エミュレータプログラミング

Contents

本書について v
本書の対象読者 v
詳細情報 v
表記 vi
Chapter 1. エミュレーター API の概要 7
API ヘッダー・ファイルの使用 8
クリティカル・セクション 8
スタック・サイズ 8
Windows x64 プラットフォームのサポート 8
サンプルプログラム 9
Chapter 2. IBM 標準 EHLLAPI、IBM 拡張 EHLLAPI、および WinHLLAPI プログラミングの概要 11
EHLLAPI の概要 11
IBM 標準 EHLLAPI 11
WinHLLAPI 11
WinHLLAPI 対 IBM® 標準 EHLLAPI 11
IBM 拡張 EHLLAPI 対 IBM 標準 EHLLAPI 12
言語 12
EHLLAPI 呼び出し形式 12
データ構造 13
メモリー割り振り 14
EHLLAPI 戻りコード 14
コンパイルとリンク 15
ダイナミック・リンク方式 16
マルチスレッド 17
表示スペース 17
IBM® 拡張 32 ビット・インターフェースの表示スペース ID 17
表示スペースのタイプ 17
表示スペースのサイズ 18
表示スペース ID 18
ホスト接続表示スペース 18
表示スペース ID の処理 18
EHLLAPI 表示スペースのプロセス間での共用 20
マウス・アクションを使用した表示スペース内でのテキストの選択、コピー、および貼り付け 23
ASCII 略号 24
のデバッグ 27
簡単な EHLLAPI サンプル・プログラム 27
標準および拡張インターフェースの考慮事項 28
ホストの自動化のシナリオ 30
Chapter 3. EHLLAPI 関数 36
ページ・レイアウトについての規則 36
前提呼び出し 36
呼び出しパラメーター 36
戻りパラメーター 36
使用上の注意 36
EHLLAPI 関数の要約 37
Allocate Communications Buffer (123) 38
Cancel File Transfer(92) 40
Change PS Window Name (106) 41
Change Switch List LT Name (105) 42
Connect for Structured Fields (120) 44
Connect Presentation Space (1) 45
Connect Window Services (101) 47
Convert Position または Convert RowCol (99) 48
Copy Field to String (34) 51
Copy OIA (13) 55
Copy Presentation Space (5) 65
Copy Presentation Space to String (8) 69
Copy String to Field (33) 74
Copy String to Presentation Space (15) 76
Copy Presentation Space to Clipboard (35) 77
Paste Clipboard to Presentation Space (36) 79
Disconnect from Structured Fields (121) 81
Disconnect Presentation Space (2) 82
Disconnect Window Service (102) 83
EditKey インターセプト 84
Find Field Length (32) 86
Find Field Position (31) 87
Free Communications Buffer (124) 89
Get Key (51) 90
Get Request Completion (125) 94
Lock Presentation Space API (60) 96
Lock Window Services API (61) 99
Pause (18) 101
Post Intercept Status (52) 102
Query Additional Field Attribute (45) 103
Query Close Intercept (42) 104
Query Communications Buffer Size (122) 105
Query Communication Event (81) 107
Query Cursor Location (7) 108
Query Field Attribute (14) 109
Query Host Update (24) 111
Query Session Status (22) 112
Query Sessions (10) 115
Query System (20) 116

Query Window Coordinates (103)....118
Read Structured Fields (126)....120
Receive File (91)....125
Release (12)....127
Reserve (11)....128
Reset System (21)....129
Search Field (30)....130
Search Presentation Space (6)....132
Send File (90)....134
Send Key (3)....135
Set Cursor (40)....144
Set Session Parameters (9)....145
Start Close Intercept (41)....155
Start Communication Notification (80)....157
Start Host Notification (23)....160
Start Keystroke Intercept (50)....162
Start Playing Macro (110)....165
Stop Close Intercept (43)....166
Stop Communication Notification (82)....167
Stop Host Notification (25)....168
Stop Keystroke Intercept (53)....169
Wait (4)....170
Window Status (104)....171
Write Structured Fields (127)....175
Chapter 4. WinHLLAPI 拡張関数....181
WinHLLAPI 関数の要約....181
WinHLLAPI の非同期関数....181
WinHLLAPIAsync....181
WinHLLAPICancelAsyncRequest....189
初期化関数と終了関数....190
WinHLLAPI Startup....190
WinHLLAPI Cleanup....191
ブロッキング・ルーチン....191
WinHLLAPIIsBlocking....191
WinHLLAPISetBlockingHook....192
WinHLLAPIUnhookBlockingHook....193
WinHLLAPICancelBlockingCall....193
Chapter 5. PCSAPI 関数....195
PCSAPI の使用方法....195
ページ・レイアウトについての規則....195
関数の形式....195
パラメーターの形式および説明....195
戻りコード....196
pcsConnectSession....196
関数の形式....196
パラメーターの形式および説明....196
戻りコード....196
pcsDisconnectSession....196
関数の形式....197
パラメーターの形式および説明....197
戻りコード....197
pcsQueryConnectionInfo....197
関数の形式....197
パラメーターの形式および説明....197
戻りコード....197
ConnectionInfo....198
例....198
pcsQueryEmulatorStatus....198
関数の形式....198
パラメーターの形式および説明....199
戻りコード....199
pcsQuerySessionList....199
関数の形式....200
パラメーターの形式および説明....200
戻りパラメーター....200
例....201
pcsQueryWorkstationProfile....201
関数の形式....201
パラメーターの形式および説明....201
戻りコード....202
pcsSetLinkTimeout....202
関数の形式....202
パラメーターの形式および説明....202
戻りコード....202
pcsStartSession....202
関数の形式....203
パラメーターの形式および説明....203
戻りコード....203
pcsStopSession....203
関数の形式....204
パラメーターの形式および説明....204
戻りコード....204
ページ設定関数....204
制約事項....204
pcsGetPageSettings....205
pcsRestorePageDefaults....208
pcsSetPageSettings....209
プリンター設定関数....213
制約事項....213
pcsGetPrinterSettings....213
pcsSetPrinterSettings....219
Chapter 6. エミュレーター・プログラミングのためのトラブルシューティング....225
Z and I Emulator for Windows ホスト画面での部分的な EHLLAPI 入力....225

HCL Z and I Emulator for Windows の VBHLLAPI サンプルが FDCC Windows Vista で実行されない....227
Appendix A. がサポートする Query Reply データ構造 EHLLAPI....229
DDM Query Reply....229
DDM アプリケーション名の自己定義パラメーター....230
PCLK プロトコル制御の自己定義パラメーター....230
基本 DDM Query Reply の形式....231
IBM 補助装置 Query Reply....232
任意 指定パラメーター....232
直接アクセス自己定義パラメーター....233
PCLK プロトコル制御の自己定義パラメーター....234
プロダクト定義 Query Reply....234
任意 指定パラメーター....234
直接アクセス自己定義パラメーター....236
文書交換アーキテクチャー Query Reply....236
Appendix B. Communication Manager/2 EHLLAPI との相違点....238
Set Session Parameter (9)....238
セット・オプション....238
戻りパラメーター....238
EAB オプション....238
Copy OIA (13)....239
Copy String to PS (15)....240
Storage Manager (17)....240
Copy String to Field (33)....240
Get Key (51)....240
Window Status (104)....240
Query Sessions (10)....241
Connect for Structured Fields (120)....241
Allocate Communications Buffer (123)....241
ASCII 略号....241
Get Request Completion (125)....241
Appendix C. 特記事項....242
商標....243
Index....244

本書について

本書は、 HCLZ and I Emulator for Windows エミュレーター高水準言語アプリケーション・プログラム・インターフェース (EHLLAPI)、 およびZ and I Emulator for Windowsセッション API (PCSAPI) を使用するために必要なプログラミング情報を提供しています。ホスト・アクセス・クラス・ライブラリーについては、「ホスト・アクセス・クラス・ライブラリー」で説明しています。

EHLLAPI/PCSAPI を、Z and I Emulator for Windows で使用すると、ユーザーおよびプログラマーは、ワークステーション・セッションで実行中のアプリケーション・プログラムから呼び出すことのできる一連の関数によって、ホスト表示スペースにアクセスすることができます。

本書では、*Windows* は、Windows® 7、Windows® 8/8.1、Windows® 10、Windows® Server 2008、および Windows® Server 2012 を指します。特定のオペレーティング・システムにのみ適用される情報は、本文中にそのことが明記されます。

本書の対象読者

本書は、本書に記載されている API を使用するアプリケーション・プログラムを 作成するプログラマーを対象としています。

読者に Windows® の知識と経験があるものと想定して説明しています。Windows® については、詳細情報 (ページ v) にある資料のリストを参照してください。

プログラマーの方は、端末から、または端末エミュレーション・ソフトウェアが インストールされているワークステーションからのホスト・システムへの接 続についても熟知している必要があります。

使用する言語およびコンパイラーについても熟知していることを前提としています。どのようにプログラムを作成し、 コンパイルし、リンクするかの情報については、詳細情報 (ページ v)を参照して、 使用する特定の言語の適切な解説書を調べてください。

詳細情報

Z and I Emulator for Windows・ライブラリーには、以下の資料が含まれています。

- *Installation Guide*
- *Quick Beginnings*
- *Emulator User's Reference*
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Client/Server Communications Programming*
- *System Management Programming*
- ホスト・アクセス・クラス・ライブラリー
- 構成ファイル解説書

印刷された資料に加えて、Z and I Emulator for Windowsとともに提供されるハイパーテキスト ・マークアップ言語 (HTML) 文書があります。

ホスト・アクセス・クラス・ライブラリー

HACL Java HTML ファイルは、組み込みオブジェクトとしてZ and I Emulator for Windowsを使用するための ActiveX/OLE 2.0 準拠アプリケーションの作成方法を説明しています。これらのファイルは、以下のパスにあるZ and I Emulator for Windowsの製品資料に付属している Docs_Admin_Aids zipped フォルダーからアクセスできます: *ZIEWin_1.0_Docs_Admin_Aids.zip\publications\en_US\doc\hacl*

関連資料としては、以下のものがあります。

- *IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書 (N:GA23-0059)
- *IBM 5250 Information Display System Functions Reference Manual* (SA21-9247)

表記

EHLLAPI 関数 (ページ 36)、PCSAPI 関数 (ページ 195)、WinHLLAPI 拡張関数 (ページ 181) の各セクションのはじめに、API 関数 関数を説明する表があります。この表は、そのセクションで説明している関数を備えている製品について、その関数 がサポートされているかどうかを示します。Yes は、ホスト・タイプについてその関数がサポートされていることを 意味し、No は、その関数がサポートされていないことを意味します。例えば、以下の表は、3270 セッションおよび VT セッションではその関数を使用できる が、5250 セッションでは使用できないことを示しています。

3270	*5250*	*VT*
はい	いいえ	はい

第 1 章. エミュレーター API の概要

IBM® Z and I Emulator for Windows 製品は、いくつかのアプリケーション・プログラミング・インターフェース (API) を備えています。それぞれのインターフェースは、特定の関数群を備えており、 目的に応じて使用することができます。ご使用のアプリケーションの機能要件に最も適した プログラミング・インターフェースを選択してください。アプリケーションによっては、希望する結果を得るために 複数のインターフェースを使用する場合もあります。プログラミング・インターフェースには、以下のものがあります。

- **エミュレーター高水準言語 API (EHLLAPI)**:このインターフェースは、ホスト画面上の文字などのエミュレーター「表示スペース」にアクセスする機能を提供します。また、ホストへのキー・ストロークの送信、ユーザーが入力したキー・ストロークのインターセプト、ホスト・セッションの状況の照会、ファイルのアップロードとダウンロードなどの機能も提供します。このインターフェースは、ユーザーが直接介入することなく、 ホスト画面を読み取ってキー・ストロークを入力する自動化操作プログラム・アプリケーションでしばしば使用します。『EHLLAPI 関数 (ページ 36)』を参照してください。
  - **IBM®標準 HLLAPI サポート**: これは、標準プログラミング・インターフェースです。これを使用することで、ホスト・エミュレーター・セッションにプログラムでアクセスすることができます。『IBM 標準 EHLLAPI、IBM 拡張 EHLLAPI、および WinHLLAPI プログラミングの概要 (ページ 11)』を参照してください。
  - **IBM®拡張された HLLAPI サポート**: このインターフェースは、IBM® 標準 HLLAPI インターフェースに基づいたものです。既存の関数のすべてを提供しますが、データ構造は変更後のものを使用します。『IBM 標準 EHLLAPI、IBM 拡張 EHLLAPI、および WinHLLAPI プログラミングの概要 (ページ 11)』を参照してください。
  - **Windows®高水準言語 API (WinHLLAPI)**: このインターフェースは、IBM® 標準 EHLLAPI とほとんど同じ関数を提供し、 さらに、Windows® 環境を利用した拡張関数をいくつか追加しています。『IBM 標準 EHLLAPI、IBM 拡張 EHLLAPI、および WinHLLAPI プログラミングの概要 (ページ 11)』を参照してください。
- ウィンドウ・ハンドルおよびポインターを受け入れる、または返す 32 ビット API は、HCL ZIEWin では正常に動作しない可能性があります。x86 と x64 プラットフォーム間でポインターやハンドルのサイズが異なるからです。

  例:

  API Start Communication Notification (80) のバイト数 (9-12) で返される「データ・ストリング」パラメーターは、x64 プラットフォームでは切り捨てられる可能性があります。
- **Z and I Emulator for Windows セッション API (PCSAPI)**: この インターフェースは、 エミュレーター・セッションを開始、停止、および制御したり、エミュレーター設定を制御したりするために使用されます。『PCSAPI 関数 (ページ 195)』を参照してください。

  Z and I Emulator for Windows Version 1.0 では、ページ設定およびプリンター設定の制御と検索を行うための関数が追加されています。『ページ設定関数 (ページ 204)』および『プリンター設定関数 (ページ 213)』を参照してください。
- **HCL Z and I Emulator for Windows ホスト・アクセス・クラス・ライブラリー (ECL)**: ECL は、アプリケーション・プログラマーおよびスクリプト言語ライターが簡単かつ迅速にホスト・アプリケーションにアクセ

スできるようにするためのオブジェクトのセットです。Z and I Emulator for Windows では、3 つの別個の ECL 層 (C++ オブジェクト、 ActiveAutomation (OLE)、および LotusScript Extension (LSX)) をサポートしています。詳しくは、*Host Access Class Library (HACL)*を参照してください。

API ヘッダー・ファイルの使用

アプリケーション・プログラムに API ヘッダー・ファイルを組み込む前に、 オペレーティング・システムのヘッダー・ファイルを組み込む必要があります。例:

```
#include <windows.h>      // Windows main header
#include "pcsapi.h"       // ZIEWin PCSAPI header
...
```

クリティカル・セクション

ご使用のプログラムでエミュレーター API を呼び出す場合、クリティカル・セクション (**EnterCriticalSection** 関数) は慎重に使用してください。クリティカル・セクション内でのエミュレーター API 呼び出しは行わないで ください。アプリケーションの 1 つのスレッドがクリティカル・セクションを設定する 際に、別のスレッドがエミュレーター API 呼び出し内にある場合、ユーザーが クリティカル・セクションを終了するまで、その呼び出しは延期されます。

API 呼び出しの処理中は、その呼び出しが完了する、または着信データを待つことが 必要となるまで、すべてのシグナル (数値演算コプロセッサー・シグナルを除く) は 遅らされます。また、アプリケーションが処理中の API 呼び出しを完了するまで、別のプロセスから発行された **TerminateProcess** は保留されます。

スタック・サイズ

エミュレーター API は、実行時に呼び出し側プログラムのスタックを使用し ます。オペレーティング・システム、アプリケーション、および API のすべてが、 ダイナミック変数と関数仮引数のためのスタック・スペースを必要とします。API 呼び出しの時点で、少なくとも 8196 バイト (8K) のスタック・スペースが 使用可能である必要があります。アプリケーション・プログラム側で、API 用のスタック・スペースが十分に 使用可能となるように保証する必要があります。

Windows x64 プラットフォームのサポート

Microsoft® Windows® Server 2008 の x64 ベースのバージョンおよび Microsoft® Windows® 8/8.1/10 x64 Edition は、ネイティブ 64 ビット・プログラムの実行に最適化されていますが、32 ビット・ドライバーも 16 ビット・アプリケーションもサポートしていません。

これらのプラットフォームの場合、Z and I Emulator for Windows は以下のライブラリーをインストールしません。

- 16 ビット API サポート:
  - 標準 EHLLAPI 16 ビット・インターフェース
  - WinHLLAPI 16 ビット・インターフェース
  - PCSAPI 16 ビット・インターフェース

サンプルプログラム

いくつかのサンプルプログラムが提供されています。Z and I Emulator for Windows API。サンプル・プログラムのインストールを選択した場合、サンプル・プログラムは \SAMPLES ディレクトリーにインストールされます。

Note: IBM Connections は、これらのファイルを「ありのまま」で提供し、明示的にも暗示的にも、商品性や特定の目的への適合性に関する黙示の保証を含むがこれらに限定されません。

サンプルプログラムファイルには、以下のソースファイルとサポートファイルが含まれています。Z and I Emulator for Windows API:

- エミュレータの高水準言語プログラミング インターフェイス (EHLLAPI)
- PCAPI 関数

以下のファイルは、\SAMPLES ディレクトリーにインストールされます。

Table 1. サンプル プログラム ファイル

ファイル名	説明
EHLAPI32.H	IBM®標準 32 ビット EHLLAPI インクルードファイル
WHLLAPI.H	WinHLLAPI 16 ビット インクルード ファイル
HAPI_C.H	EHLLAPI インクルード ファイル
PCSAPI.H	PCSAPI インクルード ファイル
PCSCALLS.LIB	標準インターフェースのインポート ライブラリ
PCSCAL32.LIB	強化されたインターフェイスのライブラリをインポート
EHLAPI32.LIB	IBM®標準 32 ビット EHLLAPI インターフェースのインポートライブラリ
WHLLAPI.LIB	WinHLLAPI 16 ビット インターフェイス用のインポート ライブラリ
WHLAPI32.LIB	WinHLLAPI 32 ビット インターフェイス用のインポート ライブラリ

以下のサブディレクトリが \SAMPLES ディレクトリに作成されます。

Table 2. サンプル プログラムのサブディレクトリ

ファイル名	説明
ECL	sample\eCL\cpp フォルダーには、HACL CPP サンプルに関連したすべてのファイルが含まれています。 sample\eCL\vb フォルダには、HACL VB.Net サンプルに関連するすべてのファイルが含まれています。

Table 2. サンプル プログラムのサブディレクトリ (continued)

ファイル名	説明
HLLSMP	EHLLAPI を使用してキーストロークを要求する方法と、VM システム (X86) にログオンする方法を示します。 ログオン、pastetext、sendkey 機能をサポートします。hllsmp.exe での上記の機能の使用の詳細については、hllsmp\ Readme.txt を参照してください。 (X64) です。

第 2 章. IBM 標準 EHLLAPI、IBM 拡張 EHLLAPI、および WinHLLAPI プログラミングの概要

本章では、高水準言語で書かれたアプリケーションに IBM® 標準 EHLLAPI (16 ビットおよび 32 ビット) 関数、WinHLLAPI (16 ビットおよび 32 ビット) 関数、 および IBM® 拡張 32 ビット EHLLAPI (EHLAPI32) 関数を組み込むのに必要な情報を提供します。呼び出し形式、メモリー割り振りに関する考慮事項、インターフェースの初期化の方法、およびアプリケーションのコンパイルとリンクの方法について詳し く説明します。また、簡単なサンプル EHLLAPI プログラム、およびこのプログラムを作成するのに 使用するコンパイル/リンクに関する命令も含まれています。この章の最後では、EHLLAPI インターフェースについて考えられる、一連の 用途 (シナリオ) を説明します。

EHLLAPI アプリケーションとは、EHLLAPI インターフェースを使用して ホスト 3270/5250/VT 表示スペースにアクセスする アプリケーション・プログラムのことです。表示スペースには、可視のエミュレーター文字データ、フィールドおよび属性データ、キー・ストローク・データ、その他の情報が 含まれます。

EHLLAPI の概要

以下は、HLLAPI プログラミング・インターフェースの概要です。

IBM 標準 EHLLAPI

EHLLAPI は、標準プログラミング・インターフェースです。これを使用することで、ホスト・エミュレーター・セッションにプログラムでアクセスすることができます。ホスト画面データ (文字および属性など) の読み取り、キー・ストロークの送信、 その他のエミュレーター関連の関数を実行するための関数が用意されています。

EHLLAPI インターフェースは、呼び出し点が 1 つのインターフェースです。1 つの呼び出し可能 API があり、これを介してすべての EHLLAPI 関数を 要求します。このインターフェースを呼び出すたびに、アプリケーションは、要求された関数を 識別する関数番号、データ・バッファーを指すポインター、データ・バッファー長 を指すポインター、および戻りコードを指すポインターを提供 します (EHLLAPI 呼び出し形式 (ページ 12)を参照してください)。

WinHLLAPI

WinHLLAPI は、よく知られている EHLLAPI.API を基にしたものです。これは、既存の関数をすべて含み、さらに、Windows® のメッセージ・ドリブン環境を利用する拡張関数が追加されています。HCL Z and I Emulator for Windows EHLLAPI インターフェースのユーザーは、WinHLLAPI 拡張関数を組み込まない場合には機能上の差に気がつきません 。

WinHLLAPI 拡張関数および EHLLAPI 形式から逸脱する関数については、 WinHLLAPI 拡張関数 (ページ 181)で説明します。共通関数の詳細については、EHLLAPI 関数 (ページ 36)を参照してください。

WinHLLAPI 対 IBM® 標準 EHLLAPI

WinHLLAPI の入り口記号は、 専有的に **WinHLLAPI** です。WinHLLAPI のインプリメンテーションに切り替えたいと考えている EHLLAPI ユーザーは、**hllapi** 標準入り口からの変更が必要です。新規ユーザーは、EHLLAPI 関数 (ペー

ジ 36)に記載されているすべての指示に従い、 さらに、標準の **hllapi** 入り口の代わりに **WinHLLAPI** 入り口を使用してください。

IBM 拡張 EHLLAPI 対 IBM 標準 EHLLAPI

IBM 拡張 EHLLAPI は、よく知られている EHLLAPI API に基づいています。これは、既存の関数をすべて含んでいますが、32 ビット環境を利用し、 変更されたデータ構造を使用します。標準インターフェースのユーザーが IBM® 拡張 32 ビット EHLLAPI への切り替えを望む場合、必要なのは 1 番目、3 番目、および 4 番目のパラメーター内の LPWORD を LPINT に変更することだけです。新規ユーザーは、以下のセクションに記載されているプロシージャーを使用してくだ さい。

言語

「Pascal」の呼び出し規則で DLL 内のエントリー・ポイントを呼び出すことができる プログラミング言語であれば、EHLLAPI 関数を実行するために使用できます。ただし、Z and I Emulator for Windows EHLLAPI ツールキットでは、 C++ 言語用のヘッダー・ファイルと関数プロトタイプだけが提供されています。他の言語を使用する場合には、データ構造レイアウトと呼び出し規則を 十分に理解することが必要となります。EHLLAPI ツールキットは、以下の C/C++ コンパイラーをサポートしています。

- Microsoft® Visual C/C++ バージョン 4.0 およびそれ以降

その他のほとんどの C/C++ コンパイラーもツールキットとともに 使用できます。

EHLLAPI C/C++ アプリケーションには、Z and I Emulator for Windows EHLLAPI ヘッダー・ファイル (HAPI_C.H) を組み込む必要があります。このファイルはデータ構造のレイアウトを定義し、EHLLAPI エントリー・ポイントの プロトタイプを提供します。

注: 16 ビットと 32 ビットのアプリケーションのデータ構造レイアウトは、 同じではありません (標準および拡張インターフェースの考慮事項 (ページ 28)を参照してください)。

EHLLAPI 呼び出し形式

EHLLAPI エントリー・ポイント (**hllapi**) は、常に以下の 4 つのパラメーターを用いて呼び出されます。

1. EHLLAPI 関数番号 (入力)
2. データ・バッファー (入出力)
3. バッファー長 (入出力)
4. 位置 (入力) ; 戻りコード (出力)

IBM® 標準 EHLLAPI のプロトタイプは次のとおりです。

```
[long hllapi (LPWORD, LPSTR, LPWORD, LPWORD);
```

IBM® 拡張 EHLLAPI のプロトタイプは次のとおりです。

```
[long hllapi (LPINT, LPSTR, LPINT, LPINT);
```

各パラメーターは、値ではなく、参照によって受け渡されます。したがって、関数呼び出しの各パラメーターは、値そのものではなく、値を指すポインターとなっている必要があります。例えば、EHLLAPI Query Session Status 関数の呼び出しの正しい例は、 以下のとおりです。

```
    #include "hapi_c.h"
    struct HLDQuerySessionStatus QueryData;
    int    Func, Len, Rc;
    long   Rc;

    memset(QueryData, 0, sizeof(QueryData)); // Init buffer
    QueryData.qsst_shortname = 'A';          // Session to query
    Func = HA_QUERY_SESSION_STATUS;          // Function number
    Len  = sizeof(QueryData);                // Len of buffer
    Rc   = 0;                                // Unused on input

    hllapi(&Func, (char *)&QueryData, &Len, &Rc);  // Call EHLLAPI
    if (Rc != 0) {                            // Check return code
      // ...Error handling
    }
```

hllapi 呼び出しのすべてのパラメーターはポインターであり、EHLLAPI 関数の戻りコードは、この関数の値としてではなく、4 番目のパラメーターの値で戻されます。例えば、次の例は正しく**ありません。**

```
    if (hllapi(&Func, (char *)&QueryData, &Len, &Rc) != 0) { // WRONG!
      // ...Error handling
    }
```

hllapi 関数は、IBM® 標準および拡張 EHLLAPI の場合には **long** データ・タイプを、WinHLLAPI の場合には **void** データ・タイプを戻すように定義されていますが、その値は未定義なので使用することはできません。

hllapi 呼び出しの 2 番目から 4 番目までのパラメーターは、 アプリケーションに情報を戻すことができます。各 EHLLAPI 関数の説明では (情報がある場合)、どのような情報がこれらの パラメーターに戻されるかを示しています。

データ構造

多くの EHLLAPI 関数は、アプリケーション・プログラムとの間の情報の受け渡し のために、定様式データ構造を使用します。各関数の説明では、データ構造のレイアウトを示します。EHLLAPI 関数との間で受け渡されるデータは、記述されているのとまったく 同じ、すなわちバイト単位で同じようにストレージ内に存在する必要があります。構造レイアウトは、IBM® 標準および WinHLLAPI の 16 ビットおよび 32 ビットのすべてのアプリケーションで同じであることに注意してください。IBM® 拡張 32 ビット・アプリケーションのデータ構造は、4 バイトの位置合わせでパックされます。

適切なデータの位置合わせとレイアウトを確保するために、提供されているヘッダー・ファイルとデータ構造の定義を使用することを特にお勧めします。次の例は技術的には可能ですが、お勧めしません。

```
    char QueryData[20];  // Not recommended
    ...
    Func = HA_QUERY_SESSION_STATUS;
    hllapi(&Func, QueryData, &Len, &Rc);
```

```
if (QueryData[13] == 'F') {
  // ...this is a 5250 session
}
```

この関数は次のように記述することをお勧めします。

```
#include "hapi_c.h"
struct HLDQuerySessionStatus QueryData;  // Recommended
...
Func = HA_QUERY_SESSION_STATUS;
hllapi(&Func, (char *)&QueryData, &Len, &Rc);
if (QueryData.qsst_sestype == 'F') {
  // ...this is a 5250 session
}
```

メモリー割り振り

EHLLAPI 関数はメモリーの割り振りや、解放は行いません。**hllapi** エントリー・ポイントを呼び出す前に、アプリケーション・プログラムはバッファー・スペースを必要とする EHLLAPI 関数に対して、そのスペースを事前に割り振る必要があります。バッファー・スペースは、次のようにダイナミック変数として事前に割り振 ることができます。

```
struct HLDQuerySessionStatus QueryBuff;
```

あるいは、次に示すように、C ライブラリーまたはオペレーティング・システム 関数を呼び出して割り振ることもできます。

```
struct HLDQuerySessionStatus *QueryBuff;
...
QueryBuff = malloc(sizeof(struct HLDQuerySessionStatus));
```

いずれの場合でも、EHLLAPI 関数を呼び出す前に十分なバッファー・スペースを 割り振ること、および不要になったバッファーを解放することは、 アプリケーション側の責任で行う必要があります。

EHLLAPI 戻りコード

EHLLAPI 関数 (**Convert Position** や **RowCol** (99) 関数を除く) は、**hllapi** 関数呼び出しの 4 番目のパラメーターに完了コードまたは戻りコードを戻します。この戻りコードは、要求した関数が正常に実行されたかどうかを示します。

それぞれの関数の説明において特に断りのない限り、次の表は、各戻りコード の値の意味を示しています。一部の関数では、これらの戻りコードの解釈が若干異なる場合があります。 詳細については、個々の関数に関する説明を参照してください。

表 3. EHLLAPI 戻りコード

戻りコード	説明
0	関数は正常に実行されました。または最後の呼び出しが発行された後、 更新されていません。
1	無効なホスト表示スペース ID が指定されました。指定されたセッションが接続されていないか、存在しないか、または、 論理プリンター・セッションです。

表 3. EHLLAPI 戻りコード (続く)

戻りコード	説明
2	パラメーター・エラーが発生したか、または無効な関数番号が指定されました。(詳細については、個々の関数を参照してください。)
4	宛先表示スペースが使用中の 状態 (X CLOCK 状態 (X []) または X SYSTEM 状態) のため関数の 実行が禁止されました。
5	戻りコード 4 で述べた理由以外の何らかの理由で関数の実行が 禁止されました。
6	無効なパラメーターの指定 (例えば、切り捨ての原因となる長さエラー) により データ・エラーが発生しました。
7	指定された表示スペースが無効です。
8	関数手順エラー (例えば、矛盾した関数の使用または前提となる関数の欠如) が 発生しました。
9	システム・エラーが発生しました。
10	この関数は EHLLAPI では使用できません。
11	このリソースは使用できません。
12	このセッションは停止しました。
24	ストリングが見つからないか、または表示スペースが定様式化 されていません。
25	キー・ストロークが入力キューにありませんでした。
26	ホスト・イベントが発生しました。詳細については、**Query Host Update** (24) を参照してください。
27	ファイル転送が、Ctrl+Break コマンドにより終了しました。
28	フィールドの長さが 0 でした。
31	キー・ストローク・キューがオーバーフローしました。キー・ストロークは失われました。
32	アプリケーションは、 通信のためにこのセッションに既に接続しています。
33	予約済み。
34	ホストに送ったメッセージが取り消されました。
35	ホストから送られたメッセージが取り消されました。
36	ホストとの接続が失われました。
37	インバウンド通信が使用不可になっています。
38	要求された関数は実行を完了していません。
39	他の DDM セッションが既に接続されています。
40	接続の切断の試行は正常に実行されましたが、切断時にまだ完了していない 非同期要求がありました。
41	要求されたバッファーは、他のアプリケーションによって使用されています。
42	一致する未処理の要求がありません。
43	API が既に他の EHLLAPI アプリケーションによってロックされていたか (ロック 要求の場合)、または API がロックされていませんでした (アンロック要求の場合)。

コンパイルとリンク

アプリケーション・プログラムは、動的リンク方法を使用してリンクすることができます。これは、ダイナミックリンクを実行することで、エントリポイントをリンクできることを意味します。この場合、アプリケーションはオペレーティングシステムコールを使用して正しい DLL をロードし、実行時にエントリポイントアドレスを取得します。

次の表に、動的ロードには .DLL を使用する必要があります。

インターフェース	エントリーポイント	DLL
IBM®標準 (64 ビット)	フラピ	EHLAPI32.DLL
IBM®拡張 (64 ビット)	フラピ	PCSHLL32.DLL
WinHLLAPI (64 ビット)	ウィンフラピ	WHLAPI32.DLL

ダイナミック・リンク方式

アプリケーションは、ダイナミック・リンク方式を使用して、実行時にオペレーティング・システムに対する呼び出しを行い、Z and I Emulator for Windows EHLLAPI モジュールをロードし、その中の **hllapi** エントリー・ポイントを見つけます。この方式では、アプリケーションにより多くのコードが必要となりますが、 アプリケーションはエラー条件をより一層制御できるようになります。例えば、Z and I Emulator for Windowsの EHLLAPI モジュールが見つからない場合、アプリケーションはユーザーに特定のエラー・メッセージを表示することができます。

ダイナミック・リンクを使用するには、アプリケーションは適切な Z and I Emulator for Windows モジュールをロードして、エントリー・ポイントを見つける必要があります。エントリー・ポイントは、 名前ではなく、その序数で見つけられるようにすることをお勧めします。その序数は、ヘッダー・ファイルに定義されています。次の 32 ビット Windows® コードは、IBM® 標準 32 ビット EHLLAPI モジュールをロードし、**hllapi** エントリー・ポイントを見つけて 、EHLLAPI 関数呼び出しを行います。

```
#include "hapi_c.h"

HMODULE Hmod;                                          // Handle of PCSHLL32.DLL
long (APIENTRY hllapi)(int *, char *, int *, int *);  // Function pointer
int HFunc, HLen, HRc;                                  // Function parameters
char HBuff[1];                                         // Function parameters

Hmod = LoadLibrary("PCSHLL32.DLL");                    // Load EHLLAPI module
if (Hmod == NULL) {
  // ... Error, cannot load EHLLAPI module
}

hllapi = GetProcAddress(Hmod, MAKEINTRESOURCE(ord_hllapi));
                                                      // Get EHLLAPI entry point
if (hllapi == NULL) {
  // ... Error, cannot find EHLLAPI entry point
}

HFunc = HA_RESET_SYSTEM;                               // Run EHLLAPI function
HLen  = 0;
HRc   = 0;
(*hllapi)(&Func, HBuff, &HLen, &HRc);
if (HRc != 0) {
```

```
    // ... EHLLAPI access error
}
```

マルチスレッド

HCL 拡張 EHLLAPI (32 ビット) とHCL®標準 EHLLAPI 16 ビットはプロセスごとに接続します。すべてのスレッドは、接続された同じホストセッションにアクセスします。接続を行うスレッドは、切断も行う必要があります。

HCL®標準 EHLLAPI (32 ビット) と WinHLLAPI はスレッド単位で接続します。各スレッドは、独自の接続を維持する必要があります。これにより、マルチスレッドプロセスは同時に複数の接続されたホストセッションへの接続を維持できます。これにより、WinHLLAPI プログラムを使用して異なるホスト間のデータを調整する場合、マルチプロセススキームの必要がなくなります。また、個々のスレッドに、必要に応じて接続したり切断したりする負荷を与えます。

表示スペース

多くの EHLLAPI 関数は、その関数に使用するホスト・エミュレーターを指示するために、表示スペース *ID (PSID)* を必要とします。(これは、短縮セッション *ID* とも呼ばれます。) 表示スペース ID は、A から Z までの範囲の 1 つの文字 です。

IBM® 拡張 32 ビット・インターフェースの表示スペース ID

IBM® 拡張 EHLLAPI アプリケーションの場合、 セッション ID は 3 バイト追加することによって拡張されています。これらの拡張セッション・バイトは、将来の互換性のためにゼロに設定しておく 必要があります。上記の設定を最も簡単に実行するには、EHLLAPI バッファーを必要な情報で埋める 前に、そのバッファーの内容をすべて 2 進ゼロに設定します。例えば、次の例は、セッション B の状況を照会するのに使用できます。

```
#include "hapi_c.h"
int HFunc, HLen, HRc;                                  // Function parameters
struct HLDPMWindowStatus StatusData;                   // Function parameters

Func = HA_PM_WINDOW_STATUS;
HLen = sizeof(StatusData);
HRc  = 0;

// Set data buffer to zeros and fill in request
memset(&StatusData, 0x00, sizeof(StatusData));
StatusData.cwin_shortname = 'B';     // Short session ID
StatusData.cwin_option    = 0x02;    // Query command

hllapi(&Func, (char *)&StatusData, &HLen, &HRc);
```

表示スペースのタイプ

エミュレーター・セッションは、ディスプレイ・セッションまたはプリンター・セッションとして構成することができます。EHLLAPI アプリケーションは、PC400 のプリンター・セッションまたは ルーター・セッションと接続することはできません。**Query Sessions (10)** 関数を使用して、特定のセッションのタイプを判別することができます。

表示スペースのサイズ

エミュレーター・ディスプレイ・セッションは、1920 バイト (24x80 の画面 サイズ) から 9920 バイト (62x160 の画面サイズ) までの画面サイズの範囲で 構成することができます。**Copy PS to String (8)** のような一部の EHLLAPI 関数は、表示スペース全体を保持できるくらいのストレージを割り振るよう、アプリケーションに要求します。特定のセッションのための表示スペースのサイズは、**Query Session Status (22)** 関数を使用して得ることができます。

表示スペース ID

EHLLAPI 関数は、一度に 1 つの表示スペースとのみ 対話します。表示スペース ID (PSID) は、関数を操作する特定の表示スペースを識別するために使用します。

関数によっては、**Connect Presentation Space** (1) 関数への、先行する呼び出しに PSID が入っている場合があります。また、別の関数では、PSID は呼び出し側のデータ・ストリング・パラメーターに入っています。

ホスト接続表示スペース

ホスト表示スペース (またはセッション) への接続は、 **Connect Presentation Space** (1) および **Disconnect Presentation Space** (2) 関数によって制御されます。いくつかの関数は、接続の状況によって実行できるかどうかが決まります。また、接続の状況は PSID の定義方法にも影響を与えます。次に、ホスト表示スペースへの接続状況を制御する方法に ついて説明します。

- 常に、ホスト接続表示スペースが 1 つも存在しないか、 ホスト接続表示スペースが 1 つだけ存在するかのいずれかになります。
- デフォルトのホスト接続表示スペースは ありません。
- 接続すると、ホスト接続表示スペースが 1 つだけ存在するようになります。接続されたホスト表示スペースは、接続関数 の呼び出しデータ・ストリング・パラメーター内で識別されます。
- 接続を行う呼び出し以降の呼び出しは、呼び出しを行うごとに切断することなく実行できます。この場合も、ホスト接続表示スペースが 1 つだけ存在します。接続されたホスト表示スペースは、この場合も接続関数 の呼び出しデータ・ストリング・パラメーター内で識別されます。
- 切断すると、ホスト接続表示スペースはなくなります。この規則は、接続を行う呼び出しを連続して行った場合にも、接続を行う呼び出しを一度だけ行った場合にも 適用されます。
- 論理プリンター・セッションに接続することはできません。

表示スペース ID の処理

PSID は、関数を操作するホスト表示スペース (またはセッション) を指定するために使用されます。PSID の処理方法は、次の 2 つの要因の影響を受けます。

1. PSID を指定するために使用される方法
    a. **Connect Presentation Space** (1) 関数への、先行する呼び出しの呼び出しデータ・ストリング・パラメーターとして。
    b. 実行する関数の呼び出しデータ・ストリングの 1 文字として。処理は、その文字が次のどちらであるかによって異なります。
        - *A* から *Z* までの文字
        - ブランクまたは NULL
2. ホスト表示スペースへの接続状況

以下の項では、これら 2 つの要因の種々の組み合わせに対する PSID の処理方法を 説明します。

接続を必要とする関数に対する PSID の処理

関数によっては、ホスト接続表示スペースとのみ対話するものが あります。これらの関数では前提呼び出しとして **Connect Presentation Space** (1) 関数が必要です。これらの関数の PSID は、**Connect Presentation Space** (1) 関数および **Disconnect Presentation Space** (2) 関数によって次のように判別されます。

- ホスト接続表示スペースが存在しない場合、これらの関数は どの表示スペースとも対話を行いません。戻りコード 1 が生成されます。
- ホスト接続表示スペースが 1 つある場合、これらの関数は、 最後の **Connect Presentation Space** (1) 関数呼び出しの呼び出しデータ・ストリング・パラメーターに指定された表示スペースと対話を行います。

接続を必要としない関数に対する PSID の処理

一部の関数は、接続されているかいないかにかかわらず、 ホスト表示スペースと対話します。これらの関数では、PSID を呼び出しデータ・ストリング・パラメーターに指定する ことができます。それらは以下のとおりです。

- **Connect Presentation Space** (1)
- **Convert Position RowCol** (99)
- **Get Key** (51)
- **Post Intercept Status** (52)
- **Query Close Intercept** (42)
- **Query Host Update** (24)
- **Query Session Status** (22)
- **Start Close Intercept** (41)
- **Start Host Notification** (23)
- **Start Keystroke Intercept** (50)
- **Stop Close Intercept** (43)
- **Stop Host Notification** (25)
- **Stop Keystroke Intercept** (53)

最初の 2 つを除く上記の関数では、次のいずれかを使用して PSID を指定できます。

- *A* から *Z* までの文字
- ブランクまたは NULL

最初の 2 つの関数では、文字を使用して PSID を指定する必要があります。

ホスト接続表示スペースが存在しない場合、次の規則が 適用されます。

- ブランクまたは NULL ではなく、文字を使用して PSID を指定すると、その関数 は任意のホスト表示スペースと対話を行うことができます。
- ブランクまたは NULL を使用して PSID を指定すると、戻りコード 1 が 生成されます。関数は実行されません。
- 文字を使用して PSID を指定すると、PS 接続要求を除き、 ホスト接続表示スペースは確立されません。

ホスト接続表示スペースが 1 つだけ存在する場合は、 次の規則が適用されます。

- 文字を使用して PSID を指定すると、その関数は任意の ホスト表示スペースと対話を行うことができます。
- ブランクまたは NULL を使用して PSID を指定すると、**Connect Presentation Space** (1) 関数に対する最後の呼び出しで識別された表示スペース内で関数が実行されます。
- 文字を使用して PSID を指定すると、PS 接続要求を除き、 ホスト接続表示スペースの確立済み PSID は変更されません。

プリンター・セッションでは、次の関数を使用できます。

- **Start Host Notification** (23)
- **Query Host Update** (24)
- **Stop Host Notification** (25)

EHLLAPI 表示スペースのプロセス間での共用

アプリケーションが共用をサポートしている場合 (つまり、アプリケーションが一緒に動作するよう開発されているか、またはお互いの動作が予測可能な場合[1])、複数の EHLLAPI アプリケーションで 1 つの表示スペースを共用することができます。共用をサポートするアプリケーションを 判別するには、EHLLAPI アプリケーションに次のタイプの 1 つを指定します。

- 監視
- 読み取り特権のある排他的書き込み
- 読み取り特権のない排他的書き込み
- 共用書き込み
- 読む

1. これは、2 つの EHLLAPI プログラムが同時に同じ表示スペースを取得しようとすることはない、表示スペースが使用可能になるまでプログラムを待ち状態にすることができるようなロジックがプログラムに組み込まれている、あるいは、アプリケーションは他のアプリケーションをロックアウトするような方法ではセッションを使用しないということです。

共用アクセスのタイプは、**Set Session Parameters** (9) 関数呼び出しで、各関数に対して次の読み取りおよび書き込み共用オプションを設定することにより定義できます。

SUPER_WRITE

アプリケーションは、共用が可能で書き込みアクセス許可を有する 他のアプリケーションに同時に同じ表示スペースに接続する ことを許可します。この関数を呼び出したアプリケーションでは、 監視タイプの関数が実行されますが、表示スペースを 共用する他のアプリケーションに対してエラーは生成されません。

WRITE_SUPER

アプリケーションは、書き込みアクセスを要求し、監視アプリケーション だけにその表示スペースへ同時に接続することを許可します。これはデフォルト値です。

WRITE_WRITE

アプリケーションは、書き込みアクセスを要求し、 その動作が予測可能なパートナーまたは他のアプリケーションに 表示スペースを共用することを許可します。

WRITE_READ

アプリケーションは、書き込みアクセスを要求し、読み取り専用関数を実行 する他のアプリケーションに表示スペースを共用することを許可します。また、表示スペースのコピーおよび他の読み取り専用操作を 通常通り実行することもできます。

WRITE_NONE

アプリケーションは、表示スペースを排他使用します。監視アプリケーションも含めて、他のアプリケーションは、表示スペースを共用できません。また、表示スペースのコピーおよび他の 読み取り専用操作を通常通り実行することもできます。

READ_WRITE

アプリケーションは、表示スペースをモニターするための読み取り アクセスのみを要求し、読み取りまたは書き込み (あるいはその両方の) 関数を 実行する他のアプリケーションに表示スペースを共用することを 許可します。また、表示スペースのコピーおよび他の読み取り専用操作を 通常通り実行することもできます。

注: 表示スペースは、1 つのプロセスのスレッドの間では 共用できません。

表 4. EHLLAPI 読み書き共用オプションの組み合わせ

呼び出すアプリケーション	Super_Write	Write_Super	Write_Write	Write_Read	Write_None	Read_Write
Super_Write	はい	はい	はい	いいえ	いいえ	はい

表 4. EHLLAPI 読み書き共用オプションの組み合わせ (続く)

呼び出すアプリケーション	Super_Write	Write_Super	Write_Write	Write_Read	Write_None	Read_Write
Write_Super (デフォルト値)	はい	いいえ	いいえ	いいえ	いいえ	いいえ
Write_Write	はい	いいえ	はい	いいえ	いいえ	はい
Write_Read	いいえ	いいえ	いいえ	いいえ	いいえ	はい
Write_None	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
Read_Write	はい	いいえ	はい	はい	いいえ	はい

互換性のある読み書きアクセス・オプションを指定することの他に、一緒に動作しても他のアプリケーションに同じ表示スペースを使用させないようにするアプリケーションでは、**Set Session Parameters** (9) 関数呼び出しの中で、キーワード KEY$nnnnnnnn をオプションで定義することができます。このキーワードにより、同じキーワードを使用しているアプリケーションだけが、表示スペースを共用することができるようになります。

注:

1. **Start Keystroke Intercept** (50) 関数は共用できません。一度に 1 つのアプリケーションだけが、キー・ストロークをトラップすることが できます。
2. **Connect To Presentation Space** (1) 関数および **Start Keystroke Intercept** (50) 関数は、 共通サブシステム関数を共用します。あるアプリケーションによって、これら 2 つの関数のうちのいずれかを共用するという要求が出され、 その要求が正常に実行されると、別のアプリケーションがこれら 2 つの関数についての要求を出した場合に 影響を受けます。例えば、アプリケーション A が KEY$abcdefgh をキーワードとして Write_Read アクセスで **Connect To Presentation Space** (1) を要求し、その要求が正常に実行された場合、アプリケーション B による **Connect To Presentation Space** (1) または **Start Keystroke Intercept** (50) についての要求は、両方のアプリケーションに互換性のある読み書きオプションを設定している場合にのみ正常に実行されます。

表 5. 前提関数およびそれに依存する関数

前提呼び出し	関数	アクセス
Allocate Communications Buffer (120)	Free Communication Buffer (120)	N/A
Connect Window Service (101)	Change PS Window Name (106) Change Switch List Name (105) Disconnect Window Service (102) Query Window Service (103) Window Status (104)	Write Read Query=Read Set=Write Write
Connect Presentation Space (1)	Copy Field to String (34) Copy OIA (13) Copy Presentation Space (5) Copy Presentation Space to String (8) Copy Presentation Space to Clipboard (35) Copy String to Field (33) Copy String to Presentation Space (15) Disconnect Presentation Space (2) Find Field Length (32) Find Field Position (31) Query Cursor	Read Read Read Read Read Write Write Write Read Read Read Read Write Write Write

表 5. 前提関数およびそれに依存する関数 (続く)

前提呼び出し	関数	アクセス
	Location (7) Query Field Attribute (14) Paste Clipboard to Presentation Space (36) Release (12) Reserve (11) Search Field (30) Search Presentation Space (6) Send key (3) Set Cursor (40) Start Playing Macro (110) Wait (4)	Read Read Read Write Write Read
Connect Structured Field (120)	Disconnect Structured Field (121) Get Request Completion (125) Read Structured Field (126) Write Structured Field (127)	N/A
Read Structured Field (126)	Get Request Completion (125)	N/A
Start Close Intercept (41)	Query Close Intercept (42) Stop Close Intercept (43)	N/A
Start Host Notification (23)	Query Host Update (24) Stop Host Notification (25)	
Start Keystroke Intercept (50)	Get Key (51) Post Intercept Status (52) Stop Keystroke Intercept (53)Send Key (3) 編集キー・ストロークが送信された場合 (編集キー・ストロークのサポートは、拡張モードで利用可能)	N/A
Write Structured Field (127)	Get Request Completion (125)	N/A

表示スペースのロック

共用表示スペースを指定した場合でも、**Lock Presentation Space API** (60) 関数または **Lock Windows® Services API** (61) 関数を使用すれば、アプリケーションは、表示スペースの排他的制御を得ることができます。これらの関数によってロックされた表示スペースを使用したい という、他のアプリケーションからの要求は、キューに入れられます。 ロックをかけた側のアプリケーションが表示スペースをアンロック すると、キューの要求は先入れ先出し法 (FIFO) に基づいて処理されます。

表示スペースをロックしたアプリケーションが、**Unlock** オプションを指定する同じ呼び出し、または **Reset System** (21) 呼び出しを使用してアンロックしない場合、そのアプリケーションが終了するか、またはセッションが停止するまで、 ロックは解除されません。

マウス・アクションを使用した表示スペース内でのテキストの選択、コピー、および貼り付け

表示スペースでは、次のマウス・アクションを使用できます。

- 左マウス・ボタンのダブルクリックによる単語の選択。
- 右マウス・ボタンのクリックによる選択された単語のコピー。
- 右マウス・ボタンのダブルクリックによるコピーされた単語の貼り付け。

ASCII 略号

ホストのキーボードで入力するキー・ストロークには、対応する ASCII 値が あります。キー・ストロークに対する **Get Key** (51) 関数の応答の仕方は、キーが定義されているか、またはキーが ASCII 値または ASCII の略号として定義されているかどうかによって異なります。

1 つのセッションのキーボードで、別のセッションが必要とする一部の コードを作成することができない場合もあります。そのようなコードを表す ASCII 略号は、**Send Key** (3) 関数のデータ・ストリング・パラメーターに入れることができます。

Send Key (3) 関数および **Get Key** (51) 関数によって、ASCII 値または使用可能なキーでは表すことができないキー・ストロークをセッションで交換することができます。キーボードから生成できる略号のセットが提供されます。これらの略号によって、ASCII 文字を使用して、ワークステーションのキーボードの 特殊ファンクション・キーを表すことができます。

シフトなしキーの略号は、エスケープ文字とその後に続く省略形からなります。また、これはシフト・キー全体、上段シフト、前面 (Alt)、Ctrl にも あてはまります。シフト付きキーの略号は、シフト・キーの略号とその後に続くシフトなしキーの 略号からなります。したがって、シフト付きキーの略号は、エスケープ文字、省略形、エスケープ文字、 省略形の 4 文字の順序になります。

デフォルトのエスケープ文字は `@` です。エスケープ文字の値は、**Set Session Parameters** (9) 関数の `ESC=c` オプションを指定して他の任意の文字に変更することもできます。ただし、以下の説明では、デフォルトのエスケープ文字を使用しています。

ASCII 文字セットにないシフト標識は、ホスト・アプリケーションに対して、 次のように 2 バイトの ASCII の略号で表されます。

上段シフト	@S
[Alt]	@A
Ctrl	@r

これらのシフトの標識に対する略号は、アプリケーションで別々に受信される ことはありません。同様に、アプリケーションがシフト標識の略号を別々に送信することもありません。シフト標識の略号には、常にシフトなし標識文字または略号が伴います。

使用する省略形によって、特殊キーの略号が覚えやすくなります。最もよく使用するキーには、英字キー・コードを使用しています。例えば、クリア・キーは *C*、 タブ・キーは *T* などです。英字の大文字と小文字は、それぞれ異なるキーの省略形となっていることに 注意してください。

以下の説明では、これらの関数の使用方法について説明します。

一般

定義済みのすべてのキーは次のいずれかで表されます。

- 1 バイトの ASCII 値。これは 256 のエレメントの ASCII 文字セットの一部です。
- 2 バイト、4 バイト、または 6 バイトの ASCII の略号。

ASCII 文字として定義したキーを表すには、その文字に対応する 1 バイト の ASCII 値を使用します。

関数として定義したキーを表すには、その関数に対応する 2 バイト、4 バイト、 または 6 バイトの ASCII の略号を使用します。例えば、後退タブ・キーを表すには、`@B` を使用します。PF1 を表すには、`@1` を使用します。入力消去を表すには、`@A@F` を使用します。次のリストを参照してください。

`@B`	左タブ	`@0`	ホーム	`@h`	PF17
`@C`	クリア	`@1`	PF1/F1	`@i`	PF18
`@D`	削除	`@2`	PF2/F2	`@j`	PF19
`@E`	Enter	`@3`	PF3/F3	`@k`	PF20
`@F`	EOF 消去	`@4`	PF4/F4	`@l`	PF21
`@H`	ヘルプ (PC400)	`@5`	PF5/F5	`@m`	PF22
`@I`	挿入	`@6`	PF6/F6	`@n`	PF23
`@J`	ジャンプ	`@7`	PF7/F7	`@o`	PF24
`@L`	カーソル左移動	`@8`	PF8/F8	`@q`	終了
`@N`	改行	`@9`	PF9/F9	`@u`	前ページ (PC400)
`@O`	スペース	`@a`	PF10/F10	`@v`	次ページ (PC400)
`@P`	印刷	`@b`	PF11/F11	`@x`	PA1
`@R`	リセット	`@c`	PF12/F12	`@y`	PA2
`@T`	右タブ	`@d`	PF13	`@z`	PA3
`@U`	カーソル上移動	`@e`	PF14	`@@`	@ (at) 記号
`@V`	カーソル下移動	`@f`	PF15	`@$`	カーソル切り替え
`@Z`	カーソル右移動				

`@A@C`	テスト (PC400)	`@A@e`	ピンク (PC/3270)
`@A@D`	ワード削除	`@A@f`	緑 (PC/3270)
`@A@E`	フィールドの終了	`@A@g`	黄 (PC/3270)
`@A@F`	入力消去	`@A@h`	青 (PC/3270)
`@A@H`	システム要求	`@A@i`	空色 (PC/3270)
`@A@I`	挿入切り替え	`@A@j`	白 (PC/3270)
`@A@J`	カーソル選択	`@A@l`	ホスト・カラーのリセット (PC/3270)
`@A@L`	高速カーソル左移動	`@A@t`	印刷 (ワークステーション)
`@A@Q`	警告	`@A@u`	ロールアップ (PC400)
`@A@R`	装置取り消し (DvCnl)	`@A@v`	ロールダウン (PC400)
`@A@T`	表示スペース印刷	`@A@y`	正方向ワード・タブ
`@A@U`	高速カーソル上移動	`@A@z`	逆方向ワード・タブ
`@A@V`	高速カーソル下移動	`@A@-`	フィールド - (PC400)

@A@Z	高速カーソル右移動	@A@+	フィールド + (PC400)
@A@9	反転表示	@A@<	レコード・バックスペース (PC400)
@A@b	下線 (PC/3270)	@S@E	ホストでの表示スペース印刷 (PC400)
@A@c	反転表示のリセット (PC/3270)	@S@x	重複
@A@d	赤 (PC/3270)	@S@y	フィールド・マーク

注:

1. 最初の表の最初の @ 記号はエスケープ文字を表します。2 番目の表の 1 つ目と 2 つ目の @ 記号はエスケープ文字です。@ 記号はデフォルトのエスケープ文字です。エスケープ文字の値は、**Set Session Parameters** (9) 関数の ESC=c オプションを使用して変更することもできます。

   エスケープ文字を # に変更すると、後退タブ・キー、Home キー、および入力消去キーを表すために使用するリテラル順序はそれぞれ、#B、#0、および #A#F になります。

   また、@ 記号を表すために使用するリテラル順序は、#@ になります。
2. 印刷画面の略号 (つまり、@P または @A@T) を送信する場合は、呼び出しデータ・ストリングの終わりにこの記号を入れてください。
3. 装置取り消しの略号 (つまり、@A@R) を送信する場合には、その略号はエラー・メッセージを伴わずに渡されます。ただし、ローカル・コピーは停止されません。

Get Key (51) 関数

端末オペレーターが ASCII 文字として定義したキーを入力すると、 ホスト・アプリケーションはその文字に対応する 1 バイトの ASCII 値を 受信します。

オペレーターが関数として定義したキーをタイプすると、ホスト・アプリケーションはその関数に対応する 2 バイト、4 バイト、または 6 バイトの ASCII の略号を 受信します。例えば、**後退タブ**・キーをタイプすると、@B を受信します。**PF1** を押すと、@1 を受信します。**入力消去** を押すと、@A@F を受信します。

オペレーターが定義済みのシフト・キーの組み合わせをタイプすると、 ホスト・アプリケーションは、ASCII 文字、または定義済みの文字や関数に対応 する 2 バイト、4 バイト、6 バイトのいずれかの ASCII の略号を受信します。

オペレーターが定義されていない個々のキーをタイプすると、**Get Key** (51) 関数が戻りコード 20 を戻し、ホスト・アプリケーションには何も送信されません。

Get Key (51) 関数は、ホスト・アプリケーションに送信されたすべての文字と略号の前に 2 つの ASCII 文字を付けます。最初の ASCII 文字は、キー・ストロークを送信する先の ホスト表示スペースの PSID です。その他の文字は、ASCII、特殊シフトまたは略号の場合には、*A*、*S*、または *M* となります。『戻りパラメーター (ページ 91)』を参照してください。

Send Key (3) 関数

ASCII 文字を別のセッションに送信するには、その文字を **Send Key** (3) 関数のデータ・ストリング・パラメーターに入れてください。

ファンクション・キーを別のセッションに送信するには、その関数の ASCII の略号を **Send Key** (3) 関数のデータ・ストリング・パラメーターに入れてください。

Send Key (3) 関数が認識されない略号をホスト・セッションに送信すると、そのキーを拒否する戻りコードが戻されます。

のデバッグ

EHLLAPI アプリケーションをデバッグする際の援助機能として、Z and I Emulator for Windowsのトレース機能を使用することができます。この機構は、すべての EHLLAPI 呼び出し、パラメーター、戻り値、および戻りコードのログを生成します。トレース機能の使用法について詳しくは、「*Administrator's Guide and Reference*」を参照してください。

簡単な EHLLAPI サンプル・プログラム

次の Windows® サンプル・アプリケーションでは、文字ストリング "Hello World!" がホスト・セッション 'A' の最初の入力フィールドに入力されます。

```
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>
#include "hapi_c.h"

int main(char **argv, int argc) {
  int HFunc, HLen, HRc;
  char HBuff[1];
  struct HLDConnectPS ConnBuff;
  // Send Key string for HOME+string+ENTER:
  char SendString[] = "@0Hello World!@E";

  HFunc = HA_RESET_SYSTEM;
  HLen  = 0;
  HRc   = 0;
  hllapi(&HFunc, HBuff, &HLen, &HRc);
  if (HRc != HARC_SUCCESS) {
    printf("Unable to access EHLLAPI.\n");
    return 1;
  }

  HFunc = HA_CONNECT_PS;
  HLen  = sizeof(ConnBuff);
  HRc   = 0;
  memset(&ConnBuff, 0x00, sizeof(ConnBuff));
  ConnBuff.stps_shortname = 'A';
  hllapi(&HFunc, (char *)&ConnBuff, &HLen, &HRc);
  switch (HRc) {
    case HARC_SUCCESS:
```

```
      case HARC_BUSY:
      case HARC_LOCKED: // All these are OK
        break;
      case HARC_INVALID_PS:
        printf("Host session A does not exist.\n");
        return 1;
      case HARC_UNAVAILABLE:
        printf("Host session A is in use by another EHLLAPI application.\n");
        return 1;
      case HARC_SYSTEM_ERROR:
        printf("System error connecting to session A.\n");
        return 1;
      default:
        printf("Error connecting to session A.\n");
        return 1;
    }

    HFunc = HA_SENDKEY;
    HLen  = strlen(SendString);
    HRc   = 0;
    hllapi(&HFunc, SendString, &HLen, &HRc);
    switch (HRc) {
      case HARC_SUCCESS:
        break;
      case HARC_BUSY:
      case HARC_LOCKED:
        printf("Send failed, host session locked or busy.\n");
        break;
      default:
        printf("Send failed.\n");
        break;
    }

    HFunc = HA_DISCONNECT_PS;
    HLen  = 0;
    HRc   = 0;
    hllapi(&HFunc, HBuff, &HLen, &HRc);

    printf("EHLLAPI program ended.\n");
    return 0;
}
```

このアプリケーションは、次のコマンドで作成することができます。

```
  nmake /a all
```

標準および拡張インターフェースの考慮事項

特定のプラットフォーム上の標準と拡張の EHLLAPI インターフェースの間 に機能上の相違はありません。しかし、他の面で重要な違いがあります。

- 拡張 EHLLAPI インターフェースは、表示スペース ID (PSID) を 1 バイトから 4 バイトに拡張します。現時点では追加バイトは使用されませんが、今後出されるバージョンの 拡張 EHLLAPI との互換性を確実にするために、ご使用のアプリケーションで 追加バイトを 2 進ゼロに設定してください。
- EHLLAPI 関数との間で受け渡されるメモリー・バッファー内の データ・エレメントの位置 (オフセット) が異なります。拡張 EHLLAPI のデータ・エレメントは、ダブルワード境界に位置合わせされます。標準 EHLLAPI のデータ・エレメントは、特定の方法で位置合わせされることはありません。EHLLAPI アプリケーションをコード化して、オフセット (バイト) 値によって バッファー内のデータを設定したり、あるいは取り出したりすることはできません。代わりに、HAPI_C.H ファイルで提供されるデータ構造を使用して、 データ・エレメントを設定したり、取り出したりすることができます。これにより、16 ビットと 32 ビットの両方のプログラムについて、 正しい位置からデータを設定したり、取り出したりすることが保証されます。

EHLLAPI データ・バッファーを 2 進ゼロで事前に埋めて、HAPI_C.H で提供される データ構造を使用すると、ソース・コードを変更することなく、標準または拡張で 稼働するようにアプリケーションをコンパイルすることができます。例えば、次のセクションのコードは、標準 EHLLAPI に対して機能しますが、 拡張 EHLLAPI に対しては機能しません。

```
   #include "hapi_c.h"
   ...
   int Func, Len, Rc;
   char Buff[18];
   char SessType;

   Func = HA_QUERY_SESSION_STATUS;   // Function
   Len  = 18;                        // Buffer length
   Rc   = 0;
   Buff[0] = 'A'                     // Session to query
   hllapi(&Func, Buff, &Len, &Rc);   // Execute function

   SessType = Buff[9];               // Get session type
   ...
```

上記の例は、拡張 EHLLAPI アプリケーションとしてコンパイルした場合は 機能しません。その理由は次のとおりです。

- このアプリケーションは、拡張セッション ID バイトをゼロに設定していません。
- この関数のバッファー長は、18 ではなく、20 です。
- セッション・タイプ標識は、データ・バッファー内のオフセット 9 ではなく、 オフセット 12 です。

同じ関数を、標準または拡張オペレーションでコンパイルした場合に、正し く機能するように書いたものを以下に示します。変更された行は、 > で示しています。

```
   #include "hapi_c.h"
   ...
   int Func, Len, Rc;
>  struct HLDQuerySessionStatus Buff;
   char SessType;

   Func = HA_QUERY_SESSION_STATUS;   // Function
>  Len  = sizeof(Buff);              // Buffer length
   Rc   = 0;
```

```
>  memset(&Buff, 0x00, sizeof(Buff));// Zero buffer
>  Buff.qsst_shortname = 'A';        // Session to query
   hllapi(&Func, (char *)&Buff, &Len, &Rc);   // Execute function

>  SessType = Buff.qsst_sestype;     // Get session type
   ...
```

ホストの自動化のシナリオ

ここで示すサンプル・シナリオは、EHLLAPI を使用して簡素化できる操作について、 概念的な情報を提供します。シナリオでは、以下の領域で EHLLAPI のプログラム式操作が実行できる 任務を示しています。

- 以下のことを含むホスト・システムの操作
  - 検索機能
  - キー・ストロークの送信
- 以下のことを含む分散処理
  - データの抽出
  - ファイル転送
- インターフェースの統合

シナリオ 1.検索機能

通常のホスト・システム・トランザクションには、次の 4 つのフェーズがあります。

1. トランザクションを開始する。
2. ホスト・システムの応答を待つ。
3. 予期した応答であるかどうかを確認するため応答を分析する。
4. 応答からデータを抽出し、使用する。

プログラム式操作では、一連の EHLLAPI 関数を使用してこれらの処置をまねる ことができます。ホスト・システム・トランザクションの正しい開始点を決定した後で、プログラム式操作は **Search Presentation Space** (6) 関数を呼び出して、どのキーワード・メッセージまたはプロンプト・メッセージを画面に表示するかを決めることができます。

次に、プログラム式操作は、**Send Key** (3) 関数を使用して、 データをホスト・システムのセッションに入力し、ホスト・システム・トランザクションを実行します。次に、プログラム式操作は、以下のことを実行できます。

- X CLOCK、X []、または X SYSTEM 条件が終了するまで待機する (あるいは、端末がロック されている場合はキーボード・ロック状態を返す) **Wait** (4) 関数を使用します。

  キーボードの使用が禁止されている場合、EHLLAPI プログラムは **Copy OIA** (13) 関数を呼び出して、エラー条件についての詳細情報を取得することができます。

- **Search Presentation Space** (6) 関数を使用して、適切な応答を受け取ったかどうかを検証するために、予期されるキーワードを検索します。
- **Copy Presentation Space to String** (8) 関数 (または、いくつかあるうちのいずれかのデータ・アクセス関数) を使用して、必要なデータを抽出します。

Search Presentation Space (6) 関数は、端末オペレーターの別のタスクをシミュレートするための重要な関数です。一部のホスト・システムは、応答するまでの間、X CLOCK、X [] または X SYSTEM モードではロック状態にならず、代わりに、すぐにキーボードをアンロック して、オペレーターが他の要求をスタックできるようにします。このような環境では、端末オペレーターは、他の何らかの表示されるプロンプト (画面のタイトルまたはラベルなど) によってデータが戻ったことを知ります。**Search Presentation Space** (6) 関数により 、EHLLAPI プログラムは待機中に表示スペースを検索することができます。また、応答待ちの間に **Pause** (18) 関数を呼び出すと、他の DOS セッションで中央演算処理装置のリソースを共用できるようになります。**Pause** (18) 関数にはオプションがあり、それによって、EHLLAPI プログラムは、ホスト・システムの更新イベントが生ずるのを待つことができます。

妥当なタイムアウト期間を過ぎてもホスト・システムのイベントが発生しない 場合、EHLLAPI プログラムは、カスタマイズされた次のようなエラー・メッセージ を呼び出すことができます。

```
No Response From Host.  Retry?
```

このような環境では、表示メッセージのわずかな変更に対しても プログラム式操作を再プログラムする必要があるので、プログラムの 改訂は重要な考慮事項になります。

例えば、端末オペレーターが

```
Enter Part Number:
```

というメッセージをプロンプトとして期待している場合に、次のメッセージを 作り出すようなアプリケーションの変更があっても、その端末オペレーターは 正しく対処できます。

```
Enter Component Number:
```

しかし、プログラム式操作はリテラル・キーワード・ストリングを探す ので、メッセージ構文の微妙な変更 (大文字と小文字の変更のような単純なもの) で あっても、プログラムは事前にプログラムされたエラー処置をとる ことになります。

シナリオ 2. キー・ストロークの送信

キー・ストロークをホスト・システムに送信するプログラムを組む際に、 注意を要する考慮事項がいくつかあります。アプリケーションの環境によっては、コマンドの発行は、 ストリングをキー入力し Enter (改行) キーを押すのと同じように簡単です。また、複数のフィールドのうちの 1 つにデータを入力できる、もっと複雑な 定様式画面を必要とするアプリケーションもあります。このような環境では、表示画面に記入する必要があるキー・ストロークを理解している必要があります。

タブ・キーの略号 (@T; 略号の詳細なリストについては、一般 (ページ 24)を参照) を使用して、フィールド間をスキップすることができます。**Send Key** (3) 関数を使用して、キー・ストロークをフィールドに送信する場合、フィールドの長さと内容を知っている必要があります。フィールドに完全に入力したときに、 次の属性バイトが自動ス

キップになっていると、カーソルは次のフィールドに移動します。次にタブを発行した場合は、さらに別のフィールドにスキップします。

同様に、キー・ストロークでフィールドを完全に埋められなかった場合、以前に入力 したデータが残ることがあります。「フィールドの終わりまで消去」(Erase EOF) コマンドを使用して、この残った データを消去してください。

シナリオ 3. 分散処理

collaborative と呼ばれる区分に入るアプリケーションもあります。このようなアプリケーションは、単一エンド・ユーザー・インターフェースを 提供しますが、その処理は、2 つ以上の異なる物理位置で実行されます。

EHLLAPI のアプリケーションは、ホスト・システムと端末ユーザー間の通信を 代行受信することによって、ホスト・システムのアプリケーションと対話する ことができます。ホスト・システムの表示スペースは、このデータを代行受信する ための手段となります。ローカルのアプリケーションは、表示スペースが更新されるたび に、またはオペレーターが AID キーを押したときにはいつでも、通知を受ける ように要求することができます。

このワークステーションのアプリケーションは、次のいずれかの方法で ホスト・システムのアプリケーションと連携することができます。

- フィールドまたは表示スペース・ベースでは、フィールドをアドレスするコピー関数 (**Copy String to Field** (33) 機能または **Copy Field to String** (34) 機能)、もしくは表示スペースからまたは表示スペースへコピーする機能 (例えば、**Copy String to Presentation Space** (15) 機能または **Copy Presentation Space to String** (8) 機能) のいずれかを使用します。
- キー・ストローク・ベースでは、**Send Key** (3) 関数を使用する。
- 大きいブロック・データに対するファイル・ベースでは、アプリケーションで EHLLAPI ファイル転送関数 (**Send File** (90) 関数または **Receive File** (91) 関数) を使用して、データまたは関数 (ロード・モジュールなど) を転送し、ローカルまたはリモートで処理する。

シナリオ 4. ファイル転送

このシナリオでは、次のように、ファイルの転送を自動化したい場合を想定します。

- 前述の検索シナリオで説明した手順を使用し、ホスト・システムの セッションへログオンして作業を開始する。
- いずれかのコピー関数 (多数の画面のデータをコピーするには非効率) を使用する代わりに、EHLLAPI プログラムでファイル転送関数の **Send File** (90) および **Receive File** (91) を呼び出してデータを転送する。
- 正常に終了した時、
    - **Send File** (90) 関数が実行を終了した場合は 、EHLLAPI プログラムは、コピー関数または **Send Key** (3) 関数を使用して、 バッチ・ジョブを発行してからログオフする。
    - **Receive File** (91) 関数が実行を終了した場合は、 EHLLAPI プログラムはローカルのアプリケーションを開始する。

シナリオ 5. 自動化

あるアプリケーションが別のアプリケーションにすべてのキー・ストロークを提供する ことができます。また、ターゲットの宛先にキー・ストロークをキーボードから 加えることができます。これを行うために、当該アプリケーションは、ターゲット・アプリケーションまたは表示スペースに向けられる可能性のある他のキー・ストローク入力ソースを (**Reserve** (11) 関数を使用して) ロックアウトし、 後から (**Release** (12) 関数を使用して) アンロックしなければならないことがあります。

任意のアプリケーションに提供されるキー・ストロークの発信元は、 アプリケーションの設計によって決まります。キー・ストロークの発信元を以下に示します。

- キーボード
- ソース・アプリケーションに組み込まれたデータ
- DOS インターフェースによって検索された 2 次記憶データ
- Z and I Emulator for Windows インターフェース

どの場合も、宛先アプリケーションに提供されるキー・ストロークは、通常の オペレーターの入力と区別がつきません。

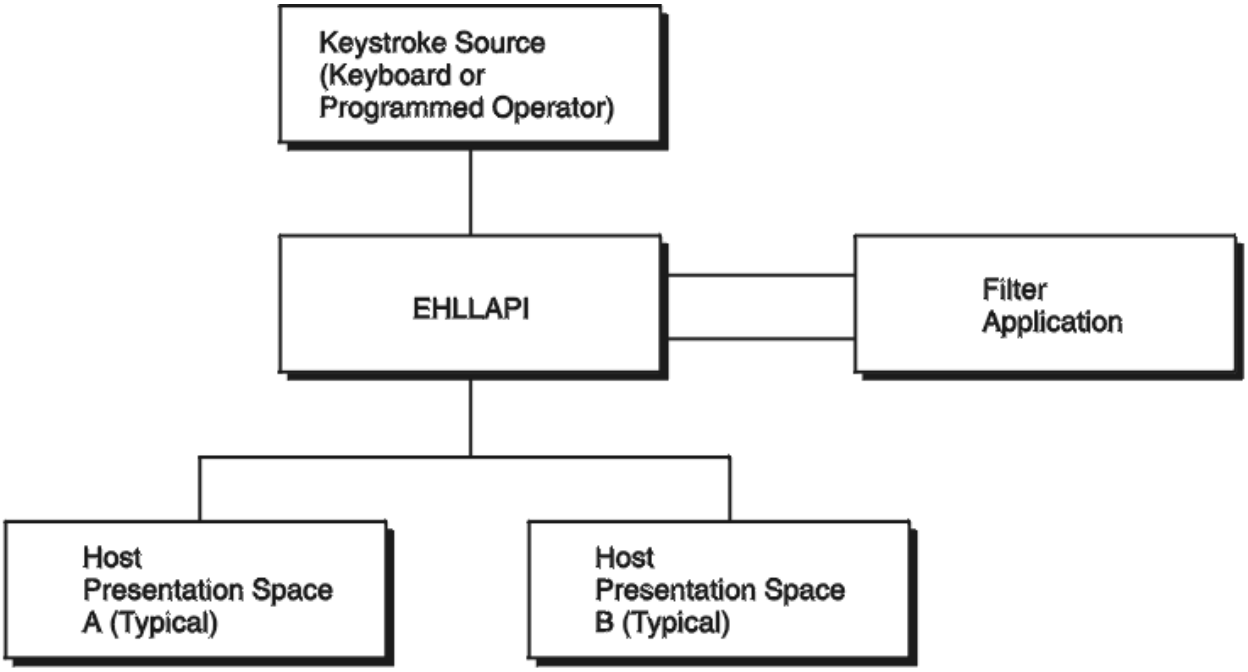
シナリオ 6. キー・ストロークのフィルター操作

フィルターとして機能するアプリケーションは、EHLLAPI から (キーボードまたは ソース・アプリケーションから) 送信された、別の宛先アプリケーションにあてた キー・ストロークを代行受信します。このキー・ストロークは、次のように処理することができます。

- 無視する (つまり、削除する)。
- 別のアプリケーションにリダイレクトする。
- 検証済み
- 変換する (例えば、大文字から小文字へ)。
- 拡張する (キーボード・マクロによって)。

図 1 : キー・ストロークのフロー (ページ 34) は、キーボード拡張環境でのキー・ストロークのフローと オブジェクトの概略を表しています。

図 1. キー・ストロークのフロー



シナリオ 7. キーボードの拡張

このシナリオでは、**拡張アプリケーション・プログラム**を作成するためにフィルターを使用します。拡張アプリケーション・プログラムとは、キーボードから入力したデータをモニターし、 ある特定の方法でそのデータを変更するプログラムです。通常、この種のアプリケーション・プログラムは**キーボード・マクロ**と呼ばれる命令を使用します。この命令は、どのキー・ストロークを探し、どんな変更を加えるかをアプリケーション・プログラムに伝えます。この変更には、キー・ストロークを抑制すること (これによって、宛先 アプリケーションではキー・ストロークが送信されなかったように見える)、ある キー・ストロークを別のキー・ストロークに置き換えること、あるいは単一の キー・ストロークを一連のキー・ストロークに置き換えることが含まれます。

EHLLAPI を使用してこの処理を行うために、次のシナリオを作成することが できます。

1. EHLLAPI アプリケーション・プログラムで **Connect Presentation Space** (1) 関数を呼び出し、そのキー・ストロークを、フィルターにかける表示スペースに接続する。
2. EHLLAPI プログラムは、次に L オプションを指定して **Start Keystroke Intercept** (50) 関数を呼び出す。これによって、すべてのキー・ストロークが、フィルター操作を行う アプリケーション・プログラムにルーティングされる。
3. フィルター操作を行うアプリケーション・プログラムで、 以下のことを行うループを定義できる。
    a. **Get Key** (51) 関数が、宛先表示スペースへ送信中のキー・ストロークのすべてを代行受信する。
    b. フィルター操作を行うアプリケーションはそれぞれのキー・ストロークを調べ、 次のようなキーボード・マクロのタスクを実行する。

- プログラム・コマンドの省略化。この結果、3 つまたは 4 つの キー・ストローク・コマンドを 1 つのキー・ストロークに凝縮することができる。
- コマンドのカスタマイズ。この結果、コマンドが覚えやすくなり、また他の ソフトウェア・パッケージとの一貫性を保つことができる。
- 契約書または頻繁に使用される書面の**定形文面**の作成。
- 異なる関数に同じキーを使用する同時実行アプリケーションのための キーボードの再編成。

例えば、フィルター操作を行うアプリケーションは、Alt+Y のようなキーの組み合わせを、カーソルを表示スペースの 2 行目の 35 桁に移動し、「テキサス州ダラス XYZ Tool Corporation」というストリングを書き込むコマンドに変換することができます。

c. キー・ストロークが受け入れられない場合、EHLLAPI プログラムは **Post Intercept Status** (52) 関数を使用してビープ音を鳴らす。

4. EHLLAPI プログラムがフィルター・ループを出た後で、**Stop Keystroke Intercept** (53) 関数を呼び出してフィルター処理を終了する。

第 3 章. EHLLAPI 関数

この章では、Z and I Emulator for Windowsの EHLLAPI の個々の関数の詳細 および EHLLAPI プログラム・サンプラーの使用方法について説明します。関数は、関数名によりアルファベット順に並べられています。標準と拡張の両方のインターフェースについて、これらの関数を説明します。

注: 本章では、WinHLLAPI、IBM® 標準 32 ビット HLLAPI および 16 ビット EHLLAPI を標準インターフェースと呼び、 IBM® 拡張 32 ビット EHLLAPI を拡張インターフェースと呼びます。

ページ・レイアウトについての規則

すべての EHLLAPI 関数呼び出しは、必要な情報を素早く検索できるように同一の 形式で表されています。書式は以下のとおりです。

- 関数名 (関数番号)
  - 前提呼び出し
  - 呼び出しパラメーター
  - 戻りパラメーター
  - 使用上の注意

前提呼び出し

「前提呼び出し」では、その関数に先行して呼び出しておく必要がある関数をリストしています。

呼び出しパラメーター

「呼び出しパラメーター」には、その EHLLAPI 関数を呼び出すためにプログラム内で定義しなければならないパラメーターをリストし、それらのパラメーターを定義する方法を説明しています。関数にパラメーターがない場合は、*NA* (不適用) と記されます。**Set Session Parameters** (9) 関数呼び出しで定義されたセッション・パラメーターの値でパラメーターをオーバーライドできる場合は、そのセッション・パラメーター名が示されています。

戻りパラメーター

「戻りパラメーター」には、その EHLLAPI 関数を呼び出した後にプログラムが受け取るはずのパラメーターをリストし、それらのパラメーターをどのように解釈するかを説明します。

使用上の注意

「使用上の注意」には、その関数に影響を与えるセッション・オプションをリストします。また、関数の使用に関する技術情報とアプリケーション開発のヒントも提供します。

EHLLAPI 関数の要約

表 6 : EHLLAPI 関数の要約 (ページ 37) は EHLLAPI 関数を要約したものです。

表 6. EHLLAPI 関数の要約

機能	3270	5250	VT
Connect Presentation Space (1) (ページ 45)	はい	はい	はい
Disconnect Presentation Space (2) (ページ 82)	はい	はい	はい
Send Key (3) (ページ 135)	はい	はい	はい
Wait (4) (ページ 170)	はい	はい	はい
Copy Presentation Space (5) (ページ 65)	はい	はい	はい
Search Presentation Space (6) (ページ 132)	はい	はい	はい
Query Cursor Location (7) (ページ 108)	はい	はい	はい
Copy Presentation Space to String (8) (ページ 69)	はい	はい	はい
Set Session Parameters (9) (ページ 145)	はい	はい	はい
Query Sessions (10) (ページ 115)	はい	はい	はい
Reserve (11) (ページ 128)	はい	はい	はい
Release (12) (ページ 127)	はい	はい	はい
Copy OIA (13) (ページ 55)	はい	はい	はい
Query Field Attribute (14) (ページ 109)	はい	はい	はい
Copy String to Presentation Space (15) (ページ 76)	はい	はい	はい
Pause (18) (ページ 101)	はい	はい	はい
Query System (20) (ページ 116)	はい	はい	はい
Reset System (21) (ページ 129)	はい	はい	はい
Query Session Status (22) (ページ 112)	はい	はい	はい
Start Host Notification (23) (ページ 160)	はい	はい	はい
Query Host Update (24) (ページ 111)	はい	はい	はい
Stop Host Notification (25) (ページ 168)	はい	はい	はい
Search Field (30) (ページ 130)	はい	はい	はい
Find Field Position (31) (ページ 87)	はい	はい	はい
Find Field Length (32) (ページ 86)	はい	はい	はい
Copy String to Field (33) (ページ 74)	はい	はい	はい
Copy Field to String (34) (ページ 51)	はい	はい	はい
Copy Presentation Space to Clipboard (35) (ページ 77)	はい	はい	はい
Paste Clipboard to Presentation Space (36) (ページ 79)	はい	はい	はい
Set Cursor (40) (ページ 144)	はい	はい	はい
Start Close Intercept (41) (ページ 155)	はい	はい	はい
Query Close Intercept (42) (ページ 104)	はい	はい	はい
Stop Close Intercept (43) (ページ 166)	はい	はい	はい

表 6. EHLLAPI 関数の要約 (続く)

機能	3270	5250	VT
Query Additional Field Attribute DRB (ページ 103)	いいえ	はい	いいえ
Start Keystroke Intercept (50) (ページ 162)	はい	はい	はい
Get Key (51) (ページ 90)	はい	はい	はい
Post Intercept Status (52) (ページ 102)	はい	はい	はい
Stop Keystroke Intercept (53) (ページ 169)	はい	はい	はい
Lock Presentation Space API (60) (ページ 96)	はい	いいえ	いいえ
Lock Window Services API (61) (ページ 99)	はい	いいえ	いいえ
Start Communication Notification (80) (ページ 157)	はい	はい	はい
Query Communication Event (81) (ページ 107)	はい	はい	はい
Stop Communication Notification (82) (ページ 167)	はい	はい	はい
Send File (90) (ページ 186)	はい	はい	いいえ
Receive File (91) (ページ 125)	はい	はい	いいえ
Cancel File Transfer(92) (ページ 40)	はい	はい	はい
Convert Position または Convert RowCol (99) (ページ 48)	はい	はい	はい
Connect Window Services (101) (ページ 47)	はい	はい	はい
Disconnect Window Service (102) (ページ 83)	はい	はい	はい
Query Window Coordinates (103) (ページ 118)	はい	はい	はい
Window Status (104) (ページ 171)	はい	はい	はい
Change Switch List LT Name (105) (ページ 42)	はい	はい	はい
Change PS Window Name (106) (ページ 41)	はい	はい	はい
Start Playing Macro (110) (ページ 165)	はい	はい	はい
Connect for Structured Fields (120) (ページ 44)	はい	いいえ	いいえ
Disconnect from Structured Fields (121) (ページ 81)	はい	いいえ	いいえ
Query Communications Buffer Size (122) (ページ 105)	はい	いいえ	いいえ
Allocate Communications Buffer (123) (ページ 38)	はい	いいえ	いいえ
Free Communications Buffer (124) (ページ 89)	はい	いいえ	いいえ
Get Request Completion (125) (ページ 94)	はい	いいえ	いいえ
Read Structured Fields (126) (ページ 120)	はい	いいえ	いいえ
Write Structured Fields (127) (ページ 175)	はい	いいえ	いいえ

Allocate Communications Buffer (123)

3270	*5250*	*VT*
はい	いいえ	いいえ

Allocate Communications Buffer 関数を使用して、 オペレーティング・システムからバッファーを取得することができます。バッファー・アドレスは、**Read Structured Fields** (126) と **Write Structured Fields** (127) の両方の関数に渡す必要があります。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	123 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	6 でなければなりません	8 である必要があります
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1–2	1–4	32 ビットまたは 16 ビットのバッファー長。(0 < size ≤ (64 KB-256 バイト)=X'FF00')
3–6	5–8	割り振られるバッファーの 32 ビット・アドレス (戻される)。

戻りパラメーター

戻りコード	**説明**
0	**Allocate Communications Buffer** 関数は正常終了しました。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
11	リソースが使用できません。(メモリーが使用できません。)

使用上の注意

1. EHLLAPI は、オペレーティング・システムのメモリー・ マネージメントからバッファーを取得し、 そのバッファー・アドレスを戻りパラメーター・ストリングに置きます。要求バッファー・サイズ (長さ) もパラメーター・ストリングに渡されます。バッファー・サイズは、1 バイトから 64 KB - 256 バイト (X'FF00' バイト) の長さです。

   バッファー・サイズに関する情報は、*"**Query Communications Buffer Size** (122)"*を参照してください。

2. この関数で取得したバッファーは、別のプロセス間では共用できません。共用しようとすると、アプリケーションで予測不可能な結果が生じることが あります。

3. EHLLAPI アプリケーションは、**Free Communications Buffer** (124) 関数を発行して、割り振られたメモリーを解放する必要があります。
4. 1 つのアプリケーションに割り振ることのできるバッファーは 10 個が 限度です。この限度に達すると、リソースが使用できないことを示す 戻りコード (RC=11) が戻されます。
5. **Reset System** (21) 関数は、この関数が割り振ったバッファーを解放します。

Cancel File Transfer(92)

3270	*5250*	*VT*
はい	はい	はい

Cancel File Transfer 関数を使用すると、現行の EHLLAPI が指定されたセッションに対して開始した、**Send File** または **Receive File** が直ちに戻されます。

前提呼び出し

Send File (90) または **Receive File** (91)

呼び出しパラメーター

	拡張インターフェース
関数番号	92 でなければなりません
データ・ストリング	ホスト表示スペースの 1 文字の短縮名。ホスト接続表示スペースへの更新要求を示すブランクまたは NULL。
長さ	4 が暗黙指定されます
PS 位置	該当なし

呼び出しデータ構造には、以下の要素が含まれます。

バイト	**定義**
1	表示スペースの 1 文字の短縮名 (PSID)
2 ~ 4	予約済み

戻りパラメーター

戻りコード	**定義**
0	関数は正常終了しました。
1	誤った PSID が指定されました。
8	この PSID に対して、先に **Start Communication Notification** (80) 関数が呼び出されていません。
9	システム・エラーが発生しました。

使用上の注意

Send File (90) および **Receive File** (91) の両方がブロッキング呼び出ししているため、この関数は、常に、異なるスレッド上で発行されなければなりません。

Change PS Window Name (106)

3270	*5250*	*VT*
はい	はい	はい

Change PS Window Name 関数を使用すると、アプリケーションは表示スペース・ウィンドウに新しい名前を指定したり、 デフォルト名にリセットすることができます。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	106 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	必ず指定してください (注を参照)。	68 でなければなりません
PS 位置	該当なし	

注: データ・ストリング長は必ず指定してください (通常、PC/3270 では 3 ~ 63、PC400 では 4 ~ 63、拡張インターフェースでは 68)。

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2	5	変更要求オプションの値。次のうちの 1 つを選択してください。 • 表示スペース・ウィンドウ名を変更する時は、X'01' • 表示スペース・ウィンドウ名をリセットする時は、X'02'
3–63	6–66	終止符バイトを含む 1 (PC/3270 の場合) または 2 (PC400 の場合) から 61 バイトの ASCII ストリング。ASCII ストリングは必ず NULL 文字で終了してください。このストリングには、NULL 文字の前に少なくとも 1 個の非 NULL 文字が必要です。

バイト		定義
	67–68	予約済み

戻りパラメーター

戻りコード	説明
0	**Change PS Window Name** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
12	セッションが停止しました。

使用上の注意

ストリングは、最初に NULL 文字が検出されたところで終了します。NULL 文字は、指定されたストリングの長さを変更します。NULL 文字が指定した長さの終わりにない場合、指定した長さのところで最後の バイトが NULL 文字に置き換えられ、データ・ストリングの残りの部分は 失われます。指定した長さより短いところで NULL 文字が検出された場合、ストリングはそこで 切り捨てられ、データ・ストリングの残りの部分は失われます。

アプリケーションが終了する前に表示スペース名のリセットに 失敗すると、出口リスト処理がその表示スペース名を リセットします。

Change Switch List LT Name (105)

3270	*5250*	*VT*
はい	はい	はい

Change Switch List LT Name 関数を使用すると、アプリケーションは選択した論理端末 (LT) に対するスイッチ・リストの変更またはリセットができます。アプリケーションは呼び出し時に、スイッチ・リストに挿入する名前を指定する 必要があります。

注: この関数は Communication Manager EHLLAPI との互換性を保つためのもので、**Change PS Window Name** (106) 関数と同じ結果になります。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	105 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	通常は 4 ~ 63 です	68 でなければなりません
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2	5	変更要求オプション。次の中から選びます。 • スイッチ・リストの LT 名を変更する時は、X'01' • スイッチ・リストの LT 名をリセットする時は、X'02'
3–63	6–66	終止符バイトを含む 2 から 61 バイトの ASCII ストリング。ASCII ストリングは必ず NULL 文字で終了してください。このストリングには、NULL 文字の前に少なくとも 1 個の非 NULL 文字が必要です。
	67–68	予約済み

戻りパラメーター

戻りコード	説明
0	**Change Switch List LT Name** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
12	セッションが停止しました。

使用上の注意

ストリングは、最初に NULL 文字が検出されたところで終了します。NULL 文字は、指定されたストリングの長さを変更します。NULL 文字が指定した長さの終わりにない場合、指定した長さのところで最後の バイトが NULL 文字に置き換えられ、データ・ストリングの残りの部分は 失われます。指定した長さより短いところで NULL 文字が検出された場合、ストリングはそこで 切り捨てられ、データ・ストリングの残りの部分は失われます。

アプリケーションが終了する前にスイッチ・リスト LT 名のリセットに失敗する と、出口リスト処理がそのスイッチ・リスト LT 名をリセットします。

Connect for Structured Fields (120)

3270	*5250*	*VT*
はい	いいえ	いいえ

Connect for Structured Fields 関数を使用すると、 アプリケーションは、エミュレーション・プログラムへの接続を確立し、 ホスト・アプリケーションとの間で構造化フィールド・データを交換することができます。ワークステーションのアプリケーションは、Query Reply (照会・ 応答) データ・フィールドを提供し、パラメーター・ストリング内でそれを 指していなければなりません。エミュレーターから戻された Destination/Origin ID はアプリケーションに 戻されます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	120 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	7 または 11	16 でなければなりません
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2–5	5–8	Query Reply (照会・応答) データ・バッファーのアドレス
6–7	9–10	戻される Destination/Origin 固有の ID (16 ビット・ワード)。
	11–12	予約済み
8–11	13–16	この位置にあるデータは EHLLAPI に無視されます。ただし、移行プログラムでこの位置にデータを指定していても、 エラーは発生しません。このデータはアプリケーションの移行時に互換性を保つために設けられています。

戻りパラメーター

戻りコード	説明
0	**Connect for Structured Fields** 関数は正常終了しました。
1	指定されたホスト表示スペースの短縮セッション ID が無効か、 またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
10	エミュレーション・プログラムはこの関数をサポートしていません。
32	アプリケーションは通信用にこのセッションに既に接続しています。(正常接続)
39	1 つの DDM セッションがこのセッションに既に接続しています。

使用上の注意

1. EHLLAPI は Destination/Origin ID (DOID) 自己定義 パラメーター (SDP) の Query Reply (照会・応答) バッファーを 走査して、Query Reply の DOID フィールドの内容を決定します。この値が X'0000' の場合は、エミュレーターが DOID をアプリケーションに割り当て、EHLLAPI は割り当てられた ID で Query Reply の DOID フィールドを埋めます。アプリケーションが Query Reply の DOID フィールドに指定した値がゼロ以外の値である場合は、ID は前もって割り当てられていないものと見なされ、 エミュレーターは指定された値をアプリケーションの DOID として割り当てます。指定された DOID が既に使用中の場合には、EHLLAPI は戻りコード 2 を戻します。
2. アプリケーションは、Query Reply データ構造をアプリケーションの専用メモリー内に作成する必要があります。EHLLAPI がサポートする Query Reply データ構造の形式と使用法についての詳細は、がサポートする Query Reply データ構造EHLLAPI (ページ 229) を参照してください。
3. Query Reply のデータについて簡単なチェックが実行されます。ID と 構造の長さについてのみ有効かどうかチェックされます。
4. ホスト・セッションごとに接続できる DDM 基本タイプは 1 つだけです。DDM 接続が Destination/Origin ID (DOID) 用に 自己定義パラメーター (SDP) をサポートしている場合は、複数接続できます。
5. 戻りコード RC=32 または RC=39 を受信した場合、アプリケーションは選択したセッションに既に接続しており、表示スペースを注意して使用する必要があります。ファイル転送、および他の EHLLAPI アプリケーションとの競合が生じる可能性があります。

Connect Presentation Space (1)

3270	*5250*	*VT*
はい	はい	はい

Connect Presentation Space 関数は、EHLLAPI アプリケーション・プログラムとホスト表示スペースとの接続を確立します。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	必ず 1 にしてください。	
データ・ストリング	ホスト表示スペースの 1 文字の短縮名。	
長さ	1 が暗黙指定されます。	4 である必要があります。
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

戻りパラメーター

Connect Presentation Space 関数は戻りコードを設定して、試行状況、および試行が正常終了した場合はホスト表示スペースの状況を示します。

戻りコード	説明
0	**Connect Presentation Space** 関数は正常終了しました。 ホスト表示スペースはアンロックされ、入力の準備ができました。
1	無効なホスト表示スペース ID が指定されました。指定されたセッションが存在しないか、または、 論理プリンター・セッションです。さらにこの戻りコードは、EHLLAPI 用の API Setting がオンに設定されていないことを示している場合があります。
4	接続には成功しましたが、ホスト表示スペースが使用中です。
5	接続には成功しましたが、 ホスト表示スペースがロックされています (入力禁止)。
9	システム・エラーが発生しました。
11	このリソースは使用できません。ホスト表示スペースは、既に他のシステム関数が使用しています。

使用上の注意

1. **Connect Presentation Space** 関数は、 `CONLOG`/`CONPHYS` セッション・オプションの影響を受けます。
2. 1 つの EHLLAPI アプリケーションが同時に複数の 表示スペースに接続することはできません。前提関数として **Connect Presentation Space** 関数を必要とする呼び出しは、現在接続されている表示スペースを使用しま

す。例えば、アプリケーションが表示スペースに A、B、C の順で 接続した場合、そのアプリケーションが関数を発行するためには B または A に 再度接続しなければなりません。

3. **Connect Presentation Space** を必要とする各スレッドに対応する **Disconnect Presentation Space** (2) があるか、 またはそれらのスレッドの 1 つが **Reset System** (21) を発行する必要があります。この機能は全スレッドに影響し、残りの接続をすべて切断します。
4. アプリケーションが共用をサポートし (つまり、それらのアプリケーションが一緒に動作できるように開発されており、予測可能な動作が示されている)、 なおかつ **Set Session Parameters** (9) 関数で設定したものと互換性がある読み取り/書き込みアクセス権とキーワードのオプションがある場合、複数の EHLLAPI アプリケーションで 1 つの表示スペースを共用できます。詳しくは、『Set Session Parameters (9) (ページ 145)』を参照してください。
5. **Connect Presentation Space** 関数と **Start Keystroke Intercept** (50) 関数は共通のサブシステム関数を共用します。このため、あるアプリケーションによって、1 つのセッションに対してこれら 2 つの関数のうちのいずれかを共用するという要求が出され、その要求が正常に実行されると、 別のアプリケーションがこれら 2 つの関数についての要求を出した場合に影響を受ける場合があります。例えば、アプリケーション A がセッションに対して、KEY$abcdefgh をキーワードとして Write アクセスで指定した **Connect Presentation Space** を要求し、その要求が正常に実行された場合、アプリケーション B による、あるセッションに対する **Connect Presentation Space** および **Start Keystroke Intercept** についての要求は、 両方のアプリケーションで互換性のある読み取り/書き込みオプションが設定されている場合にのみ、正常に実行されます。
6. 論理プリンター・セッションとして 定義したセッションには接続できません。詳しくは、*Administrator's Guide and Reference*を参照してください。

Connect Window Services (101)

3270	*5250*	*VT*
はい	はい	はい

Connect Window Services 関数を使用すると、 アプリケーションは表示スペース・ウィンドウを管理することができます。ウィンドウ・サービスでは、1 つの表示スペースに は一度に 1 つの EHLLAPI アプリケーションしか接続できません。

ウィンドウ・サービスでは、1 つの EHLLAPI アプリケーションは同時に複数の 表示スペースに接続できます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	101 でなければなりません	
データ・ストリング	ホスト表示スペースの 1 文字の短縮セッション ID。	
長さ	1 が暗黙指定されます。	4 である必要があります。

	標準インターフェース	拡張インターフェース
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

戻りパラメーター

戻りコード	説明
0	**Connect Window Services** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定されたか、 または Sessions Window Services マネージャーが接続されていません。さらにこの戻りコードは、EHLLAPI 用の API Setting がオンに設定されていないことを示している場合があります。
9	システム・エラーが発生しました。
10	エミュレーション・プログラムはこの関数をサポートしていません。
11	このリソースは使用できません。ホスト表示スペースは、既に他のシステム関数が使用しています。

使用上の注意

1. 1 つの EHLLAPI アプリケーションは同時に複数の 表示スペース・ウィンドウに接続できます。アプリケーションは、接続された複数の表示スペース・ウィンドウ を切断しなくても、それらの間で前後に移動することができます。例えば、アプリケーションが表示スペース・ウィンドウ A、B、 および C に接続している場合、同時に A、B、および C すべてにアクセス可能と なり、他のアプリケーションからは A、B、C のどれにもアクセスできません。
2. プロセスを実行するには、**Connect Window Services** 関数だけで十分です。ただし、**Connect Window Services** を必要とする各スレッドに対応する **Disconnect Window Services** (102) があるか、 またはそれらのスレッドの 1 つが **Reset System** (21) を発行する必要があります。この機能は全スレッドに影響し、残りの接続をすべて切断します。

Convert Position または Convert RowCol (99)

3270	*5250*	*VT*
はい	はい	はい

Convert Position または **Convert RowCol** 関数は、ホスト表示スペースの位置の値をディスプレイの行と桁の座標に変換したり、ディスプレイの行と桁の座標をホスト表示スペースの位置の値に変換します。この関数は、カーソル位置を変更しません。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	99 でなければなりません	
データ・ストリング	**Convert Position** 関数に対してはホスト表示スペースの短縮名および`P` (例えば、 `AP` はセッション A の表示スペース位置を変換します)、あるいは **Convert RowCol** 関数に対してはホスト表示スペースの短縮名および `R` (例えば、`AR` はセッション A の行と桁の座標を変換します)。	
長さ	データ・ストリング・パラメーターの 2 番目の文字として `R` を指定する場合は行 (Row)。有効な入力の下限値は 1、上限値はホスト表示スペースの 構成によって異なります。『使用上の注意 (ページ 50)』を参照してください。 データ・ストリング・パラメーターの 2 番目の文字として `P` を指定する場合は適用されません。	
PS 位置	データ・ストリング・パラメーターの 2 番目の文字として `R` を指定する場合は桁 (Column)。有効な入力の下限値は 1、上限値はホスト表示スペースの構成 によって異なり、24 から 43 の間です。『使用上の注意 (ページ 50)』を参照してください。 データ・ストリング・パラメーターの 2 番目の文字として `P` を指定する場合は、ホスト表示スペース位置。有効な入力の下限値は 1、上限値はホスト表示スペースの構成 によって異なり、1920 から 3564 の間です。『使用上の注意 (ページ 50)』を参照してください。	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2	5	**変換**オプション `P` または `R`
	6–8	予約済み

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

Convert Position 関数 (呼び出しデータ・ストリング内の 2 番目の文字が P) については、1 から 43 (PC/3270 の場合) または 27 (PC400 の場合) が戻されます。この値は、呼び出し PS 位置パラメーターに含まれた PS 位置を含む行の番号です。上限値は、ホスト表示スペースの構成によっては、 43 (PC/3270 の場合) または 27 (PC400 の場合) よりも小さくなります。

Convert RowCol 関数 (呼び出しデータ・ストリングの 2 番目の文字が R) の場合、0 という値は、行についての入力値 (呼び出しデータ・ストリング長パラメーター) にエラーがあることを示します。

戻りコード:

Convert Position または RowCol 関数は、4 番目の戻りパラメーターに常に戻りコードがあるという規則から除外されています。この関数に関しては、4 番目のパラメーターに戻される値は状況コードです。この状況コードには、データまたは戻りコードが含まれることがあります。予測不能な結果やエラーを避けるため、アプリケーションはこの状況コードの 処理機能を備えている必要があります。

- 4 番目のパラメーターの値が、0、9998、9999 のいずれかである場合は、 戻りコードです。
- **Convert Position** 関数 (呼び出しデータ・ストリング内の 2 番目の文字が P) の場合、1 から 132 の間の値は、呼び出し PS 位置パラメーターで渡された PS 位置を含む桁の番号です。上限値は、ホスト表示スペースの 構成によっては、132 よりも小さくなります。
- **Convert RowCol** 関数 (呼び出しデータ・ストリング内の 2 番目の文字が R) の場合、1 から 3564 の間の値は、呼び出しデータ・ストリング長パラメーターと PS 位置パラメーターでそれぞれ渡された行と桁の値に対応するホスト表示スペースの位置を表します。上限値は、ホスト表示スペースの構成によっては、3564 より も小さくなります。

以下の状況コードが定義されています。

状況コード	説明
0	これは無効な PS 位置または桁です。
>0	これは PS 位置または桁です。
9998	無効なホスト表示スペース ID が指定されたか、 またはシステム・エラーが発生しました。
9999	データ・ストリング内の 2 番目の文字が P でも R でもありません。

使用上の注意

1. 表示スペースを構成する場合は、「*Administrator's Guide and Reference*」を参照してください。
2. 表示スペースの行や桁の数を知るには、**Query Session Status** (22) 関数で戻されるデータ・ストリング・パラメーターを調べてください。『Query Session Status (22) (ページ 112)』を参照してください。

Copy Field to String (34)

3270	*5250*	*VT*
はい	はい	はい

Copy Field to String 関数は、ホスト接続の表示スペース内のフィールドからストリングへ文字を転送します。

Copy Field to String 関数は、 ホスト・ソース表示スペース内の文字を情報交換用日本工業規格コード (ASCII) に変換します。ASCII では通常は表されない属性バイトや他の文字はブランクに変換されます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	34 でなければなりません	
データ・ストリング	事前に割り振りされたターゲットのデータ・ストリング。Extended Attribute Bytes (EAB) オプションを指定して **Set Session Parameters** (9) 関数を発行した場合、 データ・ストリングの長さは少なくともフィールドの長さの 2 倍必要です。	
長さ	コピーするバイトの数 (データ・ストリングの長さ)	
PS 位置	該当フィールドを識別します。これはコピー先フィールド内のどのバイトの PS 位置でも構いません。コピー関数は常にフィールドの先頭から開始されます。	

戻りパラメーター

この関数は、データ・ストリング、データ・ストリング長、戻りコードを 戻します。

データ・ストリング:

ストリングはホスト表示スペース内の識別された フィールドからのデータを含んでいます。戻されるデータ・ストリングの先頭バイトは、ホスト表示スペース内の識別されたフィールドの先頭バイトです。戻されるデータ・ストリングのバイト数は、次のうちの小さい方によって 決まります。

- 呼び出しデータ・ストリング長パラメーターで指定したバイト数。
- ホスト表示スペース内の識別されたフィールドのバイト数。

長さ:

戻されるデータの長さ。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Copy Field to String** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター指定でエラーが発生しました。
6	コピーするデータとコピー先フィールドのサイズが等しくありません。ストリングの長さがコピーされるフィールドよりも短い場合、 データは切り捨てられます。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	ホスト表示スペースが定様式化されていません。

使用上の注意

1. フィールドの位置と長さの情報は、**フィールド位置の検索**(31) および**フィールド長の検索**(32) 機能を使用して、見つけることができます。**フィールドからストリングへのコピー**機能は、保護フィールドまたは無保護フィールドと共に使用することができます。
2. 次の条件の 1 つが発生した場合、コピーは終了します。
   - フィールドの最後に達したとき
   - 対象文字列の長さを超えた場合
3. EAB を返すことができるのは、**セッションパラメーター**(9) 機能の EAB オプションが使用されている場合です。EAB は、表示スペースの各文字に関連して、各文字の前に戻されます。
4. **「** フィールド **を** ストリング **に** コピー **」 機能** は 、 **以下** の 条件 **の** 影響 を **受ける** : セッションオプション。項目を参照5 on page 148;13 on page 151 and14 on page 152 ;17 on page 152 ; and#unique_169_Connect_42_opt18 and#unique_169_Connect_42_opt19を参照してください。

   前に説明したように、さまざまな**コピー**(5、8、34) 関数による属性の戻しは、**セッションパラメーターの設定**(9) 関数の影響を受けます。関連するセットセッションパラメータは、以下の効果があります:

   セッションパラメータの設定

   COPY 機能への影響

   NOEAB と NOEAD

   属性は返されません。テキストのみが、表示スペースからユーザー・バッファーにコピーされます。

   EAB と NOXLATE

   次の表の定義に従って属性が返されます。

EAB と XLATE

表示スペースの表示に使用されている色が返されます。色は再マップできます。したがって、XLATE と EAB が同時にオンの時、色属性は**COPY**機能によって返されるものではありません。

返される文字属性を次の表に定義します。属性のビット位置はIBM®形式であり、ビット 0 がバイトの最も左のビットです。

ホストからエミュレータに 3270 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0–1	文字の強調表示 00 = 通常 01 = ブリンク 10 = リバースビデオ 11 = 下線
2–4	文字の色 (カラーリマップでこの色の定義を上書きできます。) 000 = デフォルト 001 = 青 010 = 赤 011 = ピンク 100 = 緑 101 = ターコイズ 110 = 黄 111 = ホワイト
5–6	キャラクタ属性 00 = デフォルト値
7	予約済み

ホストからエミュレータに、5250 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0	リバースイメージ 0 = 通常イメージ 1 = イメージの反転
1	下線 0 = 下線なし 1 = 下線
2	明滅 0 = ブリンクしない 1 = ブリンク

ビット位置	意味
3	列のセパレータ 0 = 区切り文字なし 1 = セパレータ
4–7	予約済み

次の表にZ and I Emulator for Windows文字色の属性です。次の表は、EAB と XLATE が設定されている場合に適用されます。

ビット位置	意味
0–3	背景文字色 0000 = 黒 0001 = 青 0010 = 緑 0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト
4–7	文字の前景色 0000 = 黒 0001 = 青 0010 = 緑 0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト 1000 = グレー 1001 = 水色 1010 = ライトグリーン 1011 = ライトシアン 1100 = ライトレッド 1101 = ライトマゼンタ 1110 = 黄 1111 = ホワイト (高強度)

PS/2®モノクロ画面の場合、アプリケーション (ワークステーション) セッションの文字が、さまざまな濃淡で表示されます。これは、EHLLAPIユーザーがホスト・アプリケーション表示スペースを取得できるようにします。

5. この関数を使用するには、返されたデータ文字列パラメータを受信するメモリを事前に割り当てます。このメモリーを事前に割り当てるために必要な文は、アプリケーションが作成されている言語によって異なります。参照メモリー割り振り on page 14を参照してください。

Note: 5250 エミュレーションは、24 行 x 80 列の表示スペースをサポートします。場合によっては、Communication Manager 5250 エミュレーションは 25 行を表示します。これは、ホストからのエラーメッセージが表示されたとき、またはオペレータが SysReq キーを選択したときに発生します。Z and I Emulator for Windowsステータスバーの 25 行の情報を表示します。**EXTEND_PS**オプションにより、EHLLAPIアプリケーションは、 ~ と同じインターフェースを使用できます。Communication ManagerEHLLAPIこの状態が発生すると、有効な表示スペースが拡張されます。

Copy OIA (13)

3270	*5250*	*VT*
はい	はい	はい

Copy OIA 関数は、ホスト接続表示スペースから現在のオペレーター情報域 (OIA) データを戻します。

OIA は画面の下分割線の下側にあり、ワークステーションとホストとの接続に関する セッション状況情報の表示に使用されます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	13 でなければなりません	
データ・ストリング	事前に割り振りされた ターゲットのデータ・ストリング	
長さ	103	104
PS 位置	該当なし	

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

16 ビットの場合は 103 バイトのストリング、32 ビットの 場合は 104 バイトのストリング。詳しくは、戻り OIA データ・ストリングの形式 (ページ 56)を参照してください。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	OIA データが戻されます。コピー先の表示スペースはロックされていません。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	指定されたストリング長に誤りがありました。OIA データは戻されませんでした。
4	OIA データが戻されます。コピー先の表示スペースは使用中です。
5	OIA データが戻されます。コピー先の表示スペースはロックされています。(入力禁止)
9	内部システム・エラーが発生しました。OIA データは戻されませんでした。

使用上の注意

1. OIA グループは、接続したセッションの状況を表すビットで構成されています。グループは、そのグループが意味するホストの関数によって分けられています。(例えば、グループ 8 にはそのセッションで起こり得るすべての入力禁止条件を表すビットが入っています。) 各グループの状態は、高位のビットがより高い優先順位を表すような順序になっています。つまり、ビット 7 はビット 0 より優先されます。そのため、1 つのグループ内で 複数の状態が活動状態にあるときは、優先順位の最も高い状態が、そのグループ内 の活動状態です。
2. この関数を使用するには、戻りデータ・ストリング・パラメーター を受け入れるメモリーを事前に割り振ります。このメモリーの事前割り振りに必要なステートメントは、 アプリケーションに使用される言語によって異なります。詳しくは、メモリー割り振り (ページ 14)を参照してください。

戻り OIA データ・ストリングの形式

OIA データ・ストリングには以下の情報が入っています。

バイト		定義
標準	拡張	
1	1	OIA フォーマット設定バイト。値は 1 (PC/3270)、9 (PC400)、または 5 (VT) です。
2–81	2–81	ホスト・コード・ポイントの OIA イメージ。
82–103	82–103	OIA グループの標識の意味。
	104	予約済み。

PC/3270 OIA グループの標識の意味とそのイメージ

OIA イメージ・グループは、ホスト・コード・ポイントに OIA イメージを含む属性バイトのない 80 バイトの ASCII 文字列で構成されます。図 2 : ホスト表示スペース文字 (ページ 58) は、ホスト表示スペースにある 16 進コードのとそれらが表す文字を示します。戻されたデータは OIA グラフィックス文字に変換されます。OIA 標識については、「*Quick Beginnings*」を参照してください。

戻されたデータを OIA グラフィックス文字に変換する場合は次のようにします。

1. バイト 2 から 81 に戻されたデータを、画面またはプリンターで表示または 印刷します。
2. 出力が表示される装置に適用できるコード・ページ・チャートを使用して、 それぞれの文字に対応する 16 進数値を見つけます。
3. 図 2 : ホスト表示スペース文字 (ページ 58) を使用して、ステップ 2 で見つけた 16 進数値の それぞれに対応する OIA グラフィックス文字を見つけます。

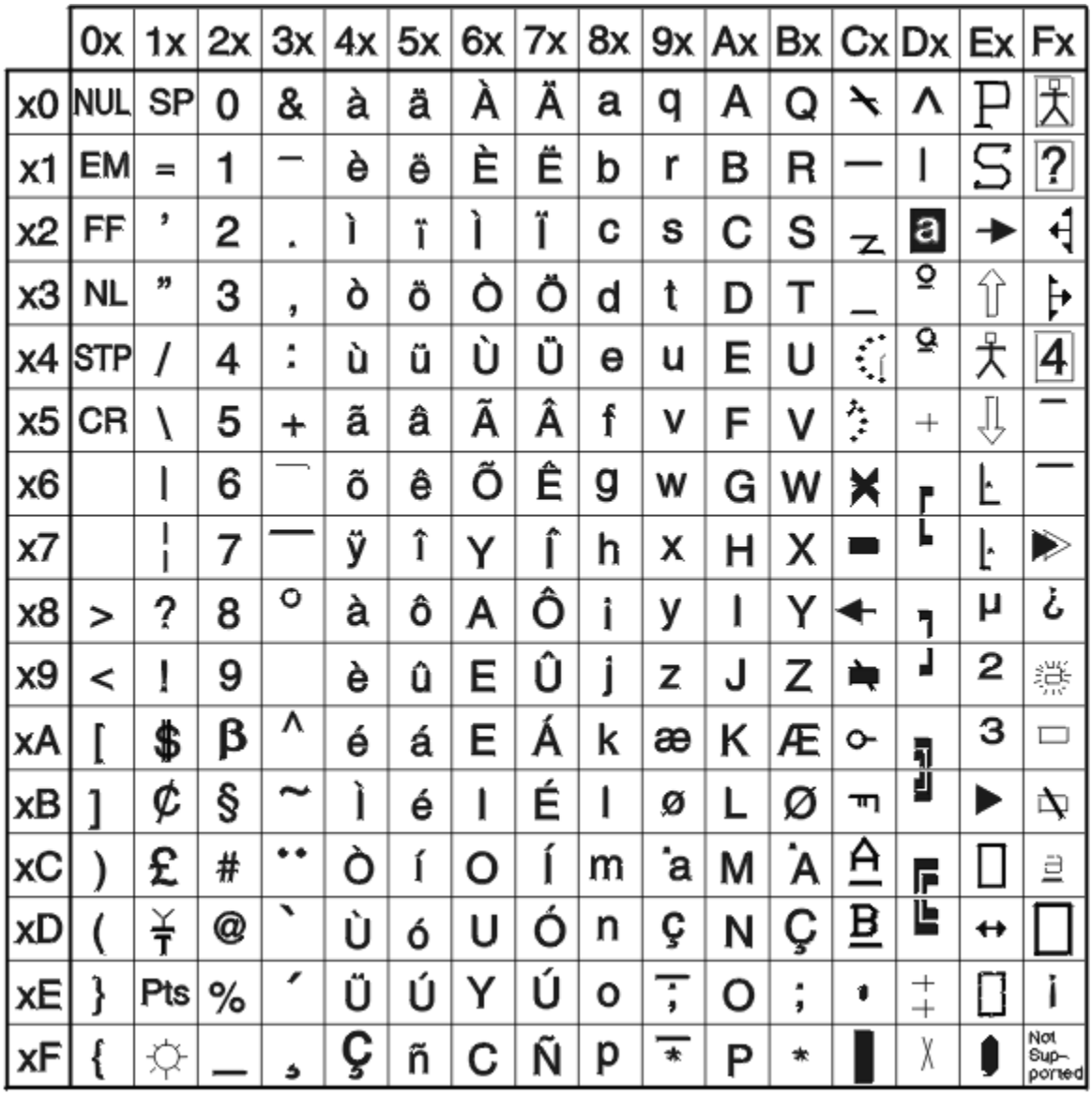
注: グループ 8 (バイト 0) 機械チェック、通信チェック、および プログラム・チェックのイメージの後には、チェックのタイプに関連する 3 桁の 数字があります。

後に X'20' が続く短縮セッション ID は 7 桁目にあります。

すべてのグループ・イメージは、メインフレーム・インタラクティブ (MFI) の 16 進コード・ポイントで表されます。

注: OIA イメージ・データ・ストリング位置から 1 を引いたものが、OIA 桁 に等しくなります。

図 2. ホスト表示スペース文字



- グループ 1 (オフセット 82): オンラインおよび画面の所有権

ビット	意味
0–1	予約済み
2	SSCP-LU セッションが表示画面を所有する
3	LU-LU セッションが表示画面を所有する
4	オンラインであり、所有されていない
5	サブシステム作動可能

ビット	意味
6–7	予約済み

- グループ 2 (オフセット 83): 文字選択

ビット	意味
0	予約済み
1	APL
3	英数字
4–5	予約済み

- グループ 3 (オフセット 84): シフト状態

ビット	意味
0	上段シフト
1	数字
2	CAPS
3–7	予約済み

- グループ 4 (オフセット 85): PSS グループ 1

ビット	意味
0–7	予約済み

- グループ 5 (オフセット 86): 強調表示グループ 1

ビット	意味
0	オペレーターが選択可能
1	フィールド継承
2–7	予約済み

- グループ 6 (オフセット 87): カラー・グループ 1

ビット	意味
0	オペレーターが選択可能
1	フィールド継承
2–7	予約済み

- グループ 7 (オフセット 88): 挿入

ビット	意味
0	挿入モード
1–7	予約済み

- グループ 8 (オフセット 89–93): 入力禁止 (5 バイト)

◦ バイト 1 (オフセット 89)

ビット	意味
0	リセット不可能なマシン・チェック
1	予約済み
2	マシン・チェック
3	通信チェック
4	プログラム・チェック
5–7	予約済み

◦ バイト 2 (オフセット 90)

ビット	意味
0	装置使用中
1	端末待ち
2	マイナス記号
3	マイナス関数
4	入力過多
5–7	予約済み

◦ バイト 3 (オフセット 91)

ビット	意味
0–2	予約済み
3	間違ったデッド・キーの組み合わせ、限定キー
4	不適当な位置
5–7	予約済み

◦ バイト 4 (オフセット 92)

ビット	意味
0–1	予約済み
2	システム待ち
3–7	予約済み

◦ バイト 5 (オフセット 93)

ビット	意味
0–7	予約済み

• グループ 9 (オフセット 94): PSS グループ 2

ビット	意味
0–7	予約済み

- グループ 10 (オフセット 95): 強調表示グループ 2

ビット	意味
0–7	予約済み

- グループ 11 (オフセット 96): カラー・グループ 2

ビット	意味
0–7	予約済み

- グループ 12 (オフセット 97): 通信エラー状況メッセージ

ビット	意味
0-6	Communications error
1–7	予約済み

- グループ 13 (オフセット 98): プリンター状況

ビット	意味
0–7	予約済み

- グループ 14 (オフセット 99): グラフィックス

ビット	意味
0–7	予約済み

- グループ 15 (オフセット 100): 予約済み
- グループ 16 (オフセット 101): 自動キーの再生/記録の状態

ビット	意味
0–7	予約済み

- グループ 17 (オフセット 102): 自動キーの打ち切り/停止の状態

ビット	意味
0–7	予約済み

- グループ 18 (オフセット 103): 拡大状態

ビット	意味
0–7	予約済み

PC400 OIA グループの標識の意味とそのイメージ

以下の表に、OIA グループの詳細をリストします。

- グループ 1 (オフセット 82): オンラインおよび画面の所有権

ビット	意味	データ・ストリングの開始位置
0–2	予約済み	
3	システム使用可能	1
4	予約済み	
5	サブシステム作動可能	
6–7	予約済み	

- グループ 2 (オフセット 83): 文字選択

ビット	意味	データ・ストリングの開始位置
0–1	予約済み	
3	英数字	
4–5	予約済み	

- グループ 3 (オフセット 84): シフト状態

ビット	意味	データ・ストリングの開始位置
0	予約済み	
1	キーボード・シフト	39
2	CAPS	
3–6	予約済み	

- グループ 4 (オフセット 85): PSS グループ 1

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 5 (オフセット 86): 強調表示グループ 1

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 6 (オフセット 87): カラー・グループ 1

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 7 (オフセット 88): 挿入

ビット	意味	データ・ストリングの開始位置
0	挿入モード	68
1–7	予約済み	

- グループ 8 (オフセット 89–93): 入力禁止 (5 バイト)

◦ バイト 1 (オフセット 89)

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

◦ バイト 2 (オフセット 90)

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

◦ バイト 3 (オフセット 91)

ビット	意味	データ・ストリングの開始位置
0–4	予約済み	
5	オペレーターの入力エラー	64
6–7	予約済み	

◦ バイト 4 (オフセット 92)

ビット	意味	データ・ストリングの開始位置
0–1	予約済み	
2	システム待ち	64
3–7	予約済み	

◦ バイト 5 (オフセット 93)

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

• グループ 9 (オフセット 94): PSS グループ 2

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

• グループ 10 (オフセット 95): 強調表示グループ 2

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

• グループ 11 (オフセット 96): カラー・グループ 2

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

• グループ 12 (オフセット 97): 通信エラー状況メッセージ

ビット	意味	データ・ストリングの開始位置
0	通信エラー	
1–5	予約済み	
7	メッセージ待ち	3

- グループ 13 (オフセット 98): プリンター状況

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 14 (オフセット 99): グラフィックス

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 15 (オフセット 100): 予約済み
- グループ 16 (オフセット 101): 自動キーの再生/記録の状態

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 17 (オフセット 102): 自動キーの打ち切り/停止の状態

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

- グループ 18 (オフセット 103): 拡大状態

ビット	意味	データ・ストリングの開始位置
0–7	予約済み	

VT ホスト OIA グループの標識の意味とそのイメージ

以下の表に、VT ホスト OIA グループの詳細をリストします。

- グループ 1 (オフセット 82): オンラインおよび画面の所有権

ビット	意味
5	サブシステム作動可能

- グループ 2 (オフセット 83): 文字選択

ビット	意味
0	上段シフト
2	CAPS

- グループ 7 (オフセット 88): 挿入

ビット	意味
0	挿入モード

OIA 行表示の一部の桁には、3270/5250 用に表示されるメッセージとは 異なる VT 用のメッセージが表示されます。個々の詳細については次の表を参照してください。

列	記号
1–7	VT220 7
	VT220 8
	VT100
	VT52
	VTANSI
9 - 12	LOCK
61 - 64	HOLD

Copy Presentation Space (5)

3270	*5250*	*VT*
はい	はい	はい

Copy Presentation Space 関数は、ホスト接続した表示スペースの内容を EHLLAPI アプリケーション・プログラム内に定義されたデータ・ストリングにコピーします。

Copy Presentation Space 関数は、ホストのコピー元の表示スペース内の文字を ASCII 形式に変換します。属性バイトやその他の通常 ASCII では表されない文字は、 ブランクに変換されます。属性バイトをブランクに変換したくない場合には、**Set Session Parameters** (9) 関数の ATTRB オプションを指定して、この変換を一時変更することができます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	5 でなければなりません	
データ・ストリング	事前にホスト表示スペースのサイズを割り振られた コピー先ストリング。このサイズはホスト表示スペースの構成によって違います。EAB オプションを指定して **Set Session Parameters** (9) 関数を発行した場合、データ・ストリングの長さは少なくとも表示スペースの長さの 2 倍必要です。	
長さ	NA (ホスト表示スペース長が暗黙指定されます)。	

	標準インターフェース	拡張インターフェース
PS 位置	NA	

戻りパラメーター

この関数は、データ・ストリング、データ・ストリング長、戻りコードを 戻します。

データ・ストリング:

接続されたホスト表示スペースの内容。

長さ:

コピーされたデータの長さ。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	ホスト表示スペースの内容は アプリケーション・プログラムにコピーされました。コピー先表示スペースは活動状態にあり、キーボードは アンロックされています。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
4	ホスト表示スペースの内容はコピーされました。接続されたホスト表示スペースはホストからの応答を 待っています。
5	ホスト表示スペースはコピーされました。キーボードはロックされています。
9	システム・エラーが発生しました。

使用上の注意

1. EAB を返すことができるのは、**セッションパラメーター**(9) 機能の EAB オプションが使用されている場合です。EAB は、表示スペースの各文字に関連して、各文字の前に戻されます。
2. **表示スペースのコピー**機能は、以下のセッションオプションの影響を受けます。
   - ATTRB/NOATTRB/NULLATTRB
   - EAB/NOEAB
   - XLate/NOXLate
   - 空白/NOBLANK
   - 表示/NODISPLAY
   - EXTEND_PS/NOEXTEND_PS

項目を参照 5 on page 148;13 on page 151 ,14 on page 152 ,15 on page 152 and17 on page 152 ; and#unique_169_Connect_42_opt18 and#unique_169_Connect_42_opt19を参照してください。

提供されたターゲット・データ・ストリングの長さでは要求されたデータを保持できない場合、予期しない結果が発生する可能性があります。

前に説明したように、さまざまな**コピー**(5、8、34) 関数による属性の戻しは、**セッションパラメーターの設定**(9) 関数の影響を受けます。関連するセットセッションパラメータは、以下の効果があります:

セッションパラメータの設定

COPY 機能への影響

NOEAB と NOEAD

属性は返されません。テキストのみが、表示スペースからユーザー・バッファーにコピーされます。

EAB と NOXLATE

次の表の定義に従って属性が返されます。

EAB と XLATE

表示スペースの表示に使用されている色が返されます。色は再マップできます。したがって、色属性は、XLATE と EAB が同時にオンの時、**コピー**機能によって返されるものではありません。

NOSO/SPACESO/SO

NOSO を指定した場合、SPACESO として機能します。表示スペースのサイズは変更されません。

返される文字属性を次の表に定義します。属性のビット位置はIBM®形式であり、ビット 0 がバイトの最も左のビットです。

ホストからエミュレータに 3270 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	**意味**
0–1	文字の強調表示 00 = 通常 01 = ブリンク 10 = リバースビデオ 11 = 下線
2–4	文字の色 (カラーリマップでこの色の定義を上書きできます。) 000 = デフォルト 001 = 青 010 = 赤 011 = ピンク 100 = 緑 101 = ターコイズ 110 = 黄 111 = ホワイト
5–6	文字属性 00 = デフォルト値

ビット位置	意味
7	予約済み

ホストからエミュレータに、5250 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0	リバースイメージ 0 = 通常イメージ 1 = イメージの反転
1	下線 0 = 下線なし 1 = 下線
2	明滅 0 = ブリンクしない 1 = ブリンク
3	列のセパレータ 0 = 区切り文字なし 1 = セパレータ
4–7	予約済み

次の表にZ and I Emulator for Windows文字色の属性です。次の表は、EAB と XLATE が設定されている場合に適用されます。

ビット位置	意味
0–3	背景文字色 0000 = 黒 0001 = 青 0010 = 緑 0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト
4–7	文字の前景色 0000 = 黒 0001 = 青 0010 = 緑

ビット位置	意味
	0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト 1000 = グレー 1001 = 水色 1010 = ライトグリーン 1011 = ライトシアン 1100 = ライトレッド 1101 = ライトマゼンタ 1110 = 黄 1111 = ホワイト (高強度)

PS/2®モノクロ画面の場合、アプリケーション (ワークステーション) セッションの文字が、さまざまな濃淡で表示されます。これは、EHLLAPIユーザーがホスト・アプリケーション表示スペースを取得できるようにします。

ホスト表示スペースの一部のみをコピーしたい場合は、表示スペースの**コピー**(8) 関数を使用します。

この関数を使用するには、返されたデータ文字列パラメータを受信するメモリを事前に割り当てます。このメモリーを事前に割り当てるために必要な文は、アプリケーションが作成されている言語によって異なります。参照メモリー割り振り on page 14を参照してください。

Note: 5250 エミュレーションは、24 行 x 80 列の表示スペースをサポートします。場合によっては、Communication Manager 5250 エミュレーションは 25 行を表示します。これは、ホストからのエラーメッセージが表示されたとき、またはオペレータが SysReq キーを選択したときに発生します。Z and I Emulator for Windows 25 行の情報を、24 行またはステータスバーに表示します。ステータスバーに情報を表示するには、ステータスバーを設定する必要があります参照*Quick Beginnings*ステータスバーの設定の詳細については、を参照してください。**EXTEND_PS**オプションを指定し、EHLLAPIアプリケーションは、 ~と同じインターフェースを使用できます。Communication ManagerEHLLAPIこの状態が発生すると、有効な表示スペースが拡張されます。

Copy Presentation Space to String (8)

3270	*5250*	*VT*
はい	はい	はい

Copy Presentation Space 関数は、ホスト接続した表示スペースの内容を EHLLAPI アプリケーション・プログラム内に定義されたデータ・ストリングにコピーします。

入力 PS 位置は、ホストの表示スペースのオフセットです。このオフセットは、左上隅 (行 1、桁 1) を位置 1、右下隅 を 3564 (ホスト表示スペースの最大表示画面サイズ) とするレイアウトを基にしています。`PS Position + (Length - 1)` の値は、ホスト表示スペースの構成サイズを超えることはできません。

Copy Presentation Space to String 関数は、ホストのコピー元の表示スペース内の文字を ASCII 形式に変換します。属性バイトやその他の通常 ASCII では表されない文字は、 ブランクに変換されます。属性バイトをブランクに変換したくない場合には、**Set Session Parameters** (9) 関数の ATTRB オプションを指定して、この変換を一時変更することができます。

前提呼び出し

Connect Presentation Space (1)。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	8 である必要があります	
データ・ストリング	事前にホスト表示スペースのサイズを割り振られた コピー先ストリング。EAB オプションを指定して **Set Session Parameters** (9) 関数を発行した場合、データ・ストリングの長さは少なくとも表示スペースの長さの 2 倍必要です。	
長さ	該当データ・ストリングの長さ。	
PS 位置	コピー先データ・ストリングの第 1 バイトの ホスト表示スペース内の位置。	

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

ホスト表示スペースの内容。

戻りコード:

以下のコードが定義されています。

戻りコード	**説明**
0	ホスト表示スペースの内容は アプリケーション・プログラムにコピーされました。コピー先表示スペースは活動状態にあり、キーボードは アンロック されています。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	指定されたストリング長に誤りがありました。あるいは、 (データ・ストリング長 - 1) + PS 位置の合計が、接続されている ホスト表示スペースのサイズを超えています。
4	ホスト表示スペースの内容はコピーされました。ホスト表示スペースはホストの応答を待っています。
5	ホスト表示スペースはコピーされました。キーボードはロック されています。

戻りコード	説明
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。

使用上の注意

1. EAB を返すことができるのは、**セッションパラメーター**(9) 機能の EAB オプションが使用されている場合です。EAB は表示スペースの各文字に関連して、それぞれの文字に続いて返されます。
2. **ストリングへの表示スペースのコピー**機能は、以下のオプションの影響を受けます。
   - ATTRB/NOATTRB/NULLATTRB
   - EAB/NOEAB
   - XLate/NOXLate
   - 空白/NOBLANK
   - 表示/NODISPLAY
   - EXTEND_PS/NOEXTEND_PS

   項目を参照5 on page 148;13 on page 151 and14 on page 152 ;15 on page 152 ;17 on page 152 ; and#unique_169_Connect_42_opt18 and#unique_169_Connect_42_opt19

   提供されたターゲット・データ・ストリングが要求されたバイト数を保持するのに十分でない場合、ターゲット・データ・ストリングの最後に達したときに、コピーは正常に終了します (RC=0、4、または 5)。

   前に説明したように、さまざまな**コピー**(5、8、34) 関数による属性の戻しは、**セッションパラメーターの設定**(9) 関数の影響を受けます。関連するセットセッションパラメータは、以下の効果があります:

   セッションパラメータの設定

   コピー機能への影響

   NOEAB と NOEAD

   属性は返されません。テキストのみが、表示スペースからユーザー・バッファーにコピーされます。

   EAB と NOXLATE

   次の表の定義に従って属性が返されます。

   EAB と XLATE

   表示スペースの表示に使用されている色が返されます。色は再マップできるため、XLATE と EAB が同時にオンの場合、色属性は**コピー**機能によって返されるものではありません。

   返される文字属性を次の表に定義します。属性のビット位置は IBM 形式であり、ビット 0 がバイトの最も左のビットです。
   - ホストからエミュレータに 3270 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0–1	文字の強調表示

ビット位置	意味
	00 = 通常 01 = ブリンク 10 = リバースビデオ 11 = 下線
2–4	文字の色 (カラーリマップでこの色の定義を上書きできます。) 000 = デフォルト 001 = 青 010 = 赤 011 = ピンク 100 = 緑 101 = ターコイズ 110 = 黄 111 = ホワイト
5–7	予約済み

- ホストからエミュレータに、5250 文字属性が返されます。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0	リバースイメージ 0 = 通常イメージ 1 = イメージの反転
1	下線 0 = 下線なし 1 = 下線
2	明滅 0 = ブリンクしない 1 = ブリンク
3	列のセパレータ 0 = 区切り文字なし 1 = セパレータ
4–7	予約済み

- ホストからエミュレータに VT 文字属性を返します。次の表は、EAB と NOXLATE が設定されている場合に適用されます。

ビット位置	意味
0-3	予約済み
4	太字 1 = オン 0 = オフ

ビット位置	意味
5	アンダースコア 1 = オン 1 = オフ
6	明滅 1 = オン 0 = オフ
7	リバース 0 = オン 1 = オフ

◦ 次の表にZ and I Emulator for Windows文字色の属性です。次の表は、EAB と XLATE が設定されている場合に適用されます。

ビット位置	意味
0–3	背景文字色 0000 = 黒 0001 = 青 0010 = 緑 0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト
4–7	文字の前景色 0000 = 黒 0001 = 青 0010 = 緑 0011 = 水色 0100 = 赤 0101 = マゼンタ 0110 = ブラウン (3270)、イエロー (5250) 0111 = ホワイト 1000 = グレー 1001 = 水色 1010 = ライトグリーン 1011 = ライトシアン 1100 = ライトレッド 1101 = ライトマゼンタ 1110 = 黄 1111 = ホワイト (高強度)

PS/2 モノクロ画面の場合、アプリケーション (ワークステーション) セッションの文字が、さまざまな濃淡で表示されます。これは、EHLLAPIユーザーがホスト・アプリケーション表示スペースを取得できるようにします。

3. この関数を使用するには、返されたデータ文字列パラメータを受信するメモリを事前に割り当てます。このメモリーを事前に割り当てるために必要な文は、アプリケーションが作成されている言語によって異なります。参照メモリー割り振り on page 14を参照してください。

Note: 5250 エミュレーションは、24 行 x 80 列の表示スペースをサポートします。場合によっては、Communication Manager 5250 エミュレーションは 25 行を表示します。これは、ホストからのエラーメッセージが表示されたとき、またはオペレータが SysReq キーを選択したときに発生します。Z and I Emulator for Windows 25 行の情報を、24 行またはステータスバーに表示します。ステータスバーに情報を表示するには、ステータスバーを設定する必要があります参照*Quick Beginnings*ステータスバーの設定の詳細については、を参照してください。**EXTEND_PS**オプションを指定し、EHLLAPIアプリケーションは、 ~ と同じインターフェースを使用できます。Communication ManagerEHLLAPIこの状態が発生すると、有効な表示スペースが拡張されます。

Copy String to Field (33)

3270	*5250*	*VT*
はい	はい	はい

Copy String to Field 関数は、文字ストリングをホスト接続表示スペース内の指定したフィールドへ転送します。この関数は、フィールド形式のホスト表示スペースでのみ使用できます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	33 でなければなりません	
データ・ストリング	ホスト表示スペース内のコピー先フィールドへ転送するデータが 入っているストリング。	
長さ	コピー元データ・ストリングの長さ (バイト数)。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	該当フィールドを識別します。これはコピー先フィールド内のどのバイトの PS 位置でも構いません。コピー関数は常にフィールドの先頭から開始されます。	

戻りパラメーター

戻りコード	説明
0	**Copy String to Field** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーターが誤っているか、 パラメーター長 0 がコピーに指定されました。
5	コピー先フィールドが保護または入力禁止になっているか、あるいは、 無効なデータ (フィールド属性など) がコピー先フィールドに送られました。
6	コピーは完了しましたが、データが切り捨てられました。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	ホスト表示スペースが定様式化されていません。

使用上の注意

1. 「**ストリングをフィールドにコピー**」機能は、以下のオプションの影響を受けます:
   - 文字列/文字列
   - EOT
   - EAB/NOEAB
   - XLate/NOXLate
   - プテータブ/NOPUTEAB

   項目を参照1 on page 147and2 on page 147 ;13 on page 151 and14 on page 152 ;18 on page 153 ; and#unique_169_Connect_42_opt18 and#unique_169_Connect_42_opt19を参照してください。
2. 転送されるストリングは、呼び出しデータ文字列パラメーターで指定されます。次の 3 つの条件のいずれかが発生すると、ストリングは終了します。
   - 「**セッションパラメータの設定**」 (9) 機能を使用して EOT モードが選択された場合、文字列内でテキスト終了 (EOT) 区切り文字が見つかったとき。 (Set Session Parameters (9) on page 145) を参照してください。
   - EOT モードでない場合、長さで指定された数に達した場合。
   - フィールドでフィールド終了が検出されたとき。

   Note: ホスト表示スペースの最後のフィールドがラップする場合、表示スペースの最後に達したときにラップが発生します。
3. キーボード・ニーモニック (**「キーの送信」**(3) ファンクションを参照) は、「**フィールドへストリングのコピー**」機能を使用して送信することはできません。
4. 転送されるデータの最初のバイトは、常に、指定された PS 位置を含むフィールドの開始に配置されます。

Note: 5250 エミュレーションは、24 行 x 80 列の表示スペースをサポートします。場合によっては、Communication Manager 5250 エミュレーションは 25 行を表示します。これは、ホストからのエラー

メッセージが表示されたとき、またはオペレータが SysReq キーを選択したときに発生します。Z and I Emulator for Windows 25 行の情報を、24 行またはステータスバーに表示します。ステータスバーに情報を表示するには、ステータスバーを設定する必要があります参照*Quick Beginnings*ステータスバーの設定の詳細については、を参照してください。**EXTEND_PS**オプションを指定し、EHLLAPIアプリケーションは、 ~と同じインターフェースを使用できます。Communication ManagerEHLLAPIこの状態が発生すると、有効な表示スペースが拡張されます。

Copy String to Presentation Space (15)

3270	*5250*	*VT*
はい	はい	はい

Copy String to Presentation Space 関数は、ASCII データ・ストリングをホスト表示スペースの PS 位置呼び出しパラメーターで指定した位置に、直接コピーします。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	15 でなければなりません。	
データ・ストリング	ホスト表示スペースにコピーする ASCII データ・ストリング。	
長さ	コピー元データ・ストリングの長さ (バイト数)。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	コピーを開始するホスト表示スペースの位置。 1 とホスト表示スペースの構成サイズの間の値です。	

戻りパラメーター

戻りコード	**説明**
0	**Copy String to Presentation Space** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーターが誤っているか、 パラメーター長 0 がコピーに指定されました。
5	コピー先表示スペースが保護または入力禁止になっているか、 あるいは、無効なデータがコピー先表示スペースに 送られました (フィールド属性バイトなど)。
6	コピーは完了しましたが、データが切り捨てられました。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。

使用上の注意

1. 「**表示スペースへストリングのコピー**」機能は、以下のオプションの影響を受けます。
   - 文字列/文字列
   - EOT
   - EAB/NOEAB
   - XLate/NOXLate
   - プテータブ/NOPUTEAB
   - EXTEND_PS/NOEXTEND_PS

   項目を参照1 on page 147and2 on page 147 ;13 on page 151 and14 on page 152 ;18 on page 153 ; and#unique_169_Connect_42_opt18 and#unique_169_Connect_42_opt19を参照してください。
2. キーボード・ニーモニック (**キーの送信**(3) 機能を参照) は、表示**スペースへのストリングのコピー**機能を使用して送信することはできません。
3. **セッションパラメータの設定**(9) 関数を使用して EOT モードが選択された場合、テキスト終了 (EOT) 区切り文字がストリング内で検出されると、ストリングは終了します。 (Set Session Parameters (9) on page 145) を参照してください。
4. **「キー (3) の送信**」ファンクションは、同じ目的を実行しますが、この関数の方がプロンプトで応答し、より素早くコマンドを入力します。**キー送信**(3) ファンクションは、端末オペレーターがキーボードからデータを入力することをエミュレートするため、大量のデータを処理するアプリケーションにとっては、処理速度が遅くなります。この関数は、ホストへのより高速な入力パスを提供します。
5. 元のデータ (コピーされたストリング) は、表示スペースのサイズを超えることはできません。
6. ホストアプリケーションによっては、この関数呼び出しにより、カーソルが予期しない位置に移動する場合があります。フィールドの入力には、この関数よりSendKey 関数の方が適切な選択です。

Note: 次の場合にのみ発生します。VTセッションまたは接続を入力します。

Copy Presentation Space to Clipboard (35)

3270	*5250*	*VT*
はい	はい	はい

Copy Presentation Space to Clipboard 関数は、ホスト接続表示スペースのすべてまたは一部を、クリップボードにコピーする場合に使用します。入力 PS 位置は、ホストの表示スペースのオフセットです。このオフセットは、左上隅 (行 1、桁 1) を位置 1、右下隅 を 3564 (ホスト表示スペースの最大表示画面サイズ) とするレイアウトを基にしています。PS 位置 + (長さ − 1) の値は、ホスト表示スペースの構成サイズを超えることはできません。

Copy Presentation Space to Clipboard はホストのコピー元の表示スペース内の文字を ASCII 形式に変換します。属性バイトやその他の通常 ASCII では表されない文字は、 ブランクに変換されます。属性バイトをブランクに変換したくない場合には、**Set Session Parameters (9)** 関数の ATTRB オプションを指定して、 この変換を一時変更することができます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	35 でなければなりません。	
データ・ストリング	事前にホスト表示スペースのサイズを割り振られた コピー先ストリング。EAB オプションを指定して **Set Session Parameters** (9) 関数を発行した場合、データ・ストリングの長さは少なくとも表示スペースの長さの 2 倍必要です。	
長さ	該当データ・ストリングの長さ	
PS 位置	コピー先データ・ストリングの第 1 バイトの ホスト表示スペース内の位置。	

戻りパラメーター

戻りコード	説明
0	ホスト表示スペースの内容はクリップボードにコピーされました。コピー先表示スペースは活動状態にあり、キーボードは アンロック されています。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	指定されたストリング長に誤りがありました。あるいは、 (データ・ストリング長 - 1) + PS 位置の合計が、接続されている ホスト表示スペースのサイズを超えています。
4	ホスト表示スペースの内容はクリップボードにコピーされました。ホスト表示スペースはホストの応答を待っています。
5	ホスト表示スペースはクリップボードにコピーされました。キーボードはロックされています。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。

使用上の注意

1. EAB を戻すには、**Set Session Parameters** (9) 関数の EAB オプションを使用します。EAB は表示スペースの各文字と関連しており、 それぞれの文字の後に戻されます。
2. **Copy Presentation Space to Clipboard** 関数は以下のオプションの影響を受けます。
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB
    - XLATE/NOXLATE
    - BLANK/NOBLANK
    - DISPLAY/NODISPLAY

- PUTEAB/NOPUTEAB
- EXTEND_PS/NOEXTEND_PS

3. データ・ストリング・バッファーは、表示スペースから取得されてクリップボードにコピーされるデータを処理するときに使用されます。指定されたデータ・ストリング・バッファーが、要求されたバイト数のデータを保持するだけの十分な大きさではない場合は、データ・ストリング・バッファーの終わりに達した時点でコピーは正常終了します (RC=0、4、または 5)。前述のように、さまざまな **Copy** (5、8、および 34) 関数により戻される属性は、**Set Session Parameters** (9) 関数の影響を受けます。関係するセッション設定パラメーターは、以下のような影響を及ぼします。

セッション設定パラメーター

Copy 機能への影響

NOEAB、NOEAD

属性は戻されません。テキストのみを表示スペースからクリップボードへコピーします。

EAB、NOXLATE

属性は以下の表に定義されたとおりに戻されます。

EAB、XLATE

表示スペースの表示に使用される色が戻されます。色は再マップできるので、XLATE と EAB が同時に ON になっている場合、 属性の色は **Copy** 関数によって戻される色ではありません。

Paste Clipboard to Presentation Space (36)

3270	*5250*	*VT*
はい	はい	はい

Paste Clipboard to Presentation Space 関数は、ホスト表示スペースにおいて PS 位置呼び出しパラメーターで指定された位置に ASCII データ・ストリングを直接貼り付けます。

前提呼び出し

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	36 でなければなりません。	
データ・ストリング	ホスト表示スペースに貼り付けるクリップボードのデータを保持するストリング・バッファー。	
長さ	貼り付けるデータの長さ (バイト)	

	標準インターフェース	拡張インターフェース
PS 位置	コピーを開始するホスト表示スペースの位置。 1 とホスト表示スペースの構成サイズの間の値です。	

戻りパラメーター

戻りコード	説明
0	**Paste Clipboard to Presentation Space** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーターが誤っているか、 パラメーター長 0 がコピーに指定されました。
5	コピー先表示スペースが保護または入力禁止になっているか、 あるいは、無効なデータがコピー先表示スペースに 送られました (フィールド属性バイトなど)。
6	コピーは完了しましたが、データが切り捨てられました。
7	コピーは完了しましたが、データが切り捨てられました。
9	システム・エラーが発生しました。

使用上の注意

1. **表示スペースへのクリップボードの貼り付け**機能は、以下のオプションの影響を受けます。
   - 文字列/文字列
   - EAB/NOEAB
   - EOT
   - XLate/NOXLate
   - プテータブ/NOPUTEAB
   - EXTEND_PS/NOEXTEND_PS

2. **セッションパラメータの設定 (9)**関数を使用して EOT モードが選択された場合、テキスト終了 (EOT) 区切り文字がストリング内で検出されると、ストリングは終了します。 (147 ページの「セッションパラメータの設定 (9)」を参照)。
3. 元のデータ (コピーされたストリング) は、表示スペースのサイズを超えることはできません。

ストリング	意味	1 バイト文字フィールド	
X'000C'	(NULL)(FF) X'00' X'0C'	(SB NULL)(SB FF) X'00' X'0C'	
X'0E000C0F'	(SO)(DB FF)(SI) X'0E' X'000C' X'0F'	-S エラー	

Note: SB は半角文字全角文字を意味します。

注: 5250 エミュレーションは、24 行 x 80 列の表示スペースをサポートします。Communication Manager 5250 エミュレーションで、25 行が表示されることがある。これは、ホストからのエラーメッセージが表示されたとき、またはオペレータが SysReq キーを選択したときに発生します。Z and I Emulator for Windows常に、24 行に同じ情

報を表示します。EHLLAPI アプリケーションは**EXTEND_PS**オプションを指定し、コミュニケーションマネージャ EHLLAPI と同じインターフェースを使用でき、この条件が発生したときに、有効な表示スペースが拡張されます。

Disconnect from Structured Fields (121)

3270	*5250*	*VT*
はい	いいえ	いいえ

Disconnect from Structured Fields 関数は、 エミュレーション・プログラムと EHLLAPI アプリケーションの間の接続を切断します。システムで終了する前に、EHLLAPI アプリケーションを エミュレーション・プログラムから切断する必要があります。以前に **Connect for Structured Fields** が発行されている場合には、EHLLAPI アプリケーションでこの関数要求を発行する必要があります。

Reset System (21) 関数も未処理の SF 接続を切断します。

前提呼び出し

Connect for Structured Fields (120)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	121 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	3 でなければなりません	8 である必要があります
PS 位置	該当なし	

データ・ストリングの内容

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
2–3	5–6	Connect for structured field (120) 関数が戻す固有の Destination/Origin ID
	7–8	予約済み。

戻りパラメーター

戻りコード	**説明**
0	**Disconnect from Structured Fields** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。

戻りコード	説明
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
40	アプリケーションは非同期要求が保留のまま切断されました。

使用上の注意

1. **Disconnect from Structured Fields** 関数が呼び出された時点で、アプリケーションが **Get Request Completion** (125) 関数呼び出しを発行すると、非同期の **Read Structured Fields** (126) や **Write Structured Fields** (127) の関数要求の未処理のものはどれも戻されます。切断呼び出しを発行した後で実行取り消しする場合は、この関数の非同期形式を 使用してください。
2. **Reset System** (21) 関数も未処理の非同期要求 (**Get Request Completion** (125) 関数をアプリケーションが検索できなかった要求) を解放します。

Disconnect Presentation Space (2)

3270	*5250*	*VT*
はい	はい	はい

Disconnect Presentation Space 関数は、EHLLAPI アプリケーション・プログラムとホスト表示スペースの間の接続を切断します。また、**Reserve** (11) 関数によりホスト表示スペースが予約済みの場合、 それは **Disconnect Presentation Space** 関数の実行時に解放されます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	2 でなければなりません	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

戻りコード	説明
0	**Disconnect Presentation Space** 関数は正常終了しました。
1	ユーザー・プログラムが現在 ホスト表示スペースに接続されていません。
9	システム・エラーが発生しました。

使用上の注意

1. **Disconnect Presentation Space** 関数が呼び出されると 、ホスト接続表示スペースと対話する関数 (**Send Key** (3)、**Wait** (4)、**Reserve** (11) および **Release** (12) 関数など) は無効になります。
2. EHLLAPI アプリケーションを終了する前に、接続されている ホスト表示スペースを切断する必要があります。
3. **Disconnect Presentation Space** 関数は、セッション・パラメーターのデフォルト値へのリセットを行いません。デフォルト値へのリセットを行うためには、EHLLAPI アプリケーションは **Reset System** (21) 関数を呼び出す必要があります。

Disconnect Window Service (102)

3270	*5250*	*VT*
はい	はい	はい

Disconnect Window Service 関数は、EHLLAPI プログラムと指定されたホスト表示スペース・ウィンドウの間のウィンドウ・サービス接続を切断します。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	102 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	1	4
PS 位置	該当なし	

データ・ストリングの内容

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

戻りパラメーター

戻りコード	**説明**
0	**Disconnect Window Service** 関数は正常終了しました。

戻りコード	説明
1	ユーザー・プログラムが Window Service に接続されていません。
9	システム・エラーが発生しました。

使用上の注意

Disconnect Window Service 関数を呼び出すと、 アプリケーションはその表示スペース・ウィンドウを管理できなくなります。

アプリケーションを終了する前に、Presentation Manager® サービス用に接続されているすべての表示スペースに対して **Disconnect Window Service** 関数を要求する必要があります。アプリケーションがウィンドウ・サービスのための接続をしたまま終了すると、 サブシステムがその接続を取り消します。

EditKey インターセプト

このフィーチャーによって、既存のすべてのキー・ストロークに加えて編集キーをインターセプトして、Windows 32 ビット環境のセッションに送信することができます。

前提条件

1. 「キーボードのカスタマイズ」ウィンドウで、Edit 関数をマップします (例: 編集コピー関数は Ctrl+C)。
2. キー・ストローク・インターセプトの開始 (50) EHLLAPI 機能を、呼び出しパラメーター・データ・ストリング値を設定して呼び出します。値は以下のとおりです。

バイト位置	目次
1	以下のいずれかの値です。 ◦特定のホスト表示スペースの短縮名 (PSID) ◦ホストに接続されたホスト表示スペースの要求を示すブランクまたはヌル
2 から 4	予約済み
5	オプション・コード文字: ◦D AID キー・ストロークのみ。 ◦L すべてのキー・ストローク。 ◦E すべてのキー・ストロークおよび編集キー。 ◦通知の非同期メッセージ・モードを要求する場合は M (Windows のみ)。M が指定された場合には、位置 13 にコード文字 D、L、または E を配置する必要があります。
6 から 8	予約済み
9 から 12	位置 5 に M が指定された場合には、メッセージを受信するウィンドウのウィンドウ・ハンドル。メッセージは、RegisterWindowMessage (PCSHLL) のゼロ以外の戻り値です。
13	位置 5 に M が指定された場合には、以下のうちのいずれかの値。

バイト位置	目次
	◦ D AID キー・ストロークのみ。 ◦ L すべてのキー・ストローク。 ◦ E すべてのキー・ストロークおよび編集キー。
14 から 16	予約済み

3. 代行受信された編集キーを取得するには、Get Key (51) EHLLAPI 関数を使用します。編集キーのデータ・ストリングで戻されるキーの略号には、 バイト位置 5 で M (キー・ストローク・タイプの略号) を使用します。続く 4 バイトは、インターセプトされた編集キーに基づいて、以下の編集キー・ニーモニックのいずれかになります。

キーの略号	代行受信されたキー
@W@C	編集 コピー
@W@D	編集 消去
@W@E	編集 コピー 追加
@W@L	編集 コピー リンク
@W@N	編集 貼り付け 次へ
@W@V	編集 貼り付け
@W@X	編集 切り抜き
@W@Z	編集 取り消し

4. 編集キーをセッションに送信するには、キーの送信 (3) EHLLAPI 機能を使用します。呼び出しパラメーターとして渡されるデータ・ストリングでは、以下の編集キー・ニーモニックを指定できます。

キーの略号	送信されるキー
@W@C	編集 コピー
@W@D	編集 消去
@W@E	編集 コピー 追加
@W@L	編集 コピー リンク
@W@N	編集 貼り付け 次へ
@W@V	編集 貼り付け
@W@X	編集 切り抜き
@W@Z	編集 取り消し

注:

1. キーの送信 (3) 機能を使用するために、キーの取得 (51) EHLLAPI 機能を呼び出す必要はありません。編集キーを処理する Get Key (51) 関数および Send Key (3) 関数の両方で、バイト位置 5 を **E** に設定して、Start Keystroke Intercept (50) を最初に呼び出す必要があります。5 番目のバイトに **M** が含まれている場合、バイト位置 13 には **E** が含まれている必要があります。
2. キー・ストローク・インターセプトの開始 (50)、キーの取得 (51)、およびキーの送信 (3) 機能の期待される戻り値は変わりません。
3. ここで記述されている前提条件に加えて、既存の資料の前提条件に従う必要があります。

Find Field Length (32)

3270	*5250*	*VT*
はい	はい	はい

Find Field Length 関数は、接続されている表示スペース内の該当フィールドの長さを戻します。この関数は、フィールド形式のホスト表示スペース内のフィールドに限り、保護フィールドや無保護フィールドを検索するのに使用できます。

この関数は、呼び出し PS 位置パラメーターを使用して識別したフィールドに 入っている文字の数を戻します。このフィールドには、該当フィールドの先頭文字から次の属性バイトの前の文字 までのすべての文字が含まれます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	32 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	該当なし	該当なし
PS 位置	注を参照	

注: PS 位置: Find を開始するホスト表示スペース内のフィールドを識別します。これは、**Find** を開始するフィールド内のどのバイトの PS 位置でも構いません。

2 文字の呼び出しデータ・ストリングの内容は次のとおりです。

コード	**説明**
ƀbまたは T ƀ	このフィールドは、
P ƀ	前のフィールド。保護、無保護のいずれでも可。
N ƀ	次のフィールド。保護または無保護。

コード	説明
NP	次の保護フィールド。
NU	次の無保護フィールド。
PP	前の保護フィールド。
PU	前の無保護フィールド。

注: b 記号は、必要なブランクを表します。

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

次の長さが有効です。

長さ	説明
= 0	戻りコード = 28 の場合、フィールド長は 0 です。戻りコード = 24 の場合、 ホスト表示スペースはフィールド・フォーマットになっていません。
> 0	ホスト表示スペース内の必要なフィールドの長さ。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Find Field Length** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター・エラーが発生しました。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	該当するフィールドがありません。
28	フィールドの長さが 0 バイトです。

使用上の注意

bbまたは rbが呼び出しデータ・ストリングとして使用される場合を除き、見つかったフィールドが **Find** を開始したフィールドと同じ場合には、戻りコード 24 が戻されます。

Find Field Position (31)

3270	*5250*	*VT*
はい	はい	はい

Find Field Position 関数は、ホスト接続されている表示スペース内の該当フィールドの開始位置を戻します。この関数は、フィールド形式のホスト表示スペース内のフィールドに限り、保護フィールドや無保護フィールドを検索するのに使用できます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	31 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	該当なし	該当なし
PS 位置	注を参照	

注: PS 位置: Find を開始するホスト表示スペース内のフィールドを識別します。これは、**Find** を開始したいフィールド内のどのバイトの PS 位置でも構いません。

2 文字の呼び出しデータ・ストリングの内容は次のとおりです。

コード	**説明**
ƀbまたは T ƀ	このフィールドは、
P ƀ	前のフィールド。保護、無保護のいずれでも可。
N ƀ	次のフィールド。保護または無保護。
NP	次の保護フィールド。
NU	次の無保護フィールド。
PP	前の保護フィールド。
PU	前の無保護フィールド。

注: ƀ 記号は、必要なブランクを表します。

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

次の長さが有効です。

長さ	**説明**
= 0	戻りコード = 28 の場合、フィールド長は 0 です。。戻りコード = 24 の場合、 ホスト表示スペースはフィールド・フォーマットになっていません。

長さ	説明
> 0	ホスト表示スペースの起点からの、要求された フィールドの相対的位置。この位置は、属性バイトの直後になるように定義されています。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Find Field Position** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター・エラーが発生しました。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	該当するフィールドがありません。
28	フィールドの長さが 0 バイトです。

使用上の注意

ƀbまたは ŧbが呼び出しデータ・ストリングとして使用される場合を除き、見つかったフィールドが **Find** を開始したフィールドと同じ場合には、戻りコード 24 が戻されます。

Free Communications Buffer (124)

3270	*5250*	*VT*
はい	いいえ	いいえ

Free Communications Buffer 関数は、アプリケーションがもう必要としないバッファーをメモリー・マネージメントに戻します。終了する前に、アプリケーションはバッファーを解放しなければなりません。

前提呼び出し

Allocate Communications Buffer (123)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	124 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	6 でなければなりません	8 である必要があります
PS 位置	該当なし	

データ・ストリングの内容

バイト		定義
標準	拡張	
1–2	1–4	必ず 0 にしてください。
3–6	5–8	バッファーのアドレス

戻りパラメーター

戻りコード	説明
0	**Free Communications Buffer** 関数は正常終了しました。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
41	要求されたバッファーは使用中です。

使用上の注意

1. アプリケーションが使用中のバッファーを解放しようとすると、 その解放要求は拒否され、戻りコード 41 が戻されます。
2. アプリケーションを終了する前に、**Allocate Communications Buffer** (123) 関数を使用して割り振られたすべての通信バッファーに対して **Free Communications Buffer** (124) 関数を要求する必要があります。
3. **Reset System** (21) 関数を使用することによっても、**Allocate Communications Buffer** (123) 関数で割り振られたバッファーを解放できます。

Get Key (51)

3270	*5250*	*VT*
はい	はい	はい

Get Key 関数を使用して、**Start Keystroke Intercept** (50) 関数によって指定したセッションからのキー・ストロークを EHLLAPI アプリケーション・プログラムで取り出し、 そのキー・ストロークを処理、受け入れ、または拒否することができます。この関数をループに配置することで、文字列を代行受信するために使用できます。

前提呼び出し

Start Keystroke Intercept (50)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	51 でなければなりません	

	標準インターフェース	拡張インターフェース
データ・ストリング	次の表を参照してください。	
長さ	8	12
PS 位置	該当なし	

データ・ストリングの内容

バイト		定義
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮名 (PSID) • ホスト接続表示スペースに対する関数呼び出しを示す、 ブランクまたは NULL 文字
	2–4	予約済み
2–8	5–11	要求されたデータのシンボリック表示用のスペース確保のためのブランク
	12	予約済み

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

次の表を参照してください。

バイト		定義
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮名 (PSID) • ホスト接続表示スペースに対する関数呼び出しを示す、 ブランクまたは NULL 文字
	2–4	予約済み
2	5	次のいずれかのオプションのコード文字: • ASCII が戻されたことを示す `A` • キー・ストローク略号を示す `M` • 特殊略号を示す `S`
3–8	6–11	この事前割り振りされた 6 バイトのバッファー・スペースは、キー・ストロークの エンキューおよびデキューのために内部的に使用されます。以下の組み合わせが可能:

バイト		定義
		• バイト 3 に ASCII 文字、バイト 4 に X'00' • バイト 3 にエスケープ文字 (@、または機能 9 の ESC=c オプションを使用して指定したその他の文字) およびバイト 4 に 1 バイトの関数の省略文字。(ASCII 略号 (ページ 24) を参照してください。) • バイト 5 ~ 8 は、戻された ASCII 略号が 2 バイトより長い場合、バイト 3 および 4 と同様になります (例えば、Attn の ASCII 略号が @A@Q の場合、バイト 5 に @、 バイト 6 に Q が入ります)。使用されない場合、バイト 5 ~ 8 は 0 (X'00') に設定されます。

説明のため、戻されるデータ・ストリングの例を以下に示します。

注: @ 記号はデフォルトのエスケープ文字です。エスケープ文字の値は、**Set Session Parameters** (9) 関数の ESC=c オプションを使用して ASCII で表される任意のキー・ストロークに設定できます。このオプションを使用して、エスケープ文字を別の文字に変更している場合は、 以下の例の中の @ 記号をその変更している文字に置き換えて考えてください。

16 ビット・インターフェースの場合

EAt

E は表示スペースの短縮名です。キー・ストロークは ASCII として返され (A)、 返されるキーは小文字の t です (バイト 4 ~ 8 = X'00')。

EM@2

E は表示スペースの短縮名です。キー・ストロークは略号として戻され、戻されるキーは PF2 です (バイト 5 ~ 8 = X'00')。

32 ビット・インターフェース

Ebbb**At**

E は表示スペースの短縮名です。キー・ストロークは ASCII として返され (A)、 返されるキーは小文字の t です(バイト 7 ~ 11 = X'00')。

Ebbb**M@2**

E は表示スペースの短縮名です。キー・ストロークは略号として戻され、 戻されるキーは PF2 です (バイト 8 ~ 11 = X'00')。

戻りコード:

以下のコードが有効です。

戻りコード	説明
0	**Get Key** 関数は正常終了しました。

戻りコード	説明
1	無効な表示スペースが指定されました。
5	**Start Keystroke Intercept** (50) 関数で AID 専用オプションが指定されました。EHLLAPI が表示スペースに無効なキーを書き込もうとするときには、このセッション・タイプでは非 AID キーは禁止されます。
8	この表示スペースに対して、先に **Start Keystroke Intercept** (50) 関数が呼び出されていません。
9	システム・エラーが発生しました。
20	定義されていないキーの組み合わせがタイプされました。
25	要求されたキー・ストロークは入力キューで使用できません。
31	キー・ストロークのキューがオーバーフローし、キー・ストロークが失われました。

使用上の注意

1. **Get Key** 関数で戻りコード 31 が発生する場合、以下のどちらかを行ってください。
   - **Start Keystroke Intercept** (50) 関数の呼び出しデータ・ストリング長パラメーターの値を増やしてください。
   - **Get Key** 関数をもっと頻繁に実行してください。

   代行受信されたキー・ストロークは、バッファー中の 3 バイトを使用します。次の代行受信されたキー・ストロークは、隣接する 3 バイトに置かれます。**Get Key** 関数がキー・ストロークを取り出す (先入れ先出し、FIFO) と、そのキー・ストロークが使用していた 3 バイトは他のキー・ストロークで使用できるようになります。バッファーのサイズを増やすか、バッファーからキー・ストロークを取り出す 頻度を増やすことによって、バッファーのオーバーフローを避けることができます。

   PC/3270 において、戻りコード 31 を除去する他の方法は、再開モードで PC/3270 エミュレーターを操作することです。
2. **Send Key** (3) 関数 を使用すれば、元のキー・ストロークと、EHLLAPI アプリケーションが必要とする可能性のある他のキー・ストロークとの両方をホスト接続表示スペースに渡すことができます。
3. キー・ストロークは非同期に受信され、**Start Keystroke Intercept** (50) 関数を使用して EHLLAPI アプリケーション・プログラムで指定したキー・ストロークのキューに入れられます。
4. **Get Key** 関数は read コマンドと同様に機能します。キー・ストロークが使用可能な場合、キー・ストロークはユーザーのアプリケーションで用意したデータ域に読み込まれます。
5. セッションに対するフィールド・サポートの場合、 アプリケーションは Enter キーなどの AID キーだけが必要かもしれません。この場合には、**Start Keystroke Intercept** (50) 関数のオプション・コードを `D` (AID キー専用を表す) に設定します。
6. この関数を使用するには、戻りデータ・ストリング・パラメーター を受け入れるメモリーを事前に割り振ります。このメモリーの事前割り振りに必要なステートメントは、 アプリケーションに使用される言語によって異なります。詳しくは、メモリー割り振り (ページ 14)を参照してください。

Get Request Completion (125)

3270	*5250*	*VT*
はい	いいえ	いいえ

Get Request Completion 関数によって、 アプリケーションは EHLLAPI に対して前回発行した非同期の関数要求の状況を判断して、データ・ストリングを再び使用する前に関数パラメーターのリストを取得できます。この関数は、**Read Structured Fields** (126) や **Write Structured Fields** (127) などの関数を前回呼び出した際に、ユーザーが非同期の (A) 完了を指定した場合に限り有効です。

Get Request Completion 関数が必要な非同期要求はそれぞれ、非同期要求から固有の ID を戻します。アプリケーションはこの ID を保管する必要があります。この ID は、必要な要求を識別するために **Get Request Completion** 関数が使用する ID です。ユーザーは、この関数を使用する際に、以下の 3 つの要求オプションを 指定できます。

1. 関数の要求 ID およびブランク以外のセッション短縮名を指定することにより、 アプリケーションは特定の非同期関数要求を照会または待機できます。
2. X'0000'という要求 ID とブランク以外のセッションの短縮名を指定することにより、アプリケーションは特定のセッションの最初に完了する非同期関数要求を照会または待機できます。

前提呼び出し

Connect Structured Fields (120) および **Allocate Communications Buffer** (123)

および

Read Structured Fields (126) または **Write Structured Fields** (127)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	125 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	14 でなければなりません	24 でなければなりません
PS 位置	該当なし	

データ・ストリングの内容

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

バイト		定義
2	5	N または W。N=NOWAIT (非待機) が要求されました。W=WAIT (待機) が要求されました。
	6–8	予約済み
3–4	9–10	関数要求 ID
5–6	11–12	予約済み
7–10	13–16	予約済み
11–12	17–20	予約済み
13–14	21–24	予約済み

Get Request Completion 関数の動作は、 パラメーター・ストリングの 2 番目の文字により異なります。その文字は以下のいずれかです。

N

非待機オプション: 特定の要求 ID が指定され、その関数が完了している場合、戻りコード 0 および戻りパラメーター (ページ 95)に定義された完了済みデータ・ストリングとともに、アプリケーションに制御が戻されます。要求 ID に 0 が指定され、該当する非同期関数が完了している場合、 戻りコード 0 および戻りパラメーター (ページ 95)に定義された完了済みデータ・ストリングと共に、 アプリケーションに制御が戻されます。

水

待機オプション: 特定の要求 ID が指定され、関数が完了していない場合、 呼び出しは関数が完了するのを待ってからアプリケーションに戻ります。要求 ID に 0 が指定され、該当する非同期関数が完了していない場合、呼び出しは 関数が完了するのを待ってから呼び出し側のアプリケーションに戻ります。アプリケーションに戻った時点で、戻りコードの値は 0 になり、 データ・ストリングは戻りパラメーター (ページ 95)に定義されたとおりに完成されます。

戻りパラメーター

バイト		定義
標準	拡張	
5–6	11–12	完了した非同期関数の関数番号 (126 または 127)。(戻り値)
7–10	13–16	完了した非同期関数呼び出しのデータ・ストリングのアドレス(アプリケーションは要求が完了するまでデータ・ストリングを再度使用することができません)。(戻り済み)
11–12	17–20	完了した非同期関数呼び出しのデータ・ストリングの長さ。(戻り済み)
13–14	21–24	完了した非同期関数呼び出しの戻りコード。(戻り済み)

戻りコード	説明
0	**Get Request Completion** 関数は正常終了しました。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。

戻りコード	説明
38	要求された関数は完了しませんでした。
42	該当する要求がありません。

戻りコード 38 と 42 には、次のような相違点があります。

1. 戻りコード 38
    a. 特定の要求 ID とセッションが要求された場合、セッションと ID の両方とも 検出されたにもかかわらず、要求が保留されている (完了状態でない)。
    b. 要求 ID ゼロと特定のセッションが要求された場合、指定されたセッションに 保留されている要求があるが、それらの要求は満たされて (完了して) いない。
    c. 要求 ID ゼロとブランク・セッションが要求された場合、保留されている要求が 見つかったが、どれも要求が満たされて (完了して) いない。
2. 戻りコード 42
    a. 特定の要求 ID とセッションが要求された場合、特定の要求 ID が 保留状態でも完了状態でも検出されない。
    b. 要求 ID ゼロと特定のセッションが要求された場合、その特定のセッションに 保留要求または完了済み要求が含まれていない。
    c. 要求 ID ゼロとブランク・セッションが要求された場合、 保留要求も完了済み要求も検出されなかった。

使用上の注意

1. この関数は、ユーザーが **Read Structured Fields** または **Write Structured Fields** などの関数を前回呼び出した際に、非同期完了 (Asynchronous の A) を指定した場合に限り有効です。
2. 戻りコードが 0 の場合、アプリケーションは、要求された非同期関数の 完了に関する情報が入っている、戻されたデータ・ストリングを検査する必要が あります。

Lock Presentation Space API (60)

3270	***5250***	***VT***
はい	いいえ	いいえ

Lock Presentation Space API 関数によって、アプリケーションは、他の Windows 32 ビット・アプリケーションに対して、表示スペース・ウィンドウの排他制御を獲得したり解放したりすることができます。ロックされている間、他のアプリケーションは表示スペース・ウィンドウに接続することはできません。

ロックを指定したこの関数が正常に処理されると、他の EHLLAPI アプリケーション から要求された EHLLAPI 表示スペース・ウィンドウ関数は、 この関数を要求したアプリケーションが表示スペースを アンロックするまで、待機させられます。ロックを行ったアプリケーションからの要求は、通常どおり処理されます。

前提呼び出し

Connect to Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	60 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	3 でなければなりません	8 である必要があります
PS 位置	該当なし	

データ・ストリングの内容

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
2	5	次のいずれかを示します。 • L - API をロックする場合。 • U - API をアンロックする場合
3	6	次のいずれかを示します。 • R - 表示スペースが既にアプリケーションによってロックされているときに戻る場合。 • Q - 表示スペースが既にアプリケーションによってロックされているときに、ロック要求をキューに入れる場合。
	7–8	予約済み。

戻りパラメーター

戻りコード	説明
0	**Lock Presentation Space API** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
43	API が既に他の EHLLAPI アプリケーションによってロックされていたか (ロック要求の場合)、または API がロックされていませんでした (アンロック要求の場合)。

使用上の注意

以下の EHLLAPI 関数は、ロックされている間は待機させられます。

- **Send Key** (3)
- **Copy Presentation Space** (5)
- **Search Presentation Space** (6)
- **Copy Presentation Space to String** (8)
- **Release** (11)
- **Reserve** (12)
- **Query Field Attribute** (14)
- **Copy String to Presentation Space** (15)
- **Search Field** (30)
- **Find Field Position** (31)
- **Find Field Length** (32)
- **Copy String to Field** (33)
- **Copy Field to String** (34)
- **Set Cursor** (40)
- **Send File** (90)
- **Copy Presentation Space to Clipboard** (35)
- **Paste Clipboard to Presentation Space** (36)
- **Receive File** (91)
- 先行の **Set Sessions Parameter** (9) 関数呼び出しで CONPHYS パラメーターが設定されている **Connect to Presentation Space** (1)

これらの待機させられた要求は、ロックが解除されるまでサービスされません。ロックが解除されると、待機させられた要求は、先入れ先出し法 (FIFO) の順序で処理されます。上記以外の EHLLAPI 関数は、ロックされていない場合と同じように実行されます。ロックを要求したアプリケーションは、以下のいずれかの方法で、表示スペース・ウィンドウをアンロックします。

- まだロックを所有している間に、表示スペースとの接続を 切断する。
- まだロックを所有している間に、**Reset System** (21) 関数を発行する。
- まだロックを所有している間に、アプリケーションを停止する。
- セッションを停止する。
- アンロック・オプションを指定した **Lock Presentation Space API** を正常に発行する。

アプリケーションを終了する前に、**Lock Presentation Space API** 関数を使用してロックしていたすべての表示スペース・ウィンドウをアンロックする必要があります。アプリケーションが未処理のロックをそのままにして終了するか 、**Reset System** (21) 関数または **Disconnect Presentation Space** (2) 関数が発行されると、ロックは解放されます。

アプリケーションでは、短時間に限り、しかも表示スペースの 排他的使用が必要なときにのみ、表示スペースをロックする ことをお勧めします。

Lock Window Services API (61)

3270	*5250*	*VT*
はい	いいえ	いいえ

Lock Window Services API 関数によって、アプリケーションは、他の Windows 32 ビット・アプリケーションに対して、表示スペース・ウィンドウの排他制御を獲得したり解放したりすることができます。ロックされている間、他のアプリケーションは表示スペース・ウィンドウに接続することはできません。

ロックを指定したこの関数が正常に処理されると、他の EHLLAPI アプリケーション から要求された EHLLAPI 表示スペース・ウィンドウ関数は、 この関数を要求したアプリケーションが表示スペースを アンロックするまで、待機させられます。ロックを行ったアプリケーションからの要求は、通常どおり処理されます。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	61 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	3 でなければなりません	8 である必要があります
PS 位置	該当なし	

データ・ストリングの内容

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
2	5	次のいずれかを示します。 • L - API をロックする場合。 • U - API をアンロックする場合
3	6	次のいずれかを示します。 • R - 表示スペースが既にアプリケーションによってロックされているときに戻る場合。 • Q - 表示スペースが既にアプリケーションによってロックされているときに、ロック要求をキューに入れる場合。
5–6	11–12	完了した非同期関数の関数番号 (126 または 127)。(戻り値)

バイト		定義
	7–8	予約済み。

戻りパラメーター

戻りコード	説明
0	**Lock Window Services API** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
38	要求された関数は完了しませんでした。
43	API が既に他の EHLLAPI アプリケーションによってロックされていたか (ロック要求の場合)、または API がロックされていませんでした (アンロック要求の場合)。

使用上の注意

以下の EHLLAPI 関数は、ロックされている間は待機させられます。

- **Window Status** (104)
- **Change Switch List Name** (105)
- **Change PS Window Name** (106)

これらの待機させられた要求は、ロックが解除されるまでサービスされません。ロックが解除されると、待機させられた要求は、先入れ先出し法 (FIFO) の順序で 処理されます。

ロックを要求したアプリケーションは、以下のいずれかの方法で、表示スペース・ウィンドウをアンロックします。

- UNLOCK オプションを指定した **Lock Window Services API** を正常に発行する。
- まだロックを所有している間に、表示スペースとの接続を 切断する。
- まだロックを所有している間に、**Reset System** (21) 関数を発行する。
- まだロックを所有している間に、アプリケーションを停止する。
- セッションを停止する。

アプリケーションを終了する前に、**Lock Window Services API** 関数を使用してロックしていたすべての表示スペース・ウィンドウをアンロックする必要があります。アプリケーションがロックを未処理のまま終了すると、サブシステムは そのロックを解放します。

アプリケーションでは、短時間に限り、しかも表示スペースの 排他的使用が必要なときにのみ、表示スペースを ロックする ことをお勧めします。

Pause (18)

3270	*5250*	*VT*
はい	はい	はい

Pause 関数は、指定した時間だけ待機します。この機能をタイミング・ループの代わりに使用して、イベントの発生を待機することができます。**Pause** 関数は、先に **Start Host Notification** (23) 関数が呼び出され、IPAUSE オプションが選択されている場合、ホストのイベントにより終了できます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	18 でなければなりません	
データ・ストリング	該当なし	
長さ	2 分の 1 秒単位で待機時間を指定します	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**定義**
0	待機時間が終了しました。
9	内部システム・エラーが発生しました。時間の結果は予測できません。
26	ホスト・セッションの表示スペースまたは OIA が既に更新されています。詳細情報を得るには、**Query Host Update** (24) 関数を使用してください。

使用上の注意

1. **Set Session Parameters** (9) 関数を使用して 、FPAUSE オプションか IPAUSE オプションを選択すると、 この関数を呼び出すときの停止時間の長さに影響を与えます。詳しくは、6 (ページ 148) を参照してください。
2. 呼び出しデータ・ストリング長パラメーターに入力される値は、 **Pause** 関数が待機する時間の最大数 (2 分の 1 秒単位) です。20 秒の休止の場合、呼び出しデータ・ストリング長パラメーターに 16 進数値の `0028` (10 進数で 40) を渡す必要があります。
3. IPAUSE オプションを使用し、Pause 関数の値がゼロの場合、Pause 関数は最大 2400 (2 分の 1 秒単位) 待機します (これより前に割り込みが発生しない場合)。FPAUSE オプションを使用し、Pause 関数の値がゼロの場合は、 この関数は即座に戻ります。

4. IPAUSE オプションを使用するときは、ホストのイベントにより一時停止の条件が満たされたら、**Query Host Update** (24) 関数を呼び出して次の **Pause** 関数の前にキューを消去してください。**Pause** 関数は、**Query Host Update** (24) 関数が完了するまで、継続して保留状態のイベントの条件を満たします。
5. **Pause** 関数の実際の最大値は `2400` です。**Pause** 関数は次のようなタスクには使用しないでください。
    - 非常に長い期間の遅延 (例えば、数時間に渡るものなど)。
    - システム時刻クロックの検査前および EHLLAPI プログラム実行の続行前の、 適切な時間 (20 分) を超える遅延。
    - **Pause** 関数で作成される時間間隔は近似値であるため、 高機能タイマーを必要とするアプリケーションを使う場合。
    - ループ内での時間間隔ゼロの設定。
6. IPAUSE 設定および割り込み可能休止は、指定された ホスト表示スペース (PS) またはオペレーター情報域 (OIA) が 更新されたかどうかを EHLLAPI アプリケーションが判別できるようにします。以下の 3 つの関数を使用します。
    - **Start Host Notification** (23)
    - **Query Host Update** (24)
    - **Stop Host Notification** (25)

    Start 関数が呼び出されるときに IPAUSE を使用して、 ホスト表示スペースまたは OIA (あるいはこの両方) が更新を受信するまで、アプリケーションを待機させることができます。受信が完了し、アプリケーションが変更点を判別するために **Query** 関数を発行できるようになった時点で、**Pause** は終了します。続いて、アプリケーションは **Search Presentation Space** (6) を発行し、予期した更新が行われたかどうか調べます。

Post Intercept Status (52)

3270	*5250*	*VT*
はい	はい	はい

Post Intercept Status 関数は、**Get Key** (51) 関数によって得られたキー・ストロークが受け入れられたか拒否されたかを Z and I Emulator for Windows エミュレーターに通知します。アプリケーションがキー・ストロークを拒否した場合には、**Post Intercept Status** 関数はビープ音を出します。

前提呼び出し

Start Keystroke Intercept (50)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	52 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	2 でなければなりません	8 である必要があります
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮名 • ホスト接続表示スペースに対する関数呼び出しを示す、 ブランクまたは NULL 文字
	2–4	予約済み
2	5	次のいずれかを示します。 • A キー・ストロークの受け入れ • R キー・ストロークの拒否
	6–8	予約済み。

戻りパラメーター

戻りコード	説明
0	**Post Intercept Status** 関数は正常終了しました。
1	無効な表示スペースが指定されました。
2	無効なセッション・オプションが指定されました。
8	この表示スペース ID に対して、先に **Start Keystroke Intercept** (50) 関数が呼び出されていません。
9	システム・エラーが発生しました。

Query Additional Field Attribute (45)

3270	*5250*	*VT*
いいえ	はい	いいえ

Query Additional Field Attribute 関数はホスト表示スペースの入力位置がある 5250 フィールドについての追加情報を戻します。この情報は定義済み構造の形式でデータ・ストリング・パラメーターに戻されます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	45 でなければなりません。	

	標準インターフェース	拡張インターフェース
データ・ストリング	8 バイト長の文字ストリング。	
長さ	8 が暗黙指定されます。	
PS 位置	コピー先フィールドを識別します。これはコピー先フィールド内のどのバイトの PS 位置でも構いません。	

呼び出しデータ・ストリングには次のものがあります。

バイト	定義
1–8	予約済み

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

関数は以下のデータ・ストリングを戻します。

バイト	定義
1–6	予約済み
7–8	以下を戻す、2 つの 8 ビット符号なし文字。 • フィールドが RTL であれば R、またフィールドが LTR であれば L。 • フィールドが大文字であれば U、フィールドが標準のケース・フィールドであれば L。

戻りコード:

以下の戻りコードが定義されています。

戻りコード	説明
0	**Query Additional Field Attribute** 関数は正常終了しました。
1	ユーザー・プログラムが 現在ホスト・セッションに接続されていません。
7	ホスト表示スペースの位置が正しくありません。
9	この位置ではフィールドが検出できませんでした。
24	フィールドが定様式化されていません。

Query Close Intercept (42)

3270	*5250*	*VT*
はい	はい	はい

Query Close Intercept 関数を使用して、アプリケーションはクローズ・オプションが選択されたかどうかを識別できます。

前提呼び出し

Start Close Intercept (41)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	42 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	必ず 1 にしてください。	4 である必要があります。
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	ホスト表示スペースの 1 文字の短縮セッション ID、 またはホスト接続されているセッションへの照会要求を示すブランクあるいは NULL。
	2–4	予約済み

戻りパラメーター

戻りコード	説明
0	クローズ割り込みは生じませんでした。
1	プレゼンテーション・ソースが正しくありません。
2	パラメーター指定でエラーが発生しました。
8	このホスト表示スペースに対して、先に **Start Close Intercept** (41) 関数が呼び出されていません。
9	システム・エラーが発生しました。
12	セッションが停止しました。
26	最後の Query Close Intercept 呼び出し後にクローズ・インターセプトが 発生しました。

Query Communications Buffer Size (122)

3270	*5250*	*VT*
はい	いいえ	いいえ

Query Communications Buffer Size 関数を使用して、 アプリケーションは、エミュレーション・プログラムがサポートするバッファーの最大サイズおよび最適サイズを判別することができます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	122 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	9 でなければなりません	20 でなければなりません
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2–3	5–8	サポートされるインバウンド・バッファーの最適サイズを 示す 16 ビット/32 ビットのフィールド (戻される値)
4–5	9–12	サポートされるインバウンド・バッファーの最大サイズを 示す 16 ビット/32 ビットのフィールド (戻される値)
6–7	13–16	サポートされるアウトバウンド・バッファーの 最適サイズを示す 16 ビット/32 ビットのフィールド (戻される値)
8–9	17–20	サポートされるアウトバウンド・バッファーの 最大サイズを示す 16 ビット/32 ビットのフィールド (戻される値)

戻りパラメーター

戻りコード	説明
0	**Query Communications Buffer Size** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
10	エミュレーション・プログラムはこの関数をサポートしていません。

使用上の注意

1. この関数を使用するようにユーザーに要求する方法はありません。これは必須の関数ではなく、アプリケーションはあらゆるシステムで 実行するように調整できます。
2. 戻されるバッファー・サイズは、実際にメディアを通じて送信されるレコードの サイズを表しています。DDM 接続の場合、**読み取り**構造化フィールドと**書き込み**構造化フィールドのデータ・バッファーに与えられる 8 バイトのヘッダーは取り除かれ、構造化フィールドの AID 値を含む 1 バイトが接頭部になります。アプリケーションは、データ・バッファー内の実際のデータ・サイズ (8 バイトのヘッダーを含めない) と、**Query Communications Buffer Size** 関数が戻すバッファー・サイズから 1 バイト引いたものを比較する必要があります。Destination/Origin 接続の場合も、**読み取り**構造化フィールドと**書き込み構造化フィールド**のデータ・バッファーに与えられる 8 バイトが取り除かれ、9 バイトがデータの接頭部になります。アプリケーションは、データ・バッファー内の実際のデータ・サイズ (8 バイトのヘッダーを含めない) と、**Query Communications Buffer Size** 関数が戻すバッファー・サイズから 9 バイト引いたものを比較する必要があります。
3. 最大バッファー・サイズは、ワークステーションのハードウェアがサポートする バイトの最大数、およびエミュレーターがサポートするバイトの最大数を表します。この最大バッファー・サイズは、ホスト側でも少なくともこれらの最大サイズを 受け入れるように構成されていれば、使用することができます。
4. 戻される最適バッファー・サイズはワークステーションのハードウェア とエミュレーターの両方がサポートする最適バイト数を表しています。ネットワーク構成によっては、下限値をこの最適バッファー・サイズ未満に設定 しているものもあります。このような場合、構造化フィールドのサポートには、エミュレーター構成 プロファイルのデータ転送バッファー・サイズの一時変更値が使用されます。エミュレーターの構成プロファイル内のバッファー・サイズの一時変更値は、**Query Communications Buffer Size** に反映されます。

Query Communication Event (81)

3270	*5250*	*VT*
はい	はい	はい

Query Communication Event 関数を使用して、 EHLLAPI プログラムは、通信イベントが発生したかどうかを判別することができます。

前提呼び出し

Start Communication Notification (80)

呼び出しパラメーター

	拡張インターフェース
関数番号	81 でなければなりません
データ・ストリング	1 文字のホスト表示スペースの短縮名、またはホスト接続されて いる表示スペースへの更新要求を示すブランクまたは NULL。
長さ	4 が暗黙指定されます

PS 位置	該当なし

呼び出しデータ構造には、以下の要素が含まれます。

バイト	定義
1	表示スペースの 1 文字の短縮名 (PSID)
2 ~ 4	予約済み

戻りパラメーター

戻りコード	定義
0	関数は正常終了しました。
1	誤った PSID が指定されました。
8	この PSID に対して、先に **Start Communication Notification** (80) 関数が呼び出されていません。
9	システム・エラーが発生しました。
21	指示された PSID が接続されました。
22	指示された PSID が切断されました。

Query Cursor Location (7)

3270	*5250*	*VT*
はい	はい	はい

Query Cursor Location 関数は、カーソル位置を戻すことにより、ホスト接続表示スペース内のカーソルの位置を示します。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	7 でなければなりません	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

カーソルのあるホスト表示スペースの位置

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Query Cursor Location** 関数は正常終了しました。
1	ユーザー・プログラムが 現在ホスト・セッションに接続されていません。
9	システム・エラーが発生しました。

Query Field Attribute (14)

3270	*5250*	*VT*
はい	はい	はい

Query Field Attribute 関数は、 ホスト表示スペースの入力位置があるフィールドの属性バイトを戻します。この情報は、戻りデータ・ストリング長パラメーターに戻されます。

PC/3270 については、次のことに注意してください。

- 画面が不定様式の場合、戻りデータ・ストリング長パラメーターには 0 が 設定されます。
- 属性バイトは C0 (16 進数) 以上です。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	必ず 14 にしてください。	
データ・ストリング	NA	
長さ	NA	
PS 位置	コピー先フィールドを識別します。これはコピー先フィールド内のどのバイトの PS 位置でも構いません。	

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

画面が定様式の場合には属性値、画面が不定様式の場合には 0 となります。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Query Field Attribute** 関数は正常終了しました。
1	ユーザー・プログラムが 現在ホスト・セッションに接続されていません。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	属性バイトがないか、またはホスト表示スペースが定様 式化されていません。

使用上の注意

戻されるフィールド属性の定義を以下の表に示します。ビット位置は、バイトの左端をビット 0 とする IBM 形式で表されます。

- 3270 フィールド属性:

ビット位置	意味
0–1	共に 1。フィールド属性バイト。
2	保護/無保護 0 = 無保護データ・フィールド 1 = 保護フィールド
3	A/N 0 = 英数字データ 1 = 数値データのみ
4–5	I/SPD 00 = 通常輝度、pen 検出不能 01 = 通常輝度、pen 検出可能 10 = 高輝度、pen 検出可能 11 = 非表示、pen 検出不能
6	予約済み
7	MDT 0 = フィールドは変更されていません。 1 = フィールドは変更されました。

- 5250 フィールド属性:

ビット位置	意味
0	フィールド属性フラグ

ビット位置	意味
	0 = フィールド属性フラグなし 1 = フィールド属性フラグあり
1	可視性 0 = 非表示 1 = 表示
2	保護/無保護 0 = 無保護データ・フィールド 1 = 保護フィールド
3	Intensity (輝度) 0 = 通常輝度 1 = 高輝度
4–6	フィールド・タイプ 000 = 英数字データ: すべての文字が使用可 001 = 英字のみ: 大文字と小文字、コンマ、ピリオド、ハイフン、 ブランク、Dup キーが使用可 010 = 数字シフト: 数字に対する自動シフト 011 = 数値データのみ: 0 ~ 9 の数字、コンマ、ピリオド、プラス、マイナス、ブランク、Dup キーが使用可 101 = 数値データのみ: 0 ~ 9 の数字、Dup キーが使用可 110 = 磁気ストライプ読み取り装置データのみ 111 = 符号付き数値データ: 0 ~ 9 の数字、プラス、マイナス、Dup キーが使用可
7	MDT 0 = フィールドは変更されていません。 1 = フィールドは変更されました。

Query Host Update (24)

3270	*5250*	*VT*
はい	はい	はい

Query Host Update 関数により、プログラム式操作は、次のいずれかにより、ホストがホスト表示スペースまたは OIA を変更したかどうか判別できます。

- **Start Host Notification** (23) 関数が呼び出された (**Query Host Update** 関数に対する初回呼び出し時のみ)
- **Query Host Update** 関数に対する前提呼び出し (初回呼び出しを除く、**Query Host Update** 関数に対するすべての呼び出し)

前提呼び出し

Start Host Notification (23)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	24 でなければなりません	
データ・ストリング	1 文字のホスト表示スペースの短縮名、またはホスト接続されて いる表示スペースへの更新要求を表すブランクあるいは NULL。	
長さ	1 が暗黙指定されます。	4 が暗黙指定されます
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

戻りパラメーター

戻りコード	定義
0	最後の呼び出し以降、更新されていません。
1	ホストの無効な表示スペースが指定されました。
8	このホスト表示スペース ID に対して、 先に **Start Host Notification** (23) 関数が呼び出されていません。
9	システム・エラーが発生しました。
21	OIA が更新されました。
22	表示スペースが更新されました。
23	OIA とホスト表示スペースが更新されました。
44	印刷は、プリンター・セッションで完了しています。

使用上の注意

該当する表示スペースをデータ・ストリングに指定しておくこと が必要です。ただし、更新の検査のためにホスト表示スペース に接続する必要はありません。

Query Session Status (22)

3270	*5250*	*VT*
はい	はい	はい

Query Session Status 関数は、セッション特有の情報を得るために使用します。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	16 ビット	**32 ビット**
関数番号	22 でなければなりません。	
データ・ストリング	該当表示スペースの短縮セッション ID (1 バイト) に、 戻りデータ用の 17 バイトを加えた、18/20 バイトのストリング。バイト位置 1 には、次のものを入れることができます。 1. ホスト接続の表示スペースに対する要求を示す ブランクまたは NULL 文字。 2. キーボード・オーナーの表示スペースに対する要求を 示す * (アスタリスク)。	
長さ	18 でなければなりません	20 でなければなりません
PS 位置	該当なし	

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2–9	5–12	セッションの正式名 (プロファイル名と同じ。またはプロファイルが設定されていない場合は、短縮名と同じ)
10	13	**Session** **タイプ** **D** 3270 表示装置 **E** 3270 印刷装置 **金** 5250 ディスプレイ

バイト		定義
		G 5250 プリンター **H** ASCII VT
11	14	セッションの特性。 以下のセッション特性ビットを含む 2 進数で表されています。 **ビット 0** EAB 0: セッションには基本属性があります。1: セッションには拡張属性があります。 **ビット 1** PSS 0: セッションはプログラム式シンボルをサポートしていません。1: セッションはプログラム式シンボルをサポートしています **ビット 2 ~ 7** 予約済み
12 ~ 13	15 ~ 16	ホスト表示スペースの行数。 2 進数で表されます。
14 ~ 15	17 ~ 18	ホスト表示スペースの桁数。 2 進数で表されます。
16 ~ 17	19 ~ 20	ホスト・コード・ページ数。 2 進数で表されます。
18		予約済み

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Query Session Status** 関数は正常終了しました。
1	ホストの無効な表示スペースが指定されました。
2	ストリング長が正しくありません。
9	システム・エラーが発生しました。

使用上の注意

1. この関数を使用するには、戻りデータ・ストリング・パラメーター を受け入れるメモリーを事前に割り振ります。このメモリーの事前割り振りに必要なステートメントは、 アプリケーションに使用される言語によって異なります。詳しくは、メモリー割り振り (ページ 14)を参照してください。

Query Sessions (10)

3270	*5250*	*VT*
はい	はい	はい

Query Sessions 関数により、各ホスト・セッションを記述している 16 バイト (標準インターフェースの場合は 12 バイト) のデータ・ストリングが戻されます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

機能	**説明**	
	標準インターフェース	拡張インターフェース
関数番号	10 でなければなりません	
データ・ストリング	16*n* (16 ビットの場合は 12*n*) バイト長以上の事前割り振りストリング (*n* =セッション数)。	
長さ	12*n* バイト	16*n* バイト
PS 位置	該当なし	

注: データ・ストリング長がセッション数に合わない場合、 戻りコードは 2 になります。

戻りパラメーター

この関数は、データ・ストリング、データ・ストリング長、戻りコードを 戻します。

データ・ストリング:

戻されるデータ・ストリングの長さは、16*n* バイト (標準インターフェースの場合は 12*n* バイト) です。 ここで、*n* はホスト・セッションの数です。記述子は、データ・ストリングと各セッション・タイプ、および ホスト・セッションの表示スペースのサイズと連結されています。

16 バイト (標準インターフェースの場合は 12 バイト) の各セッションの 記述子の形式は次のとおりです。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2–9	5–12	セッションの正式名 (プロファイル名と同じ。またはプロファイルが設定されていない場合は、短縮名と同じ)

バイト		定義
10	13	接続タイプ H=host
	14	予約済み
11–12	15 ~ 16	ホスト表示スペースのサイズ (表示形式ではなく、2 進数)。セッション・タイプが印刷セッションの場合、この値は 0 です。

長さ:

開始されるホスト・セッションの数。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Query Sessions** 関数は正常終了しました。
2	ストリング長が正しくありません。
9	システム・エラーが発生しました。

使用上の注意

1. アプリケーション・プログラムが RC=2 または RC=0 を受信した場合、 活動セッションの数はデータ・ストリング長のフィールドに戻されます。この数を使用して、アプリケーション・プログラムは必要な 最小ストリング長を判別することができます。
2. **Query Sessions** 関数は 、CFGSIZE/NOCFGZISE セッション・オプション (詳細は16 (ページ 152)を参照) および EXTEND_PS/NOEXTEND_PS オプション (詳細は20 (ページ 153)を参照) の影響を受けます。

注:

1. 5250 セッションに対して **Set Session Parameters**(9) に NOCFGSIZE が設定された場合、**Query Sessions**(10) から戻されるバイト位置 11 および 12 の表示スペースのサイズの値は、EXTEND_PS と NOEXTEND_PS のどちらを選択するかによって異なります。
2. EXTEND_PS が **Set Session Parameters**(9) で設定されたときは、**Query Sessions**(10) から戻される表示スペースのサイズに、メッセージ行がある場合にはそのサイズが含まれます。
3. NOEXTEND_PS が設定された場合には、メッセージ行の有無に かかわらず、メッセージ行のサイズは含まれません。25 行 80 桁の表示スペースの場合、このサイズは 1920 また は 2000 になります。

Query System (20)

3270	*5250*	*VT*
はい	はい	はい

EHLLAPI アプリケーション・プログラムでは、**Query System** 関数を使用して、Z and I Emulator for Windows のサポート・レベルおよびシステム関連のその他の値を判別することができます。この関数は該当するシステム・データを含むストリングを戻します。この情報の多くは、戻りコード 9 (システム・エラーが発生した) を受け取った後で IBM サポート・センターに連絡したときにサービス・コーディネーターによって使用される情報です。

この戻りストリングのバイトは戻りパラメーター (ページ 117)で定義されています。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	20 でなければなりません	
データ・ストリング	事前に割り振られた 35 バイトのストリング	36 バイト
長さ	35 でなければなりません	36 でなければなりません
PS 位置	該当なし	

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

35 バイト (16 ビットの場合) または 36 バイト (32 ビットの場合) のデータ・ストリングが戻されます。バイトは以下のように定義されます。

バイト		**定義**
標準	拡張	
1	1	EHLLAPI バージョン番号
2–3	2–3	EHLLAPI レベル番号
4–9	4–9	予約済み
10–12	10–12	予約済み
13	13	ハードウェア・ベース、 U=判別不能
14	14	プログラム・タイプ。ここで P=HCL Z and I Emulator for Windows
15 ~ 16	15 ~ 16	予約済み
17 ~ 18	17 ~ 18	2 バイトの ASCII 値によるZ and I Emulator for Windowsのバージョン/レベル
19	19	予約済み
20 ~ 23	20 ~ 23	予約済み
24 ~ 27	24 ~ 27	予約済み

バイト		定義
28 ~ 29	28 ~ 29	予約済み
	30	予約済み
30 ~ 31	31 ~ 32	2 バイトの 2 進数で表される国別コード
33 ~ 35	34 ~ 36	予約済み

戻りコード

以下のコードが定義されています。

戻りコード	説明
0	**Query System** 関数は正常終了し、 データ・ストリングが戻されました。
1	EHLLAPI がロードされていません。(PC/3270 のみ)
2	ストリング長が正しくありません。(PC/3270 のみ)
9	システム・エラーが発生しました。

使用上の注意

この関数を使用するには、戻りデータ・ストリング・パラメーター を受け入れるメモリーを事前に割り振ります。詳しくは、メモリー割り振り (ページ 14)を参照してください。

Query Window Coordinates (103)

3270	*5250*	*VT*
はい	はい	はい

Query Window Coordinates 関数は、ホスト表示スペースのウィンドウに対する座標を要求します。ウィンドウは座標レベルで戻されます。

注: (0,0) はウィンドウの最上部左方を示しています。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	103 でなければなりません	
データ・ストリング	ホスト表示スペースの 1 文字の短縮セッション ID。	
長さ	17 が暗黙指定されます	20 が暗黙指定されます
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮名 (PSID) • 現在接続されている表示スペースに対する 関数呼び出しを示す、ブランクまたは NULL 文字
	2–4	予約済み
2-17	5–20	予約済み

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

バイト		定義
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮セッション ID • 現在接続されている表示スペースに対する 関数呼び出しを示す、ブランクまたは NULL 文字
	2–4	予約済み
2–17	5–20	以下を戻す 32 ビット符号なし整数。
2–5	5–8	XLeft デスクトップ・ウィンドウに対する長方形ウィンドウの左部 X 座標の 32 ビット符号なし整数 (単位: ペル)
6–9	9–12	YBottom デスクトップ・ウィンドウに対する長方形ウィンドウの下部 Y 座標の 32 ビット符号なし整数 (単位: ペル)
10–13	13–15	XRight デスクトップ・ウィンドウに対する長方形ウィンドウの右部 X 座標の 32 ビット符号なし整数 (単位: ペル)
14–17	16–20	YTop デスクトップ・ウィンドウに対する長方形ウィンドウの上部 Y 座標の 32 ビット符号なし整数 (単位: ペル)

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Query Window Coordinates** 関数は正常終了しました。
1	ユーザー・プログラムが現在 ホスト・セッションに接続されていません。
9	システム・エラーが発生しました。
12	セッションが停止しました。

Read Structured Fields (126)

3270	*5250*	*VT*
はい	いいえ	いいえ

Read Structured Fields 関数を使用して、アプリケーションはホスト・アプリケーションから構造化フィールド・データを読み取ります。呼び出しで S (同期) を指定した場合、**Read Structured Fields** が完了するまで、アプリケーションに制御が戻されます。呼び出しで A (非同期) を指定した場合、 その呼び出しの直後に、アプリケーションに制御が戻されます。呼び出しで M (非同期メッセージ・モード) を 指定した場合、その呼び出しの直後に、アプリケーションに制御が戻されます。アプリケーションは、このメッセージを待ちます。いずれの場合 (S、A、または M) でも、 アプリケーションにより、ホストからのデータが入るバッファー・アドレスが 設定されます。

この関数が非同期に正常終了するためには、以下のステートメントを適用します。

パラメーター・リストの戻りコード・フィールドには、要求された入出力 の結果が含まれていない場合もあります。 戻りコードが 0 でない場合、要求は失敗しています。アプリケーションでは、戻りコードに応じて、適切な処置を 行わなければなりません。

この要求に対する戻りコードが 0 である場合、アプリケーションで、この関数呼び出しとともに戻される要求 ID を使用して、**Get Request Completion** 関数呼び出しを発行し、その要求 ID に関連する関数が終了したかどうかを判別します。**Get Request Completion** 関数呼び出しによって、 以下の情報が戻されます。

1. 関数要求 ID
2. 非同期要求からのデータ・ストリング・アドレス
3. データ・ストリングの長さ
4. 完了した関数の戻りコード

前提呼び出し

Connect for Structured Fields (120) および **Allocate Communication Buffer** (123)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	126 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	8、10、14 のいずれかにしてください。	20
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
2	5	S、A、M のいずれか **S =** 同期。制御は、読み取りが満たされるまでアプリケーションに戻りません。 **A =** 非同期。制御は、アプリケーション (イベント・オブジェクト待機可能) に即時に 戻されます。 **M =** 非同期。制御は、アプリケーション (メッセージ待機可能) に即時に戻されます。
	6	予約済み。
3–4	7–8	2 バイトの Destination/Origin ID
5–8	9–12	データが読み取られる 4 バイトのバッファー・アドレス。**Allocate Communications Buffer** (123) 関数を使用してバッファーを取得する必要があります。
9–10	13–16	予約済み。
11–12	17–20	バイト位置 2 に M を指定した場合、 メッセージを受け取るウィンドウのウィンドウ・ハンドルを設定してください。メッセージは、RegisterWindowMessage (「PCSHLL」) の戻り値 (0 以外) です。
13–14		この位置にあるデータは EHLLAPI で無視されます。ただし、移行プログラムでこの位置にデータを指定していても、 エラーは発生しません。このデータはアプリケーションの移行時に互換性を保つために設けられています。

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

A (非同期) を位置 5 (標準インターフェースの場合は 位置 2) に指定して、関数が正常終了した場合、以下の データ・ストリングが戻されます。

バイト		定義
標準	拡張	

バイト		定義
9–10	13–14	2 バイトの関数要求 ID。この関数要求 ID は、この関数呼び出しの完了を判別するために **Get Request Completion** (125) 関数で使用します。
	15 ~ 16	予約済み。
	17–20	EHLLAPI によってイベント・オブジェクト・アドレスが戻される 4 バイト値。アプリケーションは、このイベント・オブジェクトを待ちます。イベント・オブジェクトがクリアされたときに、アプリケーションは **Get Request Completion** (125) 関数呼び出しを発行しなければなりません (32 ビットの場合のみ)。

注: イベント・オブジェクト・アドレスは、非同期要求が正常終了する たびに戻されます。イベント・オブジェクトは再使用しないでください。要求ごとに新しいイベント・オブジェクトが戻され、 そのイベント・オブジェクトはその要求の間だけ有効です。

データ・ストリング:

M (非同期メッセージ・モード) をバイト位置 5 (16 ビットのアプリケーションの場合は 2) に指定して、 関数が正常終了した場合、以下のデータ・ストリングが戻されます。

バイト		定義
9–10	13–14	2 バイトの関数要求 ID。この関数要求 ID は、この関数呼び出しの完了を判別するために **Get Request Completion** (125) 関数で使用します。
	15 ~ 16	予約済み。
11–12	17 ~ 18	非同期メッセージ・モードのタスク ID。
	19 ~ 20	予約済み。

注: 関数が正常終了した場合、アプリケーション・ウィンドウは メッセージを受け取ります。メッセージは、RegisterWindowMessage(PCSHLL) の戻り値です。wParam パラメーターには、関数呼び出しによって戻されたタスク ID が入ります。lParam パラメーターの HIWORD には戻りコード 0 (関数の正常終了) が 、LOWORD には関数番号 126 が入ります。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Read Structured Fields** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
11	リソースが使用できません。(メモリーが使用できません。)

戻りコード	説明
35	要求はリジェクトされました。ホストからのアウトバウンド送信が取り消されました。
36	要求はリジェクトされました。ホストと接続できません。
37	関数は正常終了しましたが、ホストでのインバウンド送信が使用できません。

使用上の注意

1. 戻りコード 35。これは、ホストからのアウトバウンド送信が取り消された後に **Read Structured Fields** または **Write Structured Fields** が要求された場合に戻されます。アプリケーションで修正処置を行う必要があります。
2. 戻りコード 36。この場合、アプリケーションを エミュレーション・プログラムから切断してから再接続して、ホストとの通信を 再確立する必要があります。アプリケーションで修正処置を行う必要があります。
3. 戻りコード 37。これはホスト・インバウンドが使用不可の場合に 戻されます。**Read Structured Fields** 関数は正常に要求されました。
4. EHLLAPI では、アプリケーションごとに 20 までの非同期要求を未処理に することができます。20 を超える非同期要求が試行されると、リソースが使用不可であることを 示す戻りコード (RC=11) が戻されます。

構造化フィールド・データには、ホストからのアプリケーション構造化フィールドが含まれています。構造化フィールド・データがアプリケーションに達する前に、 EHLLAPI が構造化フィールド・ヘッダーを除去します。

構造化フィールド・データの形式は以下のとおりです。

オフセット:	長さ	目次
0	1 ワード	X'0000'。
2	1 ワード	m (メッセージの長さ。すなわち、メッセージ中のデータのバイト数。この数は 8 バイトを含むメッセージ・ヘッダーの接頭部を除いたものです)。EHLLAPI がこの値を戻します。
4	1 ワード	n (バッファー・サイズ。すなわち、8 バイトのメッセージ・ヘッダーを除くデータ・バッファーの長さを示します)。この値は、アプリケーションが設定する必要があります。
6	1 ワード	X'C000'。
8	8 バイト	先頭の (または唯一の) 構造化フィールド・メッセージの長さ。
10	1 バイト	構造化フィールド・メッセージの長さ以外の最初のバイト。
		⋮
m+7	1 バイト	構造化フィールド・メッセージの末尾のバイト。

バイト 0 から 7 は、バッファー・ヘッダーです。最初のこれらの 8 バイトは、エミュレーション・プログラムで使用されます。バッファーのユーザー・セクションは、オフセット 8 で開始されます。バイト 8 および 9 は、最初の構造化フィールド (構造化フィールド・メッセージには複数の構造化フィールドを含むことができます) 内のバイト数が入っています (バイト 8 および 9 に対する 2 バイトなど)。 バイト 8 から *m* +7 は、構造化フィールド・メッセージがホスト (複数の構造化フィールドを入れることができます) から戻される場所です。

使用中のアプリケーションは、オフセット 0 のワードを 0 に設定して、 完全なバッファーを提供しなければなりません。バッファー長は、オフセット 4 のワードになければなりません。 オフセット 6 のワードは X'C000' にする必要があります。エミュレーション・プログラムは、データ・メッセージをオフセット 8 から入れ、 メッセージの長さをオフセット 2 のワードに入れます。 バッファー長は EHLLAPI により妨害されません。

同期要求

Read Structured Fields が同期要求 (データ・ストリング中の S オプション) された際には、制御は要求が満たされた後に限りアプリケーションに戻されます。アプリケーションでは以下のことを想定できます。

- 戻りコードが正しい。
- 通信バッファー (読み取りバッファー) 内のデータが正しい。
- ホストが今後 **Read Structured Fields** 要求を処理しない。

非同期要求

Read Structured Fields が非同期要求 (データ・ストリング中の A オプション) された際には、アプリケーションでは以下のことを想定できません。

- 戻りコードが正しい。
- 通信バッファー (読み取りバッファー) 内のデータが正しい。
- ホストが今後 **Read Structured Fields** 要求を処理しない。

非同期要求がなされた際には、EHLLAPI は以下の値を戻します。

- 16 ビット要求 ID (データ・ストリングのバイト位置 13 ~ 14) (標準インターフェースの場合は 9 ~ 10)
- イベント・オブジェクトのアドレス (データ・ストリングのバイト位置 17 ~ 20)

これらは、非同期の **Read Structured Fields** 呼び出しを完了させるために使用します。

非同期の **Read Structured Fields** 関数呼び出しの結果を判別するためには、次のステップを行わなければなりません。

- EHLLAPI 戻りコードが 0 でない場合、要求は失敗しています。したがって、非同期要求は行われていません。アプリケーションでは、再び呼び出しを試みる前に、適切な処置を 行わなければなりません。
- 戻りコードが 0 ならば、アプリケーションは、**Get Request Completion** (125) 関数または **Wait For Single Object** を使用してイベント・オブジェクトが通知状態になるまで待たなければなりません。イベント・オブジェクトは再使用しないでください。イベント・オブジェクトは、**Read Structured Fields** 関数呼び出しから **Get Request Completion** (125) 関数呼び出し完了までの間で有効です。
- イベント・オブジェクトが送信状態になったら、**Get Request Completion** (125) 関数に対する呼び出しで要求 ID パラメーターとして戻された 16 ビットの要求 ID を使用してください。**Get Request Completion** (125) 関数呼び出しから戻されたデータ・ストリングには、**Read Structured Fields** 関数呼び出しの最終的な戻りコードが含まれています。

Read Structured Fields が非同期要求 (データ・ストリング中の M オプション) された際には、アプリケーションでは以下のことを想定できません。

- 戻りコードが正しい。
- 通信バッファー (読み取りバッファー) 内のデータが正しい。
- ホストが今後 **Read Structured Fields** 要求を処理しない。

M オプションを指定して非同期要求がなされた際には、EHLLAPI は以下の値を 戻します。

- 16 ビット要求 ID (データ・ストリングのバイト位置 13 ~ 14) (標準インターフェースの場合は 9 ~ 10)
- 非同期メッセージ・モードのタスク ID (データ・ストリングのバイト位置 17 ~ 18) (標準インターフェースの場合は 11 ~ 12)

これらは、非同期の **Read Structured Fields** 呼び出しを完了させるために使用します。

Receive File (91)

3270	*5250*	*VT*
はい	はい	いいえ

Receive File 関数は、ホスト・セッションからワークステーション・セッションにファイルを転送します。この機能の使用方法は、PC/3270 における RECEIVE コマンドと同様です。**Receive File** 関数は EHLLAPI アプリケーション・プログラムによって呼び出すことができます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	91 でなければなりません。	
データ・ストリング	以下の例を参照してください。	
長さ	データ・ストリングの長さ (バイト数)。EOT モードの場合、このパラメーターは一時変更されます。	

1 バイト文字セット (SBCS) のデータ・ストリングの例を以下に示します。

3270 セッション

- VM/CMS ホスト・システムからファイルを受信する場合

```
pc_filename [id:]fn ft [fm] [(option]
```

- MVS™/TSO ホスト・システムからファイルを受信する場合

```
pc_filename[id:]dataset[(member)] [/password] [option]
```

- CICS® ホスト・システムからファイルを受信する場合

```
pc_filename ［id:］host_filename ［(option］
```

5250 セッション

- iSeries™、eServer™ i5 または System i5™ ホスト・システムからファイルを受信する場合

```
pc_filename ［id:］library file member ［option］
```

戻りパラメーター

戻りコード	説明
2	パラメーター・エラーが発生したか、または EHLLAPI バッファーに対して 長すぎる (255 バイトを超過) データ・ストリング長が指定されました。ファイル転送が失敗しました。
3	ファイル転送が完了しました。
4	ファイル転送は完了し、レコードは分割されました。
9	システム・エラーが発生しました。
27	Cancel ボタンまたはタイムアウト (**Set Session Parameter** (9) 関数により設定) によってファイル転送は終了しました。
101	ファイル転送 (CICS® への/からの転送) は正常に終了しました。

システム内、またはユーザーのデータ・ストリングの指定方法に問題がある場合 には、戻りコード 2 または 9 が戻されます。

他の戻りコードも受信される場合があります。これらの戻りコードは、 ホスト転送プログラムにより生成されるメッセージ番号に関連しています。CICS® ホスト転送プログラムに転送する場合、その戻りコードから 100 を減算するとメッセージの数値部分を得ることができます。例えば、戻りコードが 101 の場合には、 ホストによりメッセージ番号 INW0001 が出されたことを示します。他のホスト転送プログラムの場合には、 そのメッセージの数字部分のみ使用してください。例えば、戻りコードが 34 の場合には、 ホストによりメッセージ番号 TRANS34 が 出されたことを示します。特定のメッセージの意味については、 ご使用のホスト転送プログラムの資料を参照してください。

EHLLAPI により報告されたオペレーティング・システムのエラー・コード は、300 より大きい数字です。エラー・コードを判別するには、300 を引いてから、オペレーティング・システム の資料でその戻りコードを調べてください。

使用上の注意

1. **Set Session Parameters** (9) 関数での 4 つのパラメーター・セットがこの関数に関連しています。それらは STRLEN/STREOT、EOT=c、QUIET/NOQUIET および TIMEOUT=c/TIMEOUT=0 の各セッション・オプションです。詳細については、1 (ページ 147) と 2 (ページ 147)、7 (ページ 149) と 8 (ページ 149) の項目を参照してください。
2. **Receive File** 関数を実行時にパスを指定しない場合、受信されたファイルは現行のサブディレクトリーに保管されます。このサブディレクトリーは、ユーザーのアプリケーションが実行されているディレクトリーです。

Release (12)

3270	*5250*	*VT*
はい	はい	はい

Release 関数は、**Reserve** (11) 関数により予約されている表示スペースに関連するキーボードをアンロックします。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	12 でなければなりません	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**説明**
0	**Release** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
9	システム・エラーが発生しました。

使用上の注意

Reserve (11) 関数を使用して予約されているホストの表示スペースに対して **Release** 関数を使用しないと、**Reset System** (21) 関数または **Disconnect Presentation Space** (2) 関数を呼び出すか、EHLLAPI アプリケーション・プログラムを終了させるまで、ユーザーはセッションからロックアウトされます。

Reserve (11)

3270	*5250*	*VT*
はい	はい	はい

Reserve 関数は、端末オペレーターからのブロック入力に対するホスト接続表示スペースに関連するキーボードをロックします。

予約済みのホスト表示スペースは、次のいずれかが 発生するまでロック状態を維持します。

- 新しいセッションに対する **Connect** (1) 関数の実行。
- **Disconnect Presentation Space** (2) 関数の実行。
- **Release** (12) 関数の実行。
- **Reset System** (21) 関数の実行。
- **Start Keystroke Intercept** (50) 関数の実行
- EHLLAPI アプリケーション・プログラムの終了

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	11 でなければなりません	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**説明**
0	**Reserve** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
5	表示スペースの使用が禁止されています。
9	システム・エラーが発生しました。

使用上の注意

1. EHLLAPI アプリケーション・プログラムで一連のトランザクションを ホストへ送信するとき、アプリケーションの処理が終了するまで、その セッションに対するアクセスを阻止する必要があります。
2. この関数でキーボードがロックされている間にユーザーが行った キーボード入力はキューに入り、セッション終了後に処理されます。
3. この関数は、マウス入力とキーボード入力の両方をロックします。マウス入力ま たはキーボード入力を使用可能にするためには、アプリケーション・プログラム で表示スペースをアンロックする必要があります。

Reset System (21)

3270	*5250*	*VT*
はい	はい	はい

Reset System 関数は、EHLLAPI を開始状態に初期化し直します。セッションのパラメーター・オプションはデフォルト値にリセットされます。イベント通知は停止します。予約済みのホスト・セッションは解放されます。ホスト表示スペースは切断されます。キー・ストロークの代行受信は使用不可になります。

Reset System 関数は、初期化時またはプログラム終了時にシステムの既知の初期化状態にリセットするために使用できます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	21 でなければなりません	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**定義**
0	**Reset System** 関数は正常終了されました。
1	EHLLAPI がロードされていません。
9	システム・エラーが発生しました。

使用上の注意

PC/3270 では、EHLLAPI がロードされているかどうかを確認するために、 この関数を使用できます。アプリケーション開始時にこの関数を呼び出し、戻りコードが 1 かどうか を確認してください。

Search Field (30)

3270	*5250*	*VT*
はい	はい	はい

Search Field 関数は、 指定されたストリングの存在について、接続されたホストの表示スペースのフィールドを検索します。ターゲット・ストリングが検出された場合、この関数は ホスト表示スペースの先頭から数えたストリングの位置 を 10 進数で戻します。(例えば、24 行 80 桁の表示スペースでは、 行 1 桁 1 の位置は番号 *1* になり、 行 5 桁 1 の位置は番号 *321* になります。)

この関数は、 フィールド形式のホスト表示スペース内のフィールドに限り、 保護フィールドまたは無保護フィールドを検索するために使用することができます。

注: フィールドがホスト表示スペースの終わりで 折り返している場合、表示スペースの終わりに到達したときに折り返します。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	30 でなければなりません。	
データ・ストリング	検索するデータ・ストリング。	
長さ	該当データ・ストリングの長さ。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	該当フィールドを識別します。`SRCHALL` に対しては、これはコピー先フィールド内のどのバイトの PS 位置でも構いません。`SRCHFROM` に対しては、これは `SRCHFRWD` の検索の開始位置または `SRCHBKWD` の検索の終了位置を示します。3 (ページ 131) の使用上の注意を参照してください。	

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

以下のコードが定義されています。

長さ	説明
= 0	ストリングが検出されませんでした。
> 0	示されたホスト表示スペースの位置でストリングが 検出されました。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Search Field** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター・エラーです。ストリング長が 0 か、または EOT モードが指定されましたが、 呼び出しデータ・ストリング内に EOT 文字がありません。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	検索ストリングが検出されなかったか、 またはホストの表示スペースが定様式化されていません。

使用上の注意

1. **Set Session Parameters** (9) 関数での 4 つのパラメーター・セットがこの関数に関連しています。それらは SRCHALL/SRCHFROM、STRLEN/STREOT、 SRCHFRWD/SRCHBKWD、および EOT=c の各セッション・オプションです。詳細については、1 (ページ 147) ~ 4 (ページ 148) の項目を参照してください。
2. **Set Session Parameters** (9) 関数を使用して、 ユーザーは検索をフィールドの先頭から終わりに向かって実行するのか (SRCHFRWD) またはその逆方向に実行するのか (SRCHBKWD) を決定することができます。
3. **Search Field** 関数は通常フィールド内全体をチェックします (SRCHALL がデフォルト・モードです)。しかし、機能 9 を使用して SRCHFROM を指定することができます。このモードでは、呼び出し PS 位置パラメーターは、該当フィールド の識別以外のことも実行します。つまり、検索の開始点または終了点も指定します。
    - SRCHFRWD オプションが有効な場合、 指示されたストリングの検索を、指定された PS 位置から開始し、 フィールドの終わりに向かって実行します。
    - SRCHBKWD オプションが有効な場合、 指示されたストリングの検索を、フィールドの終わりから開始し、 指定された PS 位置に向かって逆方向に実行します。該当ストリングが検出されない時には、検索は呼び出し PS 位置パラメーターで 指定された PS 位置で停止します。

注: 5250 エミュレーションは、24 行 80 桁の表示スペースをサポートします。Communication Manager 5250 エミュレーションが 25 番目の行を表示するインスタンスもあります。このようなことが起こるのは、ホストからのエラー・メッセージが表示される場合、またはオペレーターが SysReq キーを選択する場合です。 Z and I Emulator for Windows行 24 に 25 行目の情報を表示しますが、通常は実際の 24 行目を表示しま

す。ステータス・バーに 情報が表示される場合は、ステータス・バーがその情報用に構成されている必要があります。ステータス・バーの構成については、「*Quick Beginnings*」を参照してください。**EXTEND_PS** オプションによって、EHLLAPI アプリケーションは Communication Manager EHLLAPI と同じインターフェースを使用することができ、このような状況が発生したときに有効な表示スペースが拡張されます。

Search Presentation Space (6)

3270	*5250*	*VT*
はい	はい	はい

Search Presentation Space 関数により、EHLLAPI プログラムは指定されたストリングの存在についてホストの表示スペースをチェックします。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	6 でなければなりません。	
データ・ストリング	検索するデータ・ストリング。	
長さ	該当データ・ストリングの長さ。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	ホスト・プレゼンテーション内の、検索を開始する (SRCHFRWD オプション) 位置、または検索を終了する (SRCHBKWD オプション) 位置。SRCHALL (デフォルト) モードの場合、 このパラメーターは一時変更されます。	

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

以下のコードが定義されています。

長さ	**説明**
= 0	ストリングが検出されませんでした。
> 0	示されたホスト表示スペースの位置でストリングが 検出されました。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Search Presentation Space** 関数は正常終了しました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター指定でエラーが発生しました。
7	ホスト表示スペースの位置が正しくありません。
9	システム・エラーが発生しました。
24	検索ストリングが検出されませんでした。

使用上の注意

1. **Set Session Parameters** (9) 関数での 4 つのパラメーター・セットがこの関数に関連しています。それらは SRCHALL/SRCHFROM、STRLEN/STREOT、 SRCHFRWD/SRCHBKWD、および EOT=c の各セッション・オプションです。詳細については、1 (ページ 147) ~ 4 (ページ 148) の項目を参照してください。 ~
2. **Set Session Parameters** (9) 関数を使用して、SRCHBKWD を指定することができます。このオプションが有効である場合、検索操作はストリングの最後の出現位置を検出します。
3. **Search Presentation Space** 関数は、通常、全ホスト表示スペースをチェックします。しかし、**Set Session Parameters** (9) 関数を使用して SRCHFROM を指定することができます。このモードのとき、PS 位置パラメーターは検索の開始位置または検索の 終了位置を指定します。
   - SRCHFRWD オプションが有効な場合、指示されたストリング の検索を、指定された PS 位置から開始し、ホスト表示スペースの終わりに向かって実行します。
   - SRCHBKWD オプションが有効な場合、指示されたストリングの 検索を、表示スペースの終わりから開始し、指定された PS 位置 に向かって逆方向に実行します。該当ストリングが検出されない時には、検索は呼び出し PS 位置パラメーターで 指定された PS 位置で停止します。
4. ホスト表示スペース内に複数回出現する可能性のある キーワードを検索する場合には SRCHFROM オプションも使用できます。
5. **Search Presentation Space** 関数は、ホスト表示スペースが使用可能かどうかの判別にも使用できます。ユーザーの EHLLAPI アプリケーション・プログラムがデータ送信の前に特定のプロンプトまたはメッセージを要求している場合、**Search Presentation Space** 関数を使用して、 処理続行の前にプロンプトまたはメッセージをチェックすることができます。

注: 5250 エミュレーションは、24 行 80 桁の表示スペースをサポートします。Communication Manager 5250 エミュレーションが 25 番目の行を表示するインスタンスもあります。このようなことが起こるのは、ホストからのエラー・メッセージが表示される場合、またはオペレーターが SysReq キーを選択する場合です。 Z and I Emulator for Windows行 24 に 25 行目の情報を表示しますが、通常は実際の 24 行目を表示します。ステータス・バーに 情報が表示される場合は、ステータス・バーがその情報用に構成されている必要があります。ステータス・バーの構成については、「*Quick Beginnings*」を参照してください。**EXTEND_PS** オ

プションによって、EHLLAPI アプリケーションは Communication Manager EHLLAPI と同じインターフェースを使用することができ、このような状況が発生したときに有効な表示スペースが拡張されます。

Send File (90)

3270	*5250*	*VT*
はい	はい	いいえ

Send File 関数は、EHLLAPI を実行しているワークステーションのセッションからホストのセッションへファイルを転送するときに使用します。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	90 でなければなりません。	
データ・ストリング	以下の例を参照してください。	
長さ	該当データ・ストリングの長さ。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	必ず 0 にしてください。	

SBCS のデータ・ストリングの例を以下に示します。

3270 セッション

- VM/CMS ホスト・システムへファイルを送信する場合:

  pc_filename [*id:*]*fn ft* [*fm*] [(*option*]

- MVS/TSO ホスト・システムへファイルを送信する場合:

  pc_filename [*id:*]*dataset*[(*member*)] [/*password*] [*option*]

- CICS ホスト・システムへファイルを送信する場合:

  pc_filename [*id:*]*host_filename* [(*option*]

5250 セッション

- iSeries™、eServer™ i5、または System i5™ ホスト・システムへファイルを送信する場合:

  pc_filename [*id:*]*library file member* [*option*]

戻りパラメーター

戻りコード	説明
2	パラメーター・エラーが発生したか、または EHLLAPI バッファーに対して 長すぎる (255 バイトを超過) データ・ストリング長が指定されました。ファイル転送が失敗しました。
3	ファイル転送が完了しました。
4	ファイル転送は完了し、レコードは分割されました。
5	ワークステーション・ファイル名が有効でなかったか、見つかりませんでした。ファイル転送が取り消されました。
9	システム・エラーが発生しました。
27	Cancel ボタンまたはタイムアウト (**Set Session Parameter** (9) 関数により設定) によってファイル転送は終了しました。
101	ファイル転送 (CICS への/CICS からの転送) は正常に終了しました。

システム内、またはユーザーのデータ・ストリングの指定方法に問題がある場合 には、戻りコード 2 または 9 が戻されます。

他の戻りコードも受信される場合があります。これらの戻りコードは、 ホスト転送プログラムにより生成されるメッセージ番号に関連しています。CICS ホスト転送プログラムに転送する場合、 その戻りコードから 100 を減算するとメッセージの数値部分を得ることができます。例えば、戻りコードが 101 の場合には、 ホストによりメッセージ番号 INW0001 が出されたことを示します。他のホスト転送プログラムの場合には、 そのメッセージの数字部分のみ使用してください。例えば、戻りコードが 34 の場合には、 ホストによりメッセージ番号 TRANS34 が 出されたことを示します。特定のメッセージの意味については、 ご使用のホスト転送プログラムの資料を参照してください。

EHLLAPI により報告されたオペレーティング・システムのエラー・コード は、300 より大きい数字です。エラー・コードを判別するには、300 を引いてから、オペレーティング・システム の資料でその戻りコードを調べてください。

使用上の注意

1. **Set Session Parameters** (9) 関数での 4 つのパラメーター・セットがこの関数に関連しています。それらは QUIET/NOQUIET、STRLEN/STREOT、 TIMEOUT=c/TIMEOUT=0、 および EOT=c の各セッション・オプションです。詳細については、項目 1 (ページ 147) および 2 (ページ 147)、項目 7 (ページ 149) を参照してください。8 (ページ 149) を参照してください。

Send Key (3)

3270	*5250*	*VT*
はい	はい	はい

Send Key 関数は、1 つのキー・ストロークまたはキー・ストロークのストリングをホスト表示スペースへ送信する際に使用します。

呼び出しデータ・ストリング・パラメーターで、送信する キー・ストロークのストリングを定義します。キー・ストロークは、端末オペレーターが入力したかのように、 宛先セッションには見えます。Enter などのアテンション ID (AID) キーもすべて送信できます。入力保護されているホスト・フィールドや数字専用のホスト・フィールドはすべて、 それに対応した取り扱いが必要です。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	3 でなければなりません。	
データ・ストリング	キー・ストロークのストリング。最大 255 バイト。英大文字と英小文字の ASCII 文字は、そのまま表されます。ファンクション・キーおよびシフト・ファンクション・キーは、略号によって表されます。『キーボードの略号 (ページ 138)』を参照してください。	
長さ	送信元データ・ストリングの長さ。EOT モードの場合、このパラメーターは一時変更されます。	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**説明**
0	キー・ストロークは送信され、状況は正常です。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
2	無効なパラメーターが EHLLAPI に渡されました。
4	ホスト・セッションは使用中であったため、どのキー・ストロークも 送信できませんでした。
5	送信先セッションへの入力が禁止または拒否されたため、キー・ストローク全部を 送信することはできませんでした。
9	システム・エラーが発生しました。

使用上の注意

1. **Set Session Parameters** (9) 関数のパラメーターがこの関数に関連しています。それらは AUTORESET/NORESET、 STRLEN/STREOT、EOT=c、ESC=c、および RETRY/NORETRY の各セッション・オプションです。詳細については、1 (ページ 147) と 2 (ページ 147)、9 (ページ 150) と 10 (ページ 150)、および 19 (ページ 153) の項目を参照してください。

2. キーボードがロックされていたり、使用中の時は、キー・ストロークを ホスト・セッションへ送信することはできません。**Wait** (4) 関数でキーボードの状態をチェックすることができます。
3. ホストが使用中の場合、入力が拒否されることがあります。
4. データ・ストリングの長さは、デフォルトのデータ・ストリング長パラメーターを使用して明示的に定義する必要があります。しかし 、**Set Session Parameters** (9) 関数の EOT=c オプションで暗黙的に定義することもできます。

   データ・ストリング長を明示的に定義する場合 (1 を参照)、 アプリケーションが渡すデータ・ストリング長パラメーターの値を 計算しなければなりません。この計算では、`@E` などの複合キー・ストロークは 2 バイト、 `@A@C` などの複合キー・ストロークは 4 バイトとします。

5. 特殊制御キーを送信するには、複合文字によるコーディング法を使用します。このコーディング法では、1 つのキー・ストロークを示すために 2 から 4 文字 の ASCII 文字が使用されています。1 番目と 3 番目の文字は常にエスケープ文字です。2 番目と 4 番目の文字は常にキー・コードです。

   シーケンス `LOGON ABCDE` の後に Enter (改行) キーを押す場合は、`LOGON ABCDE@E` とコーディングします。これらのキーコードの完全なリストを キーボードの略号 (ページ 138) に示します。

   この複合コーディング法により、複雑な 16 進キー・コードを使用せずに必要な すべてのキー・ストローク・コードの ASCII ストリング表現が可能になります。

   デフォルトのエスケープ文字は `@` です。エスケープ文字の値は、**Set Session Parameters** (9) 関数の ESC=c オプションを使用して、別の文字に変更することができます。

6. パフォーマンスを向上させる必要があるユーザーは、**Send Key** (3) 関数でストロークを送信するよりも、**Copy String to Field** (33) または **Copy String to Presentation Space** (15) 関数を使用してください。ただし、**Send Key** (3) 関数だけが特殊制御キーを送信することができます。
7. この関数のパフォーマンスを向上させるためには、 **Set Session Parameters (9) (ページ 145)** の セッション・オプション 10 (ページ 150) (NORESET オプション) を参照してください。

   NORESET を指定しない場合には、リセットの略号が キー・ストローク・ストリングに接頭部として追加されます。したがって、入力禁止以外のリセット可能な状況はすべてリセットされます。

   NORESET オプションは **Reset System** (21) 関数と同じではありません。

8. AID キーを含むキー・ストローク・ストリングは複数のパスを介してホストに 送信されます。それぞれのパスは、最初の AID キー前までの (またはそれを含む) ストリングを送信します。EHLLAPI は、ストリングの長さと各パスの開始位置を調整します。ホストのアプリケーション・プログラムによっては、 任意のキー・ストロークが AID キー処理のために失われることがあります。したがって、複数の AID キーを含むキー・ストロークのリストを送信することは お勧めできません。
9. `@P` (印刷) または `@A@T` (表示スペースの印刷) 処理中、その表示スペースを更新する要求はすべて拒否されます。印刷要求の最中に表示スペースが使用中状態だったり、割り込み要求が発生した場合には、略号 @A@R (装置リセット - 表示スペースの印刷の取り消し) が要求を取り消し、状況をリセットします。

キーボードの略号

キーボードの略号はワークステーションのキーボードの特殊ファンクション・キーを 表す ASCII 文字を提供します。この省略形コードにより、これらの特殊キーが覚えやすくなります。最もよく使用されるキーは英字キー・コードです。例えば、**Clear** キーは *C*、**Tab** キーは *T* などです。

表 7 : 英大文字による略号 (ページ 138) は英大文字を使用する略号の一覧です。

表 7. 英大文字による略号

ニーモニック	意味	3270	5250	VT
@B	左タブ	はい	はい	いいえ
@C	クリア	はい	はい	いいえ
@D	削除	はい	はい	いいえ
@E	Enter	はい	はい	いいえ
@F	EOF 消去	はい	はい	いいえ
@H	ヘルプ\n	いいえ	はい	いいえ
@I	挿入	はい	はい	いいえ
@J	ジャンプ (フォーカスを設定)	はい	はい	いいえ
@L	カーソル左移動	はい	はい	はい
@N	改行	はい	はい	はい
@O	スペース	はい	はい	はい
@P	印刷	はい	はい	はい
@R	リセット	はい	はい	いいえ
@T	右タブ	はい	はい	はい
@U	カーソル上移動	はい	はい	はい
@V	カーソル下移動	はい	はい	はい
@Z	カーソル右移動	はい	はい	はい

表 8 : 数字または英小文字による略号 (ページ 138) は数字または英小文字を使用する略号の一覧です。

表 8. 数字または英小文字による略号

ニーモニック	意味	3270	5250	VT
@0	ホーム	はい	はい	いいえ
@1	PF1/F1	はい	はい	いいえ
@2	PF2/F2	はい	はい	いいえ
@3	PF3/F3	はい	はい	いいえ
@4	PF4/F4	はい	はい	いいえ
@5	PF5/F5	はい	はい	いいえ
@6	PF6/F6	はい	はい	はい
@7	PF7/F7	はい	はい	はい

表 8. 数字または英小文字による略号 (続く)

ニーモニック	意味	3270	5250	VT
@8	PF8/F8	はい	はい	はい
@9	PF9/F9	はい	はい	はい
@a	PF10/F10	はい	はい	はい
@b	PF11/F11	はい	はい	はい
@c	PF12/F12	はい	はい	はい
@d	PF13	はい	はい	はい
@e	PF14	はい	はい	はい
@f	PF15	はい	はい	はい
@g	PF16	はい	はい	はい
@h	PF17	はい	はい	はい
@i	PF18	はい	はい	はい
@j	PF19	はい	はい	はい
@k	PF20	はい	はい	はい
@l	PF21	はい	はい	いいえ
@m	PF22	はい	はい	いいえ
@n	PF23	はい	はい	いいえ
@o	PF24	はい	はい	いいえ
@q	終了	はい	はい	いいえ
@u	PageUp	いいえ	はい	いいえ
@v	PageDown	いいえ	はい	いいえ
@x	PA1	はい	はい	いいえ
@y	PA2	はい	はい	いいえ
@z	PA3	はい	はい	いいえ

表 9 : @A と @ 英大文字による略号 (ページ 139) は @A と @ 英大文字 (A ~ Z) キーの組み合わせを使用する略号の一覧です。

表 9. @A と @ 英大文字による略号

ニーモニック	意味	3270	5250	VT
@A@C	テスト	いいえ	はい	いいえ
@A@D	ワード削除	はい	はい	いいえ
@A@E	フィールドの終了	はい	はい	いいえ
@A@F	入力消去	はい	はい	いいえ
@A@H	システム要求	はい	はい	いいえ
@A@I	挿入切り替え	はい	はい	いいえ
@A@J	カーソル選択	はい	はい	いいえ
@A@L	高速カーソル左移動	はい	はい	いいえ
@A@Q	警告	はい	はい	いいえ

表 9. @A と @ 英大文字による略号 (続く)

ニーモニック	意味	3270	5250	VT
@A@R	装置の取り消し (表示スペース印刷取り消し)	はい	はい	いいえ
@A@T	表示スペース印刷	はい	はい	はい
@A@U	高速カーソル上移動	はい	はい	いいえ
@A@V	高速カーソル下移動	はい	はい	いいえ
@A@Z	高速カーソル右移動	はい	はい	いいえ

表 10 : @A と @ 英小文字による略号 (ページ 140) は、@A と @ 数字、または @A と @ 英小文字 (a ~ z) キーの組み合わせを使用する略号の一覧です。

表 10. @A と @ 英小文字による略号

ニーモニック	意味	3270	5250	VT
@A@9	反転表示	はい	はい	いいえ
@A@b	下線	はい	いいえ	いいえ
@A@c	反転表示のリセット	はい	いいえ	いいえ
@A@d	赤	はい	いいえ	いいえ
@A@e	Pink	はい	いいえ	いいえ
@A@f	緑	はい	いいえ	いいえ
@A@g	黄	はい	いいえ	いいえ
@A@h	青	はい	いいえ	いいえ
@A@i	ターコイズ	はい	いいえ	いいえ
@A@j	White	はい	いいえ	いいえ
@A@l	ホスト・カラーのリセット	はい	いいえ	いいえ
@A@t	印刷 (ワークステーション)	はい	はい	いいえ
@A@y	正方向ワード・タブ	はい	はい	いいえ
@A@z	逆方向ワード・タブ	はい	はい	いいえ

表 11 : @A と @ 特殊文字による略号 (ページ 140) は @A と @ 特殊文字の組み合わせを使用する略号の一覧です。

表 11. @A と @ 特殊文字による略号

ニーモニック	意味	3270	5250	VT
@A@-	フィールド -	いいえ	はい	いいえ
@A@+	フィールド +	いいえ	はい	いいえ
@A@<	レコード・バックスペース	いいえ	はい	いいえ

表 12 : @S (シフト)、@W (編集) および @ 英字による略号 (ページ 141) は @S、@W、および @alphabetic の小文字の組み合わせを使用する略号の 一覧です。

表 12. @S (シフト)、@W (編集) および @ 英字による略号

ニーモニック	意味	3270	5250	VT
@S@E	ホストでの表示スペース印刷	いいえ	はい	いいえ
@S@x	重複	はい	はい	いいえ
@S@y	フィールド・マーク	はい	はい	いいえ
@W@C	編集 コピー	はい	はい	はい
@W@D	編集 消去	はい	はい	はい
@W@E	編集 コピー 追加	はい	はい	はい
@W@L	編集 コピー リンク	はい	はい	はい
@W@N	編集 貼り付け 次へ	はい	はい	はい
@W@V	編集 貼り付け	はい	はい	はい
@W@X	編集 切り抜き	はい	はい	はい
@W@Z	編集 取り消し	はい	はい	はい

注: @W 編集の略号は、拡張モードの EHLLAPI 関数でのみサポートされます。EHLLAPI 関数の要約 (ページ 37)の Start Keystroke Intercept 関数を参照してください。

VT の場合のみ: 表 13 : @M、@Q、および @ 英小文字を使用した略号 (VT の場合のみ) (ページ 141)は、 @M と @ 数字または @M と @ 英小文字 (a ~ z) の組み合わせを使用する略号です。

表 13. @M、@Q、および @ 英小文字を使用した略号 (VT の場合のみ)

ニーモニック	意味	3270	5250	VT
@M@0	VT テン・キー 0	いいえ	いいえ	はい
@M@1	VT テン・キー 1	いいえ	いいえ	はい
@M@2	VT テン・キー 2	いいえ	いいえ	はい
@M@3	VT テン・キー 3	いいえ	いいえ	はい
@M@4	VT テン・キー 4	いいえ	いいえ	はい
@M@5	VT テン・キー 5	いいえ	いいえ	はい
@M@6	VT テン・キー 6	いいえ	いいえ	はい
@M@7	VT テン・キー 7	いいえ	いいえ	はい
@M@8	VT テン・キー 8	いいえ	いいえ	はい
@M@9	VT テン・キー 9	いいえ	いいえ	はい
@M@-	VT テン・キー - (負符号)	いいえ	いいえ	はい
@M@,	VT テン・キー , (コンマ)	いいえ	いいえ	はい

表 13. @M、@Q、および @ 英小文字を使用した略号 (VT の場合のみ) (続く)

ニーモニック	意味	3270	5250	VT
@M@.	VT テン・キー . (ピリオド)	いいえ	いいえ	はい
@M@e	VT テン・キー Enter	いいえ	いいえ	はい
@M@f	VT 編集 検索	いいえ	いいえ	はい
@M@i	VT 編集 挿入	いいえ	いいえ	はい
@M@r	VT 編集 除去	いいえ	いいえ	はい
@M@s	VT 編集 選択	いいえ	いいえ	はい
@M@p	VT 編集 前の画面	いいえ	いいえ	はい
@M@n	VT 編集 次の画面	いいえ	いいえ	はい
@M@a	VT PF1	いいえ	いいえ	はい
@M@b	VT PF2	いいえ	いいえ	はい
@M@c	VT PF3	いいえ	いいえ	はい
@M@d	VT PF4	いいえ	いいえ	はい
@M@h	VT 保留画面	いいえ	いいえ	はい
@M@(スペース)	制御コード NUL	いいえ	いいえ	はい
@M@A	制御コード SOH	いいえ	いいえ	はい
@M@B	制御コード STX	いいえ	いいえ	はい
@M@C	制御コード ETX	いいえ	いいえ	はい
@M@D	制御コード EOT	いいえ	いいえ	はい
@M@E	制御コード ENQ	いいえ	いいえ	はい
@M@F	制御コード ACK	いいえ	いいえ	はい
@M@G	制御コード BEL	いいえ	いいえ	はい
@M@H	制御コード BS	いいえ	いいえ	はい
@M@I	制御コード HT	いいえ	いいえ	はい
@M@J	制御コード LF	いいえ	いいえ	はい
@M@K	制御コード VT	いいえ	いいえ	はい
@M@L	制御コード FF	いいえ	いいえ	はい
@M@M	制御コード CR	いいえ	いいえ	はい
@M@N	制御コード SO	いいえ	いいえ	はい
@M@O	制御コード SI	いいえ	いいえ	はい
@M@P	制御コード DLE	いいえ	いいえ	はい
@M@Q	制御コード DC1	いいえ	いいえ	はい
@M@R	制御コード DC2	いいえ	いいえ	はい
@M@S	制御コード DC3	いいえ	いいえ	はい
@M@T	制御コード DC4	いいえ	いいえ	はい
@M@U	制御コード NAK	いいえ	いいえ	はい
@M@V	制御コード SYN	いいえ	いいえ	はい

表 13. @M、@Q、および @ 英小文字を使用した略号 (VT の場合のみ) (続く)

ニーモニック	意味	3270	5250	VT
@M@W	制御コード ETB	いいえ	いいえ	はい
@M@X	制御コード CAN	いいえ	いいえ	はい
@M@Y	制御コード EM	いいえ	いいえ	はい
@M@Z	制御コード SUB	いいえ	いいえ	はい
@M@u	制御コード ESC	いいえ	いいえ	はい
@M@v	制御コード FS	いいえ	いいえ	はい
@M@w	制御コード GS	いいえ	いいえ	はい
@M@x	制御コード RS	いいえ	いいえ	はい
@M@y	制御コード US	いいえ	いいえ	はい
@M@z	制御コード DEL	いいえ	いいえ	はい
@Q@A	VT ユーザー定義キー 6	いいえ	いいえ	はい
@Q@B	VT ユーザー定義キー 7	いいえ	いいえ	はい
@Q@C	VT ユーザー定義キー 8	いいえ	いいえ	はい
@Q@D	VT ユーザー定義キー 9	いいえ	いいえ	はい
@Q@E	VT ユーザー定義キー 10	いいえ	いいえ	はい
@Q@F	VT ユーザー定義キー 11	いいえ	いいえ	はい
@Q@G	VT ユーザー定義キー 12	いいえ	いいえ	はい
@Q@H	VT ユーザー定義キー 13	いいえ	いいえ	はい
@Q@I	VT ユーザー定義キー 14	いいえ	いいえ	はい
@Q@J	VT ユーザー定義キー 15	いいえ	いいえ	はい
@Q@K	VT ユーザー定義キー 16	いいえ	いいえ	はい
@Q@L	VT ユーザー定義キー 17	いいえ	いいえ	はい
@Q@M	VT ユーザー定義キー 18	いいえ	いいえ	はい
@Q@N	VT ユーザー定義キー 19	いいえ	いいえ	はい

表 13. @M、@Q、および @ 英小文字を使用した略号 (VT の場合のみ) (続く)

ニーモニック	意味	3270	5250	VT
@Q@0	VT ユーザー定義キー 20	いいえ	いいえ	はい
@Q@a	VT 後退タブ	いいえ	いいえ	はい
@Q@r	VT ページ消去	いいえ	いいえ	はい
@Q@s	VT 編集	いいえ	いいえ	はい

次の表は特殊文字を使用する略号の一覧です。

表 14. 特殊文字キーによる略号

ニーモニック	意味	3270	5250	VT
@@	@	はい	はい	はい
@$	カーソル切り替え (Presentation Manager®のインターフェースのみ)	はい	はい	はい
@<	Backspace	はい	はい	はい

次の文字キーは表示どおりに解釈されます。

a ~ z	!	'	'	<	}
A ~ Z	$	(	.	>	[
0–9	%	)	/	=	]
~	&	*	:	?	|
#	"	+	;	{	

Set Cursor (40)

3270	*5250*	*VT*
はい	はい	はい

Set Cursor 関数は、ホスト表示スペース内のカーソルの位置を設定する際に使用します。**Set Cursor** 関数を使用する前に、 ワークステーションのアプリケーションがホストの表示スペースに接続されていることが必要です。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	40 でなければなりません	

	標準インターフェース	拡張インターフェース
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	接続したホスト表示スペース内の、配置したいカーソルの位置	

戻りパラメーター

戻りコード	説明
0	カーソルは指定された位置に配置されました。
1	ユーザー・プログラムが現在ホスト・セッションに接続されていません。
4	セッションは使用中です。
7	1 より小さい、または接続されているホスト表示スペース・ サイズを超えるカーソル位置が指定されました。
9	システム・エラーが発生しました。

Set Session Parameters (9)

3270	*5250*	*VT*
はい	はい	はい

Set Session Parameters 関数を使用して、EHLLAPI のすべてのセッションに関して特定のデフォルト・セッション・オプションを変更することができます。EHLLAPIがロードされると、セッション・オプションのデフォルトの設定値は セッション・オプション (ページ 147) に示す表で、下線を引いた項目のようになります。これらの値のいずれか、またはすべてを、以下で説明するように、呼び出し データ・ストリング内に必要なオプションを入れて変更することができます。指定した設定値は次のいずれかが発生するまで有効です。

- 次の **Set Session Parameters** (9) 関数で新しい値を指定したために変更された。
- **Reset System** (21) 関数が実行される。
- EHLLAPI アプリケーション・プログラムが終了する。

以下にセッション・オプションの影響を受ける EHLLAPI 関数の一覧表を示します。この表に記載されていない関数は、どのセッション・オプションからも影響を 受けません。それぞれの関数に影響を及ぼすセッション・オプションは“参照個所”の欄の対応する項目に示されています。これらの項目は呼び出しパラメーター (ページ 147)以降の一覧表に示されています。

関数番号	機能名 (Function Name)	参照個所
1	**Connect Presentation Space**	**11 (ページ 150)、21 (ページ 154)、22 (ページ 154)**
3	**Send Key**	**1 (ページ 147)、2 (ページ 147)、9 (ページ 150)、10 (ページ 150)、19 (ページ 153)**
4	**待機:**	**12 (ページ 151)**

関数番号	機能名 (Function Name)	参照個所
5	Copy Presentation Space	5 (ページ 148)、13 (ページ 151)、14 (ページ 152)、15 (ページ 152)、17 (ページ 152)、20 (ページ 153)
6	Search Presentation Space	1 (ページ 147)、2 (ページ 147)、3 (ページ 148)、4 (ページ 148)
8	Copy Presentation Space to String	5 (ページ 148)、13 (ページ 151)、14 (ページ 152)、15 (ページ 152)、17 (ページ 152)、20 (ページ 153)
10	のセッションの照会	16 (ページ 152),20 (ページ 153)
15	Copy String to Presentation Space	1 (ページ 147)、2 (ページ 147)、13 (ページ 151)、14 (ページ 152)、18 (ページ 153)、20 (ページ 153)
18	休止	6 (ページ 148)
30	Search Field	1 (ページ 147)、2 (ページ 147)、3 (ページ 148)、4 (ページ 148)、20 (ページ 153)
33	Copy String to Field	1 (ページ 147)、2 (ページ 147)、13 (ページ 151)、14 (ページ 152)、18 (ページ 153)、20 (ページ 153)
34	Copy Field to String	5 (ページ 148)、13 (ページ 151)、14 (ページ 152)、17 (ページ 152)、 20 (ページ 153)
35	Copy Presentation Space to Clipboard	5 (ページ 148)、13 (ページ 151)、14 (ページ 152)、17 (ページ 152)、20 (ページ 153)
36	Paste Clipboard to Presentation Space	1 (ページ 147)、2 (ページ 147)、13 (ページ 151)、14 (ページ 152)、18 (ページ 153)、20 (ページ 153)
51	Get Key	9 (ページ 150),12 (ページ 151)
90	ファイルの送信	1 (ページ 147)、2 (ページ 147)、7 (ページ 149)、8 (ページ 149)

関数番号	機能名 (Function Name)	参照個所
91	Receive File	1 (ページ 147)、2 (ページ 147)、7 (ページ 149)、8 (ページ 149)
101	Connect Window Services	21 (ページ 154),22 (ページ 154)

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	9 でなければなりません。	
データ・ストリング	変更しようとするセッション・オプションの必要な値を含むストリング。データ・ストリングには、セッション・オプション (ページ 147)に記載されているどの値 でも入れることができます。値は、データ・ストリング行にコンマまたはブランクで区切って入れます。パラメーター・セットについては、それらが影響を及ぼす関数の項で 説明しています。	
長さ	元のデータ・ストリングの明示の長さ (STREOT オプションは使用できません)。	
PS 位置	NA	

セッション・オプション

以下の表にセッション・オプションを示します。デフォルト値には下線が付いています。

1. 次の表の値は、**Send Key** (3)、**Search Presentation Space** (6)、**Copy String to Presentation Space** (15)、**Search Field** (30)、**Copy String to Field** (33)、**Send File** (90)、および **Receive File** (91) の関数に対してどのようにデータ・ストリングの長さを定義するかを判別します。

値	説明
STRLEN	すべてのストリングに対して明示の長さが渡されています。
STREOT	ストリングの長さは明示的に指定されていません。呼び出し元のデータ・ストリングは EOT 文字で終了します。

2. 次の表の値は、EHLLAPI 関数である **Send Key** (3)、**Search Presentation Space** (6)、**Copy String to Presentation Space** (15)、**Search Field** (30)、**Copy String to Field** (33)、**Send File** (90)、および **Receive File** (91) に対して呼び出し元のデータ・ストリングのテキスト終了 (EOT) 区切り文字として使用する文字を指定する際に使用します。

値	説明
EOT=c	この値で、ストリング終了文字に EOT 文字を (STREOT モードで) 指定できます。2 進数の 0 がデフォルト値です。等号の後をブランクにしないでください。

有効値にするためには、1 バイトのリテラル文字ストリングとして、 前にブランクを付けずに `c` を入力します。このステートメントで指定した EOT 文字は、STREOT オプション (セッション・オプションの 1 を参照) が 有効な場合に限り、呼び出しデータ・ストリングの長さの判別に使用されます。

3. 次の表の値は、検索関数の **Search Presentation Space** (6) および **Search Field** (30) に影響を及ぼします。

値	説明
SRCHALL	**Search Presentation Space** (6) 関数と **Search Field** (30) 関数は、ホスト表示スペース全体またはフィールド全体を走査します。
SRCHFROM	**Search Presentation Space** (6) 関数と **Search Field** (30) 関数は、指定した PS 位置から開始 (`SRCHFRWD` のとき) するか、または指定した PS 位置で終了 (`SRCHBKWD` のとき) します。

4. 次の表の値は、検索関数の **Search Presentation Space** (6) および **Search Field** (30) に影響を及ぼします。この値により、検索方向が判別されます。

値	説明
SRCHFRWD	**Search Presentation Space** (6) 関数と **Search Field** (30) 関数は、昇順方向に実行されます。
SRCHBKWD	**Search Presentation Space** (6) 関数と **Search Field** (30) 関数は、降順方向に実行されます。検索ストリングの先頭文字が指定した範囲内にあるとき、検索の対象となります。

5. 次の表の値は、**Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、**Copy Field to String** (34) の関数について、属性バイトをどのように取り扱うかを決定します。

値	説明
NOATTRB	不明な値をすべてブランクに変換します。
ATTRB	元の値として ASCII に相当しないコードをすべて戻します。
NULLATTRB	フィールド属性をすべて NULL 文字に変換します。

6. 次の表の値は、**Pause** (18) 関数に影響を及ぼします。

値	説明
FPAUSE	全期間休止。**Pause** (18) 関数で指定した期間、休止します。
IPAUSE	割り込み可能な休止。 **Start Host Notification** (23) 関数の実行後、ホストのイベントが休止を終了させます。

7. 次の表の値は、ファイル転送関数の **Send File** (90) および **Receive File** (91) が生成するメッセージを表示するかどうかを判別します。

値	説明
NOQUIET	SEND および RECEIVE メッセージが表示されます。
QUIET	SEND および RECEIVE メッセージは表示されません。

8. 次の表のステートメントは、ファイル転送関数の **Send File** (90) および **Receive File** (91) の実行中に Z and I Emulator for Windows EHLLAPI がどれくらい待機してから Cancel を自動的に発行するかを判別します。有効にするには、`c` は英文字 J ~ N でなければならず、前にブランクがあってはなりません。

値	説明
TIMEOUT=0	約 20 秒の遅延の後に Cancel が自動的に発行されます。
TIMEOUT=c	指定した遅延の後に Cancel が自動的に発行されます。下の表にある 1 文字の標識は、Cancel を発行する前に、30 秒 サイクルを何回受け入れる必要があるかを Z and I Emulator for Windows に通知します。 **文字** **値 (分)** **1** 0.5 **2** 1.0 **3** 1.5 **4** 2.0 **5** 2.5 **6** 3.0 **7** 3.5 **8** 4.0 **9** 4.5

値	説明
	J 5.0 **K** 5.5 **L** 6.0 **月** 6.5 **N** 7.0

9. 次の表のステートメントは、キー・ストローク略号に 対するエスケープ文字を定義するために使用します。このセッション・オプションは、**Send Key** (3) 関数と **Get Key** (51) 関数に影響を及ぼします。有効値にするためには、1 バイトのリテラル文字ストリングとして、前にブランクを付けずに `c` を入力します。

値	説明
ESC=c	この値により、キー・ストローク略号に対するエスケープ文字を 指定できます (@ がデフォルト値)。等号の後をブランクにしないでください。ブランクはエスケープ文字に使用できません。

10. 次の表の値は、**Send Key** (3) 関数を使用して送信されたストリングを EHLLAPI が先に自動的にリセットするかどうかを判別します。

値	説明
AUTORESET	EHLLAPI は、リセット付きの **Send Key** (3) 関数を使用して送信されたキーの全ストリングに接頭部を付けることにより、すべての使用禁止条件のリセットを試行します。
NORESET	AUTORESET を行いません。

11. 次の表の値は、**Connect Presentation Space** (1) コマンド関数に影響を及ぼします。

値	説明
CONLOG	ワークステーション・セッションとホスト・セッションの間の論理接続を 確立します。接続中は、要求された表示スペースへジャンプしません。
CONPHYS	ワークステーション・セッションとホスト・セッションの間の物理接続を 確立します。接続中に、要求された表示スペースへジャンプします。

12. 次の表の値は、**Wait** (4) 関数と **Get Key** (51) 関数に影響を及ぼします。それぞれの値については、2 つの異なる影響があり、それぞれの関数に対応しています。

値	説明
TWAIT	**Wait** (4) 関数は、 XCLOCK (X []) または XSYSTEM でタイムアウトになるまで最大 1 分間待機します。 **Get Key** (51) 関数は、EHLLAPI アプリケーション・プログラムがキーを代行受信してから、そのアプリケーションに制御を戻します (このキーは、通常のキーまたは **Start Keystroke Intercept** (50) 関数で指定されたオプションに基づく AID キーです)。
LWAIT	**Wait** (4) 関数は、XCLOCK (X [])/XSYSTEM がクリアされるまで待機します。ホストが使用可能になるまでユーザーのアプリケーションに制御が戻らな いため、このオプションはお勧めできません。 **Get Key** (51) 関数は、EHLLAPI アプリケーション・プログラムがキーを代行受信してから、そのアプリケーションに制御を戻します (このキーは、通常のキーまたは **Start Keystroke Intercept** (50) 関数で指定されたオプションに基づく AID キーです)。
NWAIT	**Wait** (4) 関数は、状況をチェックし、即座に状況を戻します (待機なし)。 **Get Key** (51) 関数は、**Start Keystroke Intercept** (50) 関数で指定されたオプションに一致したオプションがキューにないと、4 番目のパラメーターに戻りコード 25 (キー・ストローク使用不可) を戻します。

注: NWAIT を使用するようお勧めします。

13. 次の表の値は、**Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、**Copy String to Presentation Space** (15)、**Copy String to Field** (33) および **Copy Field to String** (34) に影響を及ぼします。拡張属性バイト (EAB) には、拡張文字属性および拡張フィールド属性が含まれます。

値	説明
NOEAB	データのみを渡します (EAB なし)。
EAB	拡張属性バイトを指定して表示スペース・データを渡します。画面に表示されるそれぞれの文字については、2 バイトのデータが渡されます。このため、表示スペースの 2 倍のサイズのバッファーを 事前割り振りしておくことが必要です。例えば、2 x 1920 = 3840 (24 行 80 桁の表示スペース用)。 文字ストリングの拡張属性は、フィールド内の個々の文字の属性としてでは なく、フィールド・バイトの属性として報告される場合があります。このような場合、画面上の個々の文字または文字セットに下線が付けられる かどうかを知るには、フィールド属性バイト (画面に表示されるフィールドの 前にあるバイト) の位置を指定して CopyPStoString を実行し、そのフィールド 内の全文字に適用される EAB 情報を得てください。

注: EHLLAPI Copy PS to String を使用すると、オペレーターには見えないようにすべきテキストがコピーされます。隠しデータがあるかどうかを判別するには 、EHLLAPI Set Session Parameters 関数を使用して、NODISPLAY オプションを設定してください。これにより、EHLLAPI は非表示フィールドを NULL として戻します。データを隠すための一般的な別の手順は、前景と背景の色を同一色 (例えば、 黒色) に設定して、テキストは表示されるが人間のオペレーターには見え ないようにすることです。アプリケーションにこれを検出させる唯一の方法は、 EAB と XLATE の セッション・パラメーターを使用して PS をコピーすることです。各バイト位置の前景/背景の色が戻され、見えない文字を判別することができ ます。

14. 次の表の値は、**Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、**Copy String to Presentation Space** (15)、**Copy String to Field** (33) および **Copy Field to String** (34) に影響を及ぼします。

値	説明
NOXLATE	EAB は変換されません。
XLATE	EAB は PC のカラー・グラフィックス・アダプター (CGA) 形式に変換されます。

15. 次の表にある値は、NOATTRB と NOEAB が指定されている場合に **Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、および **Copy Presentation Space to Clipboard** (35) に影響します。

値	説明
BLANK	不明な値をすべて X'20' に変換します。
NOBLANK	不明な値をすべて X'00' に変換します。

デフォルト値は BLANK です。デフォルト値を NOBLANK に変更したい場合は 、Z and I Emulator for Windowsのユーザー・クラス・アプリケーション・データのディレクトリ ーに配置されている PCSWIN.INI ファイルに、以下のステートメントを追加し てください。

```
[API]
NullToBlank=NO
```

16. 次の表の値は、**Query Sessions** (10) により戻される表示スペースのサイズに影響を及ぼします。

値	説明
CFGSIZE	接続されている表示スペースの構成サイズを戻します。このオプションは、ホストによる構成サイズの一時変更を無視します。
NOCFGSIZE	接続されている表示スペースの現在のサイズを戻します。

17. 次の表にある値は、**Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、**Copy Field to String** (34)、および **Copy Presentation Space to Clipboard** (35) に影響します。

値	説明
DISPLAY	表示スペース内の非表示フィールドを、コピー先の バッファー域にコピーします。現行のアプリケーションは、通常通りに機能します。

値	説明
NODISPLAY	表示スペース内の非表示フィールドを、表示フィールドと 同じ方法でコピー先のバッファー域にコピーしません。非表示フィールドは、コピー先のバッファーに NULL 文字のストリングとして コピーされます。これにより、アプリケーションでは、パスワードなどの機密情報を表示せずに、 コピーしたバッファーをプレゼンテーション・ウィンドウに表示することが できます。

18. 次の表にある値は、**Copy String to Presentation Space** (15)、**Copy String to Field** (33)、および **Paste Clipboard to Presentation Space** (36) に影響します。

値	説明
NOPUTEAB	**Copy String to Presentation Space** または **Copy String to Field** のデータ・ストリングに EAB を含めません。
PUTEAB	**Copy String to Presentation Space** または **Copy String to Field** のデータ・ストリングに EAB を含めます。

このオプションは、Communication Manager/2 と互換性を持たせるためのものです。Communication Manager/2 では、**Set Session Parameters** の EAB (または EAD) が有効なときは、**Copy String to Presentation Space** または **Copy String to Field** で指定するデータ・ストリングに、EAB (または EAD) が文字データと共に含まれていなければなりません。一方、従来のZ and I Emulator for Windowsでは、EAB (または EAD) が 有効であっても、これらの関数に指定するデータ・ストリングに含まれるの は文字データのみでした。しかし、Z and I Emulator for Windowsでは、PUTEAB を設定して、データ・ ストリングの中に EAB (または EAD) を含めることにより、Communication Manager/2との互換性 を持つことができます。

19. 次の表の値は、**Send Key** (3) 関数に影響を及ぼします。キーボードがブロックまたは使用されている場合、キー・ストロークは 処理されません。4 分のタイムアウトが発生するまで、関数がキー・ストロークの再送を試行するか、 または、キーボードがブロックまたは使用されていることが確認されて即座に関数が戻るかどうかを、このオプションにより決定します。

値	説明
RETRY	キー・ストロークが送信されるか、4 分のタイムアウトが発生するまで キー・ストロークの送信を試行し続けます。
NORETRY	キーボードがブロックまたは使用されていることが確認されたら、関数は即座に戻ります。

20. 次の表にある値は **Copy Presentation Space** (5)、**Copy Presentation Space to String** (8)、**Copy String to Presentation Space** (15)、**Copy String to Field** (33)、**Copy Field to String** (34) **Search Field** (30)、**Query Sessions** (10)、**Copy Presentation Space to Clipboard** (35)、および **Paste Clipboard to Presentation Space** (36) に影響します。

値	説明
EXTEND_PS	5250 エミュレーションは、24 行 80 桁の表示スペースをサポートします。Communication Manager 5250 エミュレーションが 25 番目の行を表示するインスタンスもあります。このようなことが起こるのは、ホストからのエラー・

値	説明
	メッセージが表示される場合、またはオペレーターが SysReq キーを選択する場合です。 Z and I Emulator for Windows行 24 に 25 行目の情報を表示しますが、通常 EHLLAPI は実際の 24 行目を表示します。**EXTEND_PS** オプションによって、EHLLAPI アプリケーションは Communication Manager EHLLAPIと同じインターフェースを使用することができ、このような状況が発生したときに有効な表示スペースが拡張されます。
NOEXTEND_PS	上記の状態が発生した場合に表示スペースを拡張しません。これはデフォルト値です。

21. 次の表の値は、**Connect Presentation Space** (1) と **Connect Window Services** (101) 関数に影響を及ぼします。これらのオプションは、アプリケーションで、それが接続されている 表示スペースを他の表示スペースと 共用できるか、また共用することになるかを指定します。各 **Set Session Parameter** 呼び出しについて、次の値のうち 1 つだけを指定することができます。

値	説明
SUPER_WRITE	アプリケーションは、共用が可能で書き込みアクセス許可を有する 他のアプリケーションに同時に同じ表示スペースに接続する ことを許可します。この関数を呼び出したアプリケーションでは、 監視タイプの関数が実行されますが、表示スペースを 共用する他のアプリケーションに対してエラーは生成されません。
WRITE_SUPER	アプリケーションは、書き込みアクセスを要求し、監視アプリケーション だけにその表示スペースへ同時に接続することを許可します。これはデフォルト値です。
WRITE_WRITE	アプリケーションは、書き込みアクセスを要求し、 その動作が予測可能なパートナーまたは他のアプリケーションに 表示スペースを共用することを許可します。
WRITE_READ	アプリケーションは、書き込みアクセスを要求し、読み取り専用関数を実行 する他のアプリケーションに表示スペースを共用することを許可します。また、表示スペースのコピーおよび他の読み取り専用操作を 通常通り実行することもできます。
WRITE_NONE	アプリケーションは、表示スペースを排他使用します。監視アプリケーションも含めて、他のアプリケーションは、表示スペースを共用できません。また、表示スペースのコピーおよび他の 読み取り専用操作を通常通り実行することもできます。
READ_WRITE	アプリケーションは、表示スペースをモニターするための読み取り アクセスのみを要求し、読み取りまたは書き込み (あるいはその両方の) 関数を 実行する他のアプリケーションに表示スペースを共用することを 許可します。また、表示スペースのコピーおよび他の読み取り専用操作を 通常通り実行することもできます。

22. 次の表の値により、アプリケーションは、表示スペースの共用の要求を、パートナー・ アプリケーション (一緒に動作するよう開発されたアプリケーション) との間での 共用に制限することができます。

値	説明
NOKEY	この値を指定すると、アプリケーションは、**KEY** パラメーターを指定していない既存のアプリケーションとの互換性を持つようになります。
KEY$*nnnnnnnn*	サポートする表示スペースの共用アクセスを制限するために、 キーワードを使用します。キーワードは、正確に 8 バイトの長さでなければなりません。

戻りパラメーター

この関数はデータ・ストリング長および戻りコードを戻します。

長さ:

設定された正しいセッション・パラメーターの数。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	セッション・パラメーターが設定されました。
2	1 つ以上のパラメーターが無効でした。
9	システム・エラーが発生しました。

Start Close Intercept (41)

3270	*5250*	*VT*
はい	はい	はい

Start Close Intercept 関数を使用して、アプリケーションは、ユーザーがエミュレーター・セッション・ウィンドウから終了オプションを選択したときに生成される終了要求を代行受信できます。この関数は **Stop Close Intercept** (43) 関数が要求されるまで、クローズ要求を代行受信してそれを破棄します。

この関数を使用した後、アプリケーション・プログラムは **Query Close Intercept** (42) 関数を使用して、いつ終了要求が出されたかを確認することができます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

バイト	定義	
	標準インターフェース	拡張インターフェース

バイト	定義	
関数番号	41 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	5 または 6	12 でなければなりません
PS 位置	該当なし	

データ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
4–5		この位置にあるデータは EHLLAPI で無視されます。ただし、移行プログラムでこの位置にデータを指定していても、 エラーは発生しません。このデータはアプリケーションの移行時に互換性を保つために設けられています。
6	5	非同期メッセージ・モードを要求する場合は、M を指定 します (Windows のみ)。
	6–8	予約済み。
2–3	9–12	バイト位置 5 (16 ビットの場合は 6) に M を指定した場合、 メッセージを受け取るウィンドウのウィンドウ・ハンドルを指定してください。メッセージは、RegisterWindowMessage (PCSHLL) の 戻り値 (0 以外) です。

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

非同期メッセージ・モードをバイト位置 5 (標準インターフェースの 場合は 6) に指定しないで、関数が正常終了した場合、以下の データ・ストリングが戻されます。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–8	予約済み。
	9–12	EHLLAPIによってイベント・オブジェクト・アドレスが戻される 4 バイト値。アプリケーションは、このイベント・オブジェクトを待ちます (32 ビットのみ)。

データ・ストリング:

M (非同期メッセージ・モード) をバイト位置 5 (標準 インターフェースの場合は 6) に指定して、関数が正常終了した場合、 以下のデータ・ストリングが戻されます。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–8	予約済み
2–3	9–10	非同期メッセージ・モードのタスク ID

注: ユーザーが終了オプションを選択すると、 アプリケーション・ウィンドウはメッセージを受け取ります。メッセージは、RegisterWindowMessage(PCSHLL) の戻り値です。wParam パラメーターには、この関数呼び出しによって戻されたタスク ID が 入ります。lParam パラメーターの HIWORD には戻りコード 26 (代行受信の終了) が 入り、lParam パラメーターの LOWORD には関数番号 41 が入ります。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Start Close Intercept** 関数は正常終了しました。
1	ホストの無効な表示スペースが指定されました。
2	パラメーター・エラーが生じました。
9	システム・エラーが発生しました。
10	エミュレーション・プログラムはこの関数をサポートしていません。

使用上の注意

1. 開始要求関数が戻される場合は、戻されるイベント・オブジェクトまたはセマフォーは非通知状態になります。イベント・オブジェクトは、終了要求が出されるたびに通知状態になります。複数の終了要求イベントに対する通知を受信するためには、**SetEvent** または **Query Close Intercept** (42) 関数を使用して、 毎回イベント・オブジェクトを通知状態にすることが必要です。
2. この関数を使用した後、アプリケーション・プログラムは **Query Close Intercept** (42) 関数を使用して、いつ終了要求が出されたかを確認することができます。アプリケーションは、そのイベントが発生したことを確認するために戻される イベント・オブジェクトを待つこともできます。
3. これは排他呼び出しではありません。同じ短縮セッション ID に対して、複数のアプリケーションがこの関数を 要求することができます。
4. アプリケーションがセッションに対する終了要求を代行受信しない 場合、その後ユーザーがエミュレーター操作ダイアログから選択した終了要求が そのセッションに対する正常停止の要求となります。

Start Communication Notification (80)

3270	*5250*	*VT*
はい	はい	はい

Start Communication Notification 関数はプロセスを開始します。このプロセスによって、ユーザーの EHLLAPI アプリケーションは、指定されたセッションがホストに接続されているかどうかを判別できます。

この関数を使用した後、アプリケーションは、**Query Communication Event** (81) を使用して、 セッションが接続されているか、または切断されているかを判別できます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	拡張インターフェース
関数番号	80 でなければなりません
データ・ストリング	事前に割り振られた構造。次の表を参照してください。
長さ	16
PS 位置	該当なし

呼び出しデータ構造には、以下の要素が含まれます。

バイト	**定義**
1	表示スペースの 1 文字の短縮名 (PSID)。
2 ~ 4	予約済み
5	以下のいずれかの値です。 • 文字 C はセッションがホストから切断、またはホストに接続されたときの通知を要求します。 • 文字 A は、非同期モードの通知を要求します。A を指定した場合は、 バイト位置 9 ~ 12 にイベント・オブジェクト (Windows) のアドレスが戻されます。文字 C を、位置 13 に指定する必要があります。 • 文字 M は、非同期メッセージ・モードの通知を要求します。M を指定する場合、イベント選択文字 C をバイト位置 13 に指定する必要があります。
6-8	予約済み
9 ~ 12	バイト位置 5 に M を指定した場合、 メッセージを受け取るウィンドウのウィンドウ・ハンドルを指定してください。メッセージは、RegisterWindowMessage (PCSHLL) の戻り値 (0 以外) です。
13	バイト位置 5 が A または M の場合、このバイトは文字 C を含む必要があります。
14-16	予約済み

データ・ストリング

A (非同期モード) を呼び出しデータ構造のバイト位置 5 に指定して、関数が正常終了した場合、 以下のデータ・ストリングが戻されます。

バイト	定義
1	1 文字の表示スペース短縮名 (PSID)
2-8	予約済み
9 ~ 12	EHLLAPI によってイベント・オブジェクト・ハンドルが戻される、4 バイトの 2 進値。アプリケーションは、このイベント・オブジェクトを待ちます。

M (非同期メッセージ・モード) を呼び出しデータ構造のバイト位置 5 に指定して、 この関数が正常終了した場合、以下のデータ・ストリングが戻されます。

バイト	定義
1	1 文字の表示スペース短縮名 (PSID)
2-8	予約済み
9 ~ 10	非同期メッセージ・モードのタスク ID

セッションがアプリケーション・ウィンドウを接続または切断したとき、 メッセージを受信します。このメッセージは、RegisterWindowMessage (PCSHLL) の戻り値です。wParam には、関数呼び出しによって戻されたタスク ID が入ります。セッションがホストに接続された場合、lParam の HIWORD には `21` が入り、 セッションがホストから切断された場合、lParam の HIWORD には 22 が入ります。lParam の LOWORD には関数番号 `80` が入ります。

戻りパラメーター

戻りコード	定義
0	関数は正常終了しました。
1	誤った PSID が指定されました。
2	指定されたパラメーターに誤りがありました。
9	システム・エラーが発生しました。

使用上の注意

1. 1 つのアプリケーション・プログラムが複数のホスト・セッション でこの関数を発行することができます。セッションの通信状況を判別するために、**Query Communication Event** (81) 関数を使用することができます。
2. アプリケーションが非同期オプションを選択した場合、 そのアプリケーションは Windows SDK 呼び出し **WaitForSingleObject** を使用して、そのセッションの通信状況が変化するのを待つことができます。
3. このイベント・オブジェクトは、最初は非通知状態にあります。イベントが起こるたびに、イベント・オブジェクトは通知されます。複数のイベントに対する通知を受信するためには、アプリケーションは、Windows SDK 呼び出し **ResetEvent**、 または、関数 81 **Query Communications Event** を使用して、通知されるたびにイベント・オブジェクトを非通知状態にする必要があります。
4. 同じアプリケーションからの同じオプションを指定したこの関数への複数の呼び出しは、 無視されます。
5. この関数は 1 つのアプリケーションに対して排他的ではありません。同じセッション ID に対して、複数のアプリケーションがこの関数を要求することができます。

Start Host Notification (23)

3270	*5250*	*VT*
はい	はい	はい

Start Host Notification 関数は、指定されたホスト表示スペースまたは OIA が更新されたかどうかを EHLLAPI アプリケーション・プログラムが判別するための処理を開始します。

この関数を使用した後、アプリケーション・プログラムは **Query Host Update** (24) 関数を使用して、いつホスト・イベントが発生したかを確認することができます。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	23 でなければなりません	
データ・ストリング	事前に割り振られたストリング。次の表を参照してください。	
長さ	6 または 7 が暗黙指定されます。	16
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	以下のいずれかの値です。 • 表示スペースの 1 文字の短縮名 (PSID) • ホストに接続されたホスト表示スペースの要求を示すブランクまたはヌル
	2–4	予約済み。
2	5	以下のいずれかの値です。 • B ホスト表示スペースと OIA の両方の更新通知を要求します。 • O OIA の更新通知のみを要求します。 • P ホスト表示スペースの更新通知のみを 要求します。 • A 非同期モードの通知を要求します。 A が指定されている場合は、バイト位置 9 ~ 12 からイベント・オブジェクトのアドレスが返されます。イベント選択文字 B、O、P はバイト位置 13 に指定する必要があります。 • M 非同期メッセージ・モードの通知を要求します。

バイト		定義
		M を指定する場合、イベント選択文字 B、O、P のいずれかを バイト位置 13 (16 ビットの場合は 7) に指定します。 • E は、プリンター・セッション中の完了通知を要求する文字です。
	6–8	予約済み。
3–4	9–12	バイト位置 5 (16 ビットの場合は 2) に M を指定した場合、 メッセージを受け取るウィンドウのウィンドウ・ハンドルを指定してください。メッセージは、RegisterWindowMessage (PCSHLL) の 戻り値 (0 以外) です。
7	13	バイト位置 5 (16 ビットの場合は 2) に A または M を指定した場合、 値は次のいずれかです。 • B ホスト表示スペースと OIA の両方の更新通知を要求します。 • O OIA の更新通知のみを要求します。 • P ホスト・プレゼンテーションの更新通知のみを要求します。
	14–16	予約済み。

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

A (非同期モードの通知) をバイト位置 5 に指定し、 関数が正常終了した場合、以下のデータ・ストリングが戻されます。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–8	予約済み。
	9–12	EHLLAPI によってイベント・オブジェクト・アドレスが戻される 4 バイト値。アプリケーションは、このイベント・オブジェクトを待ちます (32 ビットのみ)。

データ・ストリング:

M (非同期メッセージ・モード) をバイト位置 5 (標準 インターフェースの場合は 2) に指定して、関数が正常終了した場合、 以下のデータ・ストリングが戻されます。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–8	予約済み
3–4	9–10	非同期メッセージ・モードのタスク ID

注: OIA または表示スペースを更新すると、 アプリケーション・ウィンドウはメッセージを受け取ります。メッセージは、RegisterWindowMessage(PCSHLL) の戻り値です。wParam パラメーターには、関数呼び出しによって戻されたタスク ID が入ります。lParam の HIWORD には戻りコード 21 (OIA の更新)、22 (ホスト表示スペースの更新)、23 (OIA とホスト表示スペースの 両方の更新) が入り、lParam パラメーターの LOWORD には関数番号 23 が入ります。

戻りコード:

以下のコードが定義されています。

戻りコード	定義
0	**Start Host Notification** 関数は正常終了しました。
1	ホストの無効な表示スペースが指定されました。
2	指定されたパラメーターに誤りがありました。
9	システム・エラーが発生しました。

使用上の注意

1. 1 つのアプリケーション・プログラムが複数のホスト・セッション でこの関数を発行することができます。**Pause** (18) 関数により、1 つ以上のホスト・セッション (PS、OIA、またはその両方) が変更された時点でアプリケーションに通知することができます。PS、OIA、またはその両方が変更されたかどうかを判別するために **Query Host Update** (24) 関数を使用することができます。
2. アプリケーションで非同期オプションを選択した場合、 アプリケーションは、そのイベントが発生した時期を判別するために戻される イベント・オブジェクトまたはセマフォーを待つこともできます。
3. イベント・オブジェクトまたはセマフォーは、 最初は非送信状態であり、該当のイベントが発生するたびに送信されます。複数のイベントに対する通知を受信するためには、アプリケーションは、**ResetEvent** または **Query Host Update** (24) 関数を使用して、通知されるたびにイベント・オブジェクトを非通知状態にする必要があります。
4. アプリケーションは、同じオプション で Start Host Notification を複数回要求することはできません。
5. これは排他呼び出しではありません。同じ短縮セッション ID に対して、複数のアプリケーションがこの関数を 要求することができます。

Start Keystroke Intercept (50)

3270	*5250*	*VT*
はい	はい	はい

Start Keystroke Intercept 関数により、 ワークステーションのアプリケーションは、端末オペレーターがセッションに送信したキー・ストロークをフィルターにかけることができます。この関数を呼び出した後、キー・ストローク・キューがオーバーフローするまで、あるいは、**Stop Keystroke Intercept** (53) 関数または **Reset System** (21) 関数が呼び出されるまで、 キー・ストロークは代行受信されて保管されます。代行受信したキー・ストロークは以下のように処理することができます。

- **Get Key** (51) 機能を介して受信し、**Send Key** (3) 機能を使用して同じセッションまたは別のセッションに送信します
- **Post Intercept Status** (52) 関数により、受け入れまたは拒否する
- **Send Key** (3) 関数で別のキー・ストロークに置き換える
- 別の処理のトリガーとして使用する。

前提呼び出し

この関数には前提呼び出しはありません。

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	50 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	キー・ストローク・バッファー・サイズ。 EHLLAPI は、このバッファーに最低 32 バイトを割り振ります。	
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものが入っています。

バイト		**定義**
標準	拡張	
1	1	以下のいずれかの値です。 • 特定のホスト表示スペースの短縮名 (PSID) • ホストに接続されたホスト表示スペースの要求を示すブランクまたはヌル
	2–4	予約済み。
2	5	**オプション・コード**文字: • D AID キー・ストロークのみ。 • L すべてのキー・ストローク。 • E 編集キーおよびすべてのキー・ストローク。(拡張モードでのみ利用可能) • 通知の非同期メッセージ・モードを要求する場合は M (Windows のみ)。 M を指定する場合、コード文字 D、L、または E (拡張モード) をバイト 位置 13 (16 ビットの場合は 7) に指定します。 前提条件: キーボードのキーは、編集関数にマップされる必要があります。例えば、Ctrl+C は編集コピー関数にマップされます。サポートされる編集

バイト		定義
		関数 について詳しくは、表 12 : @S (シフト)、@W (編集) および @ 英字による略号 (ページ 141)を参照してください。
	6–8	予約済み。
3–4	9–12	バイト位置 5 (16 ビットの場合は 2) に M を指定した場合、 メッセージを受け取るウィンドウのウィンドウ・ハンドルを指定してください。メッセージは、RegisterWindowMessage (PCSHLL) の 戻り値 (0 以外) です。
7	13	バイト位置 5 (16 ビットの場合は 2) に M を指定した場合、 値は次のいずれかです。 • D AID キー・ストロークのみ。 • L すべてのキー・ストローク。 • E 編集キーおよびすべてのキー・ストローク。(拡張モードでのみ利用可能。)
	14–16	予約済み。

データ・ストリング:

M (非同期メッセージ・モード) をバイト位置 5 (標準 インターフェースの場合は 2) に指定して、関数が正常終了した場合、 以下のデータ・ストリングが戻されます。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–8	予約済み
3–4	9–10	非同期メッセージ・モードのタスク ID

注: ユーザーがセッションへキー・ストロークを送信すると、 アプリケーション・ウィンドウはメッセージを受け取ります。メッセージは、RegisterWindowMessge (PCSHLL) の戻り値です。wParam パラメーターには、関数呼び出しによって戻されたタスク ID が入ります。lParam パラメーターの HIWORD には戻りコード 0 (関数の正常終了) が 入り、LOWORD には関数番号 50 が入ります。

戻りパラメーター

戻りコード	説明
0	**Start Keystroke Intercept** 関数は正常終了しました。
1	無効な表示スペースが指定されました。
2	無効なオプションが指定されました。
4	該当する表示スペースが使用中のため、関数は実行できません。
9	システム・エラーが発生しました。Release が使用中です。

使用上の注意

1. **Get Key** (51) 関数に対して戻りコードが 31 になる場合は、以下のどちらかを行ってください。
   - この関数の呼び出し元のデータ・ストリング長パラメーターの値を 増やしてください。
   - **Get Key** (51) 関数をより頻繁に実行してください。

   代行受信されたキー・ストロークは、バッファー中の 3 バイトを使用します。次の代行受信されたキー・ストロークは、隣接する 3 バイトに置かれます。**Get Key** (51) 関数がキー・ストローク (先入れ、先出し、FIFO) を取り出す場合は、そのキー・ストロークが使用していた 3 バイトを他のキー・ストロークで使用できるようになります。バッファーのサイズを増やすか、バッファーからキー・ストロークを取り出す 頻度を増やすことによって、バッファーのオーバーフローを避けることができます。

   PC/3270 において、戻りコード 31 を除去する他の方法は、 再開モードで PC/3270 エミュレーターを操作することです。
2. オプション・コード D が指定されている 場合、EHLLAPI は非 AID キーを、本来予定されている 表示スペースへ書き込み、AID キーのみを アプリケーションに戻します。
3. **Stop Keystroke Intercept** (53) 関数を呼び出してから、EHLLAPI アプリケーションを終了してください。この関数を呼び出さない場合は、キー・ストロークの代行 受信機能が引き続き使用可能となり、予測できない結果が発生します。

Start Playing Macro (110)

3270	*5250*	*VT*
はい	はい	はい

Start Playing Macro 関数を使用してマクロを呼び出します。マクロは接続しているセッションで実行されます。

注: このマクロは、Z and I Emulator for Windowsのユーザー・クラス・アプリケーション・データのディレクトリーに存在する必要があります。さらに、このマクロ名の関数呼び出しには拡張子を指定してはなりません。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース
関数番号	110 でなければなりません
データ・ストリング	次の表を参照してください。
長さ	マクロ名の長さに 3 を加えたもの

	標準インターフェース
PS 位置	該当なし

バイト		定義
標準	拡張	
1-2		予約済み
3-n		NULL で終了するマクロ名

戻りパラメーター

戻りコード	説明
0	**Start Playing Macro** 関数は正常終了しました。
1	プログラムが現在ホスト・セッションに接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。

Stop Close Intercept (43)

3270	*5250*	*VT*
はい	はい	はい

Stop Close Intercept 関数によりアプリケーションは **Start Close Intercept** (41) 関数をオフにすることができます。アプリケーションにより **Stop Close Intercept** 関数が発行されると、その後の終了要求は通常停止となって論理端末セッションへ送信されます。

前提呼び出し

Start Close Intercept (41)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	43 でなければなりません	
データ・ストリング	ホスト表示スペースの 1 文字の短縮セッション ID。	
長さ	1	4 である必要があります。
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)

バイト		定義
	2–4	予約済み

戻りパラメーター

戻りコード	説明
0	**Stop Close Intercept** 関数は正常終了しました。
1	ホストの無効な表示スペースが指定されました。
2	パラメーター指定でエラーが発生しました。
8	**Start Close Intercept** (41) 関数が事前に発行されていませんでした。
9	システム・エラーが発生しました。
12	セッションが停止しました。

Stop Communication Notification (82)

3270	*5250*	*VT*
はい	はい	はい

Stop Communication Notification は、指定されたセッションで何らかの通信イベントが発生したかどうかを判別する **Query Communication Event** (81) 関数を使用停止にします。

前提呼び出し

Start Communication Notification (80)

呼び出しパラメーター

	拡張インターフェース
関数番号	82 でなければなりません
データ・ストリング	1 文字のホスト表示スペースの短縮名、またはホスト接続されて いる表示スペースへの更新要求を表すブランクあるいは NULL。
長さ	4 が暗黙指定されます
PS 位置	該当なし

呼び出しデータ構造には、以下の要素が含まれます。

バイト	定義
1	表示スペースの 1 文字の短縮名 (PSID)
2 ~ 4	予約済み

戻りパラメーター

戻りコード	定義

0	関数は正常終了しました。
1	誤った PSID が指定されました。
8	この PSID に対して、先に **Start Communication Notification** (80) 関数が呼び出されていません。
9	システム・エラーが発生しました。

Stop Host Notification (25)

3270	*5250*	*VT*
はい	はい	はい

Stop Host Notification 関数は、 ホスト表示スペースまたは OIA が更新されたかどうかを判別する **Query Host Update** (24) 関数の機能を使用不可にします。この関数は、ホストのイベントが **Pause** (18) 関数に影響を及ぼすことも防ぎます。

前提呼び出し

Start Host Notification (23)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	121 でなければなりません	
データ・ストリング	以下の注を参照してください。	
長さ	1 が暗黙指定されます。	4 である必要があります。
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

注: 該当する表示スペースの 1 文字の短縮 ID、または ホストで接続する表示スペースへの要求を表すブランク あるいは NULL。

戻りパラメーター

戻りコード	**定義**
0	**Stop Host Notification** 関数は正常終了しました。
1	ホストの無効な表示スペースが指定されました。

戻りコード	定義
8	**Start Host Notification** (23) 関数が事前に発行されていませんでした。
9	システム・エラーが発生しました。

Stop Keystroke Intercept (53)

3270	*5250*	*VT*
はい	はい	はい

Stop Keystroke Intercept 関数は、アプリケーション・プログラムのキー・ストローク代行受信を終了します。

前提呼び出し

Start Keystroke Intercept (50)

呼び出しパラメーター

	標準インターフェース	拡張インターフェース
関数番号	53 でなければなりません	
データ・ストリング	該当する表示スペースの短縮名 (PSID)	
長さ	1 が暗黙指定されます。	4 である必要があります。
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		定義
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み

戻りパラメーター

戻りコード	説明
0	**Stop Keystroke Intercept** 関数は正常終了しました。
1	無効な表示スペースが指定されました。
8	この表示スペースに対して、先に **Start Keystroke Intercept** (50) 関数が呼び出されていません。
9	システム・エラーが発生しました。

Wait (4)

3270	*5250*	*VT*
はい	はい	はい

Wait 関数は、ホストで接続する表示スペースの状況をチェックします。セッションがホストの応答 (XCLOCK (X []) または XSYSTEM で示される) を待機している場合、**Wait** 関数は、待機状態が解除されたかどうかを確認するため、EHLLAPI を最大 1 分間待機させます。

前提呼び出し

Connect Presentation Space (1)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	4 である必要があります。	
データ・ストリング	該当なし	
長さ	該当なし	
PS 位置	該当なし	

戻りパラメーター

戻りコード	**定義**
0	キーボードはアンロックされており、入力できます。
1	アプリケーション・プログラムが正しいセッションに接続されていません。
4	XCLOCK (X []) または XSYSTEM の間にタイムアウトになりました。
5	キーボードがロックされています。
9	システム・エラーが発生しました。

使用上の注意

1. **Wait** 関数は、**Send Key** (3) などの関数によるホスト要求を完了するために必要な時間を提供します。 **Set Session Parameters** (9) 関数を使用して、`TWAIT`、`LWAIT`、または `NWAIT` の各オプションを要求することができます。12 (ページ 151) を参照してください。
2. この関数を使用して、ホストの OIA が使用禁止かどうかを調べることが できます。
3. ホストがキーボードをアンロックすると、**Wait** 関数が実行されます。したがって、戻りコード 0 は、トランザクションが完了したことを 必ずしも意味しません。トランザクションの完了を確認するには 、**Search Field** (30) 関数または **Search Presentation Space** (6) 関数を、**Wait** 関数と組み合わせて使用し、該当するキーワード・プロンプトを探してください。

Window Status (104)

3270	*5250*	*VT*
はい	はい	はい

Window Status 関数を使用して、アプリケーションはウィンドウの表示スペース・サイズ、位置、表示状態を照会したり変更することができます。

前提呼び出し

Connect Window Services (101)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	104 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	16 または 20	24 または 28
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)
	2–4	予約済み
2	5	要求オプションの値。 次のうちの 1 つを選択してください。 • 状況設定には X'01' **注:** セッションが複合 OLE 文書に組み込まれている場合は、この関数のセット形式 (バイト 5 = X'01') は常に 0 を戻しますが、何も行われません。 • 状況照会には X'02' • 拡張状況照会には X'03'
	6	予約済み

要求オプションの値が X'01' (状況設定) の場合:

バイト		**定義**
標準	拡張	

バイト		定義
3–4	7–8	要求オプションが 1 (状況設定) の場合には、 状況設定ビットを含む 16 ビット/32 ビットのワード。要求オプションが状況設定の場合、有効な戻り値は以下のコードです。 **X'0001'** ウィンドウ・サイズを変更します。(最小化、最大化、復元、移動は無効) **X'0002'** ウィンドウを移動します。(最小化、最大化、サイズ変更、復元は無効) **X'0004'** ZORDER ウィンドウの置き換え。 **X'0008'** ウィンドウを可視状態に設定します。 **X'0010'** ウィンドウを不可視状態に設定します。 **X'0080'** ウィンドウを活動状態にします。(ZORDER が指定されていない場合には、ウィンドウ・フォーカスを設定し、 前面に配置します。この場合、ZORDER 配置が使用されます。) **X'0100'** ウィンドウを非活動状態にします。(ZORDER も共に指定されていない場合には、ウィンドウを非活動状態にし、 一番下のウィンドウにします。この場合、ZORDER 配置が使用されます。) **X'0400'** ウィンドウを最小化します。(最大化、復元、サイズ変更、移動は無効) **X'0800'** ウィンドウを最大化します。(最小化、復元、サイズ変更、移動は無効) **X'1000'** ウィンドウを復元します。(最小化、最大化、サイズ変更、移動は無効)
5–6	9–12	ウィンドウ位置の X 座標を含む 16 ビット/32 ビットのワード。(移動オプションが設定されていない場合には無視されます。)

バイト		定義
7–8	13–16	ウィンドウ位置の Y 座標を含む 16 ビット/32 ビットのワード。(移動オプションが設定されていない場合には無視されます。)
9–10	17–20	装置単位での X ウィンドウ・サイズを含む 16 ビット/32 ビットのワード。(サイズ・オプションが設定されていない場合には無視されます。)
11–12	21–24	装置単位での Y ウィンドウ・サイズを含む 16 ビット/32 ビットのワード。(サイズ・オプションが設定されていない場合には無視されます。)
13–16	25–28	ウィンドウの相対配置のためのウィンドウ・ハンドルを 含む 16 ビット/32 ビットのワード。この 2 つのワードは設定オプション専用です。(ZORDER オプションが設定されていない場合には無視されます。) 有効な値は以下のとおりです。 X'00000003' すべての兄弟ウィンドウの前に配置。X'00000004' すべての兄弟ウィンドウの後ろに配置。

要求オプションの値が X'02' (状況照会) の場合:

バイト		定義
標準	拡張	
3–4	7–8	要求オプションが 2 (拡張状況照会) の場合には、X'0000' を含む 16 ビット/32 ビットのワード。要求オプションが状況照会の場合、可能な戻り値は以下のコードです。複数の状態も可能です。 **X'0008'** ウィンドウが可視状態です。 **X'0010'** ウィンドウは不可視状態です。 **X'0080'** ウィンドウは活動状態です。 **X'0100'** ウィンドウは非活動状態です。 **X'0400'** ウィンドウは最小化されています。 **X'0800'** ウィンドウは最大化されています。
5–6	9–12	ウィンドウ位置の X 座標を含む 16 ビット/32 ビットのワード。(移動オプションが設定されていない場合には無視されます。)
7–8	13–16	ウィンドウ位置の Y 座標を含む 16 ビット/32 ビットのワード。(移動オプションが設定されていない場合には無視されます。)

バイト		定義
9–10	17–20	装置単位での X ウィンドウ・サイズを含む 16 ビット/32 ビットのワード。(サイズ・オプションが設定されていない場合には無視されます。)
11–12	21–24	装置単位での Y ウィンドウ・サイズを含む 16 ビット/32 ビットのワード。(サイズ・オプションが設定されていない場合には無視されます。)
13–16	25–28	ウィンドウの相対配置のためのウィンドウ・ハンドルを 含む 16 ビット/32 ビットのワード。この 2 つのワードは設定オプション専用です。(ZORDER オプションが設定されていない場合には無視されます。) 有効な値は以下のとおりです。 X'00000003' すべての兄弟ウィンドウの前に配置。X'00000004' すべての兄弟ウィンドウの後ろに配置。

要求オプション値が X'03' (拡張状況照会) の場合:

バイト		定義
標準	拡張	
3–4	7–8	要求オプションが 3 (拡張状況照会) の場合には、X'0000' を含む 16 ビット/32 ビットのワード。要求オプションが拡張状況照会の場合、可能な戻り値は以下のコードです。複数の状態も可能です。 **X'0008'** ウィンドウが可視状態です。 **X'0010'** ウィンドウは不可視状態です。 **X'0080'** ウィンドウは活動状態です。 **X'0100'** ウィンドウは非活動状態です。 **X'0400'** ウィンドウは最小化されています。 **X'0800'** ウィンドウは最大化されています。
5–6	9–10	X 方向の現行のフォント・サイズを含む 16 ビット/32 ビットのワード。値はスクリーン・ペル単位です。
7–8	11–12	Y 方向の現行のフォント・サイズを含む 16 ビット/32 ビットのワード。値はスクリーン・ペル単位です。
9–12	13–16	予約済み。この値は常に 0 です。

バイト		定義
13–14	17 ~ 18	表示スペースの最初の可視文字の行番号を 含む 16 ビット/32 ビットのワード。固定サイズ・フォント・オプションが無効で、表示スペースが いくらか隠れるようにウィンドウ・サイズが変更されていない場合、 この値は通常 1 です。
15 ~ 16	19 ~ 20	表示スペースの最初の可視文字の 列番号を含む 16 ビット/32 ビットのワード。
17–20	21–24	セッションの表示スペース・ウィンドウ・ハンドルを 含む 16 ビット/32 ビットのワード。

戻りパラメーター

戻りコード	説明
0	**Window Status** 関数は正常終了しました。
1	表示スペースが正しくないか、または接続されていません。
2	無効なオプションが指定されました。
9	システム・エラーが発生しました。
12	セッションが停止しました。

使用上の注意

論理端末 (LT) のウィンドウは文字セルを使用しています。LT ウィンドウのサイズを再変更する場合、文字セルの切り捨てを防ぐために LT は 数を切り上げます。必要なサイズや位置が要求されたものとわずかに異なることがあります。設定オプションと照会オプションに従い、最終的な Presentation Manager® ウィンドウの位置とサイズを決定してください。すべての x 座標と y 座標の位置とサイズはペル単位です。

Write Structured Fields (127)

3270	*5250*	*VT*
はい	いいえ	いいえ

Write Structured Fields 関数により、アプリケーションは構造化フィールド・データをホスト・アプリケーションに書き込みます。呼び出しで S (同期) を指定した場合、 **Write Structured Fields** 関数が完了するまで、 アプリケーションに制御が戻されません。呼び出しで A (非同期) を指定した場合、 その呼び出しの直後に、アプリケーションに制御が戻されます。呼び出しで M を指定した場合、その呼び出しの直後に、 アプリケーションに制御が戻されます。アプリケーションは、このメッセージを待機します。いずれの場合 (S、A、または M) でも、アプリケーションにより、ホストに対する データが入るバッファー・アドレスが設定されます。

この関数が非同期に正常終了するためには、以下のステートメントを適用します。

パラメーター・リストの戻りコード・フィールドには、要求された入出力 の結果が含まれていない場合もあります。 戻りコードが 0 でない場合、要求は失敗しています。アプリケーションでは、戻りコードに応じて、適切な処置を 行わなければなりません。

この要求に対する戻りコードが 0 である場合、アプリケーションで、この関数呼び出しとともに戻される要求 ID を使用して、**Get Request Completion** 関数呼び出しを発行し、その要求 ID に関連する関数が終了したかどうかを判別します。**Get Request Completion** 関数呼び出しによって、 以下の情報が戻されます。

1. 関数要求 ID
2. 非同期要求からのデータ・ストリング・アドレス
3. データ・ストリングの長さ
4. 完了した関数の戻りコード

前提呼び出し

Connect for Structured Fields (120) **Allocate Communication Buffer** (123)

呼び出しパラメーター

	標準インターフェース	**拡張インターフェース**
関数番号	127 でなければなりません	
データ・ストリング	次の表を参照してください。	
長さ	8、10、14 のいずれかにしてください。	20 でなければなりません
PS 位置	該当なし	

呼び出しデータ・ストリングには次のものがあります。

バイト		**定義**
標準	拡張	
1	1	表示スペースの 1 文字の短縮名 (PSID)。
	2–4	予約済み。
2	5	S、A、M のいずれか **S =** 同期。制御は、読み取りが満たされるまでアプリケーションに戻りません。 **A =** 非同期。制御は、アプリケーション (イベント・オブジェクト待機可能) に即時に 戻されます。 **M =** 非同期。制御は、アプリケーション (メッセージ待機可能) に即時に戻されます。
	6	予約済み。
3–4	7–8	2 バイトの Destination/Origin ID

バイト		定義
5–8	9–12	データが書き込まれる 4 バイトのバッファー・アドレス。**Allocate Communications Buffer** (123) 関数を使用してバッファーを取得する必要があります。
9–10	13–16	予約済み。
11–12	17–20	「M」をバイト位置 5 (16 ビットの場合は 2) に指定した場合、メッセージを受け取るウィンドウのウィンドウ・ハンドルを設定してください。メッセージは、RegisterWindowMessage (「PCSHLL」) の戻り値 (0 以外) です。
13–14		この位置にあるデータは EHLLAPI によって無視されます。ただし、移行プログラム がこの位置にデータを指定しても、エラーは発生しません。このデータはアプリケーションの移行時に互換性を保つために設けられています。

戻りパラメーター

この関数はデータ・ストリングおよび戻りコードを戻します。

データ・ストリング:

A (非同期) をバイト位置 5 (標準インターフェースの場合 は 2) に指定して、関数が正常終了した場合、以下のデータ・ストリングが 戻されます。

バイト		定義
9–10	13–14	2 バイトの関数要求 ID。この関数要求 ID は、この関数呼び出しの完了を判別するために **Get Request Completion** (125) 関数で使用します。
	15 ~ 16	予約済み。
	17–20	EHLLAPI によってイベント・オブジェクト・アドレスが戻される 4 バイト値。アプリケーションは、このイベント・オブジェクトを待ちます。イベント・オブジェクトがクリアされる場合、アプリケーションは **Get Request Completion** (125) 関数呼び出しを発行して、**Write Structured Fields** 要求の結果を取得しなければなりません (32 ビットの場合のみ)。

注: イベント・オブジェクトは、非同期要求が正常終了するたびに 戻されます。イベント・オブジェクトは再使用しないでください。要求ごとに新しいイベント・オブジェクトが戻され、 そのイベント・オブジェクトはその要求の間だけ有効です。

データ・ストリング:

M (非同期メッセージ・モード) をバイト位置 5 (標準 インターフェースの場合は 2) に指定して、関数が正常終了した場合、 以下のデータ・ストリングが戻されます。

バイト		定義
9–10	13–14	2 バイトの関数要求 ID。この関数要求 ID は、この関数呼び出しの完了を判別するために **Get Request Completion** (125) 関数で使用します。
	15 ~ 16	予約済み。
11–12	17 ~ 18	非同期メッセージ・モードのタスク ID。
	19 ~ 20	予約済み。

注: 関数が正常終了した場合、アプリケーション・ウィンドウは メッセージを受け取ります。メッセージは、RegisterWindowMessage(PCSHLL) の戻り値です。wParam パラメーターには、関数呼び出しによって戻されたタスク ID が入ります。lParam パラメーターの HIWORD には戻りコード 0 (関数の 正常終了) が、LOWORD には関数番号 127 が入ります。

戻りコード:

以下のコードが定義されています。

戻りコード	説明
0	**Write Structured Fields** 関数は正常終了しました。
1	無効なホスト表示スペースの短縮セッション ID が指定された か、またはホスト表示スペースが接続されていません。
2	パラメーター指定でエラーが発生しました。
9	システム・エラーが発生しました。
11	リソースが使用できません。(メモリーが使用できません。)
34	インバウンドでホストに送信されたメッセージが取り消されました。
35	ホストからのアウトバウンド送信が取り消されました。
36	要求はリジェクトされました。ホストと接続できません。
37	失敗しました。ホストのインバウンド送信は使用できません。

使用上の注意

1. 戻りコード 35。これは、ホストからのアウトバウンド送信が取り消された後に **Read Structured Fields** または **Write Structured Fields** が要求された場合に戻されます。アプリケーションで修正処置を行う必要があります。
2. 戻りコード 36。この場合、アプリケーションを エミュレーション・プログラムから切断してから再接続して、ホストとの 通信を再確立する必要があります。アプリケーションで修正処置を行う必要があります。
3. 戻りコード 37。これはホスト・インバウンドが使用不可の場合に 戻されます。
4. EHLLAPI では、アプリケーションごとに 20 までの非同期要求を未処理に することができます。20 を超える非同期要求が試行されると、リソースが使用不可であることを 示す戻りコード (RC=11) が戻されます。

構造化フィールド・データの形式は以下のとおりです。

オフセット:	長さ	目次
0	1 ワード	X'0000'
2	1 ワード	m (メッセージの長さ。メッセージ中のデータのバイト数。 この数は 8 バイトを含むメッセージ・ヘッダーの接頭部を除いたものです。) この値はアプリケーションによって設定しなければなりません。
4	1 ワード	X'0000'
6	1 ワード	X'0000'
8	8 バイト	先頭の (または唯一の) 構造化フィールド・メッセージの長さ。
10	1 バイト	構造化フィールド・メッセージの長さ以外の最初のバイト。
		⋮
m+7	1 バイト	構造化フィールド・メッセージの末尾のバイト。

バイト 0 から 7 は、バッファー・ヘッダーです。最初のこれらの 8 バイトは、エミュレーション・プログラムで使用されます。バッファーのユーザー・セクションは、オフセット 8 で開始されます。バイト 8 および 9 には、最初の構造化フィールドのバイト数が含まれています。構造化フィールド・メッセージには複数の構造化フィールドを含めることができ、バイト 8 および 9 の 2 バイトを含みます。バイト 8 から *m* +7 は、ホストに送信される構造化フィールド・メッセージのために使用されます。

同期要求

Write Structured Fields が同期要求 (データ・ストリング中の S オプション) された際には、制御は要求が満たされた後に限りアプリケーションに戻されます。アプリケーションでは以下のことを想定できます。

- 戻りコードが正しい。
- 通信バッファー (読み取りバッファー) 内のデータが正しい。
- ホストが今後 **Write Structured Fields** 要求を処理しない。

非同期要求

Write Structured Fields が非同期要求 (データ・ストリング中の A オプション) された際には、 アプリケーションでは以下のことを想定できません。

- 戻りコードが正しい。
- 通信バッファー (書き込みバッファー) 内のデータが正しい。
- ホストが今後 **Write Structured Fields** 要求を処理しない。

非同期要求がなされた際には、EHLLAPI は以下の値を戻します。

- 16 ビット要求 ID (データ・ストリングのバイト位置 13 ~ 14) (標準インターフェースの場合は 9 ~ 10)
- イベント・オブジェクトのアドレス (データ・ストリングのバイト位置 17 ~ 20)

これらは、非同期の **Write Structured Fields** 呼び出しを完了させるために使用します。

非同期の **Write Structured Fields** 関数呼び出しの結果を判別するためには、次のステップを行わなければなりません。

- EHLLAPI 戻りコードが 0 でない場合、要求は失敗しています。したがって、非同期要求は行われていません。アプリケーションでは、再び呼び出しを試みる前に、適切な処置を 行わなければなりません。
- 戻りコードが 0 ならば、アプリケーションは、**Get Request Completion** (125) 関数を使用してイベント・オブジェクトが送信状態になるまで待たなければなりません。イベント・オブジェクト **Get Request Completion** (125) 関数は再使用しないでください。イベント・オブジェクトは、**Write Structured Fields** 関数呼び出しから **Get Request Completion** (125) 関数呼び出し完了までの間で有効です。
- イベント・オブジェクトが送信状態になったら、**Get Request Completion** (125) 関数に対する呼び出しで要求 ID パラメーターとして戻された 16 ビットの要求 ID を使用してください。**Get Request Completion** (125) 関数呼び出しから戻されたデータ・ストリングには、**Write Structured Fields** 関数呼び出しの最終的な戻りコードが含まれています。

非同期要求

Write Structured Fields が非同期要求 (データ・ストリング中の M オプション) された際には、 アプリケーションで以下のことを想定できません。

- 戻りコードが正しい。
- 通信バッファー (書き込みバッファー) 内のデータが正しい。
- ホストが今後 **Write Structured Fields** 要求を処理しない。

M オプションを指定して非同期要求がなされた際には、EHLLAPI は以下の値を 戻します。

- 16 ビット要求 ID (データ・ストリングのバイト位置 13 ~ 14) (標準インターフェースの場合は 9 ~ 10)
- 非同期メッセージ・モードのタスク ID (バイト位置 17 ~ 18) (標準インターフェースの場合は 11 ~ 12)

これらは、非同期の **Write Structured Fields** 呼び出しを完了させるために使用します。

第 4 章. WinHLLAPI 拡張関数

この章では、WinHLLAPI プログラミング・サポートを使用した時に提供される拡張関数 について説明します。

WinHLLAPI 関数の要約

3270、5250 および VT では、以下の WinHLLAPI 関数が使用可能です。

- Wait (4) (ページ 182)
- Start Host Notification (23) (ページ 183)
- Start Close Intercept (41) (ページ 184)
- Start Keystroke Intercept (50) (ページ 185)
- Send File (90) (ページ 186)
- Receive File (91) (ページ 188)

WinHLLAPI の非同期関数

以下のセクションでは、WinHLLAPI の非同期関数について説明します。

WinHLLAPIAsync

このエントリー・ポイントは、完了するのに長時間を要することの多い WinHLLAPI の 6 つの関数用に使用されます。WinHLLAPIAsync を用いると、関数は非同期的に立ち上げられ、 呼び出しアプリケーションの進行が続行するのを妨げることがありません。これらの関数を以下に示します:**Wait** (04)、**Start Host Notify** (23)、**Start Close Intercept** (41)、**Start Keystroke Intercept** (50)、**Send File** (90)、および **Receive File** (91) であり、 これらについてWinHLLAPI 拡張関数 (ページ 181)で説明します。

HANDLE WinHLLAPIAsync (HWIND hWnd, LPWORD *lpnFunction*, LPBYTE *lpData*, LPWORD *lpnLength*, LPWORD *lpnRetC*)*

パラメーター・リストは、関数番号の前にウィンドウ・ハンドルが必要であ ること以外は WinHLLAPI と同じです。関数は非同期で動作するため、関数の完了は登録済みメッセージによって 通知されます。ウィンドウ・ハンドルは、メッセージのターゲットとして必須です。

WinHLLAPIAsync (90 と 91 以外のすべての関数の場合) と **WinHLLAPIAsyncFileTransfer** (関数 90 と 91 の場合) のストリングを用いた **RegisterWindowsMessage()** の呼び出しを通して、WinHLLAPI アプリケーションによって登録する必要のあるメッセージが 2 つあります。その標準形式は次のとおりです。

WPARAM

元の関数呼び出しによって戻されるタスク・ハンドルが 入ります。

LPARAM

高位ワードにはエラー・コードが入り、低位ワードには 元の関数番号が入ります。

Wait (4)

この関数は、ホスト・セッションが使用禁止状態にあるかどうかを判別します。何らかの理由によって関数が使用禁止状態になっている場合、この関数は、 使用禁止状態の期限が切れた時点またはユーザーの待機期限が切れた時に、 メッセージでアプリケーションに知らせます。待機する時間は、**Set Session Parameters (9)** 関数を用いて設定されます。

前提関数

Connect Presentation Space (1)

WinHLLAPIAsync(*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode*)

呼び出しパラメーター

パラメータ	**説明**
データ・ストリング	該当なし
データのサイズ	該当なし
PS 位置	該当なし

戻りコード

コード	**説明**
WHLLOK	PS は使用禁止が解除されており、入力できます。
WHLLNOTCONNECTED	ユーザーの WinHLLAPI アプリケーションが、有効なホスト・セッションに接続されていません。
WHLLPSBUSY	使用禁止状態である間に関数がタイムアウトになりました。
WHLLNHIBITED	PS は使用禁止になっています。
SHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。
WHLLCANCEL	非同期関数が取り消されました。

注釈

Asynchronous Wait は、PS の使用禁止状態の期限が切れる時点を呼び出し アプリケーションに通知するために使用されます。使用禁止状態の期限が切れると、このバージョンの **Wait** は、**WinHLLAPIAsync** メッセージを、*hWnd* で指定されたウィンドウに通知します。セッション・オプションの **TWAIT**、**LWAIT**、および **NWAIT** は、 この関数が待機する時間の長さに影響を及ぼします。これらのセッション・オプションの詳細については、Set Session Parameters (9) (ページ 145)を参照してください。

注: セッション・パラメーターで **NWAIT** が指定され、アプリケーションが WinHLLAPI の改訂版 1.1 のインプリメンテーションを使用して登録する場合、**WINHLLAPIAsync** 呼び出しは 、**WinHLLAPI** 呼び出しと同じ

働きをし、メッセージは送信しません。改訂版 1.0 が使用されている場合、**Wait** は、PS の使用禁止状況とともに即時にメッセージを戻します。

Start Host Notification (23)

この関数を使用すると、WinHLLAPI アプリケーションに、 ホスト・セッション表示スペース (PS) または オペレーション情報域 (OIA) 内の変更を通知することができます。

前提関数

この関数には前提関数はありません。

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

呼び出しパラメーター

パラメータ	**説明**
データ・ストリング	次の形式を取る 7 バイトのストリング。 **バイト 1** 希望するホスト・セッションの短縮名セッション ID、あるいは 現行のホスト・セッションの場合はスペースまたは NULL。 **バイト 2** 通知モード。"P" (表示スペースの更新のみの場合)、"O" (OIA の更新のみの 場合)、"B" (表示スペースと OIA の両方の更新の場合)。WinHLLAPIAsync の呼び出し時、このバイト位置は "A" にすることができます。 **バイト 3 ~ 6** 使用されません。旧アプリケーションとの互換性を持たせるために用意されています。 **バイト 7** 予約済み、または WinHLLAPIAsync を使用していてバイト 2 が A である場合には、次のいずれかに置き換えられます。P (表示スペースの更新のみの場合)、O (OIA の更新のみの場合)、B (表示スペースと OIA の両方の更新の場合)。
データのサイズ	ホスト・イベント・バッファーの長さ (推奨値は 256)
PS 位置	該当なし

戻りパラメーター

パラメータ	説明
データ・ストリング	呼び出し時のデータ・ストリングと同じです。

戻りコード

コード	説明
WHLLOK	ホスト通知が使用可能になっています。
WHLLNOTCONNECTED	指定されたホスト・セッションが無効です。
WHLLPARAMETERERROR	1 つまたは複数のパラメーターが無効です。
WHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。
WHLLCANCEL	非同期関数が取り消されました。

注釈

ホスト通知は、いったん使用可能になると 、**Stop Host Notification (25)** または **WinHLLAPICancelAsyncRequest()** を呼び出すまで使用可能のままです。この関数は、ホスト通知を開始し、即時に制御をユーザー の Windows HLLAPI アプリケーションに戻します。これにより、ユーザーのアプリケーションは、ホストの更新を待機している間に他のタスクを実行することができます。更新が発生すると、この関数は、*hWnd* によって指定されたウィンドウに登録済みメッセージの **WinHLLAPIAsync** を表示して知らせます。

Start Close Intercept (41)

この関数は、ユーザー要求を代行受信して、Z and I Emulator for Windowsをクローズします。

前提関数

この関数には前提関数はありません。

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

呼び出しパラメーター

パラメータ	説明
データ・ストリング	戻されるセマフォー・アドレス用の 5 バイトのストリング。先頭バイトは、照会を行うセッションのセッション短縮名、もしくは現行セッションの場合はスペースまたは NULL。
データのサイズ	必ず指定してください。
PS 位置	該当なし

戻りパラメーター

パラメータ	説明
データ・ストリング	5 バイトのストリング。形式は次のとおりです。 **バイト 1** セッション短縮名、もしくは現行セッションの場合のスペース または NULL。 **バイト 2 ~ 5** セマフォー・アドレス。

戻りコード

コード	説明
WHLLOK	関数は正常終了しました。
WHLLNOTCONNECTED	無効な表示スペースが指定されました。
WHLLPARAMETERERROR	無効なオプションが指定されました。
WHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。
WHLLCANCEL	非同期関数が取り消されました。

注釈

ホスト通知は、いったん使用可能になると 、**Stop Close Intercept (43)** または **WinHLLAPICancelAsyncRequest ()** を呼び出すまで使用可能のままです。最初から、セマフォーは設定されています。この関数の使用後、ユーザーからのクローズ要求が破棄され、セマフォー は消去されます。

この関数は、クローズ代行受信を開始し、即時に制御を ユーザーの Windows HLLAPI アプリケーションに戻します。これにより、ユーザーのアプリケーションは、クローズ要求を待機している 間に他のタスクを実行することができます。クローズ要求が発生すると、この関数は、*hWnd* によって指定されたウィンドウに登録済みメッセージ **WinHLLAPIAsync** を表示して知らせます。

Start Keystroke Intercept (50)

この関数は、ユーザーによってセッションに送信されたキー・ストロークを 代行受信します。

前提関数

この関数には前提関数はありません。

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

呼び出しパラメーター

パラメータ	説明
データ・ストリング	6 バイトのストリング。形式は次のとおりです。 **バイト 1** セッション短縮名、もしくは現行ホスト・セッションの場合のスペース または NULL。 **バイト 2** キー・ストローク代行受信コード。"D" では、AID キー・ストロークだけが代行受信され、 "L" では、すべてのキー・ストロークが代行受信されます。 **バイト 3 ~ 6** 予約済み
データのサイズ	変数 (推奨値は 256)
PS 位置	該当なし

戻りコード

コード	説明
WHLLOK	キー・ストローク代行受信は開始されています。
WHLLNOTCONNECTED	ホスト・セッションの表示スペースが無効です。
WHLLPARAMETERERROR	1 つ以上のパラメーターが無効です。
WHLLPSBUSY	セッションは使用中です。
WHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。
WHLLCANCEL	非同期関数が取り消されました。

注釈

この関数は、キー・ストローク代行受信を開始し、即時に制御を ユーザーの Windows HLLAPI アプリケーションに戻します。これにより、ユーザーのアプリケーションは、キー・ストロークを待機してい る間に他のタスクを実行することができます。この関数は開始されると、ユーザーが PS にキーを送信するたびに 、*hWnd* によって指定されたウィンドウに **WinHLLAPIAsync** メッセージを表示して知らせます。通知後、代行受信されたキー・ストロークは、通常の EHLLAPI アプリケーション で使用できる任意の方法で処理することができます。キー・ストローク・バッファーのサイズは制限付きであるため、各キー・ストローク は処理してバッファーから除去するようにしてください。

Send File (90)

この関数は、PC からホストにファイルを転送します。

前提関数

この関数には前提関数はありません。

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

呼び出しパラメーター

パラメータ	**説明**
データ・ストリング	SEND コマンド・パラメーター。
データのサイズ	データ・ストリングの長さ。セッション・オプション EOT が指定されている場合は NA。
PS 位置	該当なし

戻りコード

コード	**説明**
WHLLOK	ファイル転送は正常に開始されました。
WHLLPARAMETERERROR	パラメーター・エラー、もしくはデータ長がゼロかまたは 255 より大きくなっています。
WHLLFTXCOMPLETE	ファイル転送が完了しました。
WHLLFTXSEGMENTED	転送は完了し、レコードはセグメント化されました。
WHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。
WHLLTRANSABORTED	ユーザーが「取り消し」ボタンをクリックしたか、あるいはタイムアウト期間が 過ぎたことが原因で、ファイル転送が異常終了しました。
WHLLFILENOTFOUND	PC ファイルが見つかりません。
WHLLFTXCOMPLETECICS	ファイル転送 (CICS への転送) は正常終了しました。
WHLLACCESSDENIED	PC ファイルへのアクセスが拒否されました。
WHLLMEMORY	メモリーが不足しています。
WHLLINVALIDENVIRONMENT	環境が無効です。

注釈

接続済みホスト・セッションごとにサポートされるファイル転送 は 1 つだけです。

この関数は、ファイル転送を開始し、即時に制御を ユーザーの Windows HLLAPI アプリケーションに戻します。これにより、ユーザーのアプリケーションは、ファイル転送が行われている 間に他のタスクを実行することができます。この関数はいったん開始されると 、**WinHLLAPIAsyncFileTransfer** メッセージを *hWnd* で指定されたウィンドウ

に定期的に表示します。これらのメッセージは、WinHLLAPI アプリケーションに転送の状況を 知らせ、転送の完了時には最終メッセージを送信します。

wParm

状況インジケーターです。高位バイトにはセッション ID が入り、低位バイトには状況が入ります。低位バイトがゼロの場合、ファイル転送はまだ進行中です。低位バイトが 1 の場合、ファイル転送は完了しています。

lParm

wParm の低位バイトがゼロ (進行中) の場合、*lParm* は、転送されたバイトの数です。*wParm* の低位バイトが 1 (完了済み) の場合 、*lParm* は完了コードです。

Receive File (91)

この関数は、PC からホストにファイルを転送します。

前提関数

この関数には前提関数はありません。

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

呼び出しパラメーター

パラメータ	**説明**
データ・ストリング	RECEIVE コマンド・パラメーター。
データのサイズ	データ・ストリングの長さ。セッション・オプション EOT が指定されている場合は NA。
PS 位置	該当なし

戻りコード

コード	**説明**
WHLLOK	ファイル転送は正常に開始されました。
WHLLPARAMETERERROR	パラメーター・エラー、もしくはデータ長がゼロかまたは 255 より大きくなっています。
WHLLFTXCOMPLETE	ファイル転送が完了しました。
WHLLFTXSEGMENTED	転送は完了し、レコードはセグメント化されました。
WHLLSYSERROR	システム・エラーが原因で、関数が失敗しました。

コード	説明
WHLLTRANSABORTED	ユーザーが「取り消し」ボタンをクリックしたか、あるいはタイムアウト期間が 過ぎたことが原因で、ファイル転送が異常終了しました。
WHLLFILENOTFOUND	PC ファイルが見つかりません。
WHLLFTXCOMPLETECICS	ファイル転送 (CICS への転送) は正常終了しました。
WHLLACCESSDENIED	PC ファイルへのアクセスが拒否されました。
WHLLMEMORY	メモリーが不足しています。
WHLLINVALIDENVIRONMENT	環境が無効です。

注釈

接続済みホスト・セッションごとにサポートされるファイル転送 は 1 つだけです。

この関数は、ファイル転送を開始し、即時に制御を ユーザーの Windows HLLAPI アプリケーションに戻します。これにより、ユーザーのアプリケーションは、ファイル転送が行われている 間に他のタスクを実行することができます。この関数はいったん開始されると 、**WinHLLAPIAsyncFileTransfer** メッセージを *hWnd* で指定されたウィンドウに定期的に表示します。これらのメッセージは、WinHLLAPI アプリケーションに転送の状況を 知らせ、転送の完了時には最終メッセージを送信します。

wParm

状況インジケーターです。高位バイトにはセッション ID が入り、低位バイトには状況が入ります。低位バイトがゼロの場合、ファイル転送はまだ進行中です。低位バイトが 1 の場合、ファイル転送は完了しています。

lParm

wParm の低位バイトがゼロ (進行中) の場合、*lParm* は、転送されたバイトの数です。*wParm* の低位バイトが 1 (完了済み) の場合 、*lParm* は完了コードです。

WinHLLAPICancelAsyncRequest

この関数は、**WinHLLAPIAsync()** 呼び出しによって立ち上げられた未解決の非同期関数を取り消します。

構文

int WinHLLAPICancelAsyncRequest (HANDLE *hAsyncTask*, WORD *wFunction***)**

パラメーター

hAsyncTask

関数が開始されたときに WinHLLAPIAsync() によって戻されるハンドル。

wFunction

取り消す非同期タスクの関数番号。このパラメーターは改訂版 1.1 では必須ですが 1.0 では必須ではなく、 オプションです。

この関数を使用すると、それ以前に WinHLLAPIAsync() の呼び出しによって 開始された非同期タスクはすべて、未解決の間に取り消されることがあります。

返品

戻り値は、指定された関数が実際に取り消されたかどうかを示します。関数が取り消されている場合、戻り値は WHLLOK (0) となります。未解決の非同期関数が取り消されていない場合、以下のコードの いずれかが戻されます。

WHLLINVALID

hAsyncTask は、有効なタスク・ハンドルではありません。

WHLLALREADY

hAsyncTask によって指定された非同期タスクは既に完了しています。

初期化関数と終了関数

次のセクションでは、WinHLLAPI プログラミング・サポートの初期化関数および終了関数について説明します。

WinHLLAPI Startup

この関数は、WinHLLAPI インプリメンテーションを用いてアプリケーションを登録する場合に使用し、他のどの WinHLLAPI インプリメンテーションの呼び出しよりも前に呼び出すことが必要です。このインプリメンテーションは、WinHLLAPI 仕様のバージョン 1.0 と 1.1 をサポートします。WinHLLAPI アプリケーションは、この関数とのバージョンの互換性 を持つようにする必要があります。

構文

int WinHLLAPIStartup(WORD *wVersionRequired,* LPWHLLAPIDATA *lpData***)**

パラメーター

wVersionRequired

これは、WinHLLAPI アプリケーションが必要とするバージョンです。低位バイトにはメジャー・バージョン番号が入り、高位バイトには マイナー・バージョン (改訂) 番号が入ります。

lpData

これは、インプリメンテーションのバージョン番号と、WinHLLAPI インプリメンテーション のプロバイダーを 記述しているストリングを受け取る WHLLAPIDATA 構造を指すポインターです。WHLLAPIDATA 構造は次のように定義されます。

```
#define  WHLLDESCRIPTION_LEN   127
typedef  struct  tagWHLLAPIDATA
```

```
{
     WORD  wVersion;
     Char  szDescription[WHLLDESCRIPTION_LEN + 1];
}WHLLAPIDATA,  *  PWHLLAPIDATA,  FAR  *LPWHLLAPIDATA;
```

返品

戻り値は、インプリメンテーションによる WinHLLAPI アプリケーションの登録が正常終了したか 失敗したかを示します。登録が正常終了している場合、戻り値は WHLLOK (ゼロ) になります。正常終了していない場合は、次のいずれかになります。

WHLLSYSNOTREADY

基盤となるネットワーク・サブシステムが使用不可であることを示します。

WHLLVERNOTSUPPORTED

このインプリメンテーションが、要求されたバージョンを提供していないことを示します。このインプリメンテーションがサポートするのは、バージョン 1.0 と 1.1 だけです。

WinHLLAPI Cleanup

WinHLLAPI 仕様では、WinHLLAPI アプリケーションがこの関数を使用して WinHLLAPI インプリメンテーションによる登録を解除することをお勧めしています。

構文

BOOL WinHLLAPICleanup()

返品

登録解除が正常終了した場合は、TRUE が戻されます。正常終了しなかった場合は FALSE が戻されます。

ブロッキング・ルーチン

以下のセクションでは、WinHLLAPI プログラミングによってサポートされるブロッキング・ルーチンについて説明します。

注: ブロッキング・ルーチンは WinHLLAPI 準拠のためにサポートされて いますが、これらのルーチンの使用はお勧めしません。非同期処理のためには、WinHLLAPIAsync 関数の使用をお勧めしています。

WinHLLAPIIsBlocking

この関数は、呼び出し WinHLLAPI アプリケーションのスレッドに、 ブロッキング呼び出しの実行が進行中であるかどうかを通知します。ブロッキング呼び出しとは、実行に長い時間がかかり、完了するまで戻らない 同期関数のことです。この WinHLLAPI のインプリメンテーションには、5 つのブロッキング呼び出しがあります。 そのブロッキング呼び出しは、**Get Key (51)**、**Wait (4)**、**Pause (18)**、**Send File (90)**、および **Receive File (91)** です。

構文

BOOL WinHLLAPIIsBlocking()

返品

WinHLLAPI アプリケーションのスレッドがブロッキング呼び出しの 中間にある場合、関数は TRUE を戻し、中間にない場合には FALSE を戻します。

注釈

デフォルトのブロッキング・フックを使用するとブロッキング呼び出し時に メッセージを処理することができるため、再度ブロッキング呼び出しを 呼び出すことが可能です。

WinHLLAPISetBlockingHook

この関数はアプリケーションで定義済みのプロシージャーを、 ブロッキング呼び出しの完了を待機している間に実行するよう設定します。ブロッキング呼び出しとは、実行に長い時間がかかり、完了するまで戻らない 同期関数のことです。この WinHLLAPI のインプリメンテーションには、5 つのブロッキング呼び出しがあります。そのブロッキング呼び出しは、**Get Key (51)**、**Wait (4)**、**Pause (18)**、**Send File (90)**、および **Receive File (91)** です。

構文

FARPROC WinHLLAPISetBlockingHook(FARPROC *lpfnBlockingHook***)**

パラメーター

lpfnBlockingHook

これは、新規ブロッキング・プロシージャーを指すポインターです。

説明

WinHLLAPI インプリメンテーションには、メッセージ・ハンドラーそのもので構成されている デフォルトのブロッキング・プロシージャーがあります。このデフォルトの機構を以下の例に示します。

```
BOOL    DefaultBlockingHook
{
      MSG msg;

      if (PeekMessage (&msg, NULL, 0, 0, xfPM_NOREMOVE))
      {
            if(msg.message = = WM_QUIT)
            {
                return FALSE;
            }
            PeekMessage (&msg, NULL, 0, 0, PM_REMOVE);
            TranslateMessage (&msg);
            DispatchMessage (&msg);
```

```
      }
return TRUE;
}
```

ブロッキング・フックは、スレッドごとに実装されます。この関数によって設定されるブロッキング・フックは、 別の **WinHLLAPISetBlockingHook()** 呼び出しによって置き換えられるまで、もしくは **WinHLLAPIUnhookBlockingHook()** の呼び出しによってデフォルトが復元されるまで、そのスレッドに対して有効のままとなります。

ブロッキング関数は、**WM_QUIT** メッセージを受け取った場合は **FALSE** を戻し、WinHLLAPI がアプリケーションに制御を戻してメッセージを処理し、正常に終了できるようにする必要があります。上記以外の場合には、関数は **TRUE** を戻す必要があります。

返品

この関数は、置き換えられるブロッキング関数を指すポインターを戻します。

WinHLLAPIUnhookBlockingHook

この関数は、呼び出しスレッド用にデフォルトのブロッキング・フックを復元します。

構文

BOOL WinHLLAPIUnhookBlockingHook()

返品

この関数は、デフォルトのブロッキング機構が正常に復元された場合 には TRUE を戻し、そうでない場合には FALSE を戻します。

WinHLLAPICancelBlockingCall

この関数は、現行スレッド内のブロッキング呼び出しの実行を取り消します。ブロッキング呼び出しとは、実行に長い時間がかかり、完了するまで戻らない 同期関数のことです。この WinHLLAPI のインプリメンテーションには、5 つのブロッキング呼び出しがあります。そのブロッキング呼び出しは、**Get Key** (51)、**Wait** (4)、**Pause** (18)、**Send File** (90)、および **Receive File** (91) です。これらのブロッキング呼び出しのいずれかが取り消されると、 その取り消された関数は WHLLCANCEL を戻します。

構文

int WinHLLAPICancelBlockingCall()

返品

戻り値は、指定された関数が実際に取り消されたかどうかを示します。関数が取り消されている場合、 戻り値は WHLLOK (0) となります。未解決のブロッキング関数がない場合は、以下の戻りコードが戻されます。

WHLLINVALID

現在実行されているブロッキング呼び出しがないことを示します。

第 5 章. PCSAPI 関数

Z and I Emulator for Windowsでは、API のセット (本書では *PCSAPI* として定義されています) が提供されています。EHLLAPI はセッションの確立後にワークステーション・アプリケーション・プログラムとホスト・システム間の対話を管理するために使用されますが、 PCSAPI は Z and I Emulator for Windows セッション自体を制御するために使用することができます。

PCSAPI の使用方法

C または C++ で PCSAPI を使用して アプリケーション・プログラムを作成することができます。PCSAPI アプリケーションを開発するには、以下を行ってください。

1. ソース・コードを用意して、適切な PCSAPI 呼び出しを 追加してください。
2. アプリケーションにヘッダー・ ファイル PCSAPI.H を入れてください。
3. ソース・コードをコンパイルしてください。
4. コンパイルしてできた .OBJ ファイルを、適切な オブジェクト・ファイルまたはライブラリーにリンクしてください。

   また、PCSAPI インポート・ライブラリー の PCSCALLS.LIB (16 ビットの場合) および PCSCAL32.LIB (32 ビットの場合) も、 このファイルにリンクしてください。

ページ・レイアウトについての規則

すべての PCSAPI 関数呼び出しは、必要な情報をす早く検索できるように 同一の形式で表されています。書式は以下のとおりです。

- 機能名 (Function Name)
  - 関数の形式
  - パラメーターの形式および説明
  - 戻りコード

関数の形式

「関数の形式」では、関数の形式を次のように示します。

TYPE FunctionName(*TYPE Parameter1, ...***)**

パラメーターの形式および説明

「パラメーターの形式および説明」には、PCSAPI 関数呼び出しで指定する各パラメーターの形式と説明のリストが示されています。

戻りコード

「戻りコード」には、PCSAPI 関数を呼び出した後、呼び出し元プログラムが受け取るコードをリストします。

pcsConnectSession

3270	*5250*	*VT*
はい	はい	はい

pcsConnectSession 関数は、 短縮セッション ID を指定してホスト・セッションとの通信を開始します。ホスト・セッションは、既に開始している必要があります。この呼び出しは、エミュレーター・セッション・パネル上の**「通信」→「接続」**メニュー項目と同等です。

関数の形式

BOOL WINAPI pcsConnectSession(*char cShortSessionID***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

戻りコード

戻りコード	**意味**
TRUE	関数が正常終了しました。
FALSE	次のいずれかを示します。 • セッションが開始されていません。 • 無効なセッション ID が指定されました。 • 呼び出しが失敗しました。

pcsDisconnectSession

3270	*5250*	*VT*
はい	はい	はい

pcsDisconnectSession 関数は、短縮セッション ID を指定してホスト・セッションとの通信を切断します。この関数は、リンクを切断するだけで、セッションを停止するものではあ りません。この呼び出しは、エミュレーター・セッション・パネル上の**「通信」→「切断」**メニュー項目と同等です。

関数の形式

BOOL WINAPI pcsDisconnectSession(*char cShortSessionID***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

戻りコード

戻りコード	意味
TRUE	関数が正常終了しました。
FALSE	次のいずれかを示します。 • セッションが開始されていません。 • 無効なセッション ID が指定されました。 • 呼び出しが失敗しました。

pcsQueryConnectionInfo

3270	*5250*	*VT*
はい	いいえ	いいえ

pcsQueryConnectionInfo 関数は、指定されたホスト・セッションの Telnet 接続についての情報を戻します。戻り情報は、 アプリケーションが提供するバッファーに戻されます。

関数の形式

BOOL WINAPI pcsQueryConnectionInfo(*char cShortSessionID, CONNECTIONINFO *ConnectionInfo***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

CONNECTIONINFO *ConnectionInfo

接続 info データが戻される CONNECTIONINFO 構造へのポインター。

戻りコード

戻りコード	意味
TRUE	関数が正常終了しました。

戻りコード	意味
FALSE	次のいずれかを示します。 • セッションが開始されていません。 • 無効なセッション ID が指定されました。 • 指定されたセッションはこの API (Telnet ではない) 用にサポートされている接続タイプではありませんでした。

ConnectionInfo

CONNECTIONINFO 構造体には、 下記の情報からなるホスト接続についての情報が入れられます。

構造	情報
ホスト名	現在接続されている Telnet ホストの名前を示します。
LU 名	現在割り当てられている LU 名を示します。
ポート番号	接続で使用される予定のホスト・ポート番号を示します。
SSL 標識	セキュア接続を示します (1 = セキュア、 = 非セキュア)。

注: この API は PCSAPI の 32 ビット・バージョンでのみ 有効で、Telnet 接続用にのみ機能します。

例

```
typedef struct_CONNECTIONINFO
{ //Description of a connection @WD06A
   char hostName[63];    //telnet host name      @WD06A
   char reserved[1];     //reserved              @wD06A
   int portNumber;       //host port number      @WD06A
   char luName[17];      //LU name               @WD06A
   char reserved2[3];    //reserved              @WD06A
   BOOL sslIndicator;    //Secure Connection     @WD06A
                           indicator
   char reserved3[256];  //reserved              @WD06A
}CONNECTIONINFO;
```

pcsQueryEmulatorStatus

3270	*5250*	*VT*
はい	はい	はい

pcsQueryEmulatorStatus 関数は、 短縮セッション ID を指定してホスト・セッションの状況を戻します。

関数の形式

ULONG WINAPI pcsQueryEmulatorStatus(*char cShortSessionID***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

戻りコード

戻りコードの値は、ビットごとに処理しなければなりません。 つまり、以下の値のうち 1 つか、または以下の値の論理 OR 演算によって処理す る必要があります。

戻りコード	値	意味
PCS_SESSION_STARTED	0x00000001	指定されたセッションが開始されました。このビットがオフの場合は、指定されたセッションが開始されていないか、 または無効なセッション ID が指定されています。
PCS_SESSION_ONLINE	0x00000002	指定されたセッションがオンライン (接続されている) です。このビットがオフの場合は、指定されたセッションがオフライン (切断されている) です。
PCS_SESSION_API_ENABLED	0x00000004	API (EHLLAPI) は指定されたセッションで使用可能です。このビットがオフの場合は、API はこのセッションで使用不可です。

pcsQuerySessionList

3270	*5250*	*VT*
はい	はい	はい

pcsQuerySessionList 関数は、現在のホスト・セッションすべてのリストを戻します。アプリケーションは PCSAPI.H ファイルに定義されているとおりの SESSINFO 構造体 の配列と、配列内のエレメント数を指定する必要があります。この関数はその構造体に、各セッションに関する情報を入れ、 見付かったセッションの数を戻します。

ホスト・セッションのエレメント数よりも配列のエレメント数の方が少ない場合は、 用意された配列のエレメントだけにデータが入れられます。配列が小さすぎる場合でも、この関数は常に実際のセッションの数を戻します。

配列エレメント数をゼロとしてアプリケーションがこの関数を呼び出して、 存在するセッションの数だけを判別することもできます。それから改めて呼び出しを行って、セッション情報を取得することができます。

関数の形式

ULONG WINAPI pcsQuerySessionList(*ULONG Count, SESSINFO *SessionList***)**

パラメーターの形式および説明

ULONG Count

SessionList 配列内のエレメントの数。

SESSINFO *SessionList

PCSAPI.H に定義されている SESSINFO 構造体の配列へのポインター。

戻りパラメーター

戻りコード

Z and I Emulator for Windows・セッションの合計数。この値は Count パラメーターよりも大きいことも小さいこともあります。

SessionList

SESSINFO 構造体の配列には、ホスト・セッションに関する情報が 入れられます。セッションは任意の順序でリスト内に配置されます。各 SESSINFO 構造体には、以下のフィールドがあります (PCSAPI32.H に 定義されています)。

名前

char とセッション ID (A-Z) を含む ULONG の共用体。Z and I Emulator for Windowsの現在のインプリメンテーションでは、下位バイト (char) のみ が使用され、他のバイトはゼロとして戻されます。

状況

セッションの現在の状況を示すビット・フラグの組み合わせ。フラグ (PCS_SESSION_*) の定義は次の表のとおりです。

状況コードの値は、ビットごとに処理しなければなりません。 つまり、以下の値のうち 1 つか、または以下の値の論理 OR 演算によって処 理する必要があります。

戻りコード	意味
PCS_SESSION_STARTED	セッションは実行中です。このフラグが立っていなければ、その他はすべて未定義です。
PCS_SESSION_ONLINE	セッションはホストへの通信を確立しました (つまり、セッションは接続されています)。
PCS_SESSION_API_ENABLED	セッションではプログラミング API が使用可能です。このフラグが立っていなければ、このセッションには EHLLAPI および ホスト・アクセス・クラス・ライブラリー API は使えません。

例

```
ULONG      NumSessions, i;  // Session counters
SESSINFO   *SessList;       // Array of session information structures
// Find out number of sessions that exist
NumSessions = pcsQuerySessionList (0,NULL);
if (NumSessions == 0) {
   printf("There are no sessions.");
   exit;
}

// Allocate array large enough for all sessions
SessList = (SESSINFO *)malloc(NumSessions * sizeof(SESSINFO));
memset(SessList, 0x00, NumSessions * sizeof(SESSINFO));

// Now read actual session info
pcsQuerySessionList(NumSessions, SessList);

for (i=0; i<NumSessions; i++) {
   if ((SessList[i].Status & PCS_SESSION_STARTED) &&
       (SessList[i].Status & PCS_SESSION_ONLINE))  {

      printf("Session %c is started and connected.",
        SessList[i].Name.ShortName);
   }
}

exit;
```

pcsQueryWorkstationProfile

3270	*5250*	*VT*
はい	はい	はい

pcsQueryWorkstationProfile 関数は、 ホスト・セッションを呼び出すために使用されるワークステーション・プロファイル名を戻します。ホスト・セッションを指定するためには、短縮セッション ID を使用する 必要があります。ワークステーション・プロファイル名は、 アプリケーションが用意する作業バッファーにコピーされます。

関数の形式

BOOL WINAPI pcsQueryWorkstationProfile(*char cShortSessionID, PSZ lpBuffer*)

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

PSZ lpBuffer

NULL 文字で終了するワークステーション・プロファイル名を コピーするための作業バッファー。バッファーは、完全修飾ファイル名を含むことができるだけの大きさで なければなりません。

戻りコード

戻りコード	意味
TRUE	関数が正常終了しました。
FALSE	次のいずれかを示します。 • セッションが開始されていません。 • 無効なセッション ID が指定されました。

pcsSetLinkTimeout

3270	*5250*	*VT*
はい	はい	はい

pcsSetLinkTimeout 関数は、SSCP 所有の Telnet リンクのアイドル・タイムアウトを設定します。この関数は、非 TN 接続、または SSCP 所有状態にない接続に対しては影響を及ぼしません。タイムアウト値がゼロに設定されると、そのリンクはタイムアウトになりません。ゼロ以外に設定された場合、SSCP 所有状態でアイドル状態になってから、 指定された分数が経過すると、タイムアウトになります (切断されます)。

関数の形式

ULONG WINAPI pcsSetLinkTimeout(*char cShortSessionID, USHORT Timeout***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

USHORT Timeout

タイムアウト値は分で指定します。ゼロという値は、タイムアウトの使用不可を意味します。

戻りコード

戻りコード	意味
PCS_SUCCESSFUL	関数が正常終了しました。
PCS_SYSTEM_ERROR	システム・エラーが発生しました。

pcsStartSession

3270	*5250*	*VT*
はい	はい	はい

pcsStartSession 関数は、指定されたワークステーション・プロファイルを使用してホスト・セッションを開始します。短縮セッション ID を指定することも可能です。

関数の形式

ULONG WINAPI pcsStartSession(*PSZ lpProfile, char cShortSessionID, USHORT fuCmdShow*)

パラメーターの形式および説明

PSZ lpProfile

ロードするプロファイルのパスと完全なファイル名。パスは任意指定ですが、完全ファイル名の指定は必須です (.ws は想定されないので、拡張子も指定してください)。

char cShortSessionID

表示スペースの短縮セッション ID。スペースまたは NULL は次に使用可能なセッション ID を示します。

USHORT fuCmdShow

ウィンドウをどのように表示させるかの指定。PCSAPI.H からの次のいずれかの値。

- PCS_HIDE
- PCS_SHOW
- PCS_MINIMIZE
- PCS_MAXIMIZE

戻りコード

戻りコード	値	意味
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。
PCS_USED_ID	2	指定された短縮セッション ID は既に使用されています。
PCS_INVALID_PROFILE	3	指定したワークステーション・プロファイルでエラーが発生したか、 またはウィンドウ・パラメーターが正しくありません。
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

pcsStopSession

3270	*5250*	*VT*
はい	はい	はい

pcsStopSession 関数は、短縮セッション ID によって指定されたホスト・セッションを停止します。

関数の形式

BOOL WINAPI pcsStopSession(*char cShortSessionID, USHORT fuSaveProfile*)

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

USHORT fuSaveProfile

このパラメーターは、次の値のいずれかです。

fuSaveProfile	値	意味
PCS_SAVE_AS_PROFILE	0	現行のプロファイルの指定に従って保管
PCS_SAVE_ON_EXIT	1	終了時にプロファイルを保管。
PCS_NOSAVE_ON_EXIT	2	終了時にプロファイルを保管しない。

戻りコード

戻りコード	意味
TRUE	関数が正常終了しました。
FALSE	次のいずれかを示します。 • セッションが開始されていません。 • 無効なセッション ID が指定されました。

ページ設定関数

このセクションで説明する各種 PCSAPI 関数を使用すると、Z and I Emulator for Windowsのエミュレーター・セッションの「**ページ設定**」の設定値を制御および検索できます。

制約事項

以下の制約事項を順守していない場合は、API が失敗します。戻りコードに、障害の理由が示されます。

- 引数 `cShortSessionID` に指定されるホスト・セッションが PDT モードになっていてはならない。
- API の呼び出し時に、ホスト・セッションが印刷中であってはならな い。
- **「ファイル」→「ページ設定」**ダイアログが使用中であってはならない。

PAGEINFO 構造体の一部のメンバーは、特定のセッション・タイプについてのみ有効、またはサポートされている場合があります。制約事項が指定されていない場合、そのメンバーは以下のセッション・タ イプで有効またはサポートされています。

- 3270 表示装置
- 3270 印刷装置
- 5250 ディスプレイ
- ASCII VT

5250 プリンター・セッションはサポートされていません。

pcsGetPageSettings

3270	*5250*	*VT*
はい	はい	はい

pcsGetPageSettings 関数は、ホスト・セッションのページ設定を検索します (**「ファイル」→「ページ設定」**ダイアログでの設定と同様)。ダイアログの**「テキスト」**タブ内の設定のみサポートされます。

関数の形式

ULONG WINAPI pcsGetPageSettings(*char cShortSessionID, PAGEINFO * const pPageInfo, ULONG * const pErrorInfo***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

PAGEINFO * const pPageInfo

ページ設定が戻される PAGEINFO 構造体へのポインター。

nFlags

構造体の有効なメンバーを示す、ビット・フラグの組み合わせ。これらのフラグを単独で使用す るか、論理和演算で処理することにより、プロパティー・ペ ージ (PCSAPI32.H に定義されています) を復元することができます。各フラグと、対応する構造体内 の有効なメンバーは以下のとおりです。

フラグ

構造体内の有効なメンバー

PCS_PAGE_CPI

nCPI

PCS_PAGE_LPI

nLPI

PCS_PAGE_FACE_NAME

szFaceName

PCS_PAGE_MPL

nMPL

PCS_PAGE_MPP

nMPP

nCPI

1 インチあたりの印刷文字数。

LOWORD は、実際の CPI 値です。

セッションで、フォント CPI が構成されている場合、HIWORD は 1 です。 フォント CPI が構成されていない場合、HIWORD は 0 です。

nLPI

1 インチあたりの印刷行数。

LOWORD は、実際の LPI 値です。

セッションで、フォント LPI が構成されている場合、HIWORD は 1 です。 フォント LPI が構成されていない場合、HIWORD は 0 です。

szFaceName

プリンター・フォントの書体名。これはヌル終了ストリングでなければなりません。

nMPL

1 ページに印刷可能な最大行数。

これは MPL (Maximum Print Lines) ともいいます。サポートされる範囲は、1 から 255 です。

nMPP

1 行に印刷可能な最大文字数。

これは MPP (Maximum Print Position) ともいいます。サポートされる範囲は、1 から 255 です。

ULONG * const pErrorInfo

使用されません。これは呼び出し側によって NULL に設定される必要があります。

戻りコード

戻りコード	値	意味
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。
PCS_INVALID_SESS_TYPE	2	該当のホスト・セッション・タイプではサポートされていません。

戻りコード	値	意味
PCS_DIALOG_IN_USE	3	ホスト・セッションの「ページ設定」ダイアログまたは「プリンター設定」ダイアログが使用中であったため失敗しました。
PCS_PRINTING	4	ホスト・セッションが印刷中であったため、ページ設定を取得できません。
PCS_PDT_MODE	5	ホスト・セッションが PDT モードになっているため、ページ設定を取 得できません。
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

例

```
{
   ULONG Rc = 0;
   PAGEINFO *PageInfo;

   PageInfo = (PAGEINFO *) malloc(sizeof(PAGEINFO));
   memset(PageInfo, 0, sizeof(PAGEINFO));

   PageInfo->nFlags = PCS_PAGE_CPI | PCS_PAGE_LPI | PCS_PAGE_FACE_NAME|
                      PCS_PAGE_MPL | PCS_PAGE_MPP;

   Rc = pcsGetPageSettings('A', PageInfo, NULL);

   if (Rc == PCS_SUCCESSFUL) {
      printf("CPI = %d,
              LPI = %d,
              FaceName = %s,
              MPL = %d,
              MPP = %d\n",
              LOWORD(PageInfo->nCPI),
              LOWORD(PageInfo->nLPI),
              PageInfo->szFaceName,
              PageInfo->nMPL,
              PageInfo->nMPP);

      if (HIWORD(PageInfo->nCPI))
         printf("FontCPI\n");
      else
         printf("No FontCPI\n");

      if (HIWORD(PageInfo->nLPI))
         printf("FontLPI\n");
      else
         printf("No FontLPI\n");

   } else
      printf("Failure. Return code = %d\n", Rc);
   free(PageInfo);
}
```

pcsRestorePageDefaults

3270	*5250*	*VT*
はい	はい	はい

pcsRestorePageDefaults 関数は、nFlags フィールドで定義されている「ページ設定」プロパティー・ページのシステム・デフォルト値を復元します。これは、**「ファイル」→「ページ設定」**ダイアログのプロパティー・ページで**「デフォルト」**をクリックするのと同じです。**「テキスト」**タブ内の設定のみサポートされます。

関数の形式

ULONG WINAPI pcsRestorePageDefaults(*char cShortSessionID, ULONG nFlags***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

ULONG nFlags

次のフラグは、指定されている**「ページ設定」**ダイアログ・プロパティー・ページの名前を記述します。このフラグをビット単位で論理和演算して 、プロパティー・ページ (PCSAPI32.H で定義されています) を復元できます。

PCS_PAGE_TEXT

このフラグは「テキスト」プロパティー・ページを記述します。これは、現在サポートされる唯一のプロパティー・ページです。

戻りコード

戻りコード	**値**	**意味**
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。
PCS_INVALID_SESS_TYPE	2	nFlags パラメーターに、ホスト・セッション・タイプとして無効なオプシ ョンが 1 つ以上含まれています。設定は復元されませんでした。
PCS_DIALOG_IN_USE	3	ホスト・セッションの「ページ設定」ダイアログまたは「プリンター設定」ダイアログが使用中であったため失敗しました。
PCS_PRINTING	4	ホスト・セッションが印刷中であったため、ページ設定を変更できません 。
PCS_PDT_MODE	5	ホスト・セッションが PDT モードになっているため、ページ設定を変更で きません。
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

例

```
{
   ULONG Rc = 0;

   Rc = pcsRestorePageDefaults('A', PCS_PAGE_TEXT);

   if (Rc != PCS_SUCCESSFUL)
      printf("Failure. Return code = %d\n", Rc);
}
```

pcsSetPageSettings

3270	*5250*	*VT*
はい	はい	はい

pcsSetPageSettings 関数は、ホスト・セッション・ページ設定を設定します。これは、**「ファイル」→「ページ設定」**ダイアログで設定を構成するのと同様です。**「テキスト」**タブ内の設定のみサポートされます。

注:

1. CPI、LPI、および FontSize は、ホスト・セッションで構成される FaceName に依存します。この API を使用して、CPI、LPI、FontSize、 および FaceName を一緒に設定する場合は、最初に FaceName を設定してか ら、依存プロパティーを設定してください。
2. この API を使用して、FaceName および依存プロパティーを別個に 呼び出して設定する場合は、最初に FaceName を設定してから、CPI、LPI、お よび FontSize を設定してください。このようにしない場合は、 FaceName を設定するたびに、CPI、LPI、および FontSize を照会し、適切な 値になっていることを確認する必要があります。
3. CPI、LPI、または FontSize を FaceName より前に設定すると、 ホスト・セッション内で CPI、LPI、または FontSize に対して異なる値 が構成される場合があります。これは、CPI、LPI、または FontSize の現行の値が、 FaceName の新しい設定値に対して有効でない場合に起こります。

関数の形式

ULONG WINAPI pcsSetPageSettings(*char cShortSessionID, const PAGEINFO * const pPageInfo, ULONG * const pErrorInfo***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

const PAGEINFO * const pPageInfo

ページ設定値が通知される PAGEINFO 構造体へのポインター。

nFlags

構造体の有効なメンバーを示す、ビット・フラグの組み合わせ。これらのフラグを単独で使用す るか、論理和演算で処理することにより、プロパティー・ペ ージ (PCSAPI32.H に定義されています) を復元することができます。各フラグと、対応する構造体内 の有効なメンバーは以下のとおりです。

フラグ	構造体内の有効なメンバー
PCS_PAGE_CPI	nCPI
PCS_PAGE_LPI	nLPI
PCS_PAGE_FACE_NAME	szFaceName
PCS_PAGE_MPL	nMPL
PCS_PAGE_MPP	nMPP

nCPI

1 インチあたりの印刷文字数。

フォント CPI を選択するには、nCPI の HIWORD を 1 に設定します。 nCPI の LOWORD は無視されます。

特定の CPI 値を選択するには、以下を行ってください。

1. nCPI の HIWORD を 0 に設定します。
2. nCPI の LOWORD を実際の CPI 値に設定します。

nLPI

1 インチあたりの印刷行数。

フォント LPI を選択するには、nLPI の HIWORD を 1 に設定します。nLPI の LOWORD は無視されます。

特定の LPI 値を選択するには、以下を行ってください。

1. nLPI の HIWORD を 0 に設定します。
2. nLPI の LOWORD を実際の LPI 値に設定します。

szFaceName

プリンター・フォントの書体名。これはヌル終了ストリングでなければなりません。

nMPL

1 ページに印刷可能な最大行数。

これは MPL (Maximum Print Lines) ともいいます。サポートされる範囲は、1 から 255 です。

nMPP

1 行に印刷可能な最大文字数。

これは MPP (Maximum Print Position) ともいいます。サポートされる範囲は、1 から 255 です。

ULONG * const pErrorInfo

API が戻りコード PCS_FAILURE で失敗した場合に、詳細なエラー情 報が記載されます。詳細なエラー情報が必要ない場合、このフラグは呼び出し側によって NULL に設定される必要があります。

これはビット・フラグの組み合わせで、正常に設定できない PAGEINFO 構造体のメンバーを記述します。これらのフラグは、PCSAPI32.H で以下のように定義されています。

フラグ

構造体内の有効なメンバー

PCS_PAGE_CPI

nCPI のみが無効です。

PCS_PAGE_LPI

nLPI のみが無効です。

PCS_PAGE_FACE_NAME

szFaceName のみが無効です。

PCS_PAGE_MPL

nMPL のみが無効です。

PCS_PAGE_MPP

nMPP のみが無効です。

戻りコード

戻りコード	値	意味
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。

戻りコード	値	意味
PCS_INVALID_SESS_TYPE	2	該当のホスト・セッション・タイプではサポートされていません。
PCS_DIALOG_IN_USE	3	ホスト・セッションの「ページ設定」ダイアログまたは「プリンター設定」ダイアログが使用中であったため失敗しました。
PCS_PRINTING	4	ホスト・セッションが印刷中であったため、ページ設定を変更できません 。
PCS_PDT_MODE	5	ホスト・セッションが PDT モードになっているため、ページ設定を変更で きません。
PCS_FAILURE	6	ホスト・セッションのページ設定が完全に適用されていません。PAGEINFO 構造体の一部またはすべてのフィールドに、無効なデータが指定されているこ とが原因として考えられます。 pErrorInfo を調べて、適用されていない設定についての詳細を確認して ください。
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

例

```
{
   ULONG Rc = 0, Error = 0;
   PAGEINFO *PageInfo;

   PageInfo = (PAGEINFO *) malloc(sizeof(PAGEINFO));
   memset(PageInfo, 0, sizeof(PAGEINFO));

   PageInfo->nFlags = PCS_PAGE_CPI | PCS_PAGE_LPI |
                      PCS_PAGE_FACE_NAME| PCS_PAGE_MPL |
                      PCS_PAGE_MPP;
   PageInfo->nCPI = MAKELONG(10, 0);
   PageInfo->nLPI = MAKELONG(8, 0);
   PageInfo->nMPL = 40;
   PageInfo->nMPP = 60;
   strcpy(PageInfo->szFaceName, "CourierPS");

   Rc = pcsSetPageSettings('A', PageInfo, &Error);

   if (Rc != PCS_SUCCESSFUL) {
      printf("Failure. Return code = %d\n", Rc);
      printf("Following members could not be set : ");

      if (Rc == PCS_FAILURE) {
         if (Error & PCS_PAGE_CPI) printf(" nCPI");
         if (Error & PCS_PAGE_LPI) printf(" nLPI");
         if (Error & PCS_PAGE_FACE_NAME) printf(" szFaceName");
         if (Error & PCS_PAGE_MPL) printf(" nMPL");
         if (Error & PCS_PAGE_MPP) printf(" nMPP");
         printf("\n");
```

```
      }
   }
   free(PageInfo);
}
```

プリンター設定関数

このセクションで説明する各種 PCSAPI 関数を使用すると、Z and I Emulator for Windowsのエミュレーター・セッションの「**プリンター設定**」の設定値を制御および検索できます。

制約事項

以下の制約事項を順守していない場合、API に障害が起こります。戻りコードに、障害の理由が示されます。

- API の呼び出し時に、ホスト・セッションが印刷中であってはならな い。
- **「ファイル」→「プリンター設定」**ダイアログが使用中であってはならない。

pcsGetPrinterSettings

3270	*5250*	*VT*
はい	はい	はい

pcsGetPrinterSettings 関数は、 ホスト・セッションのプリンター設定値を検索します (**「ファイル」→「プリンター設定」**ダイアログの設定と同様)。

関数の形式

ULONG WINAPI pcsGetPrinterSettings(*char cShortSessionID, PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

PRINTINFO * const pPrintInfo

プリンター設定値が指定される PRINTINFO 構造体へのポインター。

nFlags

0 を設定する必要があります。この値は無視されます。

nBufSize

以下のフィールドに割り振られるバッファーのサイズ。

- lpPDTFile
- lpPrtToDskAppFile

- lpPrtToDskSepFile
- lpPrinterName

単一の API 呼び出しで、複数のメンバーが検索される場合、 呼び出し側は、すべてのバッファーに同じサイズを割り振り、このメンバー にそのサイズを渡す必要があります。

このメンバーが 0 に設定されている場合、フィールドは無視されます。フィールドのバッファーに必要な最大サイズは、nSizeNeeded に戻されま す。

nSizeNeeded

このメンバーの値は、以下のフィールドに関連する条件によって決 まります。

- lpPDTFile
- lpPrtToDskAppFile
- lpPrtToDskSepFile
- lpPrinterName

条件は、以下のとおりです。

- 呼び出し側によって割り振られたバッファーのサイズが、上記にリスト したフィールドを戻すのに十分でない場合、この値は、必要なバイト数になる。
- 上記にリストしたフィールドの複数が呼び出し側によって取得され る場合、この値は、必要バッファーの最大サイズになる。
- nBufSize が呼び出し側によって 0 に設定されている場合、このメン バーには、上にリストしたフィールドのバッファーに必要な最大サイズが含 まれる。

bPromptDialog

以下の値を指定できます。

- TRUE の場合、印刷の前に「プリンター設定」ダイアログが表示されます。
- FALSE の場合、印刷の前に「プリンター設定」ダイ アログは表示されません。

bPDTMode

以下の値を指定できます。

- TRUE の場合、ホスト・セッションは PDT モードになります。
- FALSE の場合、ホスト・セッションは非 PDT モード (GDI モード) になります。

lpPDTFile

呼び出し側がこのメンバーを取得しない場合は、NULL に設定されて いる必要があります。ヌル・ポインターでない場合は、PDT ファイルが戻されます。このメンバーは、呼

び出し側によって割り振られたサイズ nBufSize のバッファーを指している必要があります。

API が戻るとき、このメンバーには以下のいずれかが含まれています。

- セッション PDT ファイルの完全修飾パス名。
- セッションで PDT ファイルが構成されていない場合は、空ストリング ("")。
- バッファー・サイズが十分でない場合は、切り捨てられたファイル名。メンバー nSizeNeeded に、必要なバッファー・サイズが含まれています。

nPrtMode

これは、接続の PrintMode を示す列挙値です。列挙データ型 PRINTMODE は、PCSAPI32.H に定義されています。 nPrtMode 設定値は、以下のいずれかでなければなりません。

- **PrtToDskAppend** (**ディスクへの印刷 - 付加**モード)

  これは、ホスト・セッションの**「プリンター設定」**→**「プリンター」**→**「ディスクへの印刷」**ダイアログで**「付加」**オプションを選択することと同じです。

- **PrtToDskSeparate** (**ディスクへの印刷 - 別個**モード)

  これは、ホスト・セッションの**「プリンター設定」**→ **「プリンター」**→ **「ディスクへの印刷」**ダイアログで**「別個」**オプションを選択することと同じです。

- **WinDefaultPrinter** (**Windows デフォルト・プリンター**・モード)

  これは、ホスト・セッションの**「プリンター設定」**ダイアログで**「Windows デフォルト・プリンターを使用」**オプションを選択することと同じです。

- **SpecificPrinter** (**特定プリンター**・モード)

  これは、ホスト・セッションの**「プリンター設定」**ダイアログで、**「Windows デフォルト・プリンターを使用」**オプションのチェックを外した状態でプリンターを選択するのと同じです。

lpPrtToDskAppFile

呼び出し側がこのメンバーを取得しない場合は、NULL に設定されて いる必要があります。ヌル・ポインターでない場合は、**Print to Disk-Append** ファイルが戻されます。このメンバーは、呼び出し側によって割り振られたサイズ nBufSize のバッファーを指している必要があります。

API が戻るとき、このメンバーには以下のいずれかが含まれています。

- セッションの **Print to Disk-Append** ファイルの完全修飾パス名。
- セッションに **Print to Disk-Append** ファイルが構成されていない場合は、空ストリング ("")。
- バッファー・サイズが十分でない場合は、切り捨てられたファイル名。nSizeNeeded メンバーに、必要なバッファー・サイズが含まれています。

lpPrtToDskSepFile

呼び出し側がこのメンバーを取得しない場合は、NULL に設定されて いる必要があります。ヌル・ポインターでない場合は、**Print to Disk-Separate** ファイルが戻されます。このメンバーは、呼び出し側によって割り振られたサイズ nBufSize のバッファーを指している必要があります。

API が戻るとき、このメンバーには以下のいずれかが含まれています。

- セッションの **Print to Disk-Separate** ファイルの完全修飾パス名。
- セッションで、**Print to Disk-Separate** ファイルが構成されていない場合は、空ストリング ("")。
- バッファー・サイズが十分でない場合は、切り捨てられたファイル名。nSizeNeeded メンバーに、必要なバッファー・サイズが含まれています。

lpPrinterName

呼び出し側がこのメンバーを取得しない場合は、NULL に設定されて いる必要があります。ヌル・ポインターでない場合は、プリンター名が戻され ます。このメンバーは、呼び出し側によって割り振られたサイズ nBufSize のバッファーを指している必要があります。

API が戻るとき、このメンバーには以下のいずれかが含まれています。

- ホスト・セッション nPrtMode が SpecificPrinter である場合は、セ ッションで構成されている特定プリンターの名前。
- ホスト・セッション nPrtMode が WinDefaultPrinter である場合は、 セッションで構成されている Windows デフォルト・プリンターの名前。
- ホスト・セッション nPrtMode が PrtToDskAppend または PrtToDskSeparate である場合は、空ストリング ("")。
- バッファー・サイズが十分でない場合は、切り捨てられたプリンター名。nSizeNeeded メンバーに、必要なバッファー・サイズが含まれています。

プリンター名の形式は次のとおりです。

```
<Printer name> on <Port Name>
```

例:

- `IBM InfoPrint 40 PS on Network Port`
- `HP LaserJet 4050 Series PCL 6 on LPT1`

ULONG * const pErrorInfo

API が戻りコード PCS_FAILURE で失敗した場合に、詳細なエラー情報が記載されます。 エラーの詳細が必要ない場合、pErrorInfo は、呼び出し側によって NULL に設定される必要があります。

以下のセクションで、PCSAPI32.H で定義されているフラグを説明します。

PRINTINFO 構造体の pErrorInfo メンバー用のフラグ

PCS_PRINT_PRINTMODE_ERROR

ホスト・セッションで PrintMode が構成されていません。

PCS_PRINT_PDTFILE_SIZEERR

lpPDTFile のバッファー・サイズが十分でないので、ファイル名が 切り捨てられます。nSizeNeeded メンバーに、PDT ファイルを戻すのに必要 となる実際のバッファー・サイズが含まれています。

PCS_PRINT_DSKAPPFILE_SIZEERR

lpPrtToDskAppFile のバッファー・サイズが十分でないので、ファ イル名が切り捨てられます。nSizeNeeded メンバーに、**Print to Disk-Append** ファイルを戻すのに必要となる実際のバッファー・サイズが含まれています。

PCS_PRINT_DSKSEPFILE_SIZEERR

lpPrtToDskSepFile のバッファー・サイズが十分でないので、ファイル名が 切り捨てられます。nSizeNeeded メンバーに、**Print to Disk-Separate** ファイルを戻すのに必要となる実際のバッファー・サイズが含まれています。

PCS_PRINT_PRINTERNAME_SIZEERR

lpPrinterName のバッファー・サイズが十分でないので、プリンター名が切り捨てられます。nSizeNeeded メンバーに、プリンター名を戻すのに必要となる実際のバッファー・サイズが含まれています。

戻りコード

戻りコード	値	意味
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。
PCS_DIALOG_IN_USE	3	ホスト・セッションの「ページ設定」ダイアログまたは「プリンター設定」ダイアログが使用中であったため失敗しました。
PCS_PRINTING	4	ホスト・セッションが印刷中であったため、プリンター設定を変更できま せんでした。後で、アプリケーションを再試行する必要があります。
PCS_FAILURE	6	一部のプリンター設定を正常に検索できませんでした。pErrorInfo に、検索できなかった設定に関する詳細なエラー情報が含まれています。

戻りコード	値	意味
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

例

```
{
   ULONG Rc = 0, Error=0, Size;
   PRINTINFO *PrintInfo;

   PrintInfo = (PRINTINFO *) malloc(sizeof(PRINTINFO));
   memset(PrintInfo, 0, sizeof(PRINTINFO));

   PrintInfo->nBufSize = 0;

   Rc = pcsGetPrinterSettings('A', PrintInfo, &Error);
   if (Rc != PCS_SUCCESSFUL)
      printf("Failure. Return code = %d\n", Rc);
   else {
      Size = PrintInfo->nSizeNeeded;
      PrintInfo->nBufSize = Size;
      PrintInfo->lpPDTFile = (char *)malloc(sizeof(char) * Size);
      PrintInfo->lpPrtToDskAppFile = (char *)malloc(sizeof(char) * Size);
      PrintInfo->lpPrtToDskSepFile = (char *)malloc(sizeof(char) * Size);
      PrintInfo->lpPrinterName = (char *)malloc(sizeof(char) * Size);
      Rc = pcsGetPrinterSettings('A', PrintInfo, &Error);

      if (Rc != PCS_SUCCESSFUL)
         printf("Failure. Return code = %d, Extended Error = 0x%08x\n", Rc, Error);
      else {
         if (PrintInfo->bPromptDialog)
            printf("PromptDialog\n");
         else
            printf("No PromptDialog\n");
         if (PrintInfo->bPDTMode)
            printf("PDT Mode\n");
         else
            printf("Not PDT Mode\n");

         switch(PrintInfo->nPrtMode) {

         case PrtToDskAppend:
            printf("Print to Disk-Append Mode\n");
            break;
         case PrtToDskSeparate:
            printf("Print to Disk-Separate Mode\n");
            break;
         case SpecificPrinter:
            printf("Specific Printer Mode\n");
            break;
         case WinDefaultPrinter:
            printf("Windows Default Printer Mode\n");
            break;
         }
         if (PrintInfo->lpPDTFile[0] == '\0')
            printf("No PDT File configured\n");
         else
```

```
            printf("PDT File = %s\n", PrintInfo->lpPDTFile);
         if (PrintInfo->lpPrtToDskAppFile[0] == '\0')
            printf("No Disk Append File configured\n");
         else
            printf("DiskAppend File=%s\n", PrintInfo->lpPrtToDskAppFile);
         if (PrintInfo->lpPrtToDskSepFile[0] == '\0')
            printf("No Disk Separate File configured\n");
         else
            printf("DiskSeparate File=%s\n", PrintInfo->lpPrtToDskSepFile);
         if ((PrintInfo->nPrtMode == SpecificPrinter) ||
             (PrintInfo->nPrtMode == WinDefaultPrinter))
            printf("Printer = %s\n", PrintInfo->lpPrinterName);
     }
     free(PrintInfo->lpPDTFile);
     free(PrintInfo->lpPrtToDskAppFile);
     free(PrintInfo->lpPrtToDskSepFile);
     free(PrintInfo->lpPrinterName);
   }
   free(PrintInfo);
}
```

pcsSetPrinterSettings

3270	*5250*	*VT*
はい	はい	はい

pcsSetPrinterSettings 関数は、ホスト・セッションのプリンター設定を制御します (**「ファイル」→「プリンター設定」**ダイアログの設定と同様)。

関数の形式

ULONG WINAPI pcsSetPrinterSettings(*char cShortSessionID, const PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo***)**

パラメーターの形式および説明

char cShortSessionID

表示スペースの短縮セッション ID。

const PRINTINFO * const pPrintInfo

プリンター設定が通知される PRINTINFO 構造体へのポインター。

nFlags

構造体の有効なメンバーを示す、ビット・フラグの組み合わせ。これらのフラグを単独で使用す るか、論理和演算で処理することにより、プロパティー・ペ ージ (PCSAPI32.H に定義されています) を復元することができます。各フラグと、対応する構造体内 の有効なメンバーは以下のとおりです。

フラグ

構造体内の有効なメンバー

PCS_PRINT_PDT

bPDTMode、lpPDTFile

PCS_PRINT_PRINTMODE

nPrtMode、lpPrtToDskAppFile、 lpPrtToDskSepFile、lpPrinterName

PCS_PRINT_PROMPT_DIALOG

bPromptDialog

nBufSize

0 を設定する必要があります。この値は無視されます。

nSizeNeeded

0 を設定する必要があります。この値は無視されます。

bPromptDialog

以下の値を指定できます。

- TRUE の場合、印刷の前に「プリンター設定」ダイアログが表示されます。
- FALSE の場合、印刷の前に「プリンター設定」ダイ アログは表示されません。

bPDTMode

以下の値を指定できます。

- TRUE の場合、接続は PDT モードに設定されます。
- FALSE の場合、接続は非 PDT モード (GDI モード) に設定されます。

lpPDTFile

bPDTMode が TRUE に設定されている場合にのみ使用されます。bPDTMode が FALSE に設定されている場合は、無視されます。

これは、PDT ファイル名を含むヌル終了ストリングで、以下のいずれかで ある必要があります。

- NULL

  現在、接続で構成されている PDT ファイルが使用されます。接続で既に構成されている PDT ファイルがない場合、API は失敗して例外を出 します。

- ファイル名 (パスなし)

  Z and I Emulator for Windows のインストール・パスの PDFPDT サブフォルダー内の lpPDTFile が使用されます。

- ファイルの完全修飾パス名

  lpPDTFile が存在しない場合、API は失敗します。

nPrtMode

これは、接続の PrintMode を示す列挙値です。列挙データ型 PRINTMODE は、PCSAPI32.H に定義されています。 nPrtMode 設定値は、以下のいずれかでなければなりません。

- **PrtToDskAppend** (**ディスクへの印刷 - 付加**モード)

  これは、ホスト・セッションの**「プリンター設定」**→**「プリンター」**→**「ディスクへの印刷」**ダイアログで**「付加」**オプションを選択することと同じです。

- **PrtToDskSeparate** (**ディスクへの印刷 - 別個**モード)

  これは、ホスト・セッションの**「プリンター設定」**→**「プリンター」**→**「ディスクへの印刷」**ダイアログで**「別個」**オプションを選択することと同じです。

- **WinDefaultPrinter** (**Windows デフォルト・プリンター**・モード)

  これは、ホスト・セッションの**「プリンター設定」**ダイアログで**「Windows デフォルト・プリンターを使用」**オプションを選択することと同じです。

- **SpecificPrinter** (**特定プリンター**・モード)

  これは、ホスト・セッションの**「プリンター設定」**ダイアログで**「Windows デフォルト・プリンターを使用」**オプションのチェックを外した状態でプリンターを選択するのと同じです。

lpPrtToDskAppFile

これは、nPrtMode が PrtToDskAppend に設定されている場合にのみ使用されます。

これは、**Print to Disk-Append** ファイル名を含むヌル終了ストリングで、以下のいずれかである必要があります。

- NULL

  現在、接続で PrtToDskAppend モード用に構成されているファイルが使用 されます。接続で既に構成されている PDT ファイルがない場合、 API は失敗します。

- ファイル名 (パスなし)

  ユーザー・クラスのアプリケーション・データ・ディレクトリー・パスを 使用してファイルの場所を検索します。ファイルが存在する場合には、それが使用されます。それ以外の場合、 印刷の完了時に作成されます。

- ファイルの完全修飾パス名

  パスにディレクトリーが存在している必要があります。存在していない場合、API は失敗します。ファイルがパス内に存在する必要はありません。

lpPrtToDskSepFile

指定できる値は以下のとおりです。

- セッションの **Print to Disk-Separate** ファイルの完全修飾パス名。
- セッションで、**Print to Disk-Separate** ファイルが構成されていない場合は、空ストリング ("")。
- バッファー・サイズが十分でない場合は、切り捨てられたファイル名。nSizeNeeded メンバーに、必要なバッファー・サイズが含まれています。

lpPrinterName

これは、nPrtMode が SpecificPrinter に設定されている場合にのみ使用されます。そうでない場合は無視されます。これは、プリンター名を含むヌル終了ストリングです。プリンターが存在しない場合、このメンバーは失敗します。

プリンター名の形式は次のとおりです。

```
<Printer name> on <Port Name>
```

例:

- `IBM InfoPrint 40 PS on Network Port`
- `HP LaserJet 4050 Series PCL 6 on LPT1`

ULONG * const pErrorInfo

API が戻りコード PCS_FAILURE で失敗した場合に、詳細なエラー情報が記載されます。 エラーの詳細が必要ない場合、pErrorInfo は、呼び出し側によって NULL に設定される必要があります。

以下のセクションで、PCSAPI32.H で定義されているフラグを説明します。

PRINTINFO 構造体の pErrorInfo メンバー用のフラグ

PCS_PRINT_PDTMODE_ERROR

以下のいずれかの理由により発生します。

- bPDTMode が TRUE、lpPDTFile が NULL に設定されている場合に、 ホスト・セッション用に構成済みの PDT ファイルがない。
- nPrtMode が PrtToDskAppend または PrtToDskSeparate に設定され、 PCS_PRINT_PDT が nFlags 内で設定されていない場合に、ホスト・セッション が PDT モードになっていない。
- nPrtMode が PrtToDskAppend または PrtToDskSeparate に設定され、 bPDTMode が FALSE に設定されている。

PCS_PRINT_PDTFILE_ERROR

lpPDTFile で指定されているファイルまたはパスが見つかりませんでした。

PCS_PRINT_PRTTODSK_FILE_ERROR

以下のいずれかの理由により発生します。

- lpPrtToDskAppFile フィールドまたは lpPrtToDskSepFile フィールドで指定されているフォルダーが存在していないか、そのフォルダ ーに書き込みアクセス権限がない。
- lpPrtToDskSepFile フィールドに拡張子が指定されている。

PCS_PRINT_PRINTMODE_ERROR

nPrtMode が正常に設定できません。これは、以下のいずれかの理由により起こります。

- nPrtMode の値が、PRINTMODE 列挙データ型の列挙型定数の 1 つでない。
- nPrtMode が PrtToDskAppend に、lpPrtToDskAppFile が NULL に設定されている場合に、ホスト・セッションに構成済みの **Print to Disk-Append** ファイルがない。
- nPrtMode が PrtToDskSeparate に、lpPrtToDskSepFile が NULL に設定されている場合に、ホスト・セッションに構成済みの **Print to Disk-Separate** ファイルがない。
- nPrtMode が SpecificPrinter に設定されている場合に、lpPrinterName フィールドで指定されているプリンターが見つからない。
- nPrtMode が WinDefaultPrinter に設定されている場合に、デフォルトの Windows® プリンターがシステムに構成されていない。
- bPDTMode が FALSE に設定されており、PCS_PRINT_PRINTMODE が nFlags に 設定されていないが、ホスト・セッション PrintMode が PrtToDskAppend または PrtToDskSeparate である。

戻りコード

戻りコード	値	意味
PCS_SUCCESSFUL	0	関数が正常終了しました。
PCS_INVALID_ID	1	無効なセッション ID が指定されました。
PCS_DIALOG_IN_USE	3	ホスト・セッションの「ページ設定」ダイアログまたは「プリンター設定」ダイアログが使用中であったため失敗しました。
PCS_PRINTING	4	ホスト・セッションが印刷中であったため、プリンター設定を変更できま せんでした。後で、アプリケーションを再試行する必要があります。
PCS_FAILURE	6	ホスト・セッションのプリンター設定が適用されていません。PRINTINFO 構造体の一部またはすべてのフィールドに、無効なデータが指定されていることが原因として考えられます。pErrorInfo にエラーの詳細が記述されています。
PCS_SYSTEM_ERROR	9	システム・エラーが発生しました。

例

```
{
   ULONG Rc = 0, Error=0;
   PRINTINFO *PrintInfo;
   char PDTFile[] = "epson.pdt";
   char SepFile[] = "DiskSep";

   PrintInfo = (PRINTINFO *) malloc(sizeof(PRINTINFO));
   memset(PrintInfo, 0, sizeof(PRINTINFO));

   PrintInfo->nFlags = PCS_PRINT_PDT | PCS_PRINT_PRINTMODE |
         PCS_PRINT_PROMPT_DIALOG;
   PrintInfo->nBufSize = 0;
   PrintInfo->nSizeNeeded = 0;
   PrintInfo->bPDTMode = TRUE;
   PrintInfo->lpPDTFile =
         (char *)malloc(sizeof(char) * (strlen(PDTFile)+1));
   strcpy(PrintInfo->lpPDTFile, PDTFile);
   PrintInfo->nPrtMode = PrtToDskSeparate;
   PrintInfo->lpPrtToDskSepFile =
         (char *)malloc(sizeof(char) * (strlen(SepFile)+1));
   strcpy(PrintInfo->lpPrtToDskSepFile, SepFile);
   PrintInfo->bPromptDialog = TRUE;
   Rc = pcsSetPrinterSettings('A', PrintInfo, &Error);
   if (Rc != PCS_SUCCESSFUL)
      printf("Failure. Return code = %d, Extended Error = 0x%08x\n", Rc, Error);
   free(PrintInfo->lpPDTFile);
   free(PrintInfo->lpPrtToDskSepFile);
   free(PrintInfo);
}
```

第 6 章. エミュレーター・プログラミングのためのトラブルシューティング

問題のトラブルシューティングを支援する以下のセルフヘルプ情報リソースとツールを使用できます。

- ご使用の製品のリリース情報で、既知の問題、次善策、およびトラブルシューティング情報を参照します。
- お客さまの問題を解決するダウンロードまたはフィックスが提供されているどうかを確認する。
- 提供されているナレッジ・ベースを検索して、問題の解決方法が既に 文書化されているかどうかを調べる。
- さらに支援が必要な場合は、HCL ソフトウェア・サポートに連絡して、問題を報告してください。

Z and I Emulator for Windows ホスト画面での部分的な EHLLAPI 入力

問題

HCL Z and I Emulator for Windowsの使用時に、 切り捨てられたコマンド・テキストがホストに送信された。

原因

EHLLAPI アプリケーションが SYSREQ キーをホストに送信し、 その後にホスト画面にコマンドを入力しようとした場合、そのコマンドの一部だけが ホストに送られる場合があります。この問題の原因は、Z and I Emulator for Windowsのホスト側での SYSREQ 処理と、EHLLAPI アプリケーションからのコマンド入力との同期が取れていないことです。

アプリケーションからホストに SYSREQ コマンドが送信されると、以下の状況が発生します。

- SSCP-LU セッションにいることを示すように、OIA が更新される。
- Z and I Emulator for Windows セッションが AO コマンド (SYSREQ) を 3270 ホストに送信する。

ホストは、SYSREQ を受信するとすぐに、0x15 コード、つまり NL (NewLine) で Z and I Emulator for Windows に応答します。Z and I Emulator for Windows は、 この NL コマンドについて、行の残りの部分をヌルで埋め、 カーソルを次行の先頭に移動するという処理を行います。

Z and I Emulator for Windows セッションがホストから NL コマンドを受信して処理を終える前であるのに、EHLLAPI アプリケーションがホスト画面に (SendKeys 機能を介して) さまざまなコマンドを入力し続けると、問題が起こります。結果として、 入力コマンドの一部がまず画面に入力され、その間に NL コマンドが処理されて カーソルが次の行に移動します。次に、コマンドの残りの部分が次行で入力されます。したがって、 切り捨てられた 2 番目のコマンド部分のみがホストに送信されることになり、誤った結果となります。

解決方法

この問題の解決策は、EHLLAPI アプリケーションに、NL コマンドの受信と処理の完了を待たせ、 その後でホスト画面へのコマンドの入力を継続させるようにすることです。SYSREQ のホスト応答が処理されたことがセッションから EHLLAPI アプリケーションに通知されると、その EHLLAPI アプリケーションは入力を 続行できます (これは、この時点でセッションが新規入力を受け入れるための正しい状態になっているためです) 。これを実現するには、次の EHLLAPI 関数呼び出しを使用します。

```
Start_Host_Notification (23)
Pause (18)
Set_Session_Parameters (9)
Query_Host_Update (24).
```

EHLLAPI アプリケーション中のコードには、以下のようなものがあり得ます。

- Sendkeys(@A@H) の呼び出し。これは、SYSREQ コマンドをセッションに送信します。
- B を指定した StartHostNotify の呼び出し。 ここで、 B は OIA と PS の両方の通知を指示します。これは、セッションの OIA または PS、あるいはその両方が ホストによって更新されたときに EHLLAPI アプリケーションに通知するよう、セッションに指示します。
- 十分なタイムアウト期間を指定した Pause の呼び出し。これにより、EHLLAPI アプリケーションは、 セッションの OIA または PS、あるいはその両方がホストによって更新されたと セッションから通知されるまで待つようになります。これは、 SYSREQ コマンドに対する、最も長く待たれているホスト応答を、 セッションが受け取るときに起こります。タイムアウト値を超えた場合で、 ホスト通知をまだ何も受信していない場合でも、Pause 機能呼び出しは戻ることに注意してください。

また、この Pause 呼び出しが機能するためには、Set_Session_Parameters (9) 機能呼び出し を使用して IPAUSE オプションを使用可能化する必要があります。これが必要な理由は、 これによって Pause API 呼び出しは、 ホストが OIA または PS、あるいはその両方の更新をセッションに通知したときには戻るようにと指示されるからです。

OIA/PS 更新 (ホスト通知) があったために Pause が戻った場合、 その戻りコードは 26 です。この場合、ホスト・コマンドを送信できる状態になっています。これ以外の場合、 もう一度ホスト応答を待つ必要があります。

EHLLAPI アプリケーションは、OIA または表示スペース (あるいはその両方) がホストによって更新されたことを知ると、 コマンドを継続できます。更新内容を検査するために、QueryHostUpdate が使用されます。更新内容を検査するために、QueryHostUpdate が使用されます。 つまり、OIA のみが更新された (戻りコード 21) のか、PS のみが更新された (戻りコード 22) のか、 それとも OIA と PS の両方が更新された (戻りコード 23) のかの検査です。

例えば、EHLLAPI コードは次の部分のようになります。

```
Send Keys(@A@H) /* Send SYSREQ command to the host */

Start Host Notification with 'B' in byte 2 /* Enable notification to EHLLAPI application
                                              when session's OIA and/or PS are updated */
```

```
Set Session Parms with IPAUSE option /* Allow Pause to be interrupted */

Label WW:

Pause for 15 seconds /* 15 secs is a sample time-out value */

retVal = Query Host Update /* Store return value of QueryHostUpdate() into retVal */

If (retVal = 21 or 22 or 23) /* OIA and/or PS was updated */

Send Keys("Your Input Command to host") /* Send input command to host */

else

goto (Label WW)

Stop Host Notification /* Disable host notification */
```

上記コードは、この問題の最適の解決策となります。 この EHLLAPI アプリケーションは、セッションが SYSREQ ホスト応答を受信して処理できるのに必要最低限の時間だけ待ち、 その後にコマンド入力を送信します。

別の解決策は、EHLLAPI アプリケーションで、SYSREQ コマンドと それに続くコマンドとの間に遅延 [例えば、Sleep(1000)] を追加することです。 これにより、セッションはホスト応答を受信して処理するのに十分な 時間を持つことになります。ただし、この解決策は、最適のものではありません。 なぜなら、遅延が短すぎたり、長すぎたりする可能性があるからです。

3270 SYSREQ 機能について 詳しくは、RFC 2355 (TN3270 拡張) を参照してください。

HCL Z and I Emulator for Windows の VBHLLAPI サンプルが FDCC Windows Vista で実行されない

問題

HCLZ and I Emulator for Windows の VBHLLAPI サンプルでは、comdlg32.ocx に備わっているコントロールが使用されますが、 この comdlg32.ocx が Microsoft Windows Vista の Federal Desktop Core Configuration (FDCC) にインストールされていません。

原因

VBHLLAPI では、Microsoft comdlg32.ocx モジュールに備わっている ActiveX コントロールおよび共通ダイアログ・コントロールが使用されます。セキュリティー上の理由により、Windows Vista の FDCC には、この特定モジュールが含まれていません。

解決方法

FDCC バージョンの Windows Vista はカスタマイズされています。 変更は推奨されません。

VBHLLAPI が含まれる HLLAPI サンプルを実行する必要がある場合は、comdlg32.ocx モジュールを標準の Windows Vista マシンから FDCC Windows Vista インストール済み環境の `\Windows\System32\` ディレクトリーにコピーする必要があります。

その後で、システムをリブートして、変更を有効にします。

付録 A. がサポートする Query Reply データ構造EHLLAPI

この付録では、EHLLAPI 構造化フィールド・インターフェースが PC/3270用にサポートする Query Reply 構造を列挙し、定義します。「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。 また、IBM ライセンス・プログラムの場合は、その特定のライセンス・プログラム用の資料を参照してください。

注:

1. EHLLAPI は、Query Reply バッファーを走査し、Destination (宛先)/Origin (発信元) ID (DOID) の 自己定義パラメーター (SDP) を見つけることにより、構造化フィールド・サポートが 確実に機能するようにします。これにより、DOID フィールドが割り当て ID で埋め込まれます。
2. アプリケーションは、Query Reply データ構造をアプリケーションの 専用メモリー内に作成する必要があります。
3. Query Reply のデータについて簡単なチェックが実行されます。ID と 構造の長さについてのみ有効かどうかチェックされます。
4. 各 Query Reply の先頭の長さ 2 バイトのフィールドは、**バイト反転していません。**
5. ホスト・セッションごとに許可される分散データ管理 (DDM) の 基本タイプは 1 つだけです。DDM 接続が DOID 用に SDP をサポートしている場合は、複数の接続が可能です。
6. アプリケーションが選択済みのセッションと既に接続していることを示すゼロ以外の戻りコードを受信した (RC 32 または 39) 場合、 表示スペースを使用する際には注意が必要です。ファイル転送、および他の EHLLAPI アプリケーションとの競合が生じる可能性があります。

DDM Query Reply

何種類かの DDM Query Reply 形式がサポートされます。形式の一部を次に示します。

表 15. DDM Query Reply の基本形式

オフセット:	長さ	コンテンツ	意味
0	1 ワード	長さ	構造体の長さ
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'95'	Query reply のタイプ
4–5	2 バイト	FLAGS	予約済み
6–7	2 バイト	LIMIN	インバウンド伝送できる DDM バイトの最大数
8–9	2 バイト	LIMOUT	アウトバウンド伝送できる DDM バイトの最大数
10	1 バイト	NSS	サブセットの ID 数
11	1 バイト	DDMSS	DDM サブセットの ID

DDM アプリケーション名の自己定義パラメーター

DDM アプリケーション名の自己定義パラメーターは、DDM 補助装置の制御を含むアプリケーションの名前をホスト・アプリケーションに提供します。アプリケーションの制御は、直接アクセス自己定義パラメーター内の DOID で識別されます。

自己定義パラメーター (SDP) は任意指定ですが、複数のアプリケーションが リモート・ワークステーションに存在し、 ホスト・アプリケーションが DDM 補助装置を明白に識別する必要がある場合には 必須です。

表 16. DDM アプリケーション名の自己定義パラメーター

オフセット:	長さ	コンテンツ	意味
0	1 バイト	長さ	パラメーター長
1	1 バイト	X'02'	DDM アプリケーション名
2–n	n-2 バイト	NAME	リモート・アプリケーション・プログラムの名前

NAME

この名前は 8 文字以下で構成され、 この名前によってホスト・アプリケーションとリモート・ワークステーションのアプリケーションを対応付けることができます。ホストとリモートのアプリケーション・ユーザーは、その名前が相手側のアプリケーションに認識されていることを確認する必要があります。

PCLK プロトコル制御の自己定義パラメーター

PCLK プロトコル制御の自己定義パラメーターは、PCLK プロトコル制御の構造化フィールド ID = X'1013' が、DDM 補助装置プロセッサーの宛先になったり発信元になったりするインバウンドとアウトバウンドの両方のデータ・ストリームで使用できることを示します。

表 17. DDM PCLK 補助装置 SDP

オフセット:	長さ	コンテンツ	意味
0	1 バイト	X'04'	パラメーター長
1	1 バイト	X'03'	PCLK プロトコル制御
2–3	2 バイト	VERS	プロトコル・バージョン

VERS

VERS に指定された値は、Query Reply が戻されるときに端末に インストールされる PCLK のバージョンを示すために使用されます。例えば、X'0001' は PCLK バージョン 1.1 を示します。

この Query Reply に対するフィールド定義については、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

基本 DDM Query Reply の形式

次の Query Reply の形式は、基本 (Base) + SDP の (自己定義パラメーター) 組み合わせの一部の例です。すべての組み合わせを示しているわけではありません。

表 18. 名前および直接アクセス SDP での基本 DDM Query Reply の形式

オフセット:	長さ	コンテンツ	意味
0	1 ワード	長さ	構造体の長さ (SDP を含む)
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'95'	Query Reply のタイプ
4–5	2 バイト	FLAGS	予約済み
6–7	2 バイト	LIMIN	インバウンド伝送できる DDM バイトの最大数
8–9	2 バイト	LIMOUT	アウトバウンド伝送できる DDM バイトの最大数
10	1 バイト	NSS	サポートされるサブセットの数
11	1 バイト	DDMSS	DDM サブセットの ID
12	1 バイト	長さ (n+2)	パラメーター長
13	1 バイト	X'02'	DDM アプリケーション名
14– (13+n)	n バイト	名前	リモート・アプリケーション・プログラムの名前
14+n	1 バイト	X'04'	パラメーター長
15+n	1 バイト	X'01'	直接アクセス ID
16+n – 17+n	2 バイト	DOID	サブシステムが割り当てた Destination/Origin ID

自己定義パラメーターはオフセット 12 および (14 + *n*) で始まります。ここで、*n* はオフセット 14 で指定されるアプリケーションの名前の長さです。

この Query Reply に対するフィールド定義については、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

表 19. 直接アクセスおよび名前 SDP での基本 DDM Query Reply の形式

オフセット:	長さ	コンテンツ	意味
0	1 ワード	長さ	構造体の長さ (SDP を含む)
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'95'	Query reply のタイプ
4–5	2 バイト	FLAGS	予約済み
6–7	2 バイト	LIMIN	インバウンド伝送できる DDM バイトの最大数

表 19. 直接アクセスおよび名前 SDP での基本 DDM Query Reply の形式 (続く)

オフセット:	長さ	コンテンツ	意味
8–9	2 バイト	LIMOUT	アウトバウンド伝送できる DDM バイトの最大数
10	1 バイト	NSS	サポートされるサブセットの数
11	1 バイト	DDMSS	DDM サブセットの ID
12	1 バイト	X'04'	パラメーター長
13	1 バイト	X'01'	直接アクセス ID
14 ~ 15	2 バイト	DOID	サブシステムが割り当てた Destination/Origin ID
16	1 バイト	長さ (n+2)	パラメーター長
17	1 バイト	X'02'	DDM アプリケーション名
16+*n* – 17+*n*	*n* バイト	名前	リモート・アプリケーション・プログラムの名前

自己定義パラメーター (SDP) はオフセット 12 および 16 で始まります。

この Query Reply に対するフィールド定義については、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

IBM 補助装置 Query Reply

補助装置 Query Reply は、IBM が定義するデータ・ストリームを使用する IBM 補助装置のサポートをホスト・アプリケーションに指示するために使用されます。 詳しくは、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

機能がサポートされると、Query Reply は Query または Query List (QCODE リスト = X'9E'、等価のもの、またはすべて) を指定する Read Partition 構造化フィールドに応答してインバウンド伝送されます。

ワークステーションが複数の補助装置をサポートしている場合 、IBM 補助装置 Query Reply は各装置用に送信されます。

任意 指定パラメーター

Query Reply の基本部分に示されるパラメーターはすべて必須です。使用されていないパラメーターは X'00' に設定されます。

自己定義パラメーターは最低 1 つ必要です。

表 20. 直接アクセス SDP での IBM 補助装置の基本形式

オフセット:	長さ	コンテンツ	意味
0–1	1 ワード	長さ	構造体の長さ (SDP を含む)

表 20. 直接アクセス SDP での IBM 補助装置の基本形式 (続く)

オフセット:	長さ	コンテンツ	意味
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'9E'	IBM 補助装置 Reply
4	1 バイト　ビット 0　1–7	FLAGS　QUERY B'1'　RES	予約済み 読み取り部分 (Query, Query List) 補助装置が Query をサポートする　予約済み、B'0' でなければなりません
5	1 バイト	FLAGS	予約済み
6–7	2 バイト	LIMIN	インバウンド伝送できる DDM バイトの最大数
8–9	2 バイト	LIMOUT	アウトバウンド伝送できる DDM バイトの最大数
10	1 バイト	TYPE X'01' X'02' その他	サポートされる補助装置のタイプ IBM 補助装置ディスプレイ IBM 補助装置プリンター 予約済み
11	1 バイト	X'04'	パラメーター長
12	1 バイト	X'01'	直接アクセス
13–14	1 ワード	DOID	サブシステムが割り当てた Destination/Origin ID

QUERY	IBM 補助装置が Read Partition (Query、Query List) の受信をサポートしていることを示すには、このビットを B'1' に設定する必要があります。これにより、ホスト・アプリケーションは補助装置にあてた Read Partition を 使用して、特性を判別できます。Destination/Origin 構造化フィールドは、Read Partition 構造化フィールドを 補助装置にあてるのに使用されます。 IBM 補助装置に対するサポートを最小レベルにすると 、Read Partition に応答して NULL の Query Reply を戻します。
LIMIN	インバウンド伝送で送信できる最大バイト数を示します。LIMIN の値が X'0000' の場合は、インバウンド伝送されるバイト数に対して使用制限がないことを示しています。
LIMOUT	アウトバウンド伝送で IBM 補助装置に送信できる最大バイト数を示します。LIMOUT の値が X'0000' の場合は、アウトバウンド伝送されるバイト数に対して使用制限がないことを示します。
タイプ	サポートされる補助装置を識別します。2 つの値が有効です。1 つは補助ディスプレイを識別し、もう 1 つは補助プリンターを識別します。その他の値は予約済みです。

IBM 補助装置プロセッサーは 2 つの自己定義パラメーター (01 および 03) をサポートします。これらのパラメーターは表 21 : IBM 補助装置直接アクセス SDP (ページ 234) で定義されています。

直接アクセス自己定義パラメーター

直接アクセス自己定義パラメーターは 、IBM 補助装置の直接アクセスで Destination/Origin 構造化フィールドに使用する ID を提供します。

この SDP は、基本 Query Reply とともに必ず必要です。

表 21. IBM 補助装置直接アクセス SDP

オフセット:	長さ	コンテンツ	意味
0	1 バイト	X'04'	パラメーター長
1	1 バイト	X'01'	直接アクセス ID
2–3	2 バイト	DOID	Destination/Origin ID

DOID

これらのバイト内の値は 、Destination/Origin 構造化フィールドの ID フィールド内で使用され、 補助装置が後に続くデータの宛先かまたは発信元かを識別します。

PCLK プロトコル制御の自己定義パラメーター

PCLK プロトコル制御の自己定義パラメーターがある場合、PCLK プロトコル制御の構造化フィールド ID = X'1013' が、IBM 補助装置プロセッサーの宛先になったり発信元になったりするインバウンドとアウトバウンドの両方のデータ・ストリームで使用できることを示します。

表 22. IBM 補助装置 PCLK SDP

オフセット:	長さ	コンテンツ	意味
0	1 バイト	X'04'	パラメーター長
1	1 バイト	X'03'	PCLK プロトコル制御
2–3	2 バイト	VERS	プロトコル・バージョン

VERS

VERS に指定された値は、Query Reply が戻されるときに端末に インストールされる PCLK のバージョンを示すために使用されます。例えば、X'0001' は PCLK バージョン 1.1 を示します。

この Query Reply に対するフィールド定義については、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

プロダクト定義 Query Reply

この照会応答は、X'9C' データ構造内の登録済みサブ ID を使用する IBM 製品によって使用されます。プロダクト定義データ・ストリーム Query Reply は、IBM プロダクト定義の データ・ストリームを使用する 3270DS ワークステーション補助装置の サポートを示します。データ・ストリームは、アーキテクチャー・レビュー・ボードなどの、識別可能な制御点を持つ形式アーキテクチャーの資料では定義されていません。

補助装置が IBM プロダクト定義のデータ・ストリームをサポートすると、この Query Reply は Query List (QCODE リスト = X'9C' またはすべて) に応答してインバウンド伝送されます。

任意 指定パラメーター

Query Reply の基本部分のすべてのパラメーター、 および直接アクセス自己定義パラメーターは必須です。

プロダクト定義 Query Reply の形式は次のとおりです。

表 23. IBM プロダクト定義 Query Reply の基本形式

オフセット:	長さ	コンテンツ	意味
0–1	1 ワード	長さ	構造体の長さ (SDP を含む)
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'9C'	IBM プロダクト定義データ・ストリーム
4–5	2 バイト	FLAGS	予約済み
6	1 バイト	REFID	参照 ID
7	1 バイト	SSID	サブセット ID
8	1 バイト	X'04'	パラメーター長
9	1 バイト	X'01'	直接アクセス
10–11	1 ワード	DOID	サブシステムが割り当てた Destination/Origin ID

プロダクト定義 Query Reply の REFID (オフセット 6) および SSID (オフセット 7) に対する有効値は次のとおりです。

表 24. IBM プロダクト定義 Query Reply に対する REFID と SSID の有効値

REFID	SSID	プロダクトおよびデータ・ストリームの情報
X'01'		5080 グラフィックス・システム: この参照 ID は、5080 グラフィックス・システム・データ・ストリームを 補助装置がサポートしていることを示します。5080 グラフィックス・アーキテクチャー、構造化フィールド、サブセット ID、DOID、および関連機能セットの記述は、「*IBM 5080 Graphics System Principles of Operation*」に定義されています。
	X'01' X'02'	5080 HGFD グラフィックス・サブセット 5080 RS232 ポート・サブセット
X'02'		WHIP API (書き込み時に SRL 名に置き換えられる) この参照 ID は、WHIP API データ・ストリームを補助装置が サポートしていることを示します。WHIP API アーキテクチャーの記述は、「*IBM RT PC Workstation Host Interface Program バージョン 1.1* ユーザーズ・ガイドおよびリファレンス・マニュアル」に定義されています。
	X'01'	WHIP サブセット 1
X'03' から X'FF'		その他の値は予約済みです。

IBM プロダクト定義プロセッサーは、直接アクセス自己定義パラメーター のみをサポートします。これは 表 25 : IBM プロダクト定義直接アクセス SDP (ページ 236) で定義されます。

直接アクセス自己定義パラメーター

直接アクセス ID 自己定義パラメーターがある場合、補助装置が Destination/Origin 構造化フィールドを使用して直接アクセスされている可能性があることを示します。プロダクト定義のデータ・ストリームを使用する複数の補助装置が サポートされている場合、固有の DOID を持つ、個別のプロダクト定義の データ・ストリーム Query Reply が提供されます。

表 25. IBM プロダクト定義直接アクセス SDP

オフセット:	長さ	コンテンツ	意味
0	1 バイト	X'04'	パラメーター長
1	1 バイト	X'01'	直接アクセス ID
2–3	2 バイト	DOID	Destination/Origin ID

DOID

これらのバイト内の値は 、Destination/Origin 構造化フィールドの ID フィールド内で使用され、 補助装置が後に続くデータの宛先かまたは発信元かを識別します。

文書交換アーキテクチャー Query Reply

この Query Reply は、サポートされている文書交換アーキテクチャー (DIA) 機能セットを示します。DIA Query Reply の形式は次のとおりです。

表 26. IBM DIA の基本形式

オフセット:	長さ	コンテンツ	意味
0	1 ワード	長さ	構造体の長さ (SDP を含む)
2	1 バイト	X'81'	Query reply ID
3	1 バイト	X'97'	IBM DIA
4–5	2 バイト	FLAGS	予約済み
6–7	2 バイト	LIMIN	インバウンド伝送できる DDM バイトの最大数
8–9	2 バイト	LIMOUT	アウトバウンド伝送できる DDM バイトの最大数
10	1 バイト	NFS	後に続く 3 バイトの機能セット ID の数
11–13	3 バイト	DIAFS	DIA 機能セット ID
14– (13+(N*3))	N*3 バイト	DIAFS	追加の DIA 機能セット ID
14+(N*3)	1 バイト	X'04'	パラメーター長
15+(N*3)	1 バイト	X'01'	直接アクセス
16+(N*3)	1 ワード	DOID	サブシステムが割り当てた Destination/Origin ID

DIA 補助装置プロセッサーは、直接アクセス自己定義パラメーター のみをサポートします。これは 表 27 : IBM プロダクト定義直接アクセス SDP (ページ 237) で定義されます。

直接アクセス ID 自己定義パラメーターがある場合、補助装置が Destination/Origin 構造化フィールドを使用して直接アクセスされている可能性があることを示します。

表 27. IBM プロダクト定義直接アクセス SDP

オフセット:	長さ	コンテンツ	意味
0	1 バイト	X'04'	パラメーター長
1	1 バイト	X'01'	直接アクセス ID
2–3	2 バイト	DOID	Destination/Origin ID

DOID

これらのバイト内の値は 、Destination/Origin 構造化フィールドの ID フィールド内で使用され、 補助装置が後に続くデータの宛先かまたは発信元かを識別します。

この Query Reply に対するフィールド定義については、「*IBM 3270* 情報表示システム・データ・ストリーム・プログラマー用解説書」を参照してください。

付録 B. Communication Manager/2 EHLLAPI との相違点

この付録では、 Z and I Emulator for Windowsの EHLLAPI と Communication Manager/2 用 EHLLAPI との相違点を説明します。

以下の EHLLAPI 関数は、同名の Communication Manager/2 とは異なっているので、使用するときは十分注意してください。

- **Set Session Parameter** (9)
- **Copy OIA** (13)
- **Copy String to PS** (15)
- **Storage Manager** (17)
- **Copy String to Field** (33)
- **Get Key** (51)
- **Window Status** (104)
- **Query Sessions** (10)
- **Connect for Structured Field** (120)
- **Allocate Communications Buffer** (123)
- ASCII 略号

Set Session Parameter (9)

セット・オプション

Z and I Emulator for Windowsでは、Communication Managerが提供している以下のセット・オプションは提供されていません。

OLDOIA、NEWOIA
COMPCASE、COMPICASE
OLD5250OIA、NEW5250OIA

戻りパラメーター

Set Session Parameter (9) 関数が終了すると、Communication Manager は第 3 パラメーターのデータ・ストリング長として、 有効なデータ・ストリングの長さを戻します。しかし、Z and I Emulator for Windowsでは、データ・ストリング長として、有効なセット・オプションの個数を戻します。

EAB オプション

Set Session Parameter (9) 関数で、EAB オプションが指定されたとき、Communication Manager/2 では **Copy PS** (5) または **Copy PS to String** (8) 関数でコピーされる EAB 属性の中の文字色の値が、カラー再マップの影響を受けます。

ただし、Z and I Emulator for Windowsでは、カラー再マップの有無にかかわらず、EAB 属性の文字色の 値は、表示スペースの内容によって決まり、カラー再マップの 影響を受けません。

Copy OIA (13)

Copy OIA (13) 関数は、Communication Manager/2 と Z and I Emulator for Windows では、以下の点が異なります。グループと桁位置についての詳しい説明は、Copy OIA (13) (ページ 55)を参照してください。

- 21 バイト目
  - Z and I Emulator for Windowsは X'F6' を返します。
  - Communication Manager/2 は X'20' を返します。
- 61 ～ 63 バイト目
  - Z and I Emulator for Windowsは、プリンター情報を戻しません。
  - Communication Manager/2 はプリンター情報を戻します。
- グループ 3: シフト状態

  Communication Manager/2 は、ビット 2 の値を戻しません。 ビット 2 は予約され、ビット 0 に上段シフトと大文字ロック (Caps Lock) の 両方がセットされます。
- グループ 8 バイト 1: 入力禁止
  - Z and I Emulator for Windowsは、ビット 6 の値 (装置作動不能) を戻しません。
  - Communication Manager/2 は、ビット 6 の値を戻すことがあります。
- グループ 8 バイト 3: 入力禁止
  - Z and I Emulator for Windowsは、ビット 1 (オペレーター許可なし) およびビット 2 (オペ レーター許可なし -f) の値を戻しません。
  - Communication Manager/2 は、ビット 1 および 2 の値を戻すことがあります。
- グループ 8 バイト 4: 入力禁止
  - Z and I Emulator for Windowsは、ビット 2 (システム待ち状態) の値を戻しません。
  - Communication Manager/2 は、ビット 2 の値を戻すことがあります。
- グループ 10: 強調表示グループ 2
  - Z and I Emulator for Windowsは、ビット 0 の値 (選択済み) を戻しません。
  - Communication Manager/2 は、ビット 0 の値を戻すことがあります。
- グループ 11: カラー・グループ 2
  - Z and I Emulator for Windowsは、ビット 0 の値 (選択済み) を戻しません。
  - Communication Manager/2 は、ビット 0 の値を戻すことがあります。
- グループ 13: プリンター状況
  - Z and I Emulator for Windowsでは、このグループは予約済みです。
  - Communication Manager/2 では、このグループを戻すことがあります。
- グループ 14: グラフィックス

  Communication Manager/2 は、ビット 0 (グラフィックス・カーソル) の値を戻しません。

Copy String to PS (15)

Communication Manager/2 では、**Set Session Parameter** (9) 関数の EAB オプションが **Copy String to PS** 関数に影響します。EAB オプションを指定すると、テキスト・データと同じサイズの属性データを、 テキスト・データと共にその関数に渡す必要があります。

Z and I Emulator for Windowsでは、EAB オプションに関係なく、渡されるデータはテキスト・デ ータのみです。Communication Manager/2 と同様のインターフェースを使用する場合は、**Set Session Parameter** (9) の `PUTEAB`オプションを使用してください。

Storage Manager (17)

Communication Manager/2 で提供されている **Storage Manager** (17) 関数は、Z and I Emulator for Windows では、サポートされません。アプリケーション用のメモリーの割り振りには、Windows® で提供されている API を使用してください。

Copy String to Field (33)

Communication Manager/2 では、**Set Session Parameter** (9) 関数の EAB オプションが指定されると、属性データが、その関数への受け渡しのデータの一部として渡されます。そのため、EAB オプションを指定すると、テキスト・データと同じサイズの 属性データを、テキスト・データと共にその関数に渡す必要があります。

Z and I Emulator for Windowsでは、EAB オプションは、**Copy String to Field** (33) 関数のデータ内容には影響しません。渡すデータはテキスト・データのみで、属性データは渡されません。Communication Manager/2 と同様のインターフェースを使用する場合は、**Set Session Parameter** (9) の `PUTEAB`オプションを使用してください。

Get Key (51)

渡されたキーのシフト状態がエミュレーター・セッションで認識されるキーや関数でない場合、Communication Manager/2 では @A、@S、または @r を使用してシフト状態を戻します。Z and I Emulator for Windows は、これらの ASCII 略号をサポートしていません。

Window Status (104)

EHLLAPI 関数 104 (PM_WINDOW_STATUS) の拡張状況照会コマンド (0x03) は、エミュレーター表示スペース・ウィンドウのハンドルを戻します。これは、関数と Communication Manager/2 のインプリメンテーションの定義によって構成されます。しかし、Z and I Emulator for Windows 版 EHLLAPI では、フレーム・ウィンドウのハンドルが戻されます。この関数を使用してZ and I Emulator for Windows 版用に書かれた、EHLLAPI アプリケーションでは、戻されたウィンドウ・ハンドルの親を使用する必要があります。

Query Sessions (10)

Communication Manager/2 では、パーソナル・コンピューター用の記述子が戻されます。しかし、Z and I Emulator for Windowsでは、記述子は戻されません。

Connect for Structured Fields (120)

Communication Manager/2 で提供されている通信接続状況のためのイベント・オブジェクトは、Z and I Emulator for Windowsにはありません。

Allocate Communications Buffer (123)

Communication Manager/2 では、要求バッファー・サイズの最大値が、 64 KB - 8 バイト (X'FFF8') です。

Z and I Emulator for Windowsでは、この値は 64 KB - 256 バイト (X'FF00') です。

ASCII 略号

次の ASCII 略号は、Z and I Emulator for Windowsではサポートされません。

略号	**意味**
@A@N	カーソル取り込み
@A@O	カーソル位置づけ
@A@X	Hexadecimal (16 進数)
@A@Y	コマンド (機能) キー
@A@a	消去しながらバックスペース
@S@A	EOL 消去
@S@B	フィールド前進
@S@C	フィールド・バックスペース
@S@D	有効文字バックスペース
@S@P	POR (送信のみ)
@S@T	タスク・マネージャーにジャンプ
@/	キューのオーバーラン (**Get Key** 関数の場合のみ)

Get Request Completion (125)

Z and I Emulator for Windowsは、ブランクまたは NULL のセッション ID をサポートしません。

付録 C. 特記事項

本書は米国 HCL が提供する製品およびサービスについて作成したものであり、 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 HCL の営業担当員にお尋ねください。本書で HCL 製品、プログラム、またはサービスに言及していても、その HCL 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、HCL の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、HCL 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

本書に記述されている主題事項に関して HCL が特許権 (特許出願を含む) を所有していることがあります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
注意: Office of the General Counsel

HCL TECHNOLOGIES LTD. 本書を特定物として現存するままの状態で提供し、 商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示 もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。HCL は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において HCL 以外の文書または HCL 以外の Web サイトに言及している場合がありますが、 便宜のため記載しただけであり、決してそれらの文書または Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この HCL 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

HCL は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
注意: Office of the General Counsel

本プログラムに関する上記の情報は、適切な使用条件の下で使用できますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、HCL 所定のプログラム契約の契約条項、HCL プログラムのご使用条件、またはそれと同等の条項に基づいて、HCL より提供されます。

本書に含まれるパフォーマンス・データは、特定の動作および環境条件下で得られたものです。実際の結果は、異なる可能性があります。

HCL 以外の製品に関する情報は、その製品の供給者もしくは公開されているその他のソースから入手したものです。HCL は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求について HCL は確証できません。HCL 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

商標

HCL、HCL ロゴおよび hcl.com は、世界の多くの国で登録された HCL Technologies Ltd. の商標または登録商標です。他の製品名およびサービス名等は、それぞれ IBM® または各社の商標である場合があります。

Index

記号

Query Reply データ構造、EHLLAPI がサポートする
  プロダクト定義 Query Reply オプション・パラメーター 234
オペレーター・サービス関数
  Send Key (3) 135
オペレーター・サービス関数
  Query Host Update (24) 111
  Query Session Status (22) 112
  Query System (20) 116
  Reset System (21) 129
  Start Host Notification (23) 160
  Stop Host Notification (25) 168
キー・ストロークの送信
  Send Key (3) 135
ダイナミック・リンク方式 16
パス、デフォルト
  Send File 135
ファイル転送機能
  Send File (90) 134
フィールド関連関数
  Search Field (30) 130
フィールド関連関数
  Find Field Position (31) 87
フィールド形式 PS 130
プレゼンテーション・サービス関数
  Disconnect Presentation Space (2) 82
  Find Field Position (31) 87
  Search Field (30) 130
プレゼンテーション・サービス関数
  Copy OIA (13) 55
ブロッキング・ルーチン 191
ホスト接続表示スペース 18
関数呼び出し
  呼び出し (入力) パラメーター 36
関数呼び出し
  ページ・レイアウトについての規則 36
  使用 36
  使用上の注意 36
  前提呼び出し 36
  戻り (出力) パラメーター 36
検索関数 30
  Search Field (30) 130
検索機能
  Search Presentation Space (6) 132
言語 12
呼び出し (入力) パラメーター
  一般 36
呼び出し、前提 36
呼び出すことができる Stop Keystroke Intercept (53) 35
指定、ストリングの 79
自動化 33
初期化関数/終了関数 190
商標 243
数字専用フィールド 136
前提呼び出し、一般 36
装置サービス関数
  Release (12) 127
  Reserve (11) 128
送信、キー・ストローク 31
属性のバイト 77
属性バイト 51, 65, 69, 74, 148
属性バイト 51
通信サービス機能
  Receive File (91) 125
  Send File (90) 134
入力保護フィールド 136
非同期関数、WinHLLAPI 181
表示サービス機能
  Copy Presentation Space to String (8) 69
  Lock Presentation space API (60) 96
  Query Cursor Location (7) 108
  Search Presentation Space (6) 132
表示サービス機能
  Copy Field to String (34) 51
  Copy Presentation Space (5) 65
  Copy String to Field (33) 74
  Find Field Length (32) 86
  Set Cursor (40) 144
表示サービス機能
  Query Additional Field Attribute (45) 103
表示スペース 17
  カーソル移動 32
  タイプ 17
  フィールド形式 87
  フィールド形式 74
  フィールド形式 130
  フィールド形式 51, 52, 86, 130
  拡張 32 ビット・インターフェース 17
  指定方法 18
  識別子
    NULL 指定子 20
    ブランク指定子 20
    関数 18
    処理方法 18
    接続を必要としない関数の処理 19
    接続を必要とする関数の処理 19
    文字指定子 20
  文字テーブル 57
表示スペース
  OIA 55
  ホスト接続 18
表示スペースのサイズ 18
表示スペースのタイプ 17
表示スペースのロック 23
表示スペース名
  最大数 18
  宣言 18
  有効な名前 18
文字、ASCII 25
文字、エスケープ 26, 92, 137, 150
文字列の指定 77, 80
文字列代行受信、Get Key (51) 90
編集キー
  インターセプト 84
戻り (出力) パラメーター、一般 36
略号
  ASCII 25
  Send Key 24
  キーボード、テーブル 138
  シフト・キー 24

数字

01, Connect Presentation Space 45
02, Disconnect Presentation Space 82
03, Send Key 93, 135
03、Send Key 163, 170
04、Wait 170
05、Copy Presentation Space 65
06、Search Presentation Space 132, 170
07、Query Cursor Location 108
08、Copy Presentation Space to String 69
09、Set Session Parameters 145
10, Query Sessions 115
101, Connect Window Services 47
  Connect Window Services (101) 47
102, Disconnect Window Service 83
103, Query Window Coordinates 118
104, Window Status 171
105, Change Switch List LT Name 42
106, Change PS Window Name 41
11, Reserve 128
110, Start Playing Macro 165
12, Release 127, 128
13, Copy OIA 55
14, Query Field Attribute 109
15, Copy String to Presentation Space 76
16/32 ビットの考慮事項 28
18, Pause 101, 168
20, Query System 116
21, Reset System 128, 129
21、Reset System 145
22, Query Session Status 112
23, Start Host Notification 101, 111, 160
24, Query Host Update 111, 168
24、Query Host Update 102
25, Stop Host Notification 168
30, Search Field 130
30、Search Field 170
31, Find Field Position 87
31、フィールド位置の検索 52
32 ビット表示スペース ID 17
32、Find Field Length 86
32、フィールド長の検索 52
33、Copy String to Field 74
34、Copy Field to String 51
35、Copy Presentation Space to Clipboard 77
36、Paste Clipboard to Presentation Space 79
40、Set Cursor 144
41、Start Close Intercept 155
42、Query Close Intercept 104
43、Stop Close Intercept 166
45、Query Additional Field Attribute 103
50、Start Keystroke Intercept 162
51、Get Key 90, 102, 163
52、Post Intercept Status 102, 163
53、Stop Keystroke Intercept 169
61、Lock PMSVC API 99
90、Send File 134
91、Receive File 125
99、Convert Position または Convert RowCol 48

A

Allocate Communications Buffer (123) 38
API ヘッダー・ファイルの使用 8
ASCII ニーモニック
  get key (51) 関数 26
  send key (3) 関数 27
ASCII 略号
  一般 24
ATTRB 148
AUTORESET 150

B

BLANK 152

C

Cancel File Transfer(92) 40
Change PS Window Name (106) 41

Change Switch List LT Name (105) 42
Connect for Structured Fields (120) 44
Connect Presentation Space (1)
一般 45
切断との対話 18
不必要な場合の関数 46
Connect Window Services (101) 47
Convert Position または Convert RowCol (99) 48
Copy Field to String (34) 32, 51
Copy OIA (13) 30, 55
Copy Presentation Space (5) 65
Copy Presentation Space to Clipboard (35) 77
Copy Presentation Space to String (8) 31, 69
Copy String to Field (33) 32, 74
Copy String to Presentation Space (15) 76

D

Disconnect from Structured Fields (121) 81
Disconnect Presentation Space (2)
一般 82
接続との対話 18
Disconnect Window Service (102) 83

E

EAB 151
EditKey インターセプト 84
EHLLAPI
関数 36
要約 37
EHLLAPI がサポートする Query Reply データ構造
DDM Query Reply
一般 229
アーキテクチャー Query Reply 236
プロダクト定義 Query Reply
直接アクセス自己定義パラメーター 236, 237
一般 229
EHLLAPI でサポートされている照会応答データ構造
IBM 補助装置 Query Reply
一般 232
EHLLAPI でサポートされる照会応答のデータ構造
製品定義の照会応答
一般 234
EHLLAPI の概要
IBM 拡張 EHLLAPI 対IBM 標準 EHLLAPI 12
IBM 標準 EHLLAPI 11
WinHLLAPI 11
WinHLLAPI 対IBM Standard EHLLAPI 11
EHLLAPI プログラミングの概要 11
EHLLAPI プログラミングの紹介 11
EHLLAPI 呼び出し形式 12
EHLLAPI 戻りコード 14
EOT 147
ESC 150

F

Find Field Length (32) 86
Find Field Position (31) 87
FPAUSE 148
Free Communications Buffer (124) 89

G

Get Key (51) 24, 90, 102, 163
Get Request Completion (125) 94

I

IBM サポート・センター 117
IPAUSE 148

L

Lock Presentation Space API (60) 96
Lock Window Services API (61) 99
LWAIT 151, 170

N

NOATTRB 148
NOBLANK 152
NOEAB 151
NOQUIET 149
NORESET 150
NOXLATE 152
NULLATTRB 148
NWAIT 151, 170

O

OIA 55, 170

P

Paste Clipboard to Presentation Space (36) 79
Pause (18) 31, 101, 168
PCSAPI
pcsConnectSession 196
pcsDisconnectSession 196
pcsGetPageSettings 205
pcsGetPrinterSettings 213
pcsQueryConnectionInfo 197
pcsQueryEmulatorStatus 198
pcsQuerySessionList 199
pcsQueryWorkstationProfile 201
pcsRestorePageDefaults 208
pcsSetLinkTimeout 202
pcsSetPageSettings 209
pcsSetPrinterSettings 219
pcsStartSession 202
pcsStopSession 203
一般 195
使用方法 195
pcsDisconnectSession 196
pcsQueryConnectionInfo 197
pcsQueryEmulatorStatus 198
pcsQuerySessionList 199
pcsQueryWorkstationProfile 201
pcsStartSession 202
pcsStopSession 203
Post Intercept Status (52) 35, 102, 163
presentation services functions
Get Request Completion (125) 94
PSID の処理
接続を必要としない関数 19
接続を必要とする関数 19

Q

Query Additional Field Attribute (45) 103, 103
Query Close Intercept (42) 104
Query Communication Event (81) 107
Query Communications Buffer Size (122) 105
Query Cursor Location (7) 108
Query Field Attribute (14) 109, 109
Query Host Update (24) 102, 111, 160, 168
Query Reply データ構造、EHLLAPI がサポートする
DDM Query Reply
DDM アプリケーション名の自己定義パラメーター 230
PCLK プロトコル制御の自己定義パラメーター 230
基本 DDM Query Reply の形式 231
IBM 補助装置 Query Reply
PCLK プロトコル制御の自己定義パラメーター 234
直接アクセス自己定義パラメーター 233
Query Session Status (22) 112
Query Sessions (10) 115
Query System (20) 116
Query Window Coordinates (103) 118
QUIET 149

R

Read Structured Fields (126) 120
Receive File (91)
デフォルト・パス、宛先ファイルに対する 127
一般 32, 125, 125, 150
RECEIVE.EXE の位置 125
Release (12) 33, 127, 128
Reserve (11) 33, 127, 128
Reset System (21) 127, 128, 129, 145

S

Search Field (30) 130, 170
Search Presentation Space (6) 31, 132, 170
Send File (90)
SEND.EXE の場所 134
デフォルト・パス、宛先ファイルに対する 135
一般 32, 134, 150
Send Key (3) 24, 93, 135, 163, 170
Set Cursor (40) 144
Set Session Parameters (9)
ストリングの仕様 147
一般 136, 145
影響を受ける機能のリスト 145
有効な入力 147
有効な入力 155
SRCHALL 148
SRCHBKWD 148
SRCHFROM 148
SRCHFRWD 148
Start Close Intercept (41) 155
Start Communication Notification (80) 157
Start Host Notification (23) 101, 111, 148, 160, 169
Start Keystroke Intercept (50) 162
Start Playing Macro (110) 165
Stop Close Intercept (43) 166
Stop Communication Notification (82) 167
Stop Host Notification (25) 168
Stop Keystroke Intercept (53) 169
STREOT 147
STRLEN 147

T

TIMEOUT 149
TWAIT 151, 170

W

Wait (4) 30, 137, 170
Window Status (104) 171
WinHLLAPI 拡張関数
非同期関数
一般 181
WinHLLAPI 拡張関数
概要 181
WinHLLAPI 拡張関数
ブロッキング・ルーチン

WinHLLAPIIsBlocking 191
WinHLLAPIUnhookBlockingHook 193
ブロッキング・ルーチン
WinHLLAPISetBlockingHook 192
初期化/終了関数
WinHLLAPI Startup 190, 191
非同期関数
WinHLLAPIAsync 181
WinHLLAPICancelAsyncRequest 189
WinHLLAPI 拡張関数
ブロッキング・ルーチン
WinHLLAPICancelBlockingCall 193
WinHLLAPI 拡張関数s
ブロッキング・ルーチン
一般 191
WinHLLAPI 拡張機能
一般 190
初期化関数/終了関数
一般 190
Write Structured Fields (127) 175

X

XLATE 152

う

ウィンドウ・サービス関数
Change PS Window Name (106) 41
Change Switch List LT Name (105) 42
ウィンドウ・サービス機能
Lock Window Services API (61) 99

え

エスケープ文字 26, 92, 137, 150
エミュレーター API の概要
Z and I Emulator for Windows セッション API (PCSAPI) 7
エミュレーター高水準言語 API (EHLLAPI) 7

お

オプション 155
オペレーター・サービス関数
Pause (18) 101
Query Sessions (10) 115
オペレーター・サービス機能
Set Session Parameters (9) 145
Wait (4) 170
オペレーター情報域
「OIA」を参照してください。 55

か

カーソル移動 32

き

キー・ストロークのフィルター操作 33
キー・ストロークの送信
略号 24
キー･ストローク代行受信、Get Key (51) 90
キーボード、セッション 24
キーボードの拡張 34
キーボードの略号
テーブル 138
一般 24

く

クリティカル・セクション 8

こ

コピー関数
Copy OIA (13) 55
Copy String to Presentation Space (15) 76
コピー機能
Copy Field to String (34) 51
Copy Presentation Space (5) 65
Copy Presentation Space to String (8) 69
コピー機能
Copy String to Field (33) 74
コンパイルおよびリンク 16
コンパイルとリンク 15
ご使用条件 243

さ

サービス 117
サンプル・プログラム、簡単な EHLLAPI の 27
サンプルプログラム 9

し

シフト・キー略号 24

す

スタック・サイズ 8
ストリングの仕様 147

せ

セッション・オプション 147
セッション・キーボード 24

そ

ソース・コード構文 29

て

ディレクトリー、デフォルト
Receive File 127
データ構造 13
デバイス・サービス機能
Get Key (51) 90
Post Intercept Status (52) 102
Start Keystroke Intercept (50) 162
Stop Keystroke Intercept (53) 169
デバッグ 27

は

パス、デフォルト
Receive File 127
パラメーター
呼び出し 36

ひ

ビープ音 102

ふ

ファイル転送 32
ファイル転送機能
Receive File (91) 125
フィールド、ホスト
数字専用 136
入力保護 136
フィールド位置の検索 (31) 52
フィールド関連関数
Query Field Attribute (14) 109
フィールド関連機能
Copy Field to String (34) 51
Find Field Length (32) 86
Query Additional Field Attribute (45) 103
フィールド関連機能
Copy String to Field (33) 74
フィールド形式の PS 52
フィールド長の検索 (32) 52
プレゼンテーション・サービス関数
Connect Presentation Space (1) 45
Copy String to Presentation Space (15) 76
Query Field Attribute (14) 109

ほ

ホスト・フィールド
数字専用 136
入力保護 136
ホストの自動化のシナリオ 30

ま

マルチスレッド 17

め

メモリー割り振り 14

り

リンク
ダイナミック・リンク方式 16
説明 15