrimType	x00	4	long int	CM_CSLIST_GETII_ACK M象在 cmi.h 中一样
UserParm	x04	4	long int	在CM_CSLIST_GETII O通过
Reserved	x08	4	long int	c
ServiceType	x0c	4	long int	在CM_CSLIST_GETII O通过

m 33. 过K器类型值 (从 cmi.h) (续)

V段名	V段偏F ？(十 y x 制)	V段长度(二 x 制)	` M	Z 容和使C
Flags	x10	4	long int	在 CM_CSLIST_GETII O通过
ServiceCount	x14	4	long int	下P服务器入Z D } ？

m 34. CM_CSLIST_GETII_ACK 原 o

V段名	V段偏F ？(十 y x 制)	V段长度(二 x 制)	` M	Z 容和使C
ProdVersion	x00	4	long int	z 品 f >
Platform	x04	4	long int	CMCsListPlatform_IWSAA
CSNameLen	x08	4	long int	下P服务器{ D $度
CSName	*	*	long int	服务器{ F(U终 a D)
CSAddrLen	*	4	long int	下P IP X址D $度
CSAddress	*	*	ASCII	m q 中服务器D IP X址: 以c分 n D . x制 IP X址:端Z
NameLen	*	4	long int	下P代理{ F D $度
AgentName	*	*	*	服务器O代理D { F(U终 a D)

m 35. CM_CSLIST_GETII_ACK 原 o 中D服务器信息 a 构

V段名	V段偏F ？(十 y x 制)	V段长度(二 x 制)	` M	Z 容和使C
PrimType	x00	4	long int	如在 cmi.h 中D CM_CSLIST_GETII_ERR
UserParm	x04	4	long int	在 CM_CSLIST_GETII O通过
Reserved	x08	4	long int	c
Errno	x0c	4	long int	错误号

配置的考G

r： 如何C = M户 k s 服务D r 值， P = v 选择:

发现

I 以 9 C SL_GetAttrs API R = r 值 (通过发 v _ P “SCOPE” t 性过K器D服务类型未定 r t 性 k s)。C API 返回K 1 0网 g 活动D服务 r P m。I 以显 > C P m供C户选择。

配置

r 值 I 通过配置M户机C = 。对Z通过 Novell M户 k s 器对 NDS P存取权D & C L 序， C r I 以在 NDS 中配置或M在其中C = 。 IntranetWare for SAA 3.0 + NDS snap-ins 和其增？？分都 a 供 x NDS 模 = ，从而 a 供多种 | 括 SLP r D M户配置值。 k N阅以下内容以C = 关Z r 及其意义D详细内容。

DA-Discovery 超时

DA-Discovery ，1 值，(G SLP_Open API O D 一 v N })，C Z X制 SLP API X须在网 g 中H待发现目 < 代理(DA)D 1 间。C发现 k s G一 v 多次 " T D k s， " R U集 y P DA 响&集中 y 需要D 1 间 I 能因为P 多种 i v 而P y ；同。如果在网 g 中；P DA，则C ，1 值 I 以 h 置为 c ，从而m > ；P x 行 DA 发现。 ，1 G以毫 k m > D。

SA 多次尝试超时

SA 多次 " T ，1 值。SL_Open API O D N } G C 来X制 SLP API (在网 g 中；P 一 v 支V k s r D DA i v 下)X须H待以发现服务、 t 性或服务类型D 1 间。在 b 种 i v 下， b 些 k s G多次 " T D， " R SLP API 会H待，1 值以 + 返回D多重响&集中在一起。 ，1 G以毫 k m > D。

管m员 o 助E息

r

r G C Z X制和管理网 g 中M户机 = 服务器存取D N } 。 | k 服务定位协议 r 相同。X须 a 供X制 r ，因为P = v 原因:

- 在网 g 、M户机 } 、服务器增 $ 1 ，增 $ D M户机 } X须分 x 存取 b 些服务器，从而 I 以减Y网 g 中D { v 通信？。
- | 允许管理员在管理组中组织C户和服务器。

r 值D意义 I 网 g 中D管理员定义。 b 些值 I m > 任何 5 e 。通 #， | G I ？E、X 理位置、或组织 x 分。

如何使C r ?

？v IWSAA 3.0 或 CSNT 6.0 服务器通过w自D配置工 _ 分配 = 一 v 或多 v r 。 9 C b 些服务器D M户机都X须配置为，S = % v X b r 内D服务器或；在 r 内D服务器。 I 以为已配置D服务(3270 和 APPC)分配；同D r 。

| 如何 k SLP * 系?

IntranetWare for SAA 3.0 和 CSNT 6.0 r 直S * 系 = SLP r 。因此，SLP 服务代理和目 < 代理需要驻 t 在支V b 些已配置过 r D网 g 中。如果允许M户基Z r 来定位服务，则须注意 r G如何作为 { e 来 k 网 g * 系D。如果在同样 9 C r D网 g 中存在未定 r 中D服务，则 b 些服务 I 以 z 足任何 r D k s， (b 些 k s I 以 1 在X + ：载加入 b 些支V未定 r 中服务D服务代理和目 < 代理)。因此，我G (议？v I = 达D服务器都P配置过D r 或 _ ；P 服务器P配置过D r 。如果目 < 代理要在 > c 网 g (相对Z向

O H 例转换)中 9 C，则 | G & C 配置为 k 配置服务器处理相同 D r 。 m 外，如果目 < 代理在网 g 中 9 C 未定 r 服务，则至 Y 要 (" 一 v 未定 r 目 < 代理。

"： 如果 SNA API M户配置为，S = 未定 r 服务器，则 + 只 P 未定 r 服务器回答。

负载 y 衡 S 权 r S

：载 y 衡加权因子允许管理员修 D 或加权？v 通信服务器 D：载 y 衡？。？v 服务器 D 因子 I 以 G；同 D。：载？G 在 0 = 100 之间 D 一 v { }，| S | 服务器 O：载 Y 分 J (100 G最 _ 值)。加权因子在 C 计 c 中为管理员 a 供 K 一 v 元 X。C 因子 D 原因 G P C D:

- 在一些 i v 下，存在其 { 一些因子，| G 对服务器 D：载 D 0 响；P 计入通信服务器 c 法 J 户之中。例如，如果通信服务器；P 专 C Z SNA 网关通信？。
- 如果 a 供 TN3270/TN5250 服务 D IntranetWare for SAA 3.0 网关 k 其 { 9 C SLP：载 y 衡 D TN 服务器 5 现 (CSNT 6.0)共存 Z 一 v 网 g 中，则 IntranetWare for SAA 3.0 因子 I 以通过 w { 以 9 % n 异。

加权因子允许管理员；选择服务器或直 S 选择服务器 9 服务器 O D：载？偏离。

服务 # e

通 E 服务器服务 # e

通信服务器服务类型 I 以 G 通信服务器 Windows NT f 或 IntranetWare for SAA 3.0。

对 Z IntranetWare for SAA 3.0，通信服务器服务类型在加载 CommExec 1 注 a 。| h v K IntranetWare for SAA 服务器 D 类 t t 性。b 些 t 性同样在其 | I IntranetWare for SAA a 供 D 服务类型 O 重 4 。

```
Release = <version/release>
```

b G 通信服务器广 f 服务 D f > 和发行 > 级 p 。| D q = G vv.rr.mm， b 里 ²vv² G主 f > 号，²rr² G次 f > 号，²mm² G修 D 级 p 。y P 号 k D 左 ` 都 I c (最多 = v) n d 。例如: f > 6，发行 > 0，修 D 级 p 0 G ²06.00.00²

```
Platform = <platform>
```

b G 先 Z 广 f 服务 D 网 g Y 作系统平 (。定义 D 值 G:

IW 服务器 9 C IntranetWare 网 g Y 作系统

NT 服务器 9 C Microsoft NT Y 作系统

OS2 服务器 9 C OS2 Y 作系统

AIX 服务器 9 C AIX Y 作系统

```
Protocol = <protocol>
```

b G I a 供 C 服务 D 服务器支 V D 协议。定义 D 值 G:

IP 服务器支 V IP O D M户，S (TCP/IP 或 UDP/IP)

IPX　服务器支V IPX O D M户，S (SPX/IPX)

```
Server name = <server name>
```

b G在 2 装期间配置D服务器{ F。 b些值只对 IW 平 (P意义。

通 E 服务器服务登G E 息

```
URL:service:commserver://<addr-spec>:<port-number>
```

t 性:

```
[(SCOPE=<string>),]
(RELEASE=06.00.00),

(PLATFORM=NT),

(PROTOCOL=IP),

(SERVERNAME=<string>)
```

从属 **LU** 服务 # e

通信服务器从 t LU 服务 a 供K通过服务器X定 API 和协议对 SNA 网 g D 3270 网关存取。C t 性反 3 K在服务器O I C D 3270 h 8、LU X、和：载信息D类型。

```
Load = <server_load>:
```

b G C来确定，S = C服务D最M：载通信服务器D：载 y 衡 } ?。 P效值D范围G 0 = 100 D { }，0 m > I 能D最M：载，100 m > 最 _ ：载。

```
LU Pool = <pool_name>,
<pool_name>/t<dev-type>,

<pool_name>/t<dev_type>, ...

<pool_name>/t<dev-type>
```

j 6 LU X D LU X { F，| I k ? v X中支V D关 * h 8类型一起在C服务O 9 C。? v 值都G一 v 记 <，b 里 Z 一 v n 牌G X D X { F，Z 二 v n 牌G C X 支V D h 8 类型。；P h 8 类型D X { F m > 在X中P未知类型D LU。 k x 定X { F 关 * D记 < 对Z ? v 支V D h 8 类型都G重 4 D。如果任何至Y为X a 供一 v PU D E 要在服务器O活动D话，x 定X | 含在G记 k s 中。P 效D dev_types 范围G:

m 36. CM_CSLIST_GETII_ERR 原 o

dev_type	含义
3270002	Lu 类型 2 模型 2
3270003	Lu 类型 2 模型 3
3270004	Lu 类型 2 模型 4
3270005	Lu 类型 2 模型 5
3270DSC	打 ! 机 LU

如果任何配置为类型D LU |含在服务器O D活动 PU E要中，则x定D h 8类型|含在G记k s中。

从属 LU 服务登G E息

```
URL: service:cs3270://<addr-spec>:<port-number>
```

t 性:

```
[(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 服务 # e

TN3270 服务a供K通过 TN3270 协议对 SNA 网关D 3270 网关存取。C t 性反3 K在服务器O I C D 3270 h 8、LU X、和：载信息D类型。 LU X和：载t 性k服务类型 cs3270 相同。

BIND, DATA, RESPONSES, SCS, SYSREQ

b些关键字 t 性h v K C服务支V D TN3270e 功能。

BIND 服务器支V SNA * S 3象功能

DATA 服务器支V非 SNA 3270 }] w

RESPONSES
服务器支V SNA 响&方=

SCS 服务器支V SNA 3270 SCS }] w

SYSREQ
服务器支V SYSREQ 键盘\钥

如果|G h v D功能I C，则|G在服务广f中a供。

```
Security = <security>
```

C字段|含服务器支V D 2全性技u。定义D值G:

NONE　C服务器； P显= 2全性技u
SSLV3　C服务器支V 2全W S字c f > 3 j准
Ciphersuites = <CipherSpec>,
<CipherSpec>, ...
<CipherSpec>
j 6来服务器支V D \ k规范。定义D值G:
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

服务支V D文5功能 RFC }。 1 0 TN3270 RFC |含 1576, 1646, 和 1647。

TN3270 服务登G E息

URL: service:tn3270://<addr-spec>:<port-number>

t性:

[(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 服务 # e

tn5250 服务 a 供通过 TN5250 协议 5250 网关对 SNA 网 g D存取。 C t 性反 3 服务器O I C D " I 存取D AS400 服务和：载信息。

```
Release = <release>
```

b G广 f 通信服务器 f > 和发行 > 。

```
Protocol = <protocol>
```

b G I a 供C服务D服务器支V D协议。定义D值G:

```
IP - 服务器支V IP ODM户，S (TCP/IP 或 UDP/IP)
Platform = <platform>
```

b G广 f 服务 y C D网 g Y作系统平 (。定义D值G:

```
IW - 服务器 9 C IntranetWare 网 g Y作系统
NT - 服务器 9 C Microsoft NT Y作系统

Server Name = <server name>
```

b G在 2 装期间配置D服务器 { F。

```
AS400 Name = <host name>
```

b G C服务G记 J C D AS400 主机。

```
Load = <INTEGER>
```

b G C来确定最M：载通信服务器D：载 y 衡 } ？。 P 效值D范围G 0 = 100。

```
Security = <security>
```

C字段 | 含服务器支V D 2 全性技 u 。 5 际值G:

NONE C服务器； P 显 = 2 全性技 u
SSLV3 C服务器支V 2 全W S 字 c f > 3 j 准
Ciphersuites = <CipherSpec>,
<CipherSpec>, ...

<CipherSpec>
ｊ 6 来服务器支ＶＤ＼ k 规范。定义
Ｄ值Ｇ:
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>
```

Ｃ字段｜含服务器支ＶＤ TN5250 功能。现 1 ； Ｐ定义功能。

```
RFC1205
```

服务支ＶＤ文 5 功能 RFC ｝。 １ 0 TN5250 Ｄ RFC ｜含 1205。

TN5250 服务登Ｇ Ｅ息

```
URL: service:tn5250://<addr-spec>:<port-number>
```

ｔ 性:

```
(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 服务 # e

csappc 服务类型 a 供 SNA APPC 存取。配置 D > X LU 定义 k C 服务一起注 a 。

```
LLU = <llu1>,<llu2>,...,<llun>
```

指定 P 效 > X LU 为在通信服务器 O 配置。

LU6.2 服务登 G E 息

```
URL: service:csappc://<addr-spec>:<port-number>
```

t 性:

```
[(SCOPE=<string>),]

(RELEASE=06.00.00),

(PLATFORM=NT),

(PROTOCOL=IP),

(SERVERNAME=<string>),

(LOAD=<integer 0 to 100>)

[,(LLU=<llu1>,<llu2>,...,<llun>)]
```

附 < E. DLL f 本 E 息

32 位 Windows DLL

下 P 32 位 Windows DLL | 含 I C Z 确定 DLL f > D 信息:

- E32APPC.DLL
- WAPPC32.DLL
- WCPIC32.DLL
- WINCSV32.DLL
- WINMS32.DLL
- WINNOF32.DLL
- WINRUI32.DLL
- WINSLI32.DLL

I C D 关键字 G:

- CompanyName
- LegalCopyright
- LegalTrademarks
- ProductName
- ProductVersion
- FileDescription
- InternalName
- FileVersion

": y P 关键字都 G ²\StringFileInfo\040904E4\² f > i D 一 ? 分, | G ; 需翻译。

I 通过 9 C L 序来检 w 信息, 或 _ I 以通过 9 C Windows Explorer 或 Windows NT Explorer , 如下:

1. C R s j 4 钮选择 DLL
2. 在 / v K % O 选择**属** T
3. 选择 f **本** j) 。

9 C C 信息, I 以写入代 k 以确定 DLL G 否来自 IBM 或其 | 公 > (CompanyName), 还 I 确定 DLL G C Z SNA API M户机还 G 服务器(ProductName)。 I 以确定 2 装哪 v DLL f > (FileVersion),和 2 装哪 v z 品 f > (ProductVersion)。

下 P 样 > C 函 } 确定 K G 否 | { D DLL G I IBM z z D:

```
//
//  Function returns TRUE if and only if given pathname is a versioned IBM DLL
//
#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;

// verify parameters aren't null
if (!pcDllPathname || !*pcDllPathname)
return FALSE;

// get size of Version Info
dwBufSize = GetFileVersionInfoSize(pcDllPathname, &dwTemp);

// no version info implies bad parameters or not versioned IBM DLL
if (!dwBufSize)
return FALSE;

// allocate a buffer for the version information (+50 for safety)
pVInfoBuffer = malloc(dwBufSize + 50);

// malloc failure
if (!pVInfoBuffer)
return FALSE;

// get version buffer filled
bRC = GetFileVersionInfo(pcDllName,dwTemp,dwBufSize,pVInfoBuffer);

// call failed
if (!bRC)
return FALSE;

// get the company name
bRC = VerQueryValue(pVInfoBuffer, TEXT(CMPNY_KEY), ReturnBuffer, ReturnBytes);

// not found or empty
if (!bRC || !dwReturnBytes)
return FALSE;

// value should begin with "IBM"
if (strncmp(pReturnBuffer, "IBM", strlen("IBM")) == 0)
   return TRUE;

return FALSE;
}
```

附 < F. " b 事项

> 资 O G 专为在@国 y a 供 D z 品和服务而 ` 写 D。 IBM I 能还未向其 | 国家 a 供 > 资 O 中 y V [D z 品、服务或功能。若想 K b 目 0 在您 y 在 X I C D z 品和服务， k 咨询 1 X D IBM 代 m。任何对 IBM z 品、L 序或服务 D 引 C " ； 5 w 或 5 > 只能 9 C IBM D z 品、L 序或服务。任何；触犯 IBM 知 6 z 权 R P 同 H 功能 D z 品、L 序或服务都 I 以 C 来代 f IBM z 品、L 序或服务。 + G， C 户 P 责任对 y 9 C D 任何非 IBM z 品、L 序或服务 x 行评估和验证。

IBM I 能已 - j k 或 } 在 j k k > 文 5 P 关 D w 项专利权。 a 供 > 文 5 " ； m > 允许您 9 C b 些专利。您 I 以 C i f 方 = + X 许 i 询寄往:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

P 关 + 字 Z (DBCS)信息 D 许 I 证 i 询， k k 您 D 国家 D IBM 知 6 z 权？E * 系或 + i 询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

下面这段 Z 容；适 C Z " 国或任何当地法 I k 本条款；；致的国 R: 国际 L 业机器公 > 以 『 AS IS 』(即“原样”)方 = a 供 > v f 物， > v f 物；_ P 任何形 = D # #，无 [G w 确 D 还 G 隐 = D，其中 | 括 + ；V 限 Z 对 X 定目 D 之合法性、I 销 [性或 J 宜性 D 隐 = # #。P 些国家；允许在 X 定 B 务中 \ x w 确 D 或隐 = D # #，因此 > y w P I 能； J C Z 您 D 国家。

> 信息 I 能会 | 含技 u O D ； + 确性或！" 错误。此处 a = D 信息会 f 1 | D； b 些 | D；合 " 至 v f 物 D 新 f > 中。IBM I 能会在；作 y w D i v 下， f 1 对 > v f 物 y h v D z 品和(或) L 序中作 D x 和(或) | D。

> L 序 D 许 I 证 V P 人，如 { 获 C P 关 D 信息以能够: （i）在 w 自 (" D L 序 k 其 | L 序（ | 括 > L 序）之间；换信息以及 （ii）相互 9 C 已；换 D 信息， k * 系:

IBM Corporation
Department TL3B/062
P.O. Box 12195
Research Triangle Park, NC 27709-2195
U.S.A.

只要服从相 & D u n 和 u 件， | 括某些 i v 下 D 6 费，M I C = b 类信息。

> 信息中 h v D X 许 L 序和 y P I C D X 许资 O G I IBM 4 U IBM M 户协议、国际 L 序 h 计许 I 证协议或任何 H 价 D 协议 a 供 D。

P关非 IBM z品D信息G从那些z品D供&L、{G发<DywW或其|公共ICD源获CD。 IBM ；PbT过那些z品， R；能确认性能、兼容性或任何关*非 IBM z品D其|ywD准确度。P关非 IBM z品D能&问b&C致函那些z品D供&L。

f权许I证:

>信息|含C源o言写D样>&CL序， b些&CL序5wKw种Y作平(ODL序h计技u。为K*发、9C、P！*销或分发遵从Y作平((样>L序在其Oi写)D&CL序h计SZD&CL序， I以以任何形=；C6费x IBM 来4制和分发b些样>L序。b些5例"；P在yPDu件下9WbT过。因此 IBM ；能#证或5>b些L序DI？性、I服务性或功能。为K*发、9C、P！*销或分发遵从 IBM &CL序h计SZD&CL序， I以以任何形=；C6费x IBM 来4制和分发b些样>L序。

？v1>、b些样>L序D任何？分或任何派zD工作， 都X须|括如下y>D1>注意B项:

(c) (您D公>{F) (年)。bv代kD一？分来源Z IBM 公>样>L序。(c) f权yP IBM 公>， d入年}。#tyP权利。

附 < G. L 标

以下 u o G IBM 公 > 在@国或其 | 国家 D L j :

ACF/VTAM
_ 级同级间 , 网
AFP
AIX
AIXwindows
Application System/400
APPN
AS/400
CallPath
CallPath/2
CallPath SwitchServer/2
CICS
Common User Access
CUA
IBM
IMS
MVS/ESA
MVS/XA
NetView
Operating System/2
OS/2
OS/400
RACF
SAA
SP
System/370
S/370
Virtual Machine/Enterprise Systems Architecture
VM/ESA
VTAM

其 | I + 星号 (**) j 6 D公 > 、 z 品和服务 { F , I 能 G 其 | 公 > D L j 或服务 j 记。

C-bus G Corollary 公 > D L j 。

ActionMedia、LANDesk、MMX、Pentium 和 ProShare G Intel 公 > 在@国和其 | 国家中 D L j 或注 a L j 。

Java 和 HotJava G Sun Microsystems 公 > D L j 。

Microsoft、Windows、Windows NT 和 Windows 95 j 志 G Microsoft 公 > D注 a L j 。

UNIX G在@国和其 | 国家 D注 a L j , I X/Open P 限公 > 独家 X 许。

PC Direct G Ziff 通信公 > D注 a L j , " R - X许 I IBM 公 > 9 C。

w }

> w引 4 汉 o 拼音，} 字，" 文字母和X b 字符 3 序排 P 。

[A]

2 全性协议
  对话级 p 33
  会话级 p 33
  伙 i LU 验证 33
  最终C户验证 33

[B]

t N } 185

[C]

v 错
  (f 31
  发M运行记 < 31
传 d 服务，支V D E 要 156
次返回 k 172
错误处理 15

[D]

G 记处理 172
w = 162
  原因 d v 暂挂 167
动词
  取消 166
  完 I 信号 167
  指定对话类型 30
动词记 <
  目 < 172
动词X制 i
  公共字段 65
  a 构 185
动词信号
  动词X制 i
    公共字段 65
  基 > 会话动词X制 i
    ALLOCATE 78
    CONFIRM 83
    CONFIRMED 87
    DEALLOCATE 89
    FLUSH 94
    GET_ATTRIBUTES 96
    PREPARE_TO_RECEIVE 99
动词信号 *(续)*
  基 > 会话动词X制 i *(续)*
    RECEIVE_AND_POST 102
    RECEIVE_AND_WAIT 107
    RECEIVE_EXPEDITED_DATA 113
    RECEIVE_IMMEDIATE 117
    REQUEST_TO_SEND 122
    SEND_CONVERSATION 124
    SEND_DATA 128
    SEND_ERROR 132
    SEND_EXPEDITED_DATA 136
    TEST_RTS 139
    TEST_RTS_AND_POST 141
  3 象会话动词X制 i
    MC_ALLOCATE 78
    MC_CONFIRMED 87
    MC_DEALLOCATE 89
    MC_FLUSH 94
    MC_GET_ATTRIBUTES 96
    MC_PREPARE_TO_RECEIVE 99
    MC_RECEIVE_AND_POST 102
    MC_RECEIVE_AND_WAIT 107
    MC_RECEIVE_EXPEDITED_DATA 113
    MC_RECEIVE_IMMEDIATE 117
    MC_REQUEST_TO_SEND 122
    MC_SEND_CONVERSATION 124
    MC_SEND_DATA 128
    MC_SEND_ERROR 132
    MC_SEND_EXPEDITED_DATA 136
    MC_TEST_RTS 139
    MC_TEST_RTS_AND_POST 141
对话
  k + 工 11
  # V类型一致 30
  i 询类型 14
  v 错 15
  %向类型 13
  发M }] 30
  S U }] 31
  确认D传] 类型 13
  }] b | 新类型 14
  选择一 v 类型 29
  3 d 12
  I 会话携带 11
对话状，
  B 务L 序 27
对话状，转移
  非 OK 返回 k 344
  4 位状， 344

对话状，转移 *(续)*
暂挂G记状， 345
AP_ERROR C法 344
RECEIVE 动词后D状，| D
primary_rc N } 345
what_rcvd N } 345
对话（conversation）
定义 t 性 19, 20
x 入分配 k s D 2 全性 21
d v 分配 k s D 2 全性 21
队 P 级 p 非阻塞支 V
三种类型队 P 37
5 w 37

[F]

发M状， 11
返回 k，次要D 172
返回 k，主要D 172
方括号
EXR 中 k s \ x 166
分段 162
否定响&
来自 EXR 动词 165
服务 TP, 指定 { F 48

[G]

公共返回 k 323
AP_ALLOCATION_ERROR 323
AP_ALLOCATION_FAILURE_NO_RETRY 323
AP_ALLOCATION_FAILURE_RETRY 323
AP_CONVERSATION_TYPE_MISMATCH 323
AP_CONVERSATION_TYPE_MIXED 324
AP_CONV_FAILURE_NO_RETRY 324
AP_CONV_FAILURE_RETRY 324
AP_DEALLOC_ABEND 324
AP_DEALLOC_ABEND_PROG 324
AP_DEALLOC_ABEND_SVC 324
AP_DEALLOC_ABEND_TIMER 324
AP_DEALLOC_NORMAL 324
AP_PIP_NOT_ALLOWED 323
AP_PIP_NOT_SPECIFIED_CORRECTLY 323
AP_PROG_ERROR_PURGING 325
AP_PROG_ERROR_TRUNC 325
AP_SVC_ERROR_NO_TRUNC 325
AP_SVC_ERROR_PURGING 325
AP_SVC_ERROR_TRUNC 325
AP_SYNC_LEVEL_NOT_SUPPORTED 323
AP_TP_BUSY 326
AP_TP_NAME_NOT_RECOGNIZED 323
AP_TRANS_PGM_NOT_AVAIL_NO_RTRY 323
AP_TRANS_PGM_NOT_AVAIL_RETRY 323
公共返回 k *(续)*
AP_UNEXPECTED_SYSTEM_ERROR 326
公共服务动词
CONVERT 276
GET_CP_CONVERT_TABLE 272
公共服务入 Z c
ACSSVC 260
GetCsvReturnCode 265
TrnsDt 266
WinCSV 261
WinCSVAsyncCSV 263
WinCSVCleanup 262
WinCSVStartup 264
公共 }] a 构 185
关 * m 155

[H]

恢 4 会话 ' \ 168
会话 9
故O恢 4 168
会话 j 6 172
I 再C D 11
携带一 v 对话 11
伙 i LU 验证 33

[J]

基 > 会话 12
基 > 会话动词X制 i
ALLOCATE 78
CONFIRM 83
CONFIRMED 87
DEALLOCATE 89
FLUSH 94
GET_ATTRIBUTES 96
PREPARE_TO_RECEIVE 99
RECEIVE_AND_POST 102
RECEIVE_AND_WAIT 107
RECEIVE_IMMEDIATE 117
REQUEST_TO_SEND 122
SEND_CONVERSATION 124
SEND_DATA 128
SEND_ERROR 132
TEST_RTS 139
TEST_RTS_AND_POST 141
检 b k，在 EXR 中 165
S U状， 11
i \ 7

[L]

类型独 " 动词X制 i
GET_TP_PROPERTIES 67

类型独 " 动词 X 制 i *(续)*
GET_TYPE 69
RECEIVE_ALLOCATE 71
TP_ENDED 74
TP_STARTED 76
礼 Z 确认 162
，S 管理器
j 6 转移 L 序 { 19
h v 17
启动 L 序 22
k x 入分配 k s 匹配
；排队 L 序 22
排队 L 序 23
w 动协议 154
_ 辑 $ 度 12

[P]

配置信息 163

[Q]

e } 163
取消动词 166
确认， k s 32

[S]

B 务 L 序
定义 19
对话状， 27
队 P 级 p 非阻塞 37
* 发 27, 33
5 w 7
系统 h 定 > X LU X 39
写 35
选择 { F 32
k & C L 序 H O 18
支 V D I 选项 h 置 35
CPI 通信 7
}]（data）
发 M 30
S U 31

[T]

X 定 }] a 构 185
通信服务器 LU 6.2
2 全性功能 33
对 Z B 务 L 序 I C D 服务 27, 29

[X]

系统 h 定 > X LU X 39
相关因子 172
相关因子 *(续)*
检 b k 165
BID D 检 b k 166
响 & 方 = 155
协议
k + 工 y C 触发器 153
w = 152
方括号 153
}] 4 S 154
写 LUA APPC L 序
w C 动，4 S b 171
过 L 入 Z c 175

[Y]

样例 LUA 通信序 P 159
一 c }] w 12
异 = 动词完 I 159
异 # 响 & 155
异 # 终止， (f 31
& C L 序子系统
支 V Z n 33
转换 33
3 象会话 12
3 象会话动词 X 制 i
MC_ALLOCATE 78
MC_CONFIRM 83
MC_CONFIRMED 87
MC_DEALLOCATE 89
MC_FLUSH 94
MC_GET_ATTRIBUTES 96
MC_PREPARE_TO_RECEIVE 99
MC_RECEIVE_AND_POST 102
MC_RECEIVE_AND_WAIT 107
MC_RECEIVE_EXPEDITED_DATA 113
MC_RECEIVE_IMMEDIATE 117
MC_REQUEST_TO_SEND 122
MC_SEND_CONVERSATION 124
MC_SEND_DATA 128
MC_SEND_ERROR 132
MC_SEND_EXPEDITED_DATA 136
MC_TEST_RTS 139
MC_TEST_RTS_AND_POST 141
I v 人通信 支 V D I 选项 h 置 35

[Z]

暂挂，处理 166
支 V D 功能管理 E 要 156
终止、异 #、(f 31
主返回 k 172
最小化 LAN 通信？ 166
最终 C 户验证 33

A

ACSSVC 260
ACTLU 160
ACTLU 信息 167
ALLOCATE 78
APPC 入 Z c
  APPC() 50
  GetAppcConfig( ) 62
  GetAppcReturnCode( ) 63
  WinAPPCCancelAsyncRequest( ) 55
  WinAPPCCancelBlockingCall( ) 56
  WinAPPCCleanup( ) 57
  WinAPPCIsBlocking( ) 58
  WinAPPCSetBlockingHook( ) 60
  WinAPPCStartup( ) 59
  WinAPPCUnhookBlockingHook( ) 61
  WinAsyncAPPC( ) 51
  WinAsyncAPPCEx( ) 53
APPC API D动词支V
  类型独 " 动词 66
  3 象会话动词 66
APPC API 支V
  队 P 级 p 非阻塞 37
  系统 h 定 > X LU X 39
  支 V D 动词 66
  支 V D I 选项 h 置 35
APPC() 50
AP_ALLOCATION_ERROR 323
AP_ALLOCATION_FAILURE_NO_RETRY 323
AP_ALLOCATION_FAILURE_RETRY 323
AP_CONVERSATION_TYPE_MISMATCH 323
AP_CONVERSATION_TYPE_MIXED 324
AP_CONV_FAILURE_NO_RETRY 324
AP_CONV_FAILURE_RETRY 324
AP_DEALLOC_ABEND 324
AP_DEALLOC_ABEND_PROGRAM 324
AP_DEALLOC_ABEND_SVC 324
AP_DEALLOC_ABEND_TIMER 324
AP_DEALLOC_NORMAL 324
AP_PIP_NOT_ALLOWED 323
AP_PIP_NOT_SPECIFIED_CORRECTLY 323
AP_PROG_ERROR_PURGING 325
AP_PROG_ERROR_TRUNC 325
AP_SECURITY_NOT_VALID 323
AP_SVC_ERROR_NO_TRUNC 325
AP_SVC_ERROR_PURGING 325
AP_SVC_ERROR_TRUNC 325
AP_SYNC_LEVEL_NOT_SUPPORTED 323
AP_TP_BUSY 326
AP_TP_NAME_NOT_RECOGNIZED 323
AP_TRANS_PGM_NOT_AVAIL_NO_RTRY 323
AP_TRANS_PGM_NOT_AVAIL_RETRY 323
AP_UNEXPECTED_SYSTEM_ERROR 326

B

BID 信息 166
BIND
  协 L N } 161
BIND 信息
  指定 TS、FM E 要 156

C

CANCEL 163
CMSLTP 功能 k 服务 TP { 48
CMSTPN 功能, k 服务 TP { 48
CONFIRM 83
CONFIRMED 87
CONVERT 276
CPI-C
  f > 41, 48
  功能 * 要 46

D

DEALLOCATE 89

F

FLUSH 94

G

GDS 12
GetAppcConfig( ) 62
GetAppcReturnCode( ) 63
GET_ATTRIBUTES 96
GET_CP_CONVERT_TABLE 272
GET_TP_PROPERTIES 67
GET_TYPE 69

I

INITSELF 160

L

LL 字段 12
LU
  从 t 9
  独 " 9
  多 v 会话 11

LU *(续)*
类型 8
{ F 9
配置 9
5 w 8
LU X 163
LU 6.2
错误处理 15
管理会话 11
* 要运 c 13
LUA
重新启动 k 重新同 = 化 152
动词 147, 157
基 > 147
兼容性 147
, S 能& 147
9 C SNA 会话
断 * 151
启动 150
停止 151
先 v u 件 149
在 LU-LU 会话O传M }] 150
e 系 a 构 156
&C L 序 147
支V D FM E要 156
支V D TS E要 156
LU, > X k 伙 i 148
LUA 通信 3 序样 > 159
RUI 会话 158
SNA c 148
LUA 动词
异 = 动词完 I 159
* 要 157
LUA_NWSAA 发 v D完 I 信号 167
LU-SSCP 会话
(" 160

N

NOTIFY 161

R

RQE D关 * 155
RTR 信息 166
RUI
支V y P FM E要 156
支V y P TS E要 156
RUI 动词
公共动词W ? 185
LUA 动词X制 q = 185
RUI_BID 190
I 功D执行 190
RUI_BID *(续)*
错误返回 k 192
RUI_BID 动词
减Y 9 C 166
RUI_BID }] a 构 189
RUI_INIT 195
I 功D执行 196
错误返回 k 196
RUI_INIT 动词
取消 167
SSCP-LU 会话 h 置后 a x 168
RUI_INIT_STATUS 202
RUI_PURGE 199
I 功D执行 199
错误返回 k 200
RUI_PURGE 动词
取消 RUI_READ 167
RUI_READ 203
I 功D执行 204
错误返回 k 205
X断 }] 204
RUI_READ 动词
取消 167
RUI_TERM 209
I 功D执行 209
RUI_TERM 动词
取消 RUI_INIT 167
取消 RUI_WRITE 167
RUI_WRITE 212
I 功D执行 213
错误返回 k 214
RUI_WRITE 动词
取消 167

S

SDT 160
SLI 入Z c 219
SLI_BID 226
I 功D执行 226
SLI_BIND_ROUTINE 251
SLI_CLOSE 231
I 功D执行 231
SLI_OPEN 234
I 功D执行 236
SLI_PURGE 240
I 功D执行 240
SLI_RECEIVE 242
I 功D执行 243
SLI_SDT_ROUTINE 255
SLI_SEND 247
SLI_STSN_ROUTINE 253

SNA
通信支V 5
一 c }] w 12
LU 类型 6.2 支V 6
SNA 检b k 161
SNA 消息
k LUA 动词D关系 159

T

TP
服务 48
需 s 启动D服务器 8
TrnsDt 266

U

UNBIND 160

W

WinAPPCCancelAsynRequest( ) 55
WinAPPCCancelBlockingCall( ) 56
WinAPPCCleanup( ) 57
WinAPPCIsBlocking( ) 58
WinAPPCSetBlockingHook( ) 60
WinAPPCStartup( ) 59
WinAPPCUnhookBlockingHook( ) 61
WinAsyncAPPC( ) 51
WinAsyncAPPCEx( ) 53
WinAsyncCSV 263
WinCSV 261
WinCSVCleanup 262
WinCSVStartup 264

读者 b { 表

eNetwork 通 E 服务器 f 本 6.0 Windows NT f 和 eNetwork 个人通 E f 本 4.2 Windows 95 f 和 Windows NT f 客户机/服务器通 E 程 r h F

SC84-0692-00

姓 {

X 址

% 位及 ? E

g 话号 k

读者 b { 表
SC84-0692-00

k 沿此线
: 下或 [起

[起 " 封 Z　　请勿使 C 钉书机　　[起 " 封 Z

在此
y O
J 票

IBM Corporation
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC
27709-9990

[起 " 封 Z　　请勿使 C 钉书机　　[起 " 封 Z

SC84-0692-00

k 沿此线
: 下或 [起

Printed in China

SC84-0692-00