Programmation de l'émulateur

A propos de ce manuel

Ce manuel fournit les informations de programmation nécessaires pour utiliser l'interface de programme d'applications de langage de haut niveau d'émulateur HCLZ and I Emulator for Windows (EHLLAPI), l' et l'API de session Z and I Emulator for Windows (PCSAPI), et. La bibliothèque de classes Host Access est décrite dans *Host Access Class Library*.

EHLLAPI/PCSAPI est utilisé avec Z and I Emulator for Windows pour fournir aux utilisateurs et aux programmeurs un moyen d'accéder à l'espace de présentation hôte avec un ensemble de fonctions pouvant être appelées à partir d'un programme d'application exécuté dans une session de poste de travail.

Dans ce manuel, *Windows* fait référence à Windows® 7, Windows® 8/8.1, Windows® 10, Windows® Server 2008 et Windows® Server 2012. Lorsque les informations concernent uniquement un système d'exploitation spécifique, cela sera indiqué dans le texte.

Qui devrait lire ce manuel

Ce manuel est destiné aux programmeurs qui écrivent des programmes d'application utilisant les API documentées dans ce livre.

Une connaissance pratique de Windows® est supposée. Pour plus d'informations sur Windows®, reportez-vous à la liste des publications sous Où trouver plus d'informations on page 2.

Le programmeur doit également être familier avec la connexion à un système hôte à partir d'un terminal ou d'un poste de travail doté d'un logiciel d'émulation de terminal.

Ce manuel suppose que vous connaissez le langage et le compilateur que vous utilisez. Pour plus d'informations sur la façon d'écrire, de compiler ou de modifier des programmes, reportez-vous à Où trouver plus d'informations on page 2 pour les références appropriées pour la langue spécifique que vous utilisez.

Où trouver plus d'informations

La bibliothèque Z and I Emulator for Windows comprend les publications suivantes :

- *Guide d'installation*
- *Guide d'initiation*
- *Référence de l'utilisateur de l'émulateur*
- *Guide d'administration et de référence*
- *Programmation d'émulateur*
- *Programmation des communications client-serveur*
- *Programmation de gestion du système*
- *Bibliothèque de classes Host Access*
- *Référence du fichier de configuration*

En plus des manuels imprimés, des documents HTML (Hypertext Markup Language) sont fournis avec Z and I Emulator for Windows :

Bibliothèque de classes Host Access

Les fichiers HTML Java HACL décrivent comment écrire une application compatible ActiveX/OLE 2.0 pour utiliser Z and I Emulator for Windows comme un objet incorporé. Ces fichiers sont accessibles à partir du dossier zippé Docs_Admin_Aids livré avec la documentation de produit Z and I Emulator for Windows dans le chemin suivant : *ZIEWin_3.0_Docs_Admin_Aids.zip\publications\fre\doc\hacl*

Voici une liste de publications connexes :

- *IBM 3270 Information Display System Data Stream Programmer's Reference*, GA23-0059
- *IBM 5250 Information Display System Functions Reference Manual*, SA21-9247

Notation

Un tableau au début de chaque section explique les fonctions API dans EHLLAPI Fonctions on page 31, Fonctions PCSAPI on page 192 et Fonctions d'extension WinHLLAPI on page 179. Il indique si une fonction est prise en charge pour les produits qui fournissent la fonction décrite dans la section. Oui signifie qu'il est pris en charge pour un type d'hôte et Non signifie qu'il n'est pas pris en charge. Par exemple, le tableau suivant indique qu'une fonction est disponible pour les sessions 3270 et VT mais pas pour les sessions 5250.

3270	*5250*	*VT*
Oui	Non	Oui

Chapter 1. Introduction aux API d'émulateur

Le produit IBM®Z and I Emulator for Windows fournit plusieurs interfaces de programmation d'application (API). Chaque interface possède un ensemble spécifique de fonctions et peut être utilisée à des fins différentes. Choisissez l'interface de programmation qui correspond le mieux aux exigences fonctionnelles de votre application. Certaines applications peuvent utiliser plusieurs interfaces pour obtenir les résultats souhaités. Les interfaces de programmation sont :

- **API de langage de haut niveau d'émulateur (EHLLAPI)** : cette interface fournit des fonctions pour accéder aux données de « l'espace de présentation » de l'émulateur telles que les caractères sur l'écran hôte. Elle fournit également des fonctions permettant d'envoyer des frappes au clavier à l'hôte, d'intercepter les frappes saisies par l'utilisateur, d'interroger l'état de la session hôte, de charger et de télécharger des fichiers, ainsi que d'autres fonctions. Cette interface est souvent utilisée pour les applications *d'opérateur automatisées* qui lisent les écrans hôtes et saisissent les touches sans intervention directe de l'utilisateur. Voir EHLLAPI Fonctions on page 31.
  - **Prise en charge d'IBM® Standard HLLAPI** : il s'agit d'une interface de programmation standard qui permet l'accès par programme à une session d'émulateur hôte. Voir Introduction à la programmation IBM Standard EHLLAPI, IBM Enhanced EHLLAPI et WinHLLAPI on page 8.
  - **Prise en charge d'IBM® Enhanced HLLAPI** : cette interface est basée sur l'interface d'IBM® Standard HLLAPI. Elle fournit toutes les fonctionnalités existantes, mais utilise des structures de données modifiées. Voir Introduction à la programmation IBM Standard EHLLAPI, IBM Enhanced EHLLAPI et WinHLLAPI on page 8.
  - **API de langage de haut niveau Windows® (WinHLLAPI)** : cette interface fournit une grande partie des mêmes fonctionnalités qu'IBM® Standard EHLLAPI et ajoute quelques extensions qui tirent parti de l'environnement Windows®. Voir Introduction à la programmation IBM Standard EHLLAPI, IBM Enhanced EHLLAPI et WinHLLAPI on page 8.
- Toutes les API 32 bits qui acceptent/renvoyent des descripteurs et des pointeurs Windows peuvent ne pas fonctionner correctement avec HCL ZIEWin en raison de la différence de taille de pointeur/descripteur entre les plateformes x86 et x64.

  Exemple :

  Le paramètre « Data String » renvoyé sous forme de numéros d'octets (9 à 12) dans la notification de communication de démarrage de l'API (80) peut être tronqué sur la plateforme x64.
- **API de session Z and I Emulator for Windows (PCSAPI)** : cette interface est utilisée pour démarrer, arrêter et contrôler les sessions et les paramètres de l'émulateur. Voir Fonctions PCSAPI on page 192.

  Pour Z and I Emulator for WindowsVersion 3.0, des fonctions ont été ajoutées pour permettre le contrôle et la récupération des paramètres de page et d'imprimante. Voir Fonctions Page Setup on page 201 et Fonctions Printer Setup on page 209.
- **Bibliothèque de classes d'accès hôte (ECL) HCLZ and I Emulator for Windows** : ECL est un ensemble d'objets qui permettent aux programmeurs d'applications et aux rédacteurs de langages de script d'accéder facilement

et rapidement aux applications hôtes. Z and I Emulator for Windows prend en charge trois couches ECL différentes (objets C++, ActiveAutomation (OLE) et LotusScript Extension (LSX)). Consultez *Bibliothèque de classes Host Access (HACL)* pour plus de détails.

Utilisation des fichiers d'en-tête de l'API

Le programme d'application doit inclure les fichiers d'en-tête du système d'exploitation avant d'inclure les fichiers d'en-tête de l'API. Par exemple :

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

Sections critiques

Utilisez les sections critiques (fonction **EnterCriticalSection**) avec précaution lorsque votre programme appelle les API d'émulateur. N'effectuez pas d'appels d'API d'émulateur dans une section critique. Si un thread d'une application établit une section critique et qu'un autre thread se trouve dans un appel d'API d'émulateur, l'appel est suspendu jusqu'à ce que vous quittiez la section critique.

Pendant le traitement d'un appel API, tous les signaux (à l'exception des signaux numériques du coprocesseur) sont retardés jusqu'à ce que l'appel soit terminé ou jusqu'à ce que l'appel doive attendre des données entrantes. De plus, la fonction **TerminateProcess** émise par un autre processus est conservée jusqu'à ce que l'application termine un appel API qu'elle est susceptible de traiter.

Taille de la pile

Les API d'émulateur utilisent la pile du programme appelant lors de leur exécution. Le système d'exploitation, l'application et l'API nécessitent tous un espace de pile pour les variables dynamiques et les paramètres de fonction. Au moins 8 196 octets (8 Ko) d'espace de pile doivent être disponibles au moment d'un appel d'API. Il est de la responsabilité du programme d'application de garantir qu'un espace de pile suffisant est disponible pour l'API.

Prise en charge de la plateforme Windows x64

Les versions x64 de Microsoft® Windows® Server 2008 et Microsoft® Windows® 8/8.1/10 Edition x64 sont optimisées pour exécuter des programmes 64 bits natifs, mais ne prennent pas en charge les pilotes 32 bits ni les applications 16 bits.

Pour ces plateformes, Z and I Emulator for Windows n'installe pas les bibliothèques suivantes.

- Prise en charge des API 16 bits :
  - Interface standard EHLLAPI 16 bits
  - Interface WinHLLAPI 16 bits
  - Interface PCSAPI 16 bits

Exemples de programmes

Plusieurs exemples de programmes sont fournis, chacun illustrant l'utilisation d'une des API de Z and I Emulator for Windows. Si vous choisissez d'installer les exemples de programmes, ils seront installés dans le répertoire \SAMPLES.

Remarque : Le présent document est livré « en l'état » sans aucune garantie explicite ou implicite. International Business Machines décline notamment toute responsabilité relative à ces informations en cas de contrefaçon ainsi qu'en cas de défaut d'aptitude à l'exécution d'un travail donné.

Les exemples de fichiers de programme incluent des fichiers source et de prise en charge pour les API Z and I Emulator for Windows suivantes :

- Interface de programmation Emulateur High-Level Language (EHLLAPI)
- Fonctions PCSAPI

Les fichiers suivants sont installés dans le répertoire \SAMPLES.

Tableau 1. Exemples de sous-répertoires du programme

Nom de fichier	Description
DDE_C.H	Fichier include DDE
EHLAPI32.H	IBM® Fichier include EHLLAPI 32 bits standard
WHLLAPI.H	Fichier include WinHLLAPI 16 bits
HAPI_C.H	Fichier include EHLLAPI
PCSAPI.H	Fichier include PCSAPI
PCSCALLS.LIB	Bibliothèque d'importation pour l'interface standard
PCSCAL32.LIB	Bibliothèque d'importation pour l'interface améliorée
EHLAPI32.LIB	Bibliothèque d'importation pour l'interface standard EHLLAPI 32 bits d'IBM®
WHLLAPI.LIB	Bibliothèque d'importation pour l'interface WinHLLAPI 16 bits
WHLAPI32.LIB	Bibliothèque d'importation pour l'interface WinHLLAPI 32 bits

Les sous-répertoires suivants sont créés dans le répertoire \SAMPLES.

Tableau 2. Exemples de sous-répertoires du programme

Nom de fichier	Description
ECL	Le dossier sample\ecl\cpp contient l'exemple lié à tous les fichiers HACL CPP.

Tableau 2. Exemples de sous-répertoires du programme (suite)

Nom de fichier	Description
	Le dossier sample\ecl\vb contient l'exemple lié à tous les fichiers HACL VB.Net.
HLLSMP	Montre comment utiliser l'interface EHLLAPI pour demander une frappe et se connecter à un système VM (X86). Il prend en charge les fonctionnalités logon, pastetext et sendkey. Reportez-vous à hllsmp\Readme.txt pour plus de détails sur l'utilisation des fonctionnalités mentionnées ci-dessus avec hllsmp.exe. (X64).

Chapter 2. Introduction à la programmation IBM Standard EHLLAPI, IBM Enhanced EHLLAPI et WinHLLAPI

Ce chapitre fournit les informations nécessaires pour intégrer les fonctions IBM® Standard EHLLAPI (16 et 32 bits), WinHLLAPI (16 et 32 bits) et IBM® Enhanced 32 bits EHLLAPI (EHLAPI32) dans des applications écrites dans un langage de haut niveau. Il fournit des détails sur le format d'appel, les considérations d'allocation de mémoire, l'initialisation des interfaces, ainsi que la compilation et la liaison des applications. Un court exemple de programme EHLLAPI et les instructions de compilation/liaison utilisées pour le construire sont également inclus. Enfin, un ensemble d'utilisations possibles de l'interface EHLLAPI (scénarios) est décrit.

Une application EHLLAPI est tout programme d'application qui utilise l'interface EHLLAPI pour accéder à l'espace de présentation hôte 3270/5250/VT. L'espace de présentation comprend les données de caractères visibles de l'émulateur, les champs et les données d'attribut, les données de frappe et d'autres informations.

Présentations EHLLAPI

Vous trouverez ci-dessous des aperçus des interfaces de programmation HLLAPI.

Norme IBM EHLLAPI

EHLLAPI est une interface de programmation standard qui permet l'accès par programme à une session d'émulateur hôte. Des fonctions sont fournies pour lire les données de l'écran hôte (telles que les caractères et les attributs), pour envoyer des frappes au clavier et pour exécuter d'autres fonctions liées à l'émulateur.

L'interface EHLLAPI est une interface à point d'appel unique. Il existe une seule API appelable via laquelle toutes les fonctions EHLLAPI sont demandées. A chaque appel à l'interface, l'application fournit un numéro de fonction qui identifie la fonction demandée, un pointeur vers un tampon de données, un pointeur vers la longueur du tampon de données et un pointeur vers un code retour (voir Format d'appel EHLLAPI on page 9).

WinHLLAPI

WinHLLAPI est basé sur le célèbre EHLLAPI.API. Il englobe toutes les fonctionnalités existantes et ajoute des extensions qui tirent parti de l'environnement Windows® piloté par messages. Les utilisateurs de l'interface HCLZ and I Emulator for Windows EHLLAPI ne remarqueront aucune différence fonctionnelle à moins qu'elle n'intègre les extensions WinHLLAPI.

Les fonctions d'extension WinHLLAPI et toutes les fonctions qui s'écartent du formulaire EHLLAPI sont décrites dans Fonctions d'extension WinHLLAPI on page 179. Pour plus d'informations sur les fonctions communes, consultez EHLLAPI Fonctions on page 31.

WinHLLAPI et IBM® Standard EHLLAPI

Le symbole d'entrée pour WinHLLAPI est, à juste titre, **WinHLLAPI**. Les utilisateurs d'EHLLAPI souhaitant passer à l'implémentation WinHLLAPI doivent modifier l'entrée standard **hllapi**. Les nouveaux utilisateurs doivent suivre toutes les instructions dans EHLLAPI Fonctions on page 31, et utilisez l'entrée **WinHLLAPI** à la place de l'entrée **hllapi** standard.

IBM Enhanced EHLLAPI et IBM Standard EHLLAPI

IBM Enhanced EHLLAPI est basé sur l'API EHLLAPI familière. Il englobe toutes les fonctionnalités existantes, mais tire parti de l'environnement 32 bits et utilise des structures de données modifiées. Les utilisateurs de l'interface standard souhaitant passer à IBM® Enhanced 32 bits EHLLAPI doivent uniquement modifier le symbole d'entrée de LPWORD à LPINT dans les premier, troisième et quatrième paramètres. Les nouveaux utilisateurs doivent utiliser les procédures des sections suivantes.

Langues

Tout langage de programmation pouvant invoquer un point d'entrée dans une DLL avec la convention d'appel « Pascal » peut être utilisé pour exécuter les fonctions EHLLAPI. Cependant, le kit d'outils Z and I Emulator for Windows EHLLAPI fournit des fichiers d'en-tête et des prototypes de fonctions uniquement pour les langages C++. Une compréhension claire de la disposition de la structure des données et des conventions d'appel est nécessaire pour utiliser n'importe quel autre langage. Le kit d'outils EHLLAPI prend en charge les compilateurs C/C++ suivants :

- Microsoft® Visual C/C++ version 4.0 et supérieure

La plupart des autres compilateurs C/C++ fonctionneront également avec le kit d'outils.
Les applications EHLLAPI C/C++ doivent inclure le fichier d'en-tête Z and I Emulator for Windows EHLLAPI (HAPI_C.H). Ce fichier définit la disposition des structures de données et fournit un prototype pour le point d'entrée EHLLAPI.

Note: La disposition de la structure des données pour les applications 16 et 32 bits n'est pas la même (voir Considérations relatives aux interfaces standard et améliorées on page 24).

Format d'appel EHLLAPI

Le point d'entrée EHLLAPI (**hllapi**) est toujours appelé avec les quatre paramètres suivants :

1. Numéro de fonction EHLLAPI (entrée)
2. Tampon de données (entrée/sortie)

3. Longueur du tampon (entrée/sortie)
4. Position (entrée) ; Code retour (sortie)

Le prototype pour IBM® Standard EHLLAPI est :

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

Le prototype pour IBM® Enhanced EHLLAPI est :

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

Chaque paramètre est transmis par *référence* et non par valeur. Ainsi, chaque paramètre de l'appel de fonction doit être un *pointeur* vers la valeur, pas la valeur elle-même. Par exemple, ce qui suit est un exemple correct d'appel de la fonction EHLLAPI Query Session Status :

```
#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 }
```

Tous les paramètres de l'appel **hllapi** sont des pointeurs et le code retour de la fonction EHLLAPI est renvoyé dans la valeur du 4ème paramètre, et non comme la valeur de la fonction. Par exemple, ce qui suit **n'est pas** correct :

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

Bien que la fonction **hllapi** soit définie pour renvoyer un type de données **long** pour IBM® Standard et Enhanced EHLLAPI, et un type de données **void** pour WinHLLAPI, sa valeur n'est pas définie et ne doit pas être utilisée.

Les deuxième à quatrième paramètres de l'appel **hllapi** peuvent renvoyer des informations à l'application. La description de chaque fonction EHLLAPI décrit les informations, le cas échéant, renvoyées dans ces paramètres.

Structures de données

De nombreuses fonctions EHLLAPI utilisent une structure de données formatée pour transmettre des informations vers ou depuis le programme d'application. La description de chaque fonction montre la disposition de la structure des données. Les données transmises vers ou depuis la fonction EHLLAPI doivent exister dans le stockage exactement comme indiqué, octet par octet. Notez que la présentation de la structure est la même pour toutes les applications IBM® Standard et WinHLLAPI 16 et 32 bits. Les structures de données pour les applications IBM® Enhanced 32 bits sont regroupées selon un alignement sur 4 octets.

Il est *fortement recommandé* d'utiliser les définitions de fichier d'en-tête et de structure de données fournies pour garantir un alignement et une présentation appropriés des données. Bien que cela soit techniquement possible, ce qui suit *n'est pas* recommandé :

```
char QueryData[20]; // Not recommended ... Func = HA_QUERY_SESSION_STATUS; hllapi(&Func, QueryData,
&Len, &Rc); if (QueryData[13] == 'F') { // ...this is a 5250 session }
```

La manière recommandée pour écrire cette fonction serait :

```
#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 }
```

Allocation de mémoire

Les fonctions EHLLAPI n'allouent ni ne libèrent de mémoire. Le programme d'application doit préallouer de l'espace tampon pour les fonctions EHLLAPI qui en ont besoin avant d'appeler le point d'entrée **hlapi**. L'espace tampon peut être préalloué en tant que variable dynamique telle que :

```
struct HLDQuerySessionStatus QueryBuff;
```

ou il peut être alloué par un appel à une bibliothèque C ou à une fonction du système d'exploitation telle que :

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

Dans tous les cas, l'application est responsable d'allouer suffisamment d'espace tampon avant d'appeler les fonctions EHLLAPI et de libérer les tampons lorsqu'ils ne sont pas nécessaires.

Codes retour EHLLAPI

Les fonctions EHLLAPI renvoient un code de fin d'exécution ou code retour dans le 4ème paramètre de l'appel de la fonction **hllapi** (sauf pour la fonction **Convert Position** ou **RowCol** (99)). Le code retour indique le succès ou l'échec de la fonction demandée.

Sauf indication contraire dans la description de chaque fonction, le tableau suivant montre la signification de chaque valeur de code retour. Certaines fonctions peuvent avoir une interprétation légèrement différente de ces codes retour. Reportez-vous aux descriptions de fonctions individuelles pour plus de détails.

Table 3. Codes retour EHLLAPI

Code retour	Explication
0	La fonction s'est exécutée avec succès ou aucune mise à jour depuis que le dernier appel a été émis.
1	Un ID d'espace de présentation hôte incorrect a été spécifié. La session spécifiée n'est pas connectée, n'existe pas ou est une session d'imprimante logique.
2	Une erreur de paramètre a été rencontrée ou un numéro de fonction incorrect a été spécifié. (Reportez-vous à la fonction individuelle pour plus de détails.)
4	L'exécution de la fonction a été inhibée, car l'espace de présentation cible était occupé, dans l'état X CLOCK (X []), ou dans l'état X SYSTEM.
5	L'exécution de la fonction a été empêchée pour une raison autre que celles indiquées dans le code de retour 4.
6	Une erreur de données s'est produite en raison de la spécification d'un paramètre incorrect (par exemple, une erreur de longueur provoquant une troncature).
7	La position spécifiée dans l'espace de présentation n'était pas valide.
8	Une erreur de procédure fonctionnelle a été rencontrée (par exemple, utilisation de fonctions conflictuelles ou fonctions prérequises manquantes).

Table 3. Codes retour EHLLAPI (continued)

Code retour	Explication
9	Une erreur système a été rencontrée.
10	Cette fonction n'est pas disponible pour EHLLAPI.
11	Cette ressource n'est pas disponible.
12	Cette session s'est arrêtée.
24	La chaîne n'a pas été trouvée ou l'espace de présentation n'est pas formaté.
25	Les frappes au clavier n'étaient pas disponibles dans la file d'attente d'entrée.
26	Un événement hôte s'est produit. Consultez **Query Host Update** (24) pour plus de détails.
27	Le transfert de fichiers a été interrompu par une commande Ctrl+Entrée.
28	La longueur du champ était de 0.
31	Débordement de la file d'attente des frappes. Les frappes ont été perdues.
32	Une application s'est déjà connectée à cette session pour les communications.
33	Réservé.
34	Le message envoyé à l'hôte a été annulé.
35	Le message envoyé par l'hôte a été annulé.
36	Le contact avec l'hôte a été perdu.
37	La communication entrante a été désactivée.
38	La fonction demandée n'a pas terminé son exécution.
39	Une autre session DDM est déjà connectée.
40	La tentative de déconnexion a réussi, mais certaines requêtes asynchrones n'étaient pas terminées au moment de la déconnexion.
41	Le tampon que vous avez demandé est utilisé par une autre application.
42	Aucune demande en attente ne correspond.
43	L'API était déjà verrouillée par une autre application EHLLAPI (sur LOCK) ou API non verrouillée (sur UNLOCK).

Compilation et liaison

Le programme d'application peut être lié à l'aide d'une méthode de liaison dynamique. Cela signifie que nous pouvons lier le point d'entrée en effectuant la liaison dynamique. Dans ce cas, l'application utilise les appels du système d'exploitation pour charger la DLL correcte et obtenir l'adresse du point d'entrée au moment de l'exécution.

Le tableau suivant montre quelle .DLL doit être utilisé pour le chargement dynamique.

Interface	Point d'entrée	DLL
IBM® Standard (32 bits/64 bits)	hllapi	EHLAPI32.DLL
IBM® Enhanced (32 bits/64 bits)	hllapi	PCSHLL32.DLL
WinHLLAPI (32 bits/64 bits)	winhllapi	WHLAPI32.DLL

Méthode de liaison dynamique

A l'aide de la méthode de liaison dynamique, l'application appelle le système d'exploitation au moment de l'exécution pour charger le module Z and I Emulator for Windows EHLLAPI et de localiser le point d'entrée **hllapi** qui s'y trouve. Cette méthode nécessite plus de code dans l'application mais donne à celle-ci un meilleur contrôle sur les conditions d'erreur. Par exemple, l'application peut afficher un message d'erreur spécifique à l'utilisateur si le module Z and I Emulator for Windows EHLLAPI est introuvable.

Pour utiliser la liaison dynamique, l'application doit charger le module Z and I Emulator for Windows approprié et localisez le point d'entrée. Il est recommandé que le point d'entrée soit localisé par son numéro ordinal et non par son nom. Le numéro ordinal est défini dans le fichier d'en-tête. Le code Windows® 32 bits suivant charge le module EHLLAPI 32 bits IBM® Standard, localise le point d'entrée **hlapi** et effectue un appel de fonction EHLLAPI.

```
#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 }
```

Traitement multitâche

IBM Enhanced EHLLAPI (32 bits) et IBM® Standard EHLLAPI 16 bits se connectent processus par processus. Toutes les unités d'exécution accèdent à la même session hôte connectée. L'unité d'exécution qui effectue les connexions doit également effectuer la déconnexion.

IBM® Standard EHLLAPI (32 bits) et WinHLLAPI se connectent unité d'exécution par unité d'exécution. Chaque unité d'exécution doit maintenir ses propres connexions. Cela permet à un processus de traitement multitâche de maintenir des connexions à plusieurs sessions hôte connectées à la fois. Cela élimine le besoin de schémas multi-processus lors de l'utilisation d'un programme WinHLLAPI pour coordonner les données entre différents hôtes. Cela impose également la charge de connexion et de déconnexion si nécessaire sur l'unité d'exécution individuelle.

Espaces de présentation

De nombreuses fonctions EHLLAPI nécessitent un *ID d'espace de présentation (PSID)* pour indiquer quelle session d'émulateur hôte doit être utilisée pour la fonction. (Ceci est également appelé *ID de session court*). Un ID d'espace de présentation est un caractère unique compris entre A et Z. Il existe un maximum de 26 sessions.

ID d'espace de présentation de l'interface d'IBM® Enhanced 32 bits

Pour les applications IBM® Enhanced EHLLAPI, l'ID de session est étendu avec trois octets supplémentaires. Ces octets de session étendue doivent être définis sur zéro pour une compatibilité future. Ceci est plus facilement réalisé

en définissant le contenu des tampons EHLLAPI sur des zéros binaires avant de les remplir avec les informations requises. Par exemple, les éléments suivants peuvent être utilisés pour interroger l'état de la session B :

```
#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);
```

Types d'espaces de présentation

Une session d'émulateur peut être configurée comme une session d'affichage ou une session d'imprimante. Les applications EHLLAPI ne peuvent pas se connecter aux sessions d'imprimante ou de routeur du PC400. La fonction **Query Sessions** (10) peut être utilisée pour déterminer le type d'une session particulière.

Taille des espaces de présentation

Une session d'affichage d'émulateur peut être configurée pour une gamme de tailles d'écran allant de 1 920 octets (taille d'écran 24 x 80) à 9 920 octets (taille d'écran 62 x 160). Certaines fonctions EHLLAPI telles que **Copy PS to String** (8) nécessitent que l'application alloue suffisamment de stockage pour contenir (éventuellement) la totalité de l'espace de présentation. La taille de l'espace de présentation pour une session donnée peut être obtenue à l'aide de la fonction **Query Session Status** (22).

ID de l'espace de présentation

EHLLAPI les fonctions interagissent avec un seul espace de présentation à la fois. L'ID d'espace de présentation (PSID) est utilisé pour identifier l'espace de présentation particulier dans lequel une fonction doit fonctionner.

Pour certaines fonctions, le PSID est contenu dans un appel précédent à la fonction **Connect Presentation Space** (1). Pour d'autres fonctions, le PSID est contenu dans le paramètre de chaîne de données appelant.

Espace de présentation connecté à l'hôte

La connexion à l'espace de présentation (ou à la session) hôte est contrôlée à l'aide des fonctions **Connect Presentation Space** (1) et **Disconnect Presentation Space** (2). Le statut de la connexion détermine si certaines fonctions peuvent être exécutées. Cela affecte également la façon dont le PSID est défini. Le texte suivant explique comment contrôler le statut de la connexion à l'espace de présentation hôte :

- A tout moment, il peut y avoir soit aucun espace de présentation connecté à l'hôte, soit il peut y avoir un seul espace de présentation connecté à l'hôte.
- Il n'y a pas d'espace de présentation connecté à l'hôte par défaut.

- Suite à une connexion, il existe un seul espace de présentation connecté à l'hôte. L'espace de présentation hôte qui est connecté est identifié dans le paramètre de chaîne de données appelant de la fonction de connexion.
- Un appel ultérieur à la connexion peut être exécuté sans déconnexion intermédiaire. Dans ce cas, il existe toujours un seul espace de présentation connecté à l'hôte. Encore une fois, l'espace de présentation hôte qui est connecté est identifié dans le paramètre de chaîne de données appelant de la fonction de connexion.
- Suite à une déconnexion, il n'y a plus d'espace de présentation connecté à l'hôte. Cette règle s'applique suite à plusieurs appels consécutifs pour se connecter ou suite à un seul appel pour se connecter.
- Vous ne pouvez pas vous connecter à une session d'imprimante logique.

Gestion de l'ID de l'espace de présentation

Le PSID est utilisé pour spécifier l'espace de présentation hôte (ou la session) dans lequel vous souhaitez qu'une fonction fonctionne. La façon dont le PSID est géré est affectée par deux facteurs :

1. La méthode utilisée pour spécifier le PSID :
    a. En tant que paramètre de chaîne de données d'appel d'un appel précédent à la fonction **Connect Presentation Space** (1)
    b. En tant que caractère dans la chaîne de données appelante de la fonction en cours d'exécution. La gestion varie selon que le caractère est :
        - Une lettre de *A* à *Z*
        - Un blanc ou une valeur Null
2. Le statut de la connexion à l'espace de présentation hôte.

Les paragraphes suivants décrivent comment le PSID est traité pour les différentes combinaisons de ces deux facteurs.

Gestion du PSID pour les fonctions nécessitant une connexion

Certaines fonctions interagissent uniquement avec l'espace de présentation connecté à l'hôte. Ces fonctions nécessitent la fonction **Connect Presentation Space** (1) comme appel préalable. Le PSID de ces fonctions est déterminé par les fonctions **Connect Presentation Space** (1) et **Disconnect Presentation Space** (2) comme suit :

- Lorsqu'il n'y a pas d'espace de présentation connecté à l'hôte, ces fonctions n'interagissent avec aucun espace de présentation. Un de code retour de 1 est généré.
- Lorsqu'il existe un espace de présentation connecté à l'hôte, ces fonctions interagissent avec l'espace de présentation spécifié dans le paramètre de chaîne de données appelant de l'appel le plus récent à la fonction **Connect Présentation Space** (1).

Gestion du PSID pour les fonctions ne nécessitant pas de connexion

Certaines fonctions peuvent interagir avec un espace de présentation hôte, qu'il soit connecté ou non. Ces fonctions vous permettent de spécifier le PSID dans le paramètre de chaîne de données appelant. Elles sont les suivantes :

- **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)

Toutes ces fonctions, sauf les deux premières, vous permettent de spécifier le PSID en utilisant soit :

- Une lettre de *A* à *Z*
- Un blanc ou une valeur Null

Les deux premières fonctions nécessitent qu'une lettre soit utilisée pour spécifier le PSID.

Lorsqu'il n'y a pas d'espace de présentation connecté à l'hôte, les règles suivantes s'appliquent :

- La fonction peut interagir avec n'importe quel espace de présentation hôte si une lettre, et non un espace ou une valeur Null, est utilisée pour spécifier le PSID.
- Si un espace ou une valeur Null est utilisé pour spécifier le PSID, un code retour de 1 est généré. La fonction ne s'exécute pas.
- L'utilisation d'une lettre pour spécifier le PSID n'établit pas d'espace de présentation connecté à l'hôte, sauf sur une demande de connexion PS.

Lorsqu'il existe un espace de présentation connecté à l'hôte, les règles suivantes s'appliquent :

- La fonction peut interagir avec n'importe quel espace de présentation hôte si une lettre est utilisée pour spécifier le PSID.
- Si un espace ou une valeur Null est utilisé pour spécifier le PSID, la fonction fonctionne dans l'espace de présentation identifié lors de l'appel le plus récent à la fonction **Connect Presentation Space** (1).
- L'utilisation d'une lettre pour spécifier le PSID ne modifie pas le PSID établi de l'espace de présentation connecté à l'hôte, sauf sur une demande de connexion PS.

Les fonctions suivantes sont disponibles pour les sessions d'impression :

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

Partage de l'espace de présentation EHLLAPI entre les processus

Plusieurs applications EHLLAPI peuvent partager un espace de présentation si les applications prennent en charge le partage (c'est-à-dire si elles ont été développées pour fonctionner ensemble ou si elles présentent un comportement prévisible[1]). Pour déterminer quelles applications prennent en charge le partage, les applications EHLLAPI sont spécifiées comme l'un des types suivants :

- Surveillance
- Ecriture exclusive avec privilège de lecture autorisé
- Ecriture exclusive sans privilège de lecture autorisé
- Super écriture
- Lu

Le type d'accès partagé peut être défini en spécifiant les options de partage de lecture et d'écriture suivantes pour chaque fonction dans l'appel de fonction **Set Session Parameters** (9) :

SUPER_WRITE

L'application permet à d'autres applications autorisant le partage et disposant d'autorisations d'accès en écriture de se connecter simultanément au même espace de présentation. L'application d'origine exécute des fonctions de type supervision mais ne crée pas d'erreurs pour les autres applications partageant l'espace de présentation.

WRITE_SUPER

L'application nécessite un accès en écriture et permet uniquement aux applications de supervision de se connecter simultanément à son espace de présentation. Il s'agit de la valeur par défaut.

WRITE_WRITE

L'application nécessite un accès en écriture et permet à des applications partenaires ou à d'autres applications ayant un comportement prévisible de partager l'espace de présentation.

1. Cela signifie que deux programmes EHLLAPI ne se disputeront pas le même espace de présentation en même temps, ou qu'il existe une logique dans ces programmes qui permettra au programme d'attendre que le PS soit disponible, ou que les applications n'utilisent jamais la session d'une manière qui verrouillerait d'autres applications.

WRITE_READ

L'application nécessite un accès en écriture et permet à d'autres applications qui exécutent des fonctions en lecture seule de partager l'espace de présentation. L'application est également autorisée à copier l'espace de présentation et à effectuer d'autres opérations en lecture seule comme d'habitude.

WRITE_NONE

L'application a l'usage exclusif de l'espace de présentation. Aucune autre application n'est autorisée à partager l'espace de présentation, y compris les applications de supervision. L'application est autorisée à copier l'espace de présentation et à effectuer des opérations en lecture seule comme d'habitude.

READ_WRITE

L'application nécessite uniquement un accès en lecture pour surveiller l'espace de présentation et permet à d'autres applications effectuant des fonctions de lecture ou d'écriture, ou les deux, de partager l'espace de présentation. L'application est également autorisée à copier l'espace de présentation et à effectuer d'autres opérations en lecture seule comme d'habitude.

Note: Le partage de l'espace de présentation n'est pas disponible entre les unités d'exécution d'un processus.

Table 4. Combinaisons d'options de partage de lecture et d'écriture EHLLAPI

Application appelante	SUPER_WRITE	WRITE_SUPER	WRITE_WRITE	WRITE_READ	WRITE_NONE	READ_WRITE
SUPER_WRITE	Oui	Oui	Oui	Non	Non	Oui
Write_Super (par défaut)	Oui	Non	Non	Non	Non	Non
WRITE_WRITE	Oui	Non	Oui	Non	Non	Oui
WRITE_READ	Non	Non	Non	Non	Non	Oui
WRITE_NONE	Non	Non	Non	Non	Non	Non
READ_WRITE	Oui	Non	Oui	Oui	Non	Oui

En plus de spécifier des options d'accès en lecture et en écriture compatibles, les applications conçues pour fonctionner ensemble mais ne pouvant pas permettre à d'autres personnes de travailler dans le même espace de présentation peuvent éventuellement définir un mot clé, KEY$nnnnnnnn, dans l'appel de fonction **Set Session Parameters** (9). Ce mot-clé permet uniquement aux applications qui utilisent le même mot clé de partager l'espace de présentation.

Note:

1. La fonction **Start Keystroke Intercept** (50) n'est pas partageable. Une seule application à la fois peut intercepter les frappes au clavier.
2. Les fonctions **Connect To Presentation Space** (1) et **Start Keystroke Intercept** (50) partagent des fonctions de sous-système communes. Les demandes réussies d'une application visant à partager l'une ou l'autre de ces fonctions peuvent affecter les demandes de ces deux fonctions par d'autres applications. Par exemple, si l'application A demande avec succès à se connecter à l'espace de présentation (**Connect To Presentation Space**) (1) avec un accès Write_Read et KEY$abcdefgh comme mot clé, une demande de l'application B pour se connecter à l'espace de présentation (**Connect To Presentation Space**) (1) ou démarrer l'interception de frappe (**Start Keystroke Intercept**) (50) aboutit uniquement si les deux applications ont défini des options de lecture et d'écriture compatibles.

Table 5. Fonctions prérequises et fonctions dépendantes associées

Appel prérequis	Fonctions	Accès
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 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 Read Read Write Write Write Read Read Read Read Write Write Write 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) if edit keystrokes are to be sent (edit keystroked support is available in Enhanced Mode)	N/A
Write Structured Field (127)	Get Request Completion (125)	N/A

Verrouillage de l'espace de présentation

Une application, même si elle est spécifiée avec un espace de présentation partagé, peut obtenir un contrôle exclusif d'un espace de présentation à l'aide des fonctions **Lock Presentation Space API** (60) ou **Lock Windows® Services API** (61). Les demandes des autres applications visant à utiliser un espace de présentation verrouillé par ces fonctions sont mises en file d'attente et traitées dans l'ordre première entrée, première sortie (FIFO) lorsque l'application d'origine déverrouille l'espace de présentation.

Si l'application qui a verrouillé l'espace de présentation ne le déverrouille pas en utilisant le même appel avec une option **Unlock** ou un appel **Reset System** (21), le verrouillage est supprimé lorsque l'application se termine ou que la session s'arrête.

Utilisation des actions de la souris pour sélectionner, copier et coller du texte dans l'espace de présentation

Les actions de souris suivantes peuvent être utilisées dans l'espace de présentation.

- Sélectionnez un mot en double-cliquant sur le bouton gauche de la souris.
- Copiez un mot sélectionné en cliquant sur le bouton droit de la souris.
- Collez un mot copié en double-cliquant sur le bouton droit de la souris.

Mnémoniques ASCII

Les frappes provenant d'un clavier hôte peuvent avoir une valeur ASCII correspondante. La réponse de la fonction **Get Key** (51) à une frappe dépend du fait que la clé est définie et également du fait que la clé est définie comme une valeur ASCII ou un mnémonique ASCII.

Le clavier d'une session peut ne pas être capable de produire certains codes nécessaires à une autre session. Les mnémoniques ASCII qui représentent ces codes peuvent être inclus dans le paramètre de chaîne de données de la fonction **Send Key** (3).

Les capacités de la fonction **Send Key** (3) et de la fonction **Get Key** (51) permettent aux sessions d'échanger des frappes qui pourraient ne pas être représentées par des valeurs ASCII ou par une clé disponible. Un ensemble de mnémoniques qui peut être généré à partir d'un clavier est fourni. Ces mnémoniques vous permettent d'utiliser des caractères ASCII pour représenter les clés de fonction spéciales du clavier du poste de travail.

Les mnémoniques pour les clés non décalées se composent du caractère d'échappement suivi d'une abréviation. Cela est également vrai pour les clés de décalage elles-mêmes, Maj supérieur, Alt et Ctrl. Les mnémoniques pour les clés décalées comprennent le mnémonique de la clé de décalage suivi du mnémonique de la clé non décalée. Par conséquent, le mnémonique d'une clé décalée est une séquence de 4 caractères de caractère d'échappement, abréviation, caractère d'échappement, abréviation.

Le caractère d'échappement par défaut est `@`. Vous pouvez remplacer la valeur du caractère d'échappement par n'importe quel autre caractère avec l'option `ESC=c` de la fonction **Set Session Parameters** (9). Le texte suivant utilise cependant le caractère d'échappement par défaut.

Les indicateurs de décalage qui ne font pas partie du jeu de caractères ASCII sont représentés à l'application hôte par des mnémoniques ASCII de 2 octets comme suit :

Décalage supérieur	@S
Alt	@A
Ctrl	@r

Les mnémoniques de ces indicateurs de décalage ne sont jamais reçus séparément par une application. De même, ils ne sont jamais envoyés séparément par une application. Les mnémoniques des indicateurs de décalage sont toujours accompagnés d'un caractère ou d'un mnémonique non indicateur de décalage.

Les abréviations utilisées facilitent la mémorisation des mnémoniques des clés spéciales. Un code de clé alphabétique a été utilisé pour les clés les plus courantes. Par exemple, la clé Clear est *C*, la clé Tab est *T*, et ainsi de suite. Veuillez noter que les caractères alphabétiques majuscules et minuscules sont des abréviations mnémoniques pour différentes touches.

Le texte suivant décrit l'utilisation de ces fonctions.

Général

Toutes les clés définies sont représentées par :

- Une valeur ASCII de 1 octet faisant partie du jeu de caractères ASCII de 256 éléments, ou
- Un mnémonique ASCII de 2, 4 ou 6 octets

Pour représenter une clé définie comme un caractère ASCII, une valeur ASCII de 1 octet correspondant à ce caractère est utilisée.

Pour représenter une clé définie comme une fonction, un mnémonique ASCII de 2, 4 ou 6 octets correspondant à cette fonction est utilisé. Par exemple, pour représenter la clé de retour arrière, `@B` est utilisé. Pour représenter PF1, `@1` est utilisé. Pour représenter Erase Input, `@A@F` est utilisé. Consultez les listes suivantes :

`@B`	Tabulation gauche	`@0`	Domicile	`@h`	PF17
`@C`	Effacer	`@1`	PF1/F1	`@i`	PF18
`@D`	Supprimer	`@2`	PF2/F2	`@j`	PF19
`@E`	Entrée	`@3`	PF3/F3	`@k`	PF20
`@F`	Effacer EOF	`@4`	PF4/F4	`@l`	PF21
`@H`	Aide (PC400)	`@5`	PF5/F5	`@m`	PF22
`@I`	Insérer	`@6`	PF6/F6	`@n`	PF23
`@J`	Sauter	`@7`	PF7/F7	`@o`	PF24
`@L`	Curseur vers la gauche	`@8`	PF8/F8	`@q`	Fin
`@N`	Nouvelle ligne	`@9`	PF9/F9	`@u`	Page HAUT (PC400)

@O	Espace	@a	PF10/F10	@v	Page Bas (PC400)
@P	Imprimer	@b	PF11/F11	@x	PA1
@R	Réinitialiser	@c	PF12/F12	@y	PA2
@T	Tabulation droite	@d	PF13	@z	PA3
@U	Curseur vers le haute	@e	PF14	@@	Symbole @ (arobase)
@V	Curseur vers le bas	@f	PF15	@$	Curseur alterné
@Z	Curseur vers la droite				

@A@C	Test (PC400)	@A@E	Rose (PC/3270)
@A@D	Supprimer un mot	@A@F	Vert (PC/3270)
@A@E	Zone Quitter	@A@g	Jaune (PC/3270)
@A@F	Effacer l'entrée	@A@H	Bleu (PC/3270)
@A@H	Demande du système	@A@I	Turquoise (PC/3270)
@A@I	Insérer une bascule	@A@J	Blanc (PC/3270)
@A@J	Sélection à l'aide du curseur	@A@L	Réinitialiser la couleur de l'hôte (PC/3270)
@A@L	Curseur vers la gauche rapidement	@A@T	Imprimer (ordinateur personnel)
@A@Q	Attention	@A@U	Remonter (PC400)
@A@R	Annuler l'appareil	@A@V	Descendre (PC400)
@A@T	Imprimer l'espace de présentation	@A@Y	Tabulation mot en avant
@A@U	Curseur vers le haut rapidement	@A@Z	Tabulation mot en arrière
@A@V	Curseur vers le bas rapidement	@A@-	Champ - (PC400)
@A@Z	Curseur vers la droite rapidement	@A@+	Champ + (PC400)
@A@9	Vidéo inversée	@A@<	Enregistrer le retour arrière (PC400)
@A@b	Souligner (PC/3270)	@S@E	Imprimer l'espace de présentation sur l'hôte (PC400)
@A@C	Réinitialiser la vidéo inversée (PC/3270)	@S@x	Dup
@A@D	Rouge (PC/3270)	@S@y	Zone Marquer

Note:

1. Le premier symbole @ du premier tableau représente le caractère d'échappement. Le premier et le deuxième symbole @ du deuxième tableau sont le caractère d'échappement. Le symbole @ est le caractère d'échappement par défaut. Vous pouvez modifier la valeur du caractère d'échappement à l'aide de l'option ESC=c de la fonction **Set Session Parameters** (9).

   Si vous remplacez le caractère d'échappement par #, les séquences littérales utilisées pour représenter les touches Backtab, Home et Erase Input deviennent respectivement #B, #0 et #A#F.

   De plus, la séquence littérale utilisée pour représenter le symbole @ devient #@.

2. Si vous envoyez le mnémonique pour impression d'écran (c'est-à-dire `@P` ou `@A@T`), placez-le à la fin de la chaîne de données appelante.
3. Si vous envoyez le mnémonique pour l'annulation de l'appareil (c'est-à-dire `@A@R`), il est transmis sans message d'erreur ; cependant, la copie locale n'est pas arrêtée.

Fonction Get Key (51)

Si l'opérateur du terminal saisit une clé définie comme un caractère ASCII, l'application hôte reçoit une valeur ASCII de 1 octet qui correspond à ce caractère.

Si l'opérateur saisit une clé définie comme fonction, l'application hôte reçoit un mnémonique ASCII de 2, 4 ou 6 octets qui correspond à cette fonction. Par exemple, si la touche **Backtab** est saisie, `@B` est reçu. Si **PF1** est enfoncé, `@1` est reçu. Si **Erase Input** est enfoncé, `@A@F` est reçu.

Si l'opérateur saisit une combinaison de touches Maj définie, l'application hôte reçoit le caractère ASCII ou le mnémonique ASCII de 2, 4 ou 6 octets qui correspond au caractère ou à la fonction définie.

Si l'opérateur saisit une clé individuelle qui n'est pas définie, la fonction **Get Key** (51) renvoie un code retour de 20 et rien n'est envoyé à l'application hôte.

La fonction **Get Key** (51) préfixe tous les caractères et mnémoniques envoyés à l'application hôte avec deux caractères ASCII. Le premier caractère ASCII est le PSID de l'espace de présentation hôte auquel les frappes sont envoyées. L'autre caractère est respectivement un *A*, *S* ou *M* pour ASCII, décalage spécial ou mnémonique. VoirParamètres de retour on page 86.

Fonction send key (3)

Pour envoyer un caractère ASCII à une autre session, incluez ce caractère dans le paramètre de chaîne de données de la fonction **Send Key** (3).

Pour envoyer une clé de fonction à une autre session, incluez le mnémonique ASCII de cette fonction dans le paramètre de chaîne de données de la fonction **Send Key** (3).

Si la fonction **Send Key** (3) envoie un mnémonique non reconnu à la session hôte, un code de retour rejetant la clé peut en résulter.

Débogage

Pour faciliter le débogage des applications EHLLAPI, la fonction de trace de Z and I Emulator for Windows peut être utilisé. Cette fonction produira un journal de tous les appels, paramètres, valeurs de retour et codes de retour EHLLAPI. Pour plus d'informations sur l'utilisation de la fonction de Trace, reportez-vous à *Guide d'administration et de référence*.

Un exemple de programme EHLLAPI simple

L'exemple d'application Windows® suivant entrera la chaîne de caractères « Hello World ! » dans le premier champ de saisie de la session hôte « A ».

```
#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; }
```

L'application peut être construite avec la commande suivante :

```
nmake /a all
```

Considérations relatives aux interfaces standard et améliorées

Il n'y a aucune différence fonctionnelle entre les interfaces EHLLAPI standard et améliorées sur une plateforme donnée. Il existe cependant d'autres différences importantes :

- L'interface EHLLAPI améliorée étend l'ID de l'espace de présentation (PSID) de 1 octet à 4 octets. Actuellement, les octets supplémentaires ne sont pas utilisés, mais votre application doit les définir sur des zéros binaires pour garantir la compatibilité avec les futures versions de l'interface EHLLAPI améliorée.
- La position (décalage) des éléments de données dans les tampons mémoire transmis vers et depuis les fonctions EHLLAPI est différente. Les éléments de données dans l'interface EHLLAPI améliorée sont alignés sur les limites double-mot. Les éléments de données de l'interface EHLLAPI standard ne sont pas alignés de manière particulière. Les applications EHLLAPI ne doivent pas être codées pour définir ou récupérer des données dans les tampons par valeurs de décalage (octets). Au lieu de cela, les structures de données fournies dans le fichier HAPI_C.H doivent être utilisées pour définir et récupérer des éléments de données. Cela garantira que les données sont définies et récupérées à partir de la position correcte pour les programmes 16 et 32 bits.

En pré-remplissant les tampons de données EHLLAPI avec des zéros binaires et en utilisant les structures de données fournies dans HAPI_C.H, une application peut être compilée pour un fonctionnement standard ou amélioré sans aucune modification du code source. Par exemple, la section suivante du code fonctionnerait pour l'interface EHLLAPI standard mais échouerait pour l'interface EHLLAPI améliorée :

```
#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 ...
```

L'exemple ci-dessus échouerait s'il était compilé en tant qu'application EHLLAPI améliorée, car :

- L'application ne définit pas les octets de l'ID de session étendu sur zéro.
- La longueur du tampon pour cette fonction est de 20 et non de 18.
- L'indicateur de type de session n'est pas au décalage 9 dans le tampon de données, il est au décalage 12.

Ce qui suit est la même fonction écrite pour fonctionner correctement si elle est compilée pour un fonctionnement standard ou amélioré. Les lignes modifiées sont indiquées par un > :

```
#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 ...
```

Scénarios d'automatisation de l'hôte

Les exemples de scénarios présentés ici fournissent des informations conceptuelles sur les activités qui peuvent être facilitées à l'aide d'EHLLAPI. Les scénarios traitent des tâches que votre opérateur programmé EHLLAPI peut effectuer dans ces domaines :

- Fonctionnement du système hôte, notamment :
  - Fonction de recherche
  - Envoi de frappes
- Traitement distribué, comprenant :
  - Extraction de données
  - Transfert de fichier
- Intégration d'interfaces

Scénario 1. Une fonction de recherche

Il y a quatre phases dans une transaction typique du système hôte :

1. Démarrage de la transaction
2. En attente de la réponse du système hôte
3. Analyse de la réponse pour voir s'il s'agit de la réponse attendue
4. Extraction et utilisation des données de la réponse

Votre opérateur programmé peut utiliser une série de fonctions EHLLAPI pour imiter ces actions. Après avoir déterminé le point de départ correct pour la transaction du système hôte, l'opérateur programmé peut appeler

la fonction **Search Presentation Space** (6) pour déterminer quels messages de mot clé ou messages d'invite se trouvent sur l'écran d'affichage.

Ensuite, l'opérateur programmé peut utiliser la fonction **Send Key** (3) pour saisir des données dans une session du système hôte et saisir une transaction du système hôte. L'opérateur programmé peut alors :

- Utilisez la fonction **Wait** (4) qui attend la fin de la condition X CLOCK, X [] ou X SYSTEM (ou renvoie une condition de verrouillage du clavier si le terminal s'est verrouillé).

  Si le clavier est inhibé, votre programme EHLLAPI peut appeler la fonction **Copy OIA** (13) pour obtenir plus d'informations sur la condition d'erreur.
- Utilisez la fonction **Search Presentation Space** (6) pour rechercher un mot clé attendu pour valider que la réponse appropriée a été reçue.
- Utilisez la fonction **Copy Presentation Space to String** (8) (ou l'une des nombreuses fonctions d'accès aux données) pour extraire les données souhaitées.

La fonction **Search Presentation Space** (6) est essentielle pour simuler une autre tâche de l'opérateur du terminal. Certains systèmes hôtes ne restent pas verrouillés en mode X CLOCK, X [] ou X SYSTEM jusqu'à ce qu'ils répondent ; au lieu de cela, ils déverrouillent rapidement le clavier et permettent à l'opérateur d'empiler d'autres demandes. Dans cet environnement, l'opérateur du terminal dépend d'une autre invite visuelle pour savoir que les données ont été renvoyées (peut-être un titre d'écran ou un libellé). La fonction **Search Presentation Space** (6) permet à votre programme EHLLAPI de rechercher l'espace de présentation en attendant. De plus, en attendant une réponse, l'appel de la fonction **Pause** (18) autorise d'autres sessions DOS pour partager la ressource de l'unité centrale. La fonction **Pause** (18) possède une option qui permet à votre programme EHLLAPI d'attendre qu'un événement de mise à jour du système hôte se produise.

Si aucun événement du système hôte ne se produit après un délai d'attente raisonnable, votre programme EHLLAPI peut appeler un message d'erreur personnalisé tel que :

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

Dans cet environnement, les révisions du programme deviennent des considérations très importantes, car l'opérateur programmé doit être reprogrammé pour des changements même mineurs dans les messages affichés.

Par exemple, si un opérateur de terminal attend le message :

```
Entrez le numéro de pièce :
```

à titre d'invite, il sera probablement en mesure de répondre correctement à un changement d'application qui produit le message :

```
Entrez le numéro de composant :
```

Cependant, étant donné que l'opérateur programmé recherche une chaîne de mots clés littérale, des changements subtils dans la syntaxe du message, même aussi triviaux que les majuscules par rapport aux minuscules, peuvent amener le programme à entreprendre une action d'erreur préprogrammée.

Scénario 2. Envoi de frappes

Plusieurs considérations nécessitent une attention particulière lors de la conception de programmes qui envoient des frappes au clavier au système hôte. Dans certains environnements d'application, émettre une commande est aussi simple que de taper une chaîne et d'appuyer sur Entrée. D'autres applications impliquent des écrans au format plus complexe dans lesquels les données peuvent être saisies dans l'un des nombreux champs. Dans cet environnement, vous devez comprendre les frappes nécessaires pour remplir l'écran d'affichage.

Le mnémonique de la touche Tab (@T ; voir Général on page 21 pour une liste complète des mnémoniques) peut être utilisé pour passer d'un champ à l'autre. Lorsque vous envoyez des frappes au clavier vers un champ à l'aide de la fonction **Send Key** (3), vous devez connaître la longueur et le contenu du champ. Si vous remplissez complètement les champs et que l'octet d'attribut suivant est autoskip, votre curseur sera alors déplacé vers le champ suivant. Si vous émettez ensuite une tabulation, vous passerez à un autre champ.

De même, si vos frappes ne remplissent pas complètement le champ, il se peut qu'il reste des données provenant d'une saisie précédente. Vous devez utiliser la commande EOF (Erase End of Field) pour effacer ces données résiduelles.

Scénario 3. Traitement distribué

Certaines applications entrent dans la catégorie dite *collaborative*. Ces applications fournissent une seule interface utilisateur final, mais leur traitement est effectué à deux ou plusieurs emplacements physiques différents.

Une application EHLLAPI peut interagir avec les applications du système hôte en interceptant la communication entre le système hôte et l'utilisateur du terminal. L'espace de présentation du système hôte est le véhicule utilisé pour intercepter ces données. L'application locale peut demander à être avertie à chaque fois que l'espace de présentation est mis à jour ou à chaque fois qu'une touche AID est enfoncée par l'opérateur.

Cette application de poste de travail peut alors coopérer avec une application du système hôte de l'une des manières suivantes :

- Sur la base d'un champ ou d'un espace de présentation, en utilisant soit les fonctions de copie qui traitent les champs (fonction **Copy String to Field** (33) ou la fonction **Copy Field to String** (34)), soit les fonctions qui vous permettent de copier depuis et vers des espaces de présentation (par exemple, la fonction **Copy String to Presentation Space** (15) ou la fonction **Copy Presentation Space to String** (8)).
- Sur la base d'une frappe au clavier, en utilisant la fonction **Send Key** (3).
- Sur une base de fichier, pour de gros blocs de données. Vous pouvez demander à votre application d'utiliser la fonctionnalité de transfert de fichiers EHLLAPI (à l'aide de la fonction **Send File** (90) ou de la fonction **Receive File** (91)) pour transférer des données ou des fonctions (telles que des modules de chargement) et les traiter localement ou à distance.

Scénario 4. Transfert de fichier

Dans ce scénario, supposons que vous souhaitiez automatiser un transfert de fichiers :

- Vous pouvez commencer par utiliser la procédure décrite dans le scénario de recherche précédent pour vous connecter à une session du système hôte.
- Au lieu d'utiliser l'une des fonctions de copie (qui sont inefficaces pour copier de nombreux écrans de données), votre programme EHLLAPI pourrait appeler les fonctions de transfert de fichiers **Send File** (90) et **Receive File** (91) pour transférer des données.
- En cas de réussite :
    - Si l'exécution de la fonction **Send File** (90) est terminée, votre programme EHLLAPI peut soumettre un travail par lots à l'aide d'une fonction de copie ou de la fonction **Send Key** (3) avant de vous déconnecter.
    - Si l'exécution de la fonction **Receive File** (91) est terminée, votre programme EHLLAPI peut démarrer une application locale.

Scénario 5. Automatisation

Une application peut fournir toutes les frappes d'une autre application ou peut intercaler les frappes vers la destination cible avec celles du clavier. Parfois, pour ce faire, l'application doit verrouiller les autres sources de saisie au clavier qui pourraient être destinées à une application ou un espace de présentation cible (à l'aide de la fonction **Reserve** (11)) et ensuite la déverrouiller (à l'aide de la fonction **Release** (12)).

L'origine des frappes présentées dans n'importe quelle application est déterminée par la conception de l'application. Les frappes au clavier peuvent provenir des éléments suivants :

- Le clavier
- Les données intégrées à l'application source
- Le stockage secondaire récupéré via l'interface DOS
- L'interface Z and I Emulator for Windows

Dans tous les cas, les frappes fournies à l'application cible ne peuvent pas être distinguées de la saisie ordinaire de l'opérateur.

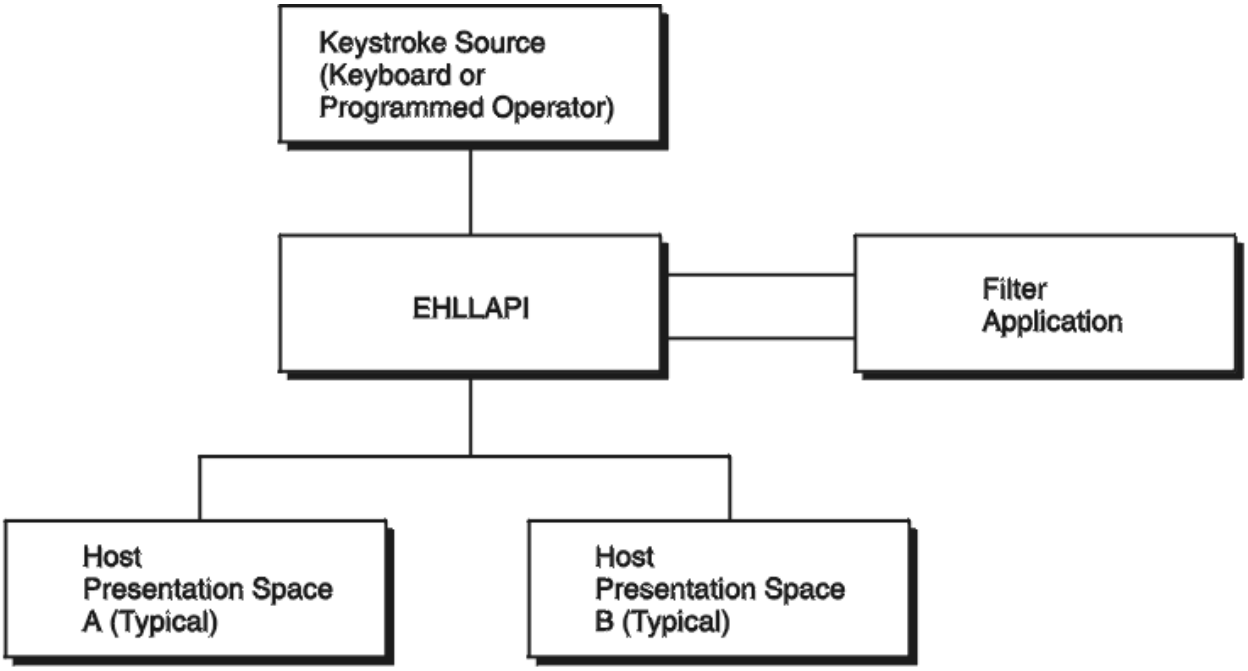
Scénario 6. Filtrage des frappes

Une application qui agit comme un filtre peut intercepter une frappe provenant d'EHLLAPI (soit du clavier, soit d'une application source) qui est ciblée vers une autre destination. La frappe peut alors être :

- Ignorée (c'est-à-dire supprimée)
- Redirigée vers une autre application
- Validé
- Convertie (par exemple, majuscules en minuscules)
- Améliorée (grâce aux macros du clavier)

Figure 1: Flux des frappes on page 29 fournit une représentation simplifiée du flux de frappe et des objets dans un environnement d'amélioration du clavier.

Figure 1. Flux des frappes



Scénario 7. Amélioration du clavier

Ce scénario utilise le filtrage pour créer un **programme d'application d'amélioration**. Un programme d'application d'amélioration est un programme qui surveille les données provenant du clavier et les modifie d'une manière spécifiée. En règle générale, ces programmes d'application utilisent des instructions appelées **macros de clavier**, qui leur indiquent les frappes à rechercher et les modifications à apporter. Le changement peut impliquer la suppression d'une frappe (de sorte qu'elle apparaisse à l'application cible comme si elle n'avait jamais été envoyée), le remplacement d'une frappe par une autre ou le remplacement d'une frappe unique par une série de frappes.

Pour ce faire à l'aide d'EHLLAPI, vous pouvez construire ce scénario :

1. Votre programme d'application EHLLAPI appelle la fonction **Connect Presentation Space** (1) pour se connecter à l'espace de présentation dont les frappes doivent être filtrées.
2. Votre programme EHLLAPI appelle ensuite la fonction **Start Keystroke Intercept** (50) en spécifiant l'option L. Cela entraîne l'acheminement de toutes les frappes vers le programme d'application de filtrage.
3. Le programme d'application de filtrage peut désormais définir une boucle dans laquelle :
    a. La fonction **Get Key** (51) intercepte toutes les frappes envoyées vers l'espace de présentation cible.
    b. L'application de filtrage examine chaque frappe et exécute une tâche de macro clavier, telle que :

- Abréviation des commandes de programme afin que les commandes à trois ou quatre touches puissent être condensées en une seule frappe
- Personnalisation des commandes afin qu'elles soient plus faciles à retenir ou cohérentes avec d'autres progiciels
- Création de **textes standard** pour les contrats ou les lettres fréquemment utilisées
- Réorganisation du clavier pour les applications simultanées qui utilisent les mêmes touches pour différentes fonctions

Par exemple, l'application de filtrage peut convertir une combinaison de touches telle que Alt+Y en une commande pour déplacer le curseur vers la colonne 35 de la deuxième ligne de l'espace de présentation et écrire la chaîne « XYZ Tool Corporation, Dallas, Texas ».

c. Si une frappe est rejetée, votre programme EHLLAPI peut provoquer l'émission d'un bip, en utilisant la fonction **Post Intercept Status** (52).

4. Une fois que votre programme EHLLAPI quitte la boucle de filtrage, la fonction **Stop Keystroke Intercept** (53) met fin au processus de filtrage.

Chapter 3. EHLLAPI Fonctions

Ce chapitre décrit chaque fonction Z and I Emulator for WindowsEHLLAPI individuelle en détail et explique comment utiliser l'échantillonneur de programme EHLLAPI. Les fonctions sont classées par ordre alphabétique de nom. Les fonctions sont expliquées pour les interfaces standard et améliorées.

Note: Dans tout ce chapitre, WinHLLAPI, IBM® Standard 32 bits HLLAPI et 16 bits EHLLAPI sont appelés interface standard, et IBM® Enhanced 32 bits EHLLAPI est appelé interface améliorée.

Conventions de mise en page

Tous les appels de fonction EHLLAPI sont présentés dans le même format afin que vous puissiez récupérer rapidement les informations dont vous avez besoin. Son format est le suivant :

- Nom de la fonction (numéro de la fonction)
  - Appels prérequis
  - Paramètres d'appel
  - Paramètres de retour
  - Notes sur l'utilisation de cette fonction

Appels prérequis

« Appels prérequis » répertorie tous les appels qui doivent être effectués avant d'appeler la fonction en cours de discussion.

Paramètres d'appel

« Call Parameters » répertorie les paramètres qui doivent être définis dans votre programme pour appeler la fonction EHLLAPI abordée et explique comment ces paramètres doivent être définis. Si un paramètre n'est jamais utilisé par une fonction, alors *NA* (non applicable) est répertorié. Si un paramètre peut être remplacé par certaines valeurs des paramètres de session définis avec des appels à la fonction **Set Session Parameters** (9), ces paramètres de session sont nommés.

Paramètres de retour

« Paramètres de retour » répertorie les paramètres qui doivent être reçus par votre programme après un appel à la fonction EHLLAPI discutée et explique comment interpréter ces paramètres.

Notes sur l'utilisation de cette fonction

« Notes sur l'utilisation de cette fonction » répertorie toutes les options de session qui affectent la fonction en discussion. Il fournit également des informations techniques sur l'utilisation de la fonction et des conseils de développement d'applications.

Résumé des fonctions EHLLAPI

Table 6: Résumé des fonctions EHLLAPI on page 32 est le résumé des fonctions EHLLAPI :

Table 6. Résumé des fonctions EHLLAPI

Fonctionnalité	3270	5250	VT
Connect Presentation Space (1) on page 41	Oui	Oui	Oui
Disconnect Presentation Space (2) on page 77	Oui	Oui	Oui
Send Key (3) on page 131	Oui	Oui	Oui
Wait (4) on page 167	Oui	Oui	Oui
Copy Presentation Space (5) on page 59	Oui	Oui	Oui
Search Presentation Space (6) on page 128	Oui	Oui	Oui
Query Cursor Location (7) on page 104	Oui	Oui	Oui
Copy Presentation Space to String (8) on page 64	Oui	Oui	Oui
Set Session Parameters (9) on page 142	Oui	Oui	Oui
Query Sessions (10) on page 111	Oui	Oui	Oui
Reserve (11) on page 124	Oui	Oui	Oui
Release (12) on page 123	Oui	Oui	Oui
Copy OIA (13) on page 50	Oui	Oui	Oui
Query Field Attribute (14) on page 105	Oui	Oui	Oui
Copy String to Presentation Space (15) on page 70	Oui	Oui	Oui
Pause (18) on page 96	Oui	Oui	Oui
Query System (20) on page 113	Oui	Oui	Oui
Reset System (21) on page 125	Oui	Oui	Oui
Query Session Status (22) on page 108	Oui	Oui	Oui
Start Host Notification (23) on page 157	Oui	Oui	Oui
Query Host Update (24) on page 107	Oui	Oui	Oui
Stop Host Notification (25) on page 165	Oui	Oui	Oui
Search Field (30) on page 126	Oui	Oui	Oui
Find Field Position (31) on page 82	Oui	Oui	Oui
Find Field Length (32) on page 81	Oui	Oui	Oui
Copy String to Field (33) on page 69	Oui	Oui	Oui

Table 6. Résumé des fonctions EHLLAPI (continued)

Fonctionnalité	3270	5250	VT
Copy Field to String (34) on page 46	Oui	Oui	Oui
Copy Presentation Space to Clipboard (35) on page 72	Oui	Oui	Oui
Paste Clipboard to Presentation Space (36) on page 74	Oui	Oui	Oui
Set Cursor (40) on page 141	Oui	Oui	Oui
Start Close Intercept (41) on page 152	Oui	Oui	Oui
Query Close Intercept (42) on page 100	Oui	Oui	Oui
Stop Close Intercept (43) on page 163	Oui	Oui	Oui
Interroger un attribut de champ supplémentaire DRB on page 99	Non	Oui	Non
Start Keystroke Intercept (50) on page 160	Oui	Oui	Oui
Get Key (51) on page 85	Oui	Oui	Oui
Post Intercept Status (52) on page 98	Oui	Oui	Oui
Arrêter l'interception de frappe (53) on page 166	Oui	Oui	Oui
Lock Presentation space API (60) on page 92	Oui	Non	Non
Lock Window Services API (61) on page 94	Oui	Non	Non
Start Communication Notification (80) on page 155	Oui	Oui	Oui
Query Communication Event (81) on page 103	Oui	Oui	Oui
Arrêter la notification de communication (82) on page 164	Oui	Oui	Oui
Send File (90) on page 184	Oui	Oui	Non
Receive File (91) on page 121	Oui	Oui	Non
Cancel File Transfer (92) on page 35	Oui	Oui	Oui
Convert Position or Convert RowCol (99) on page 44	Oui	Oui	Oui
Connect Window Services (101) on page 42	Oui	Oui	Oui
Disconnect Window Service (102) on page 78	Oui	Oui	Oui
Query Window Coordinates (103) on page 114	Oui	Oui	Oui
Window Status (104) on page 168	Oui	Oui	Oui
Change Switch List LT Name (105) on page 37	Oui	Oui	Oui
Change PS Window Name (106) on page 36	Oui	Oui	Oui
Start Playing Macro (110) on page 162	Oui	Oui	Oui
Connect for Structured Fields (120) on page 39	Oui	Non	Non
Disconnect from Structured Fields (121) on page 75	Oui	Non	Non
Query Communications Buffer Size (122) on page 101	Oui	Non	Non
Allocate Communications Buffer (123) on page 34	Oui	Non	Non
Free Communications Buffer (124) on page 84	Oui	Non	Non
Get Request Completion (125) on page 89	Oui	Non	Non
Read Structured Fields (126) on page 116	Oui	Non	Non

Table 6. Résumé des fonctions EHLLAPI (continued)

Fonctionnalité	3270	5250	VT
Write Structured Fields (127) on page 173	Oui	Non	Non

Allocate Communications Buffer (123)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Allocate Communications Buffer** obtient un tampon du système d'exploitation. Une adresse de tampon doit être transmise à la fois aux fonctions **Read Structured Fields** (126) et **Write Structured Fields** (127).

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 123	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 6	Doit être 8
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1–2	1–4	Longueur de tampon de 32 bits ou 16 bits. (0 < taille ≤ (64 Ko-256 octets)=X'FF00')
3–6	5–8	Adresse de tampon allouée sur 32 bits (renvoyée)

Paramètres de retour

Code retour	Explication
0	La fonction **Allocate Communications Buffer** a réussi.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
11	Ressource non disponible (mémoire non disponible).

Notes sur l'utilisation de cette fonction

1. Le EHLLAPI obtient un tampon de la gestion de la mémoire du système d'exploitation et place l'adresse du tampon dans la chaîne de paramètres de retour. La taille (longueur) du tampon demandée est également transmise dans la chaîne de paramètres. La taille de la mémoire tampon peut aller de 1 octet à 64 Ko moins 256 octets (X'FF00' octets).

   Consultez *"**Query Communications Buffer Size** (122)"* pour plus d'informations sur la taille du tampon.
2. Les tampons obtenus à l'aide de cette fonction ne doivent pas être partagés entre différents processus. Si cela est tenté, les applications connaîtront des résultats imprévisibles.
3. Une application EHLLAPI doit émettre une fonction **Free Communications Buffer** (124) pour libérer la mémoire allouée.
4. Un maximum de 10 tampons peuvent être alloués à une application. Si cette limite est atteinte, un code retour pour ressource indisponible (RC=11) sera renvoyé.
5. La fonction **Reset System** (21) libère les tampons alloués par cette fonction.

Cancel File Transfer (92)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Cancel File Transfer** entraîne le retour immédiat de toute fonction **Send File** ou **Receive File** lancé par EHLLAPI pour la session spécifiée.

Appels prérequis

Send File (90) ou **Receive File** (91)

Paramètres d'appel

	Interface améliorée
Numéro de fonction	Doit être 92
Chaîne de données	Nom court à 1 caractère de l'espace de présentation hôte. Un espace vide ou une valeur Null indique une demande de mise à jour de l'espace de présentation connecté à l'hôte.
Longueur	4 est implicite
Position PS	NA

La structure de données appelante contient ces éléments

Octet	**Définition**
1	Un nom court d'espace de présentation (PSID) à 1 caractère

Octet	Définition
2-4	Réservé

Paramètres de retour

Code retour	Définition
0	La fonction a réussi
1	Un PSID incorrect a été spécifié
8	Aucun appel préalable à la fonction **Start Communication Notification** (80) n'a été appelé pour le PSID
9	Une erreur système a été rencontrée

Notes sur l'utilisation de cette fonction

Etant donné que **Send File** (90) et **Receive File** (91) bloquent les appels, cette fonction doit toujours être émise sur une unité d'exécution différente.

Change PS Window Name (106)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Change PS Window Name** permet à l'application de spécifier un nouveau nom pour la fenêtre de l'espace de présentation ou de réinitialiser la fenêtre de l'espace de présentation au nom par défaut.

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 106	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être spécifié (Voir note.)	Doit être 68
Position PS	NA	

Note: La longueur de la chaîne de données doit être spécifiée (normalement 3 à 63 pour PC/3270, 4–63 pour PC400, 68 pour une interface améliorée).

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2	5	Une valeur d'option de demande de modification, sélectionnez l'une des valeurs suivantes : • X'01' pour changer le nom de la fenêtre de l'espace de présentation. • X'02' pour réinitialiser le nom de la fenêtre de l'espace de présentation.
3–63	6–66	Une chaîne ASCII de 1 (pour PC/3270) ou 2 (pour PC400) à 61 octets, y compris un octet de terminaison. La chaîne ASCII doit se terminer par un caractère NULL. Cette chaîne doit contenir au moins un caractère non NULL suivi d'un caractère NULL.
	67–68	Réservé

Paramètres de retour

Code retour	Explication
0	La fonction **Change PS Window Name** a réussi.
1	Un ID de session court de l'espace de présentation hôte incorrect a été spécifié ou l'espace de présentation hôte n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
12	La session s'est arrêtée.

Notes sur l'utilisation de cette fonction

Une chaîne se termine au premier caractère NULL trouvé. Le caractère NULL remplace la longueur de chaîne spécifiée. Si le caractère NULL ne se trouve pas à la fin de la longueur spécifiée, le dernier octet de la longueur spécifiée est remplacé par un caractère NULL et le reste de la chaîne de données est perdu. Si le caractère NULL est trouvé avant la longueur spécifiée, la chaîne est tronquée à ce stade et le reste de la chaîne de données est perdu.

Si l'application ne parvient pas à réinitialiser le nom de l'espace de présentation avant de quitter, le traitement de la liste de sortie réinitialise le nom.

Change Switch List LT Name (105)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Change Switch List LT Name** permet à l'application de modifier ou de réinitialiser une liste de commutateurs pour un terminal logique (LT) sélectionné. L'application doit préciser lors de l'appel le nom à insérer dans la liste des commutateurs.

Note: Ceci est pour la compatibilité avec Communication ManagerEHLLAPI, et a le même résultat que la fonction **Change PS Window Name** (106).

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 105	
Chaîne de données	Voir le tableau suivant	
Longueur	Normalement 4 à 63	Doit être 68
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2	5	Une option de demande de changement ; sélectionnez : • X'01' pour changer le nom LT d'une liste de commutateurs • X'02' pour réinitialiser le nom LT d'une liste de commutateurs
3–63	6–66	Une chaîne ASCII de 2 à 61 octets comprenant un octet de terminaison. La chaîne ASCII doit se terminer par un caractère NULL. Cette chaîne doit contenir au moins un caractère non NULL suivi d'un caractère NULL.
	67–68	Réservé

Paramètres de retour

Code retour	**Explication**
0	La fonction **Change Switch List LT Name** a réussi.
1	Un ID de session court de l'espace de présentation hôte incorrect a été spécifié ou l'espace de présentation hôte n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.

Code retour	Explication
9	Une erreur système s'est produite.
12	La session s'est arrêtée.

Notes sur l'utilisation de cette fonction

Une chaîne se termine au premier caractère NULL trouvé. Le caractère NULL remplace la longueur de chaîne spécifiée. Si le caractère NULL ne se trouve pas à la fin de la longueur spécifiée, le dernier octet de la longueur spécifiée est remplacé par un caractère NULL et le reste de la chaîne de données est perdu. Si le caractère NULL est trouvé avant la longueur spécifiée, la chaîne est tronquée à ce stade et le reste de la chaîne de données est perdu.

Si l'application ne parvient pas à réinitialiser le nom LT de la liste de commutateurs avant de quitter, le traitement de la liste de sortie réinitialise le nom.

Connect for Structured Fields (120)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Connect for Structured Fields** permet à une application d'établir une connexion au programme d'émulation pour échanger des données de champ structurées avec une application hôte. L'application du poste de travail doit fournir le champ de données Query Reply et doit pointer vers celui-ci dans la chaîne de paramètres. L'ID de destination/origine renvoyé par l'émulateur sera renvoyé à l'application.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 120	
Chaîne de données	Voir le tableau suivant	
Longueur	7 ou 11	Doit être 16
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2–5	5–8	Adresse du tampon de données de réponse à la requête

Octet		Définition
6–7	9–10	ID unique de destination/origine. (mot de 16 bits, renvoyé)
	11–12	Réservé
8–11	13–16	Les données à ces positions sont ignorées par EHLLAPI. Cependant, aucune erreur n'est générée si le programme de migration contient des données à ces positions. Ces données sont acceptées pour assurer la compatibilité avec les applications en migration.

Paramètres de retour

Code retour	Explication
0	La fonction **Connect for Structured Fields** a réussi.
1	Un ID de session court de l'espace de présentation hôte spécifié n'était pas valide ou l'espace de présentation hôte n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
10	La fonction n'est pas prise en charge par le programme d'émulation.
32	Une application s'est déjà connectée à cette session pour les communications (connexion réussie).
39	Une session DDM est déjà connectée à cette session.

Notes sur l'utilisation de cette fonction

1. EHLLAPI analyse les tampons de réponse à la requête pour le paramètre autodéfinissant (SDP) de l'ID de destination/origine (DOID) afin de déterminer le contenu du champ DOID de la réponse à la requête. Si cette valeur est X'0000', l'émulateur attribuera un DOID à l'application et EHLLAPI remplira le champ DOID de la réponse à la requête avec l'ID attribué. Si la valeur spécifiée par l'application dans le champ DOID de la réponse à la requête est une valeur différente de zéro, l'émulateur attribuera la valeur spécifiée comme DOID de l'application, en supposant que l'ID n'a pas été précédemment attribué. Si le DOID spécifié est déjà utilisé, un code retour de 2 sera renvoyé par EHLLAPI.
2. L'application doit créer les structures de données de réponse à la requête dans la mémoire privée de l'application. Reportez-vous à Structures de données de réponse à la requête prises en charge par EHLLAPI on page 222, pour les formats et utilisations détaillés des structures de données de réponse aux requêtes prises en charge par EHLLAPI.
3. Seule une vérification superficielle est effectuée sur les données de réponse à la requête. Seule la validité de l'ID et de la longueur de la structure est vérifiée.

4. Une seule connexion de type de base DDM est autorisée par session hôte. Si la connexion DDM prend en charge le paramètre autodéfinissant (SDP) pour l'ID d'origine de destination (DOID), plusieurs connexions sont autorisées.
5. Si le code retour RC=32 ou RC=39 est reçu, une application est déjà connectée à la session sélectionnée et l'utilisation de cet espace de présentation doit être abordée avec prudence. Des conflits avec le transfert de fichier et d'autres applications EHLLAPI peuvent en résulter.

Connect Presentation Space (1)

3270	***5250***	***VT***
Oui	Oui	Oui

La fonction **Connect Presentation Space** établit une connexion entre votre programme d'application EHLLAPI et l'espace de présentation hôte.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 1	
Chaîne de données	Nom court à 1 caractère de l'espace de présentation hôte	
Longueur	1 est implicite	Doit être 4
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

La fonction **Connect Présentation Space** définit le code retour pour indiquer le statut de la tentative et, en cas de réussite, le statut de l'espace de présentation hôte.

Code retour	**Explication**
0	La fonction **Connect Présentation Space** a réussi ; l'espace de présentation hôte est déverrouillé et prêt à être saisi.

Code retour	Explication
1	Un ID d'espace de présentation hôte incorrect a été spécifié. La session spécifiée n'existe pas ou est une session d'imprimante logique. Ce code retour peut également signifier que le paramètre API pour EHLLAPI n'est pas activé.
4	La connexion a été établie, mais l'espace de présentation hôte est occupé.
5	La connexion a été établie avec succès, mais l'espace de présentation hôte est verrouillé (entrée interdite).
9	Une erreur système a été rencontrée.
11	Cette ressource n'est pas disponible. L'espace de présentation hôte est déjà utilisé par une autre fonction système.

Notes sur l'utilisation de cette fonction

1. La fonction **Connect Présentation Espace** est affectée par l'option de session `CONLOG`/`CONPHYS`.
2. Une application EHLLAPI ne peut pas être connectée simultanément à plusieurs espaces de présentation. Les appels nécessitant la fonction **Connect Presentation Space** comme prérequis utilisent l'espace de présentation actuellement connecté. Par exemple, si une application est connectée aux espaces de présentation A, B et C dans cet ordre, l'application doit se connecter à nouveau à B ou A pour émettre des fonctions.
3. Chaque unité d'exécution qui demande une fonction **Connect Presentation Space** doit avoir une fonction **Disconnect Presentation Space** (2) correspondante, ou l'une des unités d'exécution doit émettre une fonction **Reset System** (21), qui affecte toutes les unités d'exécution et déconnecte toutes les connexions restantes.
4. Plusieurs applications EHLLAPI peuvent partager un espace de présentation, si les applications prennent en charge le partage (c'est-à-dire si elles ont été développées pour fonctionner ensemble et si elles présentent un comportement prévisible) et disposent d'un accès en lecture/écriture et d'options de mots clés compatibles, comme défini dans la fonction **Set Sessions Parameters** (9). Pour plus d'informations, voir Set Session Parameters (9) on page 142.
5. Etant donné que les fonctions **Connect Presentation Space** et **Start Keystroke Intercept** (50) partagent des fonctions de sous-système communes, les demandes réussies d'une application visant à partager l'une ou l'autre de ces fonctions pour la même session peuvent affecter la demande de ces deux fonctions par d'autres applications. Par exemple, si l'application A demande avec succès une fonction **Connect Presentation Space** pour une session avec un accès Write_Read et KEY$abcdefgh comme mot-clé, une demande de l'application B pour la fonction **Connect Presentation Space** pour une session et **Start Keystroke Intercept** aboutit uniquement si les deux applications ont défini des options de lecture/écriture compatibles.
6. Vous ne pouvez pas vous connecter à une session définie comme session d'imprimante logique. Consultez *Guide d'administration et de référence* pour plus d'informations.

Connect Window Services (101)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Connect Window Services** permet à l'application de gérer les fenêtres de l'espace de présentation. Seulement une application EHLLAPI à la fois peut être connectée à un espace de présentation pour les services Windows.

Une application EHLLAPI peut se connecter simultanément à plusieurs espaces de présentation pour les services Windows.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 101	
Chaîne de données	ID de session court à 1 caractère de l'espace de présentation hôte	
Longueur	1 est implicite	Doit être 4
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

Code retour	**Explication**
0	La fonction **Connect Window Services** a réussi.
1	Un ID de session court de l'espace de présentation hôte incorrect a été spécifié ou le gestionnaire des services de fenêtre de sessions n'était pas connecté. Ce code retour peut également signifier que le paramètre API pour EHLLAPI n'est pas activé.
9	Une erreur système s'est produite.
10	La fonction n'est pas prise en charge par le programme d'émulation.
11	Cette ressource n'est pas disponible. L'espace de présentation hôte est déjà utilisé par une autre fonction système.

Notes sur l'utilisation de cette fonction

1. Une application EHLLAPI peut être connectée à plusieurs fenêtres d'espace de présentation en même temps. L'application peut faire des allers-retours entre les fenêtres de l'espace de présentation connectées sans avoir à se déconnecter. Par exemple, si une application est connectée aux fenêtres de l'espace de présentation A, B et C, l'application peut accéder à A, B et C en même temps, et les autres applications ne peuvent pas accéder à A, B ou C.
2. Une fonction **Connect Window Services** est suffisante pour le processus. Cependant, chaque unité d'exécution qui demande une fonction **Connect Window Services** doit avoir une fonction **Disconnect Window Services** (102) correspondante, ou l'une des unités d'exécution doit émettre une fonction **Reset System** (21), qui affecte toutes les unités d'exécution et déconnecte toutes les connexions restantes.

Convert Position or Convert RowCol (99)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Convert Position** ou **Convert RowCol** convertit la valeur de position de l'espace de présentation hôte en coordonnées de ligne et de colonne d'affichage ou convertit les coordonnées de ligne et de colonne d'affichage en valeur de position de l'espace de présentation hôte. Cette fonction ne modifie pas la position du curseur.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 99	
Chaîne de données	Nom court de l'espace de présentation hôte *et*`P` pour la fonction **Convert Position** (par exemple, `AP` convertit la position de l'espace de présentation de la session A) ; *ou* Nom court de l'espace de présentation hôte et `R` pour la fonction **Convert RowCol** par exemple, `AR` convertit les coordonnées de ligne et de colonne de la session A).	
Longueur	Ligne, lorsque `R` est spécifié comme deuxième caractère dans le paramètre de chaîne de données. La limite inférieure pour une entrée valide est 1. La limite supérieure des entrées valides dépend de la configuration de votre espace de présentation hôte. Voir Notes sur l'utilisation de cette fonction on page 46.	

	Interface standard	Interface améliorée
	NA lorsque P est spécifié comme deuxième caractère dans le paramètre de chaîne de données.	
Position PS	Colonne, lorsque R est spécifié comme deuxième caractère dans le paramètre de chaîne de données. La limite inférieure pour une entrée valide est 1. La limite supérieure des entrées valides varie de 24 à 43 selon la configuration de votre espace de présentation hôte. Voir Notes sur l'utilisation de cette fonction on page 46. Position de l'espace de présentation hôte, lorsque P est spécifié comme deuxième caractère dans le paramètre de chaîne de données. La limite inférieure pour une entrée valide est 1. La limite supérieure des entrées valides va de 1 920 à 3 564 selon la configuration de votre espace de présentation hôte. Voir Notes sur l'utilisation de cette fonction on page 46.	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2	5	Option**Convert** P ou R
	6–8	Réservé

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Pour la fonction **Convert Position** (P comme deuxième caractère dans la chaîne de données appelante), un nombre compris entre 1 et 43 (pour PC/3270) ou 27 (pour PC400) est renvoyé. Cette valeur est le numéro de la ligne qui contient la position PS contenue dans le paramètre de position PS appelant. La limite supérieure peut être inférieure à 43 (pour PC/3270) ou 27 (pour PC400) en fonction de la configuration de l'espace de présentation hôte.

Pour la fonction **Convert RowCol** (R comme deuxième caractère dans la chaîne de données appelante), une valeur de 0 indique une erreur dans la valeur d'entrée pour la ligne (paramètre de longueur d'appel).

Code retour :

La fonction **Convert Position or RowCol** est l'exception à la règle selon laquelle le quatrième paramètre de retour contient toujours un code retour. Pour cette fonction, la valeur renvoyée dans le quatrième paramètre est appelée code de statut. Ce code de statut peut contenir des données ou un code retour.

Votre application doit prévoir le traitement de ce code de statut pour éviter des résultats imprévisibles ou une erreur.

- Si la valeur du quatrième paramètre est 0, 9998 ou 9999, il s'agit d'un code retour.
- Pour la fonction **Convert Position** (`P` comme deuxième caractère de la chaîne de données appelante), une valeur comprise entre 1 et 132 est le numéro de la colonne qui contient la position PS transmise dans le paramètre PS Position appelant. La limite supérieure peut être inférieure à 132 selon la configuration de l'espace de présentation hôte.
- Pour la fonction **Convert RowCol** (`R` comme deuxième caractère de la chaîne de données appelante), une valeur comprise entre 1 et 3 564 représente la position de l'espace de présentation hôte qui correspond aux valeurs de ligne et de colonne transmises dans les paramètres de longueur d'appel et de position PS, respectivement. La limite supérieure peut être inférieure à 3 564 selon la configuration de l'espace de présentation hôte.

Les codes de statut suivants sont définis :

Code de statut	Explication
0	Il s'agit d'une position ou d'une colonne PS incorrecte.
>0	Il s'agit de la position ou de la colonne PS.
9 998	Un ID d'espace de présentation hôte incorrect a été spécifié ou une erreur système s'est produite.
9999	Le caractère 2 de la chaîne de données n'est ni P ni R.

Notes sur l'utilisation de cette fonction

1. Pour configurer votre espace de présentation, reportez-vous à *Guide d'administration et de référence*
2. Pour savoir combien de lignes et de colonnes se trouvent dans votre espace de présentation, examinez le paramètre de chaîne de données renvoyé pour la fonction **Query Session Status** (22). Voir Query Session Status (22) on page 108.

Copy Field to String (34)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy Field to String** transfère les caractères d'un champ de l'espace de présentation connecté à l'hôte vers une chaîne.

La fonction **Copy Field to String** traduit les caractères de l'espace de présentation de la source hôte en ASCII (American National Standard Code for Information Interchange). Les octets d'attribut et autres caractères non représentés en ASCII sont normalement traduits en espaces.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 34	
Chaîne de données	Chaîne de données cible préallouée. Lorsque la fonction **Set Session Parameters** (9) avec l'option EAB (Extended Attribute Bytes) est émise, la longueur de la chaîne de données doit être au moins deux fois la longueur du champ.	
Longueur	Nombre d'octets à copier (la longueur de la chaîne de données).	
Position PS	Identifie le champ cible. Il peut s'agir de la position PS de n'importe quel octet dans le champ cible. La copie commence toujours au début du champ.	

Paramètres de retour

Cette fonction renvoie une chaîne de données, une longueur et un code retour.

Chaîne de données :

Chaîne contenant les données du champ identifié dans l'espace de présentation hôte. Le premier octet de la chaîne de données renvoyée est le début du champ identifié dans l'espace de présentation hôte. Le nombre d'octets dans la chaîne de données renvoyée est déterminé par le plus petit des deux :

- Nombre d'octets spécifié dans le paramètre de longueur d'appel
- Nombre d'octets dans le champ identifié dans l'espace de présentation hôte

Longueur :

La longueur des données renvoyées.

Code retour :

Les codes suivants sont définis :

Code retour	**Explication**
0	La fonction **Copy Field to String** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification des paramètres.
6	Les données à copier et le champ cible n'ont pas la même taille. Les données sont tronquées si la longueur de la chaîne est inférieure au champ copié.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.

Code retour	Explication
24	Espace de présentation hôte non formaté.

Notes sur l'utilisation de cette fonction

1. Les informations sur la position et la longueur du champ peuvent être trouvées à l'aide des fonctions **Find Field Position** (31) et **Find Field Length** (32). La fonction **Copy Field to String** peut être utilisée avec des champs protégés ou non, mais uniquement dans un espace de présentation hôte *formaté par champ*.
2. La copie est terminée lorsque l'une des conditions suivantes est remplie :
   - Lorsque la fin du champ est atteinte
   - Lorsque la longueur de la chaîne cible est dépassée
3. Un EAB peut être renvoyé lorsque l'option EAB de la fonction **Set Session Parameters** (9) est utilisée. EAB est lié à chaque caractère de l'espace de présentation et est renvoyé avant chaque caractère.
4. La fonction **Copy Field to String** est affectée par les options de session `ATTRB`/`NOATTRB`/`NULLATTRB`, `EAB`/`NOEAB`, `XLATE`/`NOXLATE`, `DISPLAY`/`NODISPLAY`, `DISPLAY`/`NODISPLAY`. Voir les éléments 5 on page 145, 13 on page 148, 14 on page 149 et 17 on page 149 pour plus d'informations.
   Comme indiqué précédemment, le retour des attributs par les différentes fonctions de **copie** (5, 8 et 34) est affecté par la fonction **Set Session Parameters** (9). Les paramètres de session définis impliqués ont l'effet suivant :

   Définir le paramètre de session

   Effet sur la fonction de copie

   NOEAB et NOEAD

   Les attributs ne sont pas renvoyés. Seul le texte est copié de l'espace de présentation vers le tampon utilisateur.

   EAB et NOXLATE

   Les attributs sont renvoyés tels que définis dans les tableaux suivants.

   EAB et XLATE

   Les couleurs utilisées pour l'affichage de l'espace de présentation sont renvoyées. Les couleurs peuvent être remappées ; les couleurs des attributs ne sont donc pas celles renvoyées par les fonctions **Copy** lorsque XLATE et EAB sont activés en même temps.

   Les attributs de caractères renvoyés sont définis dans les tableaux suivants. Les positions des bits d'attribut sont au format IBM®, le bit 0 étant le bit le plus à gauche de l'octet.

   Les attributs de caractères 3270 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0–1	Mise en évidence des caractères 00 = Normale 01 = Clignotement

Position des bits	Signification
	10 = Vidéo inversée 11 = Soulignement
2–4	Couleur des caractères (le remappage des couleurs peut remplacer cette définition de couleur.) 000 = Par défaut 001 = Bleu 010 = Rouge 011 = Rose 100 = Vert 101 = Turquoise 110 = Jaune 111 = Blanc
5–6	Attributs de police 00 = Valeur par défaut
7	Réservé

Les attributs de caractères 5250 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0	Image inversée 0 = Image normale 1 = Image inversée
1	Soulignement 0 = Pas de soulignement 1 = Soulignement
2	Clignotement 0 = Aucun clignotement 1 = Clignotement
3	Séparateur de colonnes 0 = Aucun séparateur 1 = Séparateur
4–7	Réservé

Le tableau suivant montre les attributs de couleur des caractères Z and I Emulator for Windows. Le tableau suivant s'applique lorsque EAB et XLATE sont définis.

Position des bits	Signification
0–3	Couleurs des caractères d’arrière-plan 0000 = Noir 0001 = Bleu 0010 = Vert

Position des bits	Signification
	0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc
4–7	Couleurs des caractères de premier plan 0000 = Noir 0001 = Bleu 0010 = Vert 0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc 1000 = Gris 1001 = Bleu clair 1010 = Vert clair 1011 = Cyan clair 1100 = Rouge clair 1101 = Magenta clair 1110 = Jaune 1111 = Blanc (haute intensité)

Pour un écran monochrome PS/2®, les caractères de la session d'application (poste de travail) apparaissent sous différentes nuances de gris. Ceci est nécessaire pour donner aux utilisateurs leurs couleurs remappées dans la session d'application EHLLAPI afin qu'ils puissent obtenir ce qu'ils voient dans leurs espaces de présentation d'application hôte.

5. Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la 25e ligne sur la barre d'état. Par l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Copy OIA (13)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy OIA** renvoie les données actuelles de la zone d'informations de l'opérateur (OIA) à partir de l'espace de présentation connecté à l'hôte.

L'OIA est situé sous la ligne de séparation inférieure de l'écran et est utilisé pour afficher des informations sur l'état de la session concernant la connexion entre le poste de travail et l'hôte.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 13	
Chaîne de données	Chaîne de données cible préallouée	
Longueur	103	104
Position PS	NA	

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Une chaîne de 103 octets pour 16 bits et une chaîne de 104 octets pour 32 bits. Consultez Format de la chaîne de données OIA renvoyée on page 52 pour plus d'informations.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	Les données OIA sont renvoyées. L'espace de présentation cible est déverrouillé.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification de la longueur de la chaîne. Les données OIA n'ont pas été renvoyées.
4	Les données OIA sont renvoyées. L'espace de présentation cible est occupé.
5	Les données OIA sont renvoyées. L'espace de présentation cible est verrouillé. (Entrée bloquée)
9	Une erreur système interne a été rencontrée. Les données OIA n'ont pas été renvoyées.

Notes sur l'utilisation de cette fonction

1. Le groupe OIA se compose des bits qui affichent le statut des sessions connectées. Le groupe est classé selon la fonction hôte représentée. (Par exemple, le groupe 8 comprend les bits qui affichent toutes les conditions d'interdiction d'entrée dans la session.) Les états de chaque groupe sont classés de manière à ce que les bits de poids fort représentent les indicateurs de priorité plus élevée. Autrement dit, le bit 7 a la priorité sur le bit 0. Par conséquent, si plusieurs états sont actifs au sein d'un groupe, l'état ayant la priorité la plus élevée est l'état actif au sein de ce groupe.
2. Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Format de la chaîne de données OIA renvoyée

La chaîne de données OIA contient les informations suivantes :

Octet		**Définition**
Standard	Etendu	
1	1	L'octet de format OIA. La valeur est 1 (PC/3270), 9 (PC400) ou 5 (VT).
2–81	2–81	L'image OIA dans les points de code hôte.
82–103	82–103	Signification des indicateurs du groupe OIA.
	104	Réservé.

PC/3270 Signification des indicateurs du groupe OIA et son image

Le groupe d'images OIA se compose d'une chaîne de caractères ASCII de 80 octets sans octets d'attribut qui contient l'image OIA dans les points de code hôte. Figure 2: Caractères de l'espace de présentation hôte on page 53 affiche les codes hexadécimaux trouvés dans l'espace de présentation de l'hôte et les caractères qu'ils représentent. Les données renvoyées peuvent être traduites en caractères graphiques OIA. Reportez-vous à *Guide d'initiation* pour plus d'informations sur les indicateurs de l'OIA.

Pour traduire les données renvoyées en caractères graphiques OIA, procédez comme suit :

1. Imprimez les données renvoyées dans les octets 2 à 81 sur l'écran ou sur une imprimante.
2. A l'aide du tableau des pages de codes applicable à l'appareil sur lequel la sortie apparaît, recherchez la valeur hexadécimale correspondant à chaque caractère.
3. En utilisant Figure 2: Caractères de l'espace de présentation hôte on page 53, recherchez le caractère graphique OIA correspondant à chaque valeur hexadécimale trouvée à l'étape 2.

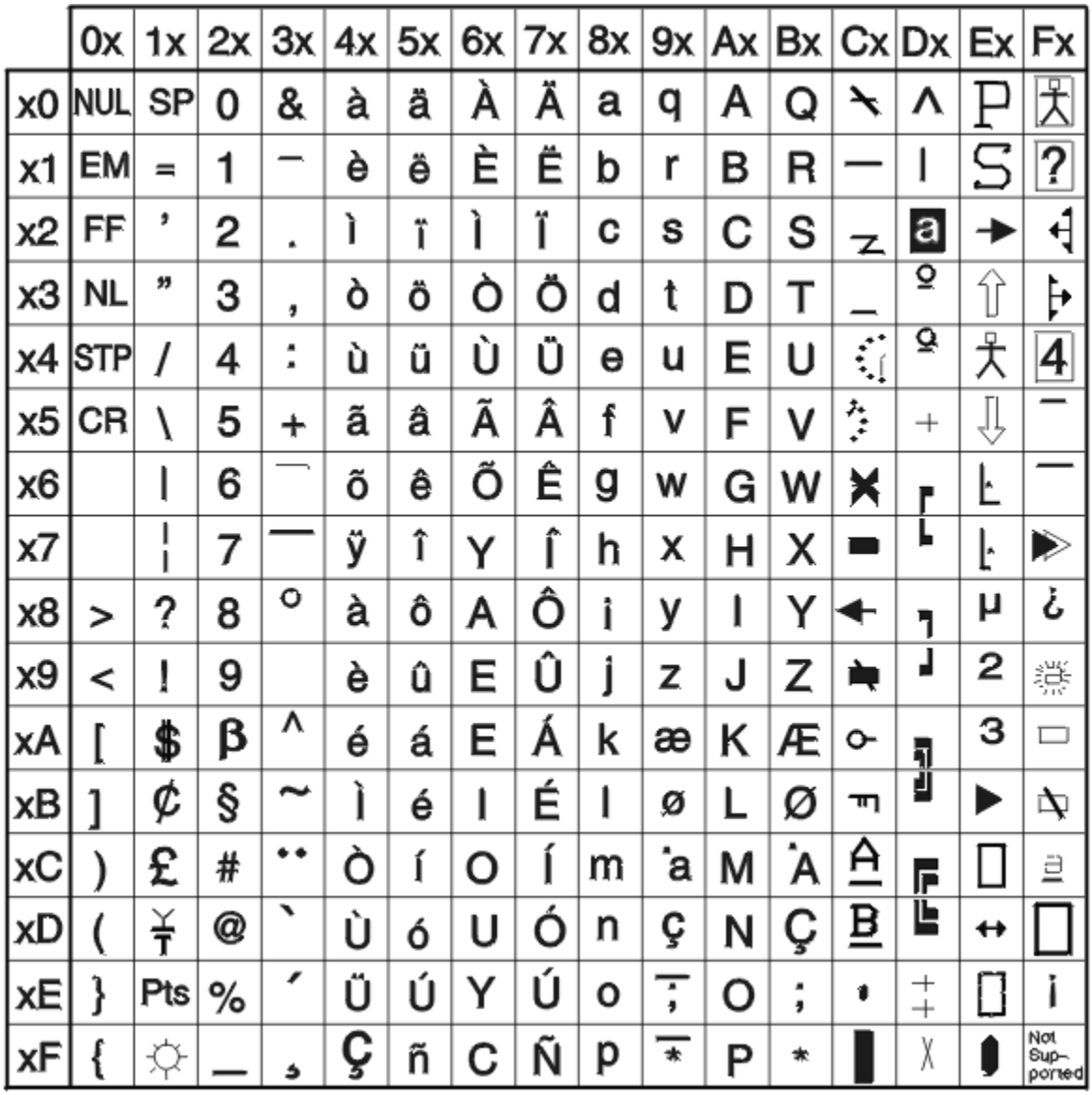
Note: Les images de contrôle de machine, de communications et de programme du groupe 8 (octet 0) sont suivies d'un numéro à trois chiffres lié au type de contrôle.

L'ID de session court suivi de X'20' se trouve dans la colonne 7.

Toutes les images de groupe sont représentées par des points de code hexadécimaux MFI (Main Frame Interactive).

Note: La position de la chaîne de données d'image OIA moins 1 position est égale à la colonne OIA.

Figure 2. Caractères de l'espace de présentation hôte



- Groupe 1 (Décalage 82) : propriété en ligne et sur écran

Bit	Signification
0–1	Réservé
2	La session SSCP-LU possède un écran
3	La session LU-LU possède un écran
4	En ligne et sans propriétaire
5	Sous-système prêt
6–7	Réservé

- Groupe 2 (Décalage 83) : sélection de caractères

Bit	Signification
0	Réservé
1	APL
3	Alphanumérique
4–5	Réservé

- Groupe 3 (Décalage 84) : état de décalage

Bit	Signification
0	Décalage supérieur
1	Numérique
2	CAPS
3–7	Réservé

- Groupe 4 (Décalage 85) : PSS groupe 1

Bit	Signification
0–7	Réservé

- Groupe 5 (Décalage 86) : mettre en évidence le groupe 1

Bit	Signification
0	Opérateur sélectionnable
1	Champ hérité
2–7	Réservé

- Groupe 6 (Décalage 87) : groupe de couleurs 1

Bit	Signification
0	Opérateur sélectionnable
1	Champ hérité
2–7	Réservé

- Groupe 7 (Décalage 88) : insérer

Bit	Signification
0	Mode insertion

Bit	Signification
1–7	Réservé

- Groupe 8 (Décalage 89-93) : entrée interdite (5 octets)
  - Octet 1 (Décalage 89)

Bit	Signification
0	Contrôle de machine non réinitialisable
1	Réservé
2	Contrôle de machine
3	Contrôle de communications
4	Contrôle de programme
5–7	Réservé

  - Octet 2 (Décalage 90)

Bit	Signification
0	Appareil occupé
1	Attente du terminal
2	Symbole moins
3	Fonction moins
4	Trop d'éléments entrés
5–7	Réservé

  - Octet 3 (Décalage 91)

Bit	Signification
0–2	Réservé
3	Combinaison de touches mortes incorrecte, clé limitée.
4	Mauvais endroit
5–7	Réservé

  - Octet 4 (Décalage 92)

Bit	Signification
0–1	Réservé
2	Attente du système
3–7	Réservé

  - Octet 5 (Décalage 93)

Bit	Signification
0–7	Réservé

- Groupe 9 (Décalage 94) : PSS groupe 2

Bit	Signification
0–7	Réservé

- Groupe 10 (Décalage 95) : mettre en évidence le groupe 2

Bit	Signification
0–7	Réservé

- Groupe 11 (Décalage 96) : groupe de couleurs 2

Bit	Signification
0–7	Réservé

- Groupe 12 (Décalage 97) : rappel d'erreur de communication

Bit	Signification
0-6	Erreur de communication
1–7	Réservé

- Groupe 13 (Décalage 98) : état de l'imprimante

Bit	Signification
0–7	Réservé

- Groupe 14 (Décalage 99) : graphiques

Bit	Signification
0–7	Réservé

- Groupe 15 (Décalage 100) : réservé
- Groupe 16 (Décalage 101) : état de lecture/enregistrement automatique des touches

Bit	Signification
0–7	Réservé

- Groupe 17 (Décalage 102) : état de sortie/arrêt automatique de la clé

Bit	Signification
0–7	Réservé

- Groupe 18 (Décalage 103) : état étendu

Bit	Signification
0–7	Réservé

PC400 Signification des indicateurs du groupe OIA et son image

Les détails du groupe OIA sont répertoriés dans les tableaux suivants.

- Groupe 1 (Décalage 82) : propriété en ligne et sur écran

Bit	Signification	Position de départ de la chaîne de données
0–2	Réservé	
3	Système disponible	1
4	Réservé	

Bit	Signification	Position de départ de la chaîne de données
5	Sous-système prêt	
6–7	Réservé	

- Groupe 2 (Décalage 83) : sélection de caractères

Bit	Signification	Position de départ de la chaîne de données
0–1	Réservé	
3	Alphanumérique	
4–5	Réservé	

- Groupe 3 (Décalage 84) : état de décalage

Bit	Signification	Position de départ de la chaîne de données
0	Réservé	
1	Maj du clavier	39
2	CAPS	
3–6	Réservé	

- Groupe 4 (Décalage 85) : PSS groupe 1

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

- Groupe 5 (Décalage 86) : mettre en évidence le groupe 1

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

- Groupe 6 (Décalage 87) : groupe de couleurs 1

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

- Groupe 7 (Décalage 88) : insérer

Bit	Signification	Position de départ de la chaîne de données
0	Mode insertion	68
1–7	Réservé	

- Groupe 8 (Décalage 89-93) : entrée interdite (5 octets)
  - Octet 1 (Décalage 89)

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

  - Octet 2 (Décalage 90)

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

  - Octet 3 (Décalage 91)

Bit	Signification	Position de départ de la chaîne de données
0–4	Réservé	
5	Erreur de saisie de l'opérateur	64
6–7	Réservé	

◦ Octet 4 (Décalage 92)

Bit	Signification	Position de départ de la chaîne de données
0–1	Réservé	
2	Attente du système	64
3–7	Réservé	

◦ Octet 5 (Décalage 93)

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 9 (Décalage 94) : PSS groupe 2

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 10 (Décalage 95) : mettre en évidence le groupe 2

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 11 (Décalage 96) : groupe de couleurs 2

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 12 (Décalage 97) : rappel d'erreur de communication

Bit	Signification	Position de départ de la chaîne de données
0	Erreur de communication	
1–5	Réservé	
7	Message en attente	3

• Groupe 13 (Décalage 98) : état de l'imprimante

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 14 (Décalage 99) : graphiques

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

• Groupe 15 (Décalage 100) : réservé

• Groupe 16 (Décalage 101) : état de lecture/enregistrement automatique des touches

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

- Groupe 17 (Décalage 102) : état de sortie/arrêt automatique de la clé

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

- Groupe 18 (Décalage 103) : état étendu

Bit	Signification	Position de départ de la chaîne de données
0–7	Réservé	

VT Signification des indicateurs du groupe hôte OIA et son image

Les détails du groupe OIA hôtes VT sont répertoriés dans les tableaux suivants.

- Groupe 1 (Décalage 82) : propriété en ligne et sur écran

Bit	Signification
5	Sous-système prêt

- Groupe 2 (Décalage 83) : sélection de caractères

Bit	Signification
0	Décalage supérieur
2	CAPS

- Groupe 7 (Décalage 88) : insérer

Bit	Signification
0	Mode insertion

Certaines colonnes de la ligne OIA affichent des messages différents pour VT par rapport aux messages affichés pour 3270/5250. Consultez le tableau suivant pour plus de détails.

Colonne	Symbole
1–7	VT220 7
	VT220 8
	VT100
	VT52
	VTANSI
9 - 12	VERROUILLE
61 - 64	HOLD

Copy Presentation Space (5)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy Presentation Space** copie le contenu de l'espace de présentation connecté à l'hôte dans une chaîne de données que vous définissez dans votre programme d'application EHLLAPI.

La fonction **Copy Presentation Space** traduit les caractères de l'espace de présentation source hôte en ASCII. Les octets d'attribut et autres caractères non représentés en ASCII sont normalement traduits en espaces. Si vous ne souhaitez pas que les octets d'attribut soient traduits en blancs, vous pouvez remplacer cette traduction avec l'option ATTRB sous la fonction **Set Session Parameters** (9).

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 5	
Chaîne de données	Chaîne cible pré-allouée correspondant à la taille de votre espace de présentation hôte. Cela peut varier en fonction de la configuration de votre espace de présentation hôte. Lorsque la fonction **Set Session Parameters** (9) avec l'option EAB est émise, la longueur de la chaîne de données doit être au moins deux fois supérieure à la longueur de l'espace de présentation.	
Longueur	NA (la longueur de l'espace de présentation hôte est implicite).	
Position PS	NA.	

Paramètres de retour

Cette fonction renvoie une chaîne de données, une longueur et un code retour.

Chaîne de données :

Contenu de l'espace de présentation hôte connecté.

Longueur :

Longueur des données copiées.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	Le contenu de l'espace de présentation hôte a été copié dans le programme d'application. L'espace de présentation cible était actif et le clavier était déverrouillé.
1	Votre programme n'est pas connecté à une session hôte.
4	Le contenu de l'espace de présentation hôte a été copié. L'espace de présentation hôte connecté attendait la réponse de l'hôte.
5	L'espace de présentation de hôte a été copié. Le clavier était verrouillé.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Un EAB peut être renvoyé lorsque l'option EAB de la fonction **Set Session Parameters** (9) est utilisée. EAB est lié à chaque caractère de l'espace de présentation et est renvoyé avant chaque caractère.
2. La fonction **Copy Presentation Space** est affectée par les options de session suivantes :
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB
    - XLATE/NOXLATE
    - BLANK/NOBLANK
    - DISPLAY/NODISPLAY
    - EXTEND_PS/NOEXTEND_PS

Voir les éléments 5 on page 145, 13 on page 148, 14 on page 149, 15 on page 149 et 17 on page 149 pour plus d'informations.

Si la chaîne de données cible fournie n'est pas suffisamment longue pour contenir les données demandées, des résultats imprévisibles peuvent se produire.

Comme indiqué précédemment, le retour des attributs par les différentes fonctions de **copie** (5, 8 et 34) est affecté par la fonction **Set Session Parameters** (9). Les paramètres de session définis impliqués ont l'effet suivant :

Définir le paramètre de session

Effet sur la fonction de copie

NOEAB et NOEAD

Les attributs ne sont pas renvoyés. Seul le texte est copié de l'espace de présentation vers le tampon utilisateur.

EAB et NOXLATE

Les attributs sont renvoyés tels que définis dans les tableaux suivants.

EAB et XLATE

Les couleurs utilisées pour l'affichage de l'espace de présentation sont renvoyées. Les couleurs peuvent être remappées ; les couleurs des attributs ne sont donc pas celles renvoyées par les fonctions **Copy** lorsque XLATE et EAB sont activés en même temps.

NOSO/SPACESO/SO

Lorsque NOSO est spécifié, il fonctionne comme SPACESO. La taille de l'espace de présentation n'est pas modifiée.

Les attributs de caractères renvoyés sont définis dans les tableaux suivants. Les positions des bits d'attribut sont au format IBM®, le bit 0 étant le bit le plus à gauche de l'octet.

Les attributs de caractères 3270 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0–1	Mise en évidence des caractères 00 = Normale 01 = Clignotement 10 = Vidéo inversée 11 = Soulignement
2–4	Couleur des caractères (le remappage des couleurs peut remplacer cette définition de couleur.) 000 = Par défaut 001 = Bleu 010 = Rouge 011 = Rose 100 = Vert 101 = Turquoise 110 = Jaune 111 = Blanc
5–6	Attribut de caractère 00 = Valeur par défaut
7	Réservé

Les attributs de caractères 5250 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0	Image inversée 0 = Image normale 1 = Image inversée
1	Soulignement 0 = Pas de soulignement 1 = Soulignement

Position des bits	Signification
2	Clignotement 0 = Aucun clignotement 1 = Clignotement
3	Séparateur de colonnes 0 = Aucun séparateur 1 = Séparateur
4–7	Réservé

Le tableau suivant montre les attributs de couleur des caractères Z and I Emulator for Windows. Le tableau suivant s'applique lorsque EAB et XLATE sont définis.

Position des bits	Signification
0–3	Couleurs des caractères d’arrière-plan 0000 = Noir 0001 = Bleu 0010 = Vert 0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc
4–7	Couleurs des caractères de premier plan 0000 = Noir 0001 = Bleu 0010 = Vert 0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc 1000 = Gris 1001 = Bleu clair 1010 = Vert clair 1011 = Cyan clair 1100 = Rouge clair 1101 = Magenta clair 1110 = Jaune 1111 = Blanc (haute intensité)

Pour un écran monochrome PS/2®, les caractères de la session d'application (poste de travail) apparaissent sous différentes nuances de gris. Ceci est nécessaire pour donner aux utilisateurs leurs couleurs remappées dans la session d'application EHLLAPI afin qu'ils puissent obtenir ce qu'ils voient dans leurs espaces de présentation d'application hôte.

Si vous souhaitez copier uniquement une partie de l'espace de présentation hôte, utilisez la fonction **Copy Presentation Space to String** (8).

Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la ligne 25 sur la ligne 24 ou sur la barre d'état. Pour que les informations soient affichées sur la barre d'état, la barre d'état doit être configurée. Consultez *Guide d'initiation* pour plus d'informations sur la configuration de la barre d'état. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Copy Presentation Space to String (8)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy Presentation Space to String** permet de copier tout ou partie de l'espace de présentation connecté à l'hôte dans une chaîne de données que vous définissez dans votre programme d'application EHLLAPI.

La position PS d'entrée est le décalage dans l'espace de présentation hôte. Ce décalage est basé sur une mise en page dans laquelle le coin supérieur gauche (ligne 1/colonne 1) est l'emplacement 1 et le coin inférieur droit est 3564, ce qui correspond à la taille d'écran maximale pour l'espace de présentation hôte. La valeur de `PS Position (Position PS) + (Longueur – 1)` ne peut pas dépasser la taille configurée de votre espace de présentation hôte.

La fonction **Copy Presentation Space to String** traduit les caractères de l'espace de présentation source hôte en ASCII. Les octets d'attribut et autres caractères non représentés en ASCII sont normalement traduits en espaces. Si vous ne souhaitez pas que les octets d'attribut soient traduits en blancs, vous pouvez remplacer cette traduction avec l'option ATTRB sous la fonction **Set Session Parameters** (9).

Appels prérequis

Connect Presentation Space (1).

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 8	
Chaîne de données	Chaîne cible pré-allouée correspondant à la taille de votre espace de présentation hôte. Lorsque la fonction **Set Session Parameters** (9) avec l'option EAB est émise, la longueur de la chaîne de données doit être au moins deux fois supérieure à la longueur de l'espace de présentation.	
Longueur	Longueur de la chaîne de données cible.	
Position PS	Positionnez dans l'espace de présentation hôte le premier octet de votre chaîne de données cible.	

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Contenu de l’espace de présentation hôte.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	Le contenu de l'espace de présentation hôte a été copié dans le programme d'application. L'espace de présentation cible était actif et le clavier était déverrouillé.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification de la longueur de la chaîne, ou la somme de (Longueur - 1) + position PS est supérieure à la taille de l'espace de présentation de l'hôte connecté.
4	Le contenu de l'espace de présentation hôte a été copié. L'espace de présentation hôte attendait la réponse de l'hôte.
5	L'espace de présentation de hôte a été copié. Le clavier était verrouillé.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Un EAB peut être renvoyé lorsque l’option EAB de la fonction **Set Session Parameters** (9) est utilisée. EAB est lié à chaque caractère de l'espace de présentation et est renvoyé après chaque caractère.
2. La fonction **Copy Presentation Space to String** est affectée par les options suivantes :
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB

- XLATE/NOXLATE
- BLANK/NOBLANK
- DISPLAY/NODISPLAY
- EXTEND_PS/NOEXTEND_PS

.

Si la chaîne de données cible fournie n'est pas suffisamment grande pour contenir le nombre d'octets demandé, la copie se termine avec succès (RC=0, 4 ou 5) lorsque la fin de la chaîne de données cible est atteinte.

Comme indiqué précédemment, le retour des attributs par les différentes fonctions de **copie** (5, 8 et 34) est affecté par la fonction **Set Session Parameters** (9). Les paramètres de session définis impliqués ont l'effet suivant :

Définir le paramètre de session

Effet sur la fonction de copie

NOEAB et NOEAD

Les attributs ne sont pas renvoyés. Seul le texte est copié de l'espace de présentation vers le tampon utilisateur.

EAB et NOXLATE

Les attributs sont renvoyés tels que définis dans les tableaux suivants.

EAB et XLATE

Les couleurs utilisées pour l'affichage de l'espace de présentation sont renvoyées. Les couleurs peuvent être remappées, de sorte que les couleurs d'attribut ne sont pas celles renvoyées par les fonctions de **copie** lorsque XLATE et EAB sont activés en même temps.

Les attributs de caractères renvoyés sont définis dans les tableaux suivants. Les positions des bits d'attribut sont au format IBM, le bit 0 étant le bit le plus à gauche de l'octet.

- Les attributs de caractères 3270 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	**Signification**
0–1	Mise en évidence des caractères 00 = Normale 01 = Clignotement 10 = Vidéo inversée 11 = Soulignement
2–4	Couleur des caractères (le remappage des couleurs peut remplacer cette définition de couleur.) 000 = Par défaut 001 = Bleu 010 = Rouge 011 = Rose

Position des bits	Signification
	100 = Vert 101 = Turquoise 110 = Jaune 111 = Blanc
5–7	Réservé

- Les attributs de caractères 5250 sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0	Image inversée 0 = Image normale 1 = Image inversée
1	Soulignement 0 = Pas de soulignement 1 = Soulignement
2	Clignotement 0 = Aucun clignotement 1 = Clignotement
3	Séparateur de colonnes 0 = Aucun séparateur 1 = Séparateur
4–7	Réservé

- Les attributs de caractères VT sont renvoyés de l'hôte à l'émulateur. Le tableau suivant s'applique lorsque EAB et NOXLATE sont définis.

Position des bits	Signification
0-3	Réservé
4	Gras 1 = Activé 0 = Désactivé
5	Tiret de soulignement 1 = Activé 1 = Désactivé
6	Clignotement 1 = Activé 0 = Désactivé
7	Inverser 0 = Activé 1 = Désactivé

- Le tableau suivant montre les attributs de couleur des caractères Z and I Emulator for Windows. Le tableau suivant s'applique lorsque EAB et XLATE sont définis.

Position des bits	Signification
0–3	Couleurs des caractères d'arrière-plan 0000 = Noir 0001 = Bleu 0010 = Vert 0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc
4–7	Couleurs des caractères de premier plan 0000 = Noir 0001 = Bleu 0010 = Vert 0011 = Cyan 0100 = Rouge 0101 = Magenta 0110 = Marron (3270), Jaune (5250) 0111 = Blanc 1000 = Gris 1001 = Bleu clair 1010 = Vert clair 1011 = Cyan clair 1100 = Rouge clair 1101 = Magenta clair 1110 = Jaune 1111 = Blanc (haute intensité)

Pour un écran monochrome PS/2, les caractères de la session d'application (poste de travail) apparaissent sous différentes nuances de gris. Ceci est nécessaire pour donner aux utilisateurs leurs couleurs remappées dans la session d'application EHLLAPI afin qu'ils puissent obtenir ce qu'ils voient dans leurs espaces de présentation d'application hôte.

3. Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un

message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la ligne 25 sur la ligne 24 ou sur la barre d'état. Pour que les informations soient affichées sur la barre d'état, la barre d'état doit être configurée. Consultez *Guide d'initiation* pour plus d'informations sur la configuration de la barre d'état. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Copy String to Field (33)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy String to Field** transfère une chaîne de caractères dans un champ spécifié dans l'espace de présentation connecté à l'hôte. Cette fonction ne peut être utilisée que dans un espace de présentation hôte *formaté par champ*.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 33	
Chaîne de données	Chaîne contenant les données à transférer vers un champ cible dans l'espace de présentation hôte.	
Longueur	Longueur, en nombre d'octets, de la chaîne de données source. Annulé si en mode EOT.	
Position PS	Identifie le champ cible. Il peut s'agir de la position PS de n'importe quel octet dans le champ cible. La copie commence toujours au début du champ.	

Paramètres de retour

Code retour	**Explication**
0	La fonction **Copy String to Field** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Erreur de paramètre ou longueur nulle pour la copie.
5	Le champ cible a été protégé ou inhibé, ou des données incorrectes ont été envoyées au champ cible (comme un attribut de champ).
6	La copie est terminée, mais les données sont tronquées.

Code retour	Explication
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	Espace de présentation hôte non formaté.

Notes sur l'utilisation de cette fonction

1. La fonction **Copy String to Field** est affectée par les options suivantes :
   - STRLEN/STREOT
   - EOT
   - EAB/NOEAB
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB

   Voir les éléments 1 on page 144, 2 on page 144, 13 on page 148, 14 on page 149 et 18 on page 150 pour plus d'informations.
2. La chaîne à transférer est spécifiée avec le paramètre de chaîne de données appelante. La chaîne se termine lorsque l'une de ces trois conditions est remplie :
   - Lorsqu'un délimiteur de fin de texte (EOT) est rencontré dans la chaîne si le mode EOT a été sélectionné à l'aide de la fonction **Set Session Parameters** (9). (Voir Set Session Parameters (9) on page 142).
   - Lorsque le nombre spécifié dans la longueur est atteint si vous n'êtes pas en mode EOT.
   - Lorsqu'une fin de champ est rencontrée dans le champ.

   Note: Si le champ à la fin de l'espace de présentation hôte est renvoyé à la ligne, celui-ci se produit lorsque la fin de l'espace de présentation est atteinte.
3. Les mnémoniques du clavier (voir la fonction **Send Key** (3)) ne peuvent pas être envoyés à l'aide de la fonction **Copy String to Field**.
4. Le premier octet des données à transférer est toujours placé au début du champ contenant la position PS spécifiée.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la ligne 25 sur la ligne 24 ou sur la barre d'état. Pour que les informations soient affichées sur la barre d'état, la barre d'état doit être configurée. Consultez *Guide d'initiation* pour plus d'informations sur la configuration de la barre d'état. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Copy String to Presentation Space (15)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy String to Presentation Space** copie une chaîne de données ASCII directement dans l'espace de présentation hôte à l'emplacement spécifié par le paramètre d'appel de position PS.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 15.	
Chaîne de données	Chaîne de données ASCII à copier dans l'espace de présentation hôte.	
Longueur	Longueur, en nombre d'octets, de la chaîne de données source. Annulé si en mode EOT.	
Position PS	Positionnez dans l'espace de présentation hôte pour commencer la copie, une valeur comprise entre 1 et la taille configurée de votre espace de présentation hôte.	

Paramètres de retour

Code retour	**Explication**
0	La fonction **Copy String to Presentation Space** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Erreur de paramètre ou longueur nulle pour la copie.
5	L'espace de présentation cible est protégé ou inhibé, ou des données incorrectes ont été envoyées à l'espace de présentation cible (comme un octet d'attribut de champ).
6	La copie était terminée, mais les données étaient tronquées.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. La fonction **Copy String to Presentation Space** est affectée par les options suivantes :
    - STRLEN/STREOT
    - EOT
    - EAB/NOEAB
    - XLATE/NOXLATE

- PUTEAB/NOPUTEAB
- EXTEND_PS/NOEXTEND_PS

Voir les éléments1 on page 144, 2 on page 144, 13 on page 148, 14 on page 149 et 18 on page 150 pour plus d'informations.

2. Les mnémoniques du clavier (voir la fonction **Send Key** (3)) ne peuvent pas être envoyés à l'aide de la fonction **Copy String to Presentation Space**.
3. La chaîne se termine lorsqu'un délimiteur de fin de texte (EOT) est rencontré dans la chaîne si le mode EOT a été sélectionné à l'aide de la fonction **Set Session Parameters** (9). (Voir Set Session Parameters (9) on page 142).
4. Bien que la fonction **Send Key** (3) accomplisse le même objectif, cette fonction répond par l'invite et saisit une commande plus rapidement. Etant donné que la fonction **Send Key** (3) émule l'opérateur du terminal saisissant les données à partir du clavier, sa vitesse de traitement est lente pour une application fonctionnant avec beaucoup de données. Cette fonction fournit un chemin d'entrée plus rapide vers l'hôte.
5. Les données originales (la chaîne copiée) ne peuvent pas dépasser la taille de l'espace de présentation.
6. Cet appel de fonction peut provoquer un déplacement du curseur vers une position inattendue avec certaines applications hôtes. Une fonction SendKey peut être un meilleur choix pour remplir un champ que cette fonction.

Note: Cela ne se produit qu'avec des sessions VT ou des connexions à un hôte ASCII.

Copy Presentation Space to Clipboard (35)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Copy Presentation Space to Clipboard** permet de copier tout ou partie de l'espace de présentation connecté à l'hôte dans le presse-papiers. La position PS d'entrée est le décalage dans l'espace de présentation hôte. Ce décalage est basé sur une mise en page dans laquelle le coin supérieur gauche (ligne 1/colonne 1) est l'emplacement 1 et le coin inférieur droit est 3564, ce qui correspond à la taille d'écran maximale pour l'espace de présentation hôte. La valeur de PS Position (Position PS) + (Longueur – 1) ne peut pas dépasser la taille configurée de votre espace de présentation hôte.

La fonction **Copy Presentation Space to Clipboard** traduit les caractères de l'espace de présentation source hôte en ASCII. Les octets d'attribut et autres caractères non représentés en ASCII sont normalement traduits en espaces. Si vous ne souhaitez pas que les octets d'attribut soient traduits en blancs, vous pouvez remplacer cette traduction avec l'option ATTRB sous la fonction **Set Sesssion Parameters** (9).

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 35.	
Chaîne de données	Chaîne cible pré-allouée correspondant à la taille de votre espace de présentation hôte. Lorsque la fonction **Set Session Parameters** (9) avec l'option EAB est émise, la longueur de la chaîne de données doit être au moins deux fois supérieure à la longueur de l'espace de présentation.	
Longueur	Longueur de la chaîne de données cible	
Position PS	Positionnez dans l'espace de présentation hôte le premier octet de votre chaîne de données cible.	

Paramètres de retour

Code retour	Explication
0	Le contenu de l'espace de présentation hôte a été copié dans le presse-papiers. L'espace de présentation cible était actif et le clavier était déverrouillé.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification de la longueur de la chaîne, ou la somme de (Longueur - 1) + position PS est supérieure à la taille de l'espace de présentation de l'hôte connecté.
4	Le contenu de l'espace de présentation hôte a été copié dans le presse-papiers. L'espace de présentation hôte attendait la réponse de l'hôte.
5	L'espace de présentation hôte a été copié dans le presse-papiers. Le clavier était verrouillé.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Un EAB peut être renvoyé lorsque l'option EAB de la fonction **Set Session Parameters** (9) est utilisée. EAB est lié à chaque caractère de l'espace de présentation et est renvoyé après chaque caractère.
2. La fonction **Copy Presentation Space to Clipboard** est affectée par les options suivantes :
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB
    - XLATE/NOXLATE
    - BLANK/NOBLANK
    - DISPLAY/NODISPLAY
    - PUTEAB/NOPUTEAB
    - EXTEND_PS/NOEXTEND_PS

3. Le tampon de chaîne de données est utilisé pour traiter les données extraites de l'espace de présentation et copiées dans le presse-papiers. Si le tampon de chaîne de données fourni n'est pas suffisamment grand pour contenir le nombre d'octets demandé, la copie se termine avec succès (RC=0, 4 ou 5) lorsque la fin du tampon de chaîne de données est atteinte. Comme indiqué précédemment, le retour des attributs par les différentes fonctions de **copie** (5, 8 et 34) est affecté par la fonction **Set Session Parameters (9)**. Les paramètres de session définis impliqués ont l'effet suivant :

   Définir le paramètre de session

   Effet sur la fonction de copie

   NOEAB et NOEAD

   Les attributs ne sont pas renvoyés. Seul le texte est copié de l'espace de présentation vers le presse-papiers.

   EAB et NOXLATE

   Les attributs sont renvoyés tels que définis dans les tableaux suivants.

   EAB et XLATE

   Les couleurs utilisées pour l'affichage de l'espace de présentation sont renvoyées. Les couleurs peuvent être remappées, de sorte que les couleurs d'attribut ne sont pas celles renvoyées par les fonctions de **copie** lorsque XLATE et EAB sont activés en même temps.

Paste Clipboard to Presentation Space (36)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Paste Clipboard to Presentation Space** colle une chaîne de données ASCII directement dans l'espace de présentation hôte à l'emplacement spécifié par le paramètre d'appel de position PS.

Appels prérequis

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 36.	
Chaîne de données	Tampon de chaîne qui contient les données du presse-papiers à coller dans l'espace de présentation hôte.	
Longueur	Longueur, en nombre d'octets à coller	
Position PS	Positionnez dans l'espace de présentation hôte pour commencer la copie, une valeur comprise entre 1 et la taille configurée de votre espace de présentation hôte.	

Paramètres de retour

Code retour	Explication
0	La fonction **Paste Clipboard to Presentation Space** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Erreur de paramètre ou longueur nulle pour la copie
5	L'espace de présentation cible est protégé ou inhibé, ou des données incorrectes ont été envoyées à l'espace de présentation cible (comme un octet d'attribut de champ).
6	La copie était terminée, mais les données étaient tronquées.
7	La copie était terminée, mais les données étaient tronquées.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. La fonction **Paste Clipboard to Presentation Space** est affectée par les options suivantes :
   - STRLEN/STREOT
   - EAB/NOEAB
   - EOT
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB
   - EXTEND_PS/NOEXTEND_PS

2. La chaîne se termine lorsqu'un délimiteur de fin de texte (EOT) est rencontré dans la chaîne si le mode EOT a été sélectionné à l'aide de la fonction **Set Session Parameters** (9). (Voir « Set Session Parameters (9) » à la page 147).
3. Les données originales (la chaîne copiée) ne peuvent pas dépasser la taille de l'espace de présentation.

Chaîne	Significations	Champ de caractère à un octet	
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 error	
Note: SB signifie des caractères à un octet.			

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche toujours les mêmes informations sur la 24ème ligne. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication Manager EHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Disconnect from Structured Fields (121)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Disconnect from Structured Fields** supprime la connexion entre le programme d'émulation et l'application EHLLAPI. L'application EHLLAPI doit se déconnecter du programme d'émulation avant de quitter le système. L'application EHLLAPI doit émettre cette demande de fonction si une précédente fonction **Disconnect from Structured Fields** a été émise.

La fonction **Reset System** (21) déconnectera également toutes les connexions SF en attente.

Appels prérequis

Connect for Structured Fields (120)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 121	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 3	Doit être 8
Position PS	NA	

Contenu de la chaîne de données

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
2–3	5–6	ID unique de destination/origine renvoyé par Connect for structured field (120).
	7–8	Réservé.

Paramètres de retour

Code retour	**Explication**
0	La fonction **Disconnect from Structured Fields** a réussi.
1	Un ID de session court de l'espace de présentation hôte spécifié n'était pas valide ou n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
40	Déconnecté avec des demandes asynchrones en attente.

Notes sur l'utilisation de cette fonction

1. Lorsqu'une fonction **Disconnect from Structured Fields** est appelée, toutes les demandes de fonction asynchrones **Read Structured Fields** (126) ou **Write Structured Fields** (127) en attente sont renvoyées si l'application émet l'appel de fonction **Get Request Completion** (125). Utilisez la forme asynchrone de cette fonction lors du nettoyage après avoir émis un appel Disconnect.
2. La fonction **Reset System** (21) libérera également toutes les demandes asynchrones en attente (demandes qui n'ont pas été récupérées par l'application à l'aide de la fonction **Get Request Completion** (125)).

Disconnect Presentation Space (2)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Disconnect Presentation Space** supprime la connexion entre votre programme d'application EHLLAPI et l'espace de présentation hôte. De plus, si un espace de présentation hôte est réservé à l'aide de la fonction **Reserve** (11), il est libéré lors de l'exécution de la fonction **Disconnect Presentation Space**.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 2	
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Code retour	**Explication**
0	La fonction **Disconnect Presentation Space** a réussi.
1	Votre programme n'était actuellement pas connecté à l'espace de présentation hôte.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Après l'appel de la fonction **Disconnect Presentation Space**, les fonctions qui interagissent avec l'espace de présentation connecté à l'hôte ne sont plus valides (par exemple, les fonctions **Send Key** (3), **Wait** (4), **Reserve** (11) et **Release** (12)).
2. Votre application EHLLAPI doit se déconnecter de l'espace de présentation hôte avant de quitter.
3. La fonction **Disconnect Presentation Space** ne réinitialise pas les paramètres de session aux valeurs par défaut. Votre application EHLLAPI doit appeler la fonction **Reset System** (21) pour y parvenir.

Disconnect Window Service (102)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Disconnect Window Service** déconnecte la connexion des services Windows entre le programme EHLLAPI et la fenêtre de l'espace de présentation hôte spécifiée.

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 102	
Chaîne de données	Voir le tableau suivant	
Longueur	1	4
Position PS	NA	

Contenu de la chaîne de données

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

Code retour	**Explication**
0	La fonction **Disconnect Window Service** a réussi.

Code retour	Explication
1	Votre programme n'est pas connecté aux services de la fenêtre.
9	Une erreur système s'est produite.

Notes sur l'utilisation de cette fonction

Après l'appel de la fonction **Disconnect Window Service**, votre application ne gère plus la fenêtre de l'espace de présentation.

Avant de quitter l'application, vous devez demander une fonction **Disconnect Window Service** pour tous les espaces de présentation qui ont été connectés pour les services Presentation Manager®. Si l'application se termine avec une connexion en attente pour les services de la fenêtre, le sous-système annule la connexion en attente.

Interception EditKey

Cette fonctionnalité vous permet d'intercepter les clés d'édition en plus de toutes les frappes existantes et de les envoyer à une session dans un environnement Windows 32 bits.

Conditions préalables

1. Mappez les fonctions d'édition dans la fenêtre Customize Keyboard (par exemple Ctrl+C pour la fonction d'édition de copie).
2. Appelez la fonction EHLLAPI Start Keystroke Intercept (50) avec la valeur de chaîne de données du paramètre d'appel définie. Les valeurs sont les suivantes :

Position de l'octet	Contenus
1	Correspond à l'une des valeurs suivantes : • Un nom court d'espace de présentation hôte spécifique (PSID) • Un espace vide ou Null indiquant une demande d'espace de présentation hôte connecté à l'hôte
2 à 4	Réservé
5	Un caractère de code d'option : • D pour les frappes AID uniquement • L pour toutes les frappes • E pour toutes les frappes et touches d'édition • M pour demander le mode message asynchrone de la notification (Windows uniquement). Si M est spécifié, un caractère de code D ou L, ou E doit être placé en position 13
6 à 8	Réservé
9 à 12	Si M est spécifié en position 5, le descripteur de la fenêtre qui reçoit le message. Le message est une valeur de retour non nulle de RegisterWindowMessage (PCSHLL).
13	Si M est spécifié en position 5, une des valeurs suivantes :

Position de l'octet	Contenus
	• D pour les frappes AID uniquement • L pour toutes les frappes • E pour toutes les frappes et touches d'édition
14 à 16	Réservé

3. Pour obtenir les clés d'édition interceptées, utilisez la fonction EHLLAPI Get Key (51). Le mnémonique de clé renvoyé dans la chaîne de données pour les touches d'édition aura M (mnémonique de type frappe) à la position du 5ème octet. Les 4 octets suivants auront l'un des mnémoniques de clé d'édition suivants basés sur la clé d'édition interceptée :

Mnémonique de clé	Clé interceptée
@W@C	Modifier Copier
@W@D	Modifier Effacer
@W@E	Modifier Copier Ajouter
@W@L	Modifier Copier Liaison
@W@N	Modifier Coller Suivant
@W@V	Modifier Coller
@W@X	Modifier Couper
@W@Z	Modifier Annuler

4. Pour envoyer des clés d'édition à la session, utilisez la fonction EHLLAPI Send Key (3). La chaîne de données transmise en tant que paramètre d'appel peut spécifier les mnémoniques de clé d'édition suivants :

Mnémonique de clé	Clé envoyée
@W@C	Modifier Copier
@W@D	Modifier Effacer
@W@E	Modifier Copier Ajouter
@W@L	Modifier Copier Liaison
@W@N	Modifier Coller Suivant
@W@V	Modifier Coller
@W@X	Modifier Couper
@W@Z	Modifier Annuler

Note:

1. Il n'est pas nécessaire d'appeler la fonction EHLLAPI Get Key (51) pour utiliser la fonction Send Key (3). Pour que les fonctions Get Key (51) et Send Key (3) gèrent les clés d'édition, vous devez d'abord appeler Start Keystroke Intercept (50) avec la position du 5ème octet définie sur **E**. Si le 5ème octet contient **M**, alors la position 13 doit contenir **E**.
2. Les valeurs de retour attendues pour les fonctions Start Keystroke Intercept (50), Get Key (51) et Send Key (3) n'ont pas changé.
3. Toutes les conditions préalables de la documentation existante doivent être respectées ainsi que les conditions préalables documentées ici.

Find Field Length (32)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Find Field Length** renvoie la longueur d'un champ cible dans l'espace de présentation connecté. Cette fonction peut être utilisée pour rechercher des champs protégés ou non, mais uniquement dans un espace de présentation hôte *formaté par champ*.

Cette fonction renvoie le nombre de caractères contenus dans le champ identifié à l'aide du paramètre position d'appel PS. Cela inclut tous les caractères depuis le début du champ cible jusqu'au caractère précédant l'octet d'attribut suivant.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 32	
Chaîne de données	Voir le tableau suivant	
Longueur	NA	NA
Position PS	Voir la note	

Note: Position PS : identifie le champ dans l'espace de présentation hôte à partir duquel démarrer la **recherche**. Il peut s'agir de la position PS de n'importe quel octet dans le champ dans lequel vous souhaitez que la **recherche** commence.

La chaîne de données appelante à 2 caractères peut contenir :

Code	Explication
␣␣ ou T␣	Ce champ
P␣	Le champ précédent, protégé ou non.
N␣	Le champ suivant, protégé ou non
NP	Le champ suivant protégé
NU	Le champ suivant non protégé
PP	Le champ protégé précédent
PU	Le champ non protégé précédent

Note: Le symbole ␣ représente un espace requis.

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Les longueurs suivantes sont valides :

Longueur	Explication
= 0	Lorsque le code de retour = 28, la longueur du champ est 0. Lorsque le code de retour = 24, l'espace de présentation hôte n'est pas formaté en champ.
> 0	Longueur de champ obligatoire dans l'espace de présentation hôte.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Find Field Length** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur de paramètre a été rencontrée.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	Aucun champ de ce type n'a été trouvé.
28	Longueur du champ de 0 octet.

Notes sur l'utilisation de cette fonction

Sauf quand ␣␣ or T␣ est utilisée comme chaîne de données appelante, si le champ trouvé est le même que le champ à partir duquel la **recherche** a démarré, un code de retour de 24 est renvoyé.

Find Field Position (31)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Find Field Position** renvoie la position de début d'un champ cible dans l'espace de présentation connecté à l'hôte. Cette fonction peut être utilisée pour rechercher des champs protégés ou non, mais uniquement dans un espace de présentation hôte *formaté par champ*.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 31	
Chaîne de données	Voir le tableau suivant	
Longueur	NA	NA
Position PS	Voir la note	

Note: Position PS : identifie le champ dans l'espace de présentation hôte à partir duquel démarrer la **recherche**. Il peut s'agir de la position PS de n'importe quel octet dans le champ dans lequel vous souhaitez que la **recherche** commence.

La chaîne de données appelante à 2 caractères peut contenir :

Code	**Explication**
␣␣ ou T␣	Ce champ
P␣	Le champ précédent, protégé ou non
N␣	Le champ suivant, protégé ou non
NP	Le champ suivant protégé
NU	Le champ suivant non protégé
PP	Le champ protégé précédent
PU	Le champ non protégé précédent

Note: Le symbole ␣ représente un espace requis.

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Les longueurs suivantes sont valides :

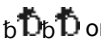
Longueur	Explication
= 0	Lorsque le code de retour = 28, la longueur du champ est 0. Lorsque le code de retour = 24, l'espace de présentation hôte n'est pas formaté en champ.
> 0	Position relative du champ demandé par rapport à l'origine de l'espace de présentation hôte. Cette position est définie comme étant la première position après l'octet d'attribut.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Find Field Position** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur de paramètre a été rencontrée.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	Aucun champ de ce type n'a été trouvé.
28	Longueur du champ de 0 octet.

Notes sur l'utilisation de cette fonction

Sauf quand  est utilisée comme chaîne de données appelante, si le champ trouvé est le même que le champ à partir duquel la **recherche** a démarré, un code de retour de 24 est renvoyé.

Free Communications Buffer (124)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Free Communication Buffer** (124) renvoie à la gestion de la mémoire d'un tampon qui n'est plus nécessaire à l'application. L'application doit libérer le tampon avant de quitter le système.

Appels prérequis

Allocate Communications Buffer (123)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 124	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 6	Doit être 8
Position PS	NA	

Contenu de la chaîne de données

Octet		Définition
Standard	Etendu	
1–2	1–4	Doit être 0
3–6	5–8	L'adresse du tampon

Paramètres de retour

Code retour	Explication
0	La fonction **Free Communications Buffer** a réussi.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
41	Le tampon est en cours d'utilisation.

Notes sur l'utilisation de cette fonction

1. Si l'application tente de libérer un tampon en cours d'utilisation, la demande de libération sera refusée et un code de retour de 41 sera renvoyé.
2. Une application doit demander la fonction **Free Communications Buffer** (124) avant de quitter tous les tampons de communication qui ont été alloués à l'aide de la fonction **Allocate Communications Buffer** (123).
3. La fonction **Reset System** (21) libérera les tampons alloués par la fonction **Allocate Communications Buffer** (123).

Get Key (51)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Get Key** permet à votre programme d'application EHLLAPI de récupérer une frappe d'une session spécifiée par la fonction **Start Keystroke Intercept** (50) et traite, accepte ou rejette cette frappe. En plaçant cette fonction dans une boucle, vous pouvez l'utiliser pour intercepter une chaîne.

Appels prérequis

Start Keystroke Intercept (50)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 51	
Chaîne de données	Voir le tableau suivant	
Longueur	8	12
Position PS	NA	

Contenu de la chaîne de données

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Un nom court d'espace de présentation (PSID) à 1 caractère • Un espace vide ou Null indiquant un appel de fonction pour la présentation connectée à l'hôte
	2–4	Réservé
2–8	5–11	Espaces contenant de l'espace pour la représentation symbolique des données demandées
	12	Réservé

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Voir le tableau suivant :

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes :

Octet		Définition
		• Un nom court d'espace de présentation (PSID) à 1 caractère • Un espace vide ou Null indiquant un appel de fonction pour la présentation connectée à l'hôte
	2–4	Réservé
2	5	Un caractère de code d'option, l'un des caractères suivants : • `A` pour ASCII renvoyé • `M` pour mnémonique de frappe • `S` pour mnémonique spécial
3–8	6–11	Ces 6 octets de l'espace tampon pré-alloué sont utilisés en interne pour mettre et retirer les frappes au clavier. Les combinaisons possibles incluent : • L'octet 3 contient un caractère ASCII et l'octet 4 contient X'00' • L'octet 3 contient le caractère d'échappement (soit `@`, soit un autre caractère spécifié à l'aide de l'option `ESC=c` de la fonction 9) et l'octet 4 contient une abréviation d'un octet pour une fonction. (Voir Mnémoniques ASCII on page 20) • Les octets 5 à 8 peuvent être similaires aux octets 3 et 4 si le mnémonique ASCII renvoyé est plus long que 2 octets (par exemple, si le mnémonique ASCII représente Attn `@A@Q`, l'octet 5 contient `@` et l'octet 6 contient `Q`). S'ils ne sont pas utilisés, les octets 5 à 8 sont mis à zéro (X'00').

Pour plus de clarté, quelques exemples de chaînes de données renvoyées sont fournis ci-dessous :

Note: Le symbole `@` est le caractère d'échappement par défaut. La valeur du caractère d'échappement peut être définie sur n'importe quelle frappe représentée en ASCII à l'aide de l'option `ESC=c` de la fonction **Set Session Parameters** (9). Si le caractère d'échappement a été remplacé par un autre caractère à l'aide de cette option, le symbole `@` dans les exemples suivants est remplacé par l'autre caractère.

Interface 16 bits

`EAt`

`E` est le nom court de l'espace de présentation. Les frappes sont renvoyées au format ASCII (A) et la clé renvoyée est la lettre minuscule t. (Octets 4 à 8 = X'00').

`EM@2`

`E` est le nom court de l'espace de présentation. Les frappes sont renvoyées sous forme de mnémoniques et la clé renvoyée est PF2 (octets 5 à 8 = X'00').

Interface 32 bits

E b b b b At

E est le nom court de l'espace de présentation. Les frappes sont renvoyées au format ASCII (A) et la clé renvoyée est la lettre minuscule t. (Octets 7 à 11 = X'00').

E b b b b M@2

E est le nom court de l'espace de présentation. Les frappes sont renvoyées sous forme de mnémoniques et la clé renvoyée est PF2 (octets 8 à 11 = X'00').

Code retour :

Les codes suivants sont valides :

Code retour	Explication
0	La fonction **Get Key** a réussi.
1	Un espace de présentation incorrect a été spécifié.
5	Vous avez spécifié l'option AID uniquement sous la fonction **Start Keystroke Intercept** (50), et les touches non-AID sont inhibées par ce type de session lorsque EHLLAPI essaie d'écrire des clés incorrectes dans l'espace de présentation.
8	Aucune fonction **Start Keystroke Intercept** (50) antérieure n'a été appelée pour cet espace de présentation.
9	Une erreur système a été rencontrée.
20	Une combinaison de clés non définie a été saisie.
25	Les frappes demandées ne sont pas disponibles dans la file d'attente d'entrée.
31	La file d'attente des frappes a débordé et les frappes ont été perdues.

Notes sur l'utilisation de cette fonction

1. Si un code retour 31 apparaît pour la fonction **Get Key**, soit :
   - Augmentez la valeur du paramètre de longueur d'appel pour la fonction **Start Keystroke Intercept** (50), ou
   - Exécutez la fonction **Get Key** plus fréquemment.

   Une frappe interceptée occupe 3 octets dans le tampon. La prochaine frappe interceptée est placée dans les trois octets adjacents. Lorsque la fonction **Get Key** récupère une frappe (première entrée, première sortie, FIFO), les trois octets qu'elle occupait sont rendus disponibles pour une autre frappe. En augmentant la taille du tampon ou la vitesse à laquelle les frappes sont récupérées du tampon, vous pouvez éliminer le dépassement de la mémoire tampon.

   Pour le PC/3270, une autre façon d'éliminer le code retour 31 est d'utiliser l'émulateur PC/3270en mode reprise.
2. Vous pouvez utiliser la fonction **Send Key** (3) pour transmettre à la fois les frappes originales et toutes les autres dont votre application EHLLAPI peut avoir besoin pour l'espace de présentation connecté à l'hôte.

3. Les frappes arrivent de manière asynchrone et sont mises en file d'attente dans la file d'attente de frappe que vous avez fournie dans votre programme d'application EHLLAPI à l'aide de la fonction **Start Keystroke Intercept** (50).
4. La fonction **Get Key** se comporte comme une lecture. Lorsque des frappes au clavier sont disponibles, elles sont lues dans la zone de données que vous avez fournie dans votre application.
5. Dans le cas d'une prise en charge de champ d'une session, l'application peut s'intéresser uniquement aux touches AID, par exemple la touche Entrée. Si tel est le cas, le code d'option de fonction **Start Keystroke Intercept** (50) doit être défini sur `D` (c'est-à-dire pour les touches AID uniquement).
6. Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Get Request Completion (125)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Get Request Completion** permet à une application de déterminer le statut d'une demande de fonction asynchrone précédente émise vers le EHLLAPI et pour obtenir la liste des paramètres de fonction avant d'utiliser à nouveau la chaîne de données. Cette fonction n'est valide que si l'utilisateur a spécifié l'achèvement asynchrone (A) lors d'un appel de fonction précédent tel que **Read Structured Fields** (126) ou **Write Structured Fields** (127).

Chaque requête asynchrone nécessitant la fonction **Get Request Completion** renverra un ID unique de la requête asynchrone. L'application doit enregistrer cet identifiant. Cet ID est l'identification utilisée par la fonction **Get Request Completion** pour identifier la demande souhaitée. L'utilisateur dispose de trois options de demande utilisant cette fonction :

1. L'application peut interroger ou attendre une demande de fonction asynchrone spécifique en fournissant l'ID de demande de cette fonction et un nom abrégé de session non vide.
2. L'application peut interroger ou attendre la première demande de fonction asynchrone terminée pour une session spécifiée en fournissant un ID de demande X'0000' et un nom abrégé de session non vide.

Appels prérequis

Connect Structured Fields (120) et **Allocate Communications Buffer** (123)

et

Read Structured Fields (126) ou **Write Structured Fields** (127)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 125	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 14	Doit être 24
Position PS	NA	

Contenu de la chaîne de données

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2	5	N ou WN=NOWAIT est requis W=WAIT est requis
	6–8	Réservé
3–4	9–10	ID de demande de fonction.
5–6	11–12	Réservé
7–10	13–16	Réservé
11–12	17–20	Réservé
13–14	21–24	Réservé

La fonction **Get Request Completion** se comporte différemment selon le deuxième caractère de la chaîne de paramètres, qui est l'un des caractères suivants :

N

Option Nowait : si un ID de demande spécifique a été fourni et que la fonction est terminée, le contrôle sera renvoyé à l'application avec un code retour de zéro et une chaîne de données complétée comme défini dans Paramètres de retour on page 91. Si un ID de demande de zéro a été fourni et qu'une fonction asynchrone éligible est terminée, le contrôle sera renvoyé à l'application avec un code retour de zéro et une chaîne de données complétée comme défini dans Paramètres de retour on page 91.

W

Option d'attente : si un ID de demande spécifique a été fourni et que la fonction n'est pas terminée, l'appel attendra que la fonction soit terminée avant de revenir à l'application. Si l'ID de demande fourni était zéro et qu'aucune fonction asynchrone éligible n'est terminée, l'appel attendra la fin d'une fonction avant de revenir à l'application appelante. Au retour, la valeur du code retour sera zéro et la chaîne de données sera complétée comme défini dans Paramètres de retour on page 91.

Paramètres de retour

Octet		Définition
Standard	Etendu	
5–6	11–12	Numéro de fonction de la fonction asynchrone terminée (126 ou 127). (renvoyé)
7–10	13–16	Adresse de la chaîne de données de l'appel de fonction asynchrone terminé. (L'application ne doit pas réutiliser la chaîne de données tant que la demande n'est pas terminée). (renvoyé)
11–12	17–20	Longueur de la chaîne de données de l’appel de fonction asynchrone terminé. (renvoyé)
13–14	21–24	Code retour de l’appel de fonction asynchrone terminé. (renvoyé)

Code retour	Explication
0	La fonction **Get Request Completion** a réussi.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système a été rencontrée.
38	La fonction demandée n'était pas terminée.
42	Aucune demande correspondante n'a été trouvée.

Il existe quelques différences entre les codes de retour 38 et 42 :

1. Code retour 38
    a. Si un ID de demande et une session spécifiques ont été demandés, la session et l'ID ont été trouvés, mais la demande est en attente (pas dans un état terminé).
    b. Si un ID de demande nul et une session spécifique ont été demandées, la session spécifiée a des demandes en attente, mais elles ne sont pas satisfaites (complètes).
    c. Si un ID de demande nul et une session vide étaient demandés, des demandes en attente étaient trouvées mais aucune n'était satisfaite (complète).
2. Code retour 42
    a. Si un ID de demande et une session spécifiques ont été demandés, l'ID de demande spécifique n'a pas été trouvé dans un état en attente ou terminé.
    b. Si un ID de demande nul et une session spécifique ont été demandées, la session spécifique ne contient aucune demande en attente ou terminée.
    c. Si un ID de demande nul et une session vide ont été demandés, aucune demande en attente ou terminée n'a été trouvée.

Notes sur l'utilisation de cette fonction

1. Cette fonction n'est valide que si l'utilisateur a spécifié l'achèvement asynchrone (A pour Asynchrone) lors d'un appel de fonction précédent tel que **Read Structured Fields** ou **Write Structured Fields**.
2. Si le code de retour est 0, l'application doit vérifier la chaîne de données renvoyée pour obtenir des informations relatives à l'achèvement de la fonction asynchrone demandée.

Lock Presentation space API (60)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Lock Presentation Space API** permet à l'application d'obtenir ou de libérer le contrôle exclusif de la fenêtre de l'espace de présentation sur d'autres applications Windows 32 bits. Lorsqu'elle est verrouillée, aucune autre application ne peut se connecter à la fenêtre de l'espace de présentation.

Le traitement réussi de cette fonction avec la fonction Lock entraîne les fonctions de la fenêtre de l'espace de présentation EHLLAPI demandées depuis autres applications EHLLAPI d'être mises en file d'attente jusqu'à ce que l'application demandeuse déverrouille l'espace de présentation. Les demandes de l'application de verrouillage sont traitées normalement.

Appels prérequis

Connect to Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 60	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 3	Doit être 8
Position PS	NA	

Contenu de la chaîne de données

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
2	5	Un des caractères suivants :

Octet		Définition
		• L pour verrouiller l'API. • U pour déverrouiller l'API.
3	6	Un des caractères suivants : • R pour revenir si l'espace de présentation est déjà verrouillé par une application. • Q pour mettre en file d'attente la demande de verrouillage si l'espace de présentation est déjà verrouillé par une application.
	7–8	Réservé.

Paramètres de retour

Code retour	Explication
0	La fonction **Lock Presentation Space API** a réussi.
1	Un ID de session court d'espace de présentation hôte incorrect a été spécifié ou n'a pas été connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système a été rencontrée.
43	L'API était déjà verrouillée par une autre application EHLLAPI (sur LOCK) ou API non verrouillée (sur UNLOCK).

Notes sur l'utilisation de cette fonction

Les fonctions EHLLAPI suivantes sont mises en file d'attente lorsqu'un verrouillage est en vigueur :

- **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)
- **Connect to Presentation Space** (1) avec le paramètre CONPHYS défini lors d'un précédent appel de fonction **Set Sessions Parameter** (9).

Ces demandes en file d'attente ne sont pas traitées tant que le verrouillage n'est pas supprimé. Lorsque le verrouillage est supprimé, les demandes en file d'attente sont traitées dans l'ordre première entrée, première sortie (FIFO). Les fonctions EHLLAPI non répertoriées sont exécutées comme s'il n'y avait pas de verrouillage. L'application requérante déverrouille la fenêtre de l'espace de présentation par l'une des méthodes suivantes :

- Déconnexion de l'espace de présentation tout en étant propriétaire du verrouillage.
- Emission de la fonction **Reset System** (21) tout en étant propriétaire du verrouillage.
- Arrêt de l'application tout en étant propriétaire du verrouillage.
- Arrêt de la session.
- Emission réussie de l'**API Lock Presentation Space** avec l'option Unlock.

Avant de quitter l'application, vous devez déverrouiller toutes les fenêtres de l'espace de présentation qui ont été verrouillées avec la fonction API **Lock Presentation Space**. Si l'application se ferme avec des verrouillages en attente, ou si une fonction **Reset System** (21) ou **Disconnect Presentation Space** (2) est émise, les verrouillages sont libérés.

Il est recommandé que les applications verrouillent l'espace de présentation uniquement pendant de courtes périodes et uniquement lorsque l'utilisation exclusive de l'espace de présentation est requise.

Lock Window Services API (61)

3270	***5250***	***VT***
Oui	Non	Non

La fonction **Lock Window Services API** permet à l'application d'obtenir ou de libérer le contrôle exclusif de la fenêtre de l'espace de présentation sur d'autres applications Windows 32 bits. Lorsqu'elle est verrouillée, aucune autre application ne peut se connecter à la fenêtre de l'espace de présentation.

Le traitement réussi de cette fonction avec la fonction Lock entraîne les fonctions de la fenêtre de l'espace de présentation EHLLAPI demandées depuis autres applications EHLLAPI d'être mises en file d'attente jusqu'à ce que l'application demandeuse déverrouille l'espace de présentation. Les demandes de l'application de verrouillage sont traitées normalement.

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 61	
Chaîne de données	Voir le tableau suivant.	
Longueur	Doit être 3	Doit être 8
Position PS	NA	

Contenu de la chaîne de données

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
2	5	Un des caractères suivants : • L pour verrouiller l'API. • U pour déverrouiller l'API.
3	6	Un des caractères suivants : • R pour revenir si l'espace de présentation est déjà verrouillé par une application. • Q pour mettre en file d'attente la demande de verrouillage si l'espace de présentation est déjà verrouillé par une application.
5–6	11–12	Numéro de fonction de la fonction asynchrone terminée (126 ou 127). (renvoyé)
	7–8	Réservé.

Paramètres de retour

Code retour	**Explication**
0	La fonction **Lock Window Services API** a réussi.
1	Un ID de session court d'espace de présentation hôte incorrect a été spécifié ou n'a pas été connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système a été rencontrée.
38	La fonction demandée n'était pas terminée.
43	L'API était déjà verrouillée par une autre application EHLLAPI (sur LOCK) ou API non verrouillée (sur UNLOCK).

Notes sur l'utilisation de cette fonction

Les fonctions EHLLAPI suivantes sont mises en file d'attente lorsqu'un verrouillage est en vigueur :

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

Ces demandes en file d'attente ne sont pas traitées tant que le verrouillage n'est pas supprimé. Lorsque le verrouillage est supprimé, les demandes en file d'attente sont traitées dans l'ordre première entrée, première sortie (FIFO).

L'application requérante déverrouille la fenêtre de l'espace de présentation par l'une des méthodes suivantes :

- Emission réussie de l'**API Lock Window Services** avec l'option UNLOCK (Déverrouiller).
- Déconnexion de l'espace de présentation tout en étant propriétaire du verrouillage.
- Emission de la fonction **Reset System** (21) tout en étant propriétaire du verrouillage.
- Arrêt de l'application tout en étant propriétaire du verrouillage.
- Arrêt de la session.

Avant de quitter l'application, vous devez déverrouiller toutes les fenêtres de l'espace de présentation qui ont été verrouillées avec la fonction API **Lock Window Services**. Si l'application se ferme avec des verrouillages en attente, le sous-système libère les verrouillages.

Il est recommandé que les applications verrouillent l'espace de présentation uniquement pendant de courtes périodes et uniquement lorsque l'utilisation exclusive de l'espace de présentation est requise.

Pause (18)

3270	***5250***	***VT***
Oui	Oui	Oui

La fonction **Pause** attend pendant une durée spécifiée. Il doit être utilisé à la place des *boucles de synchronisation* pour attendre qu'un événement se produise. Une fonction **Pause** peut être terminée par un événement hôte si une fonction **Start Host Notification** (23) a été appelée et que l'option IPAUSE est sélectionnée.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 18	
Chaîne de données	NA	

	Interface standard	Interface améliorée
Longueur	Contient la durée de la pause par incréments d'une demi-seconde	
Position PS	NA	

Paramètres de retour

Code retour	Définition
0	Le délai d'attente a expiré.
9	Une erreur système interne a été rencontrée. Les résultats temporels sont imprévisibles.
26	L'espace de présentation de la session hôte ou OIA a été mis à jour. Utilisez la fonction **Query Host Update** (24) pour obtenir plus d'informations.

Notes sur l'utilisation de cette fonction

1. La sélection de l'option FPAUSE ou IPAUSE à l'aide de la fonction **Set Session Parameters** (9) affecte la durée de la pause que vous obtenez lorsque vous appelez cette fonction. Voir l'élément 6 on page 145 pour plus d'informations.
2. La valeur saisie dans le paramètre de longueur d'appel est le nombre maximum d'intervalles d'une demi-seconde que la fonction **Pause** attend. Pour une pause de 20 secondes, une valeur hexadécimale de `0028` (décimal 40) doit être transmise dans le paramètre de longueur d'appel.
3. Si vous utilisez l'option IPAUSE et que la valeur de pause est nulle, la fonction attend jusqu'à 2 400 intervalles d'une demi-seconde, à moins d'être interrompue plus tôt. Si vous utilisez l'option FPAUSE et que la valeur de pause est nulle, la fonction retourne immédiatement.
4. Si vous utilisez l'option IPAUSE, une fois qu'une pause a été satisfaite par un événement hôte, vous devez appeler la fonction **Query Host Update** (24) pour effacer la file d'attente avant la prochaine fonction **Pause**. La fonction **Pause** continuera à se contenter de l'événement en attente jusqu'à ce que la fonction **Query Host Update** (24) soit terminée.
5. Une valeur maximale pratique pour la fonction **Pause** est `2400`. Vous ne devez pas utiliser la fonction **Pause** pour ce type de tâches :
    - Retard sur des durées très longues (de plusieurs heures par exemple).
    - Retard de plus d'une durée modérée (20 minutes) avant de vérifier l'horloge du système et de poursuivre l'exécution de votre programme EHLLAPI.
    - Avec des applications nécessitant une minuterie haute résolution, car l'intervalle de temps créé par une fonction **Pause** est approximatif.
    - Définissez l'intervalle de temps à zéro dans une boucle.
6. La fonction IPAUSE définie et la pause interrompue permettent à une application EHLLAPI de déterminer si l'espace de présentation de l'hôte (PS) ou la zone d'informations de l'opérateur (OIA) spécifié est mis à jour. Les trois fonctions suivantes sont utilisées :

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

En utilisant IPAUSE lorsque la fonction **Start** est appelée, vous pouvez faire attendre une application jusqu'à ce que l'espace de présentation hôte ou l'OIA (ou les deux) reçoive une mise à jour. Lorsque la réception est terminée et que l'application peut émettre la fonction **Query** pour déterminer les modifications, **Pause** se termine. Ensuite, l'application émet la fonction **Search Presentation Space** (6) pour vérifier si la mise à jour attendue a eu lieu.

Post Intercept Status (52)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Post Intercept Status** informe l'émulateur Z and I Emulator for Windows qu'une frappe obtenue via la fonction **Get Key** (51) a été acceptée ou rejetée. Lorsque l'application rejette une frappe, la fonction **Post Intercept Status** émet un bip.

Appels prérequis

Start Keystroke Intercept (50)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 52	
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 2	Doit être 8
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Le nom court d’une lettre de l’espace de présentation. • Un espace vide ou Null indiquant un appel de fonction pour l'espace de présentation connectée à l'hôte.
	2–4	Réservé
2	5	Un des caractères suivants :

Octet		Définition
		• A pour la frappe acceptée. • R pour la frappe rejetée.
	6–8	Réservé.

Paramètres de retour

Code retour	Explication
0	La fonction **Post Intercept Status** a réussi.
1	Un espace de présentation incorrect a été spécifié.
2	Une option de session incorrecte a été spécifiée.
8	Aucune fonction **Start Keystroke Intercept** (50) antérieure n'a été appelée pour cet ID d'espace de présentation.
9	Une erreur système a été rencontrée.

Query Additional Field Attribute (45)

3270	*5250*	*VT*
Non	Oui	Non

La fonction **Query Additional Field Attribute** renvoie des informations supplémentaires sur le champ 5250 contenant la position de l'espace de présentation de l'hôte d'entrée. Ces informations sont renvoyées dans le paramètre de chaîne de données sous la forme d'une structure définie.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 45.	
Chaîne de données	Chaîne de caractères de 8 octets.	
Longueur	8 est implicite.	
Position PS	Identifie la cible. Il peut s'agir de la position PS de n'importe quel octet dans le champ cible.	

La chaîne de données appelante peut contenir :

Octet	Définition
1–8	Réservé

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

La fonction renvoie la chaîne de données suivante.

Octet	Définition
entre 1 et 6	Réservé
7–8	Deux caractères non signés de 8 bits qui renvoient : • R si le champ est RTL et L si le champ est LTR. • U si le champ est en majuscule et L si le champ est un champ en casse normale.

Code retour :

Les codes de retour suivants sont définis :

Code retour	Explication
0	La fonction **Query Additional Field Attribute** a réussi.
1	Votre programme n'est actuellement pas connecté à une session hôte.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Aucun champ n'a été trouvé à cette position.
24	Le champ n'est pas formaté.

Query Close Intercept (42)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Close Intercept** permet à l'application de déterminer si l'option de fermeture a été sélectionnée.

Appels prérequis

Start Close Intercept (41)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 42	
Chaîne de données	Voir le tableau suivant.	
Longueur	Doit être 1	Doit être 4

	Interface standard	Interface améliorée
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	ID de session court à 1 caractère de l'espace de présentation hôte, ou un espace vide ou Null indiquant une demande d'interrogation de la session connectée à l'hôte
	2–4	Réservé

Paramètres de retour

Code retour	Explication
0	Aucun événement d'interception de fermeture ne s'est produit.
1	La source de présentation n'était pas valide.
2	Une erreur a été commise lors de la spécification des paramètres.
8	Aucune fonction **Start Close Intercept** (41) antérieure n'a été appelée pour cet espace de présentation hôte.
9	Une erreur système s'est produite.
12	La session s'est arrêtée.
26	Une interception de fermeture s’est produite depuis le dernier appel d’interception de fermeture de la requête.

Query Communications Buffer Size (122)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Query Communications Buffer Size** (122) permet à une application de déterminer à la fois les tailles de tampon maximale et optimale prises en charge par le programme d'émulation.

Appels prérequis

Il n’y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 122	

	Interface standard	Interface améliorée
Chaîne de données	Voir le tableau suivant	
Longueur	Doit être 9	Doit être 20
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2–3	5–8	Champ de 16 ou 32 bits pour la taille optimale du tampon entrant pris en charge (valeur renvoyée)
4–5	9–12	Champ de 16 ou 32 bits pour la taille maximale du tampon entrant pris en charge (valeur renvoyée)
6–7	13–16	Champ de 16 ou 32 bits pour la taille optimale du tampon sortant pris en charge (valeur renvoyée)
8–9	17–20	Champ de 16 ou 32 bits pour la taille maximale du tampon sortant pris en charge (valeur renvoyée)

Paramètres de retour

Code retour	Explication
0	La fonction **Query Communications Buffer Size** a réussi.
1	Un ID de session court de l'espace de présentation hôte spécifié n'était pas valide ou n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
10	La fonction n'était pas prise en charge par le programme d'émulation.

Notes sur l'utilisation de cette fonction

1. Il n'y a aucun moyen d'obliger l'utilisateur à utiliser cette fonction. Il ne s'agit pas d'une fonction obligatoire afin que l'application puisse être adaptée pour fonctionner sur n'importe quel système.
2. Les tailles de tampon renvoyées représentent les tailles d'enregistrement réellement transmises sur le support. Pour une connexion DDM, l'en-tête de 8 octets fourni dans le tampon de données des fonctions **Read** et **Write Structured Fields** est supprimé et 1 octet contenant la valeur AID du champ structuré est préfixé. L'application doit comparer la taille des données réelles dans la mémoire tampon de données (qui n'inclut pas l'en-tête de 8 octets) avec les tailles de mémoire tampon renvoyées par la **taille du tampon de communication de requête** moins 1 octet. Pour les connexions destination/origine, l'en-tête de 8 octets fourni dans le tampon de données des fonctions **Read** et **Write Structured Fields** est supprimé et 9 octets sont ensuite préfixés aux

données. L'application doit comparer la taille des données réelles dans la mémoire tampon de données (qui n'inclut pas l'en-tête de 8 octets) avec les tailles de mémoire tampon renvoyées par la **taille du tampon de communication de requête** moins 9 octets.

3. Les tailles de tampon maximales renvoyées représentent le nombre maximal d'octets pris en charge par le matériel du poste de travail et par l'émulateur. La taille maximale de la mémoire tampon ne peut être utilisée que si l'hôte est également configuré pour accepter au moins ces tailles maximales.
4. Les tailles de tampon optimales renvoyées représentent le nombre optimal d'octets pris en charge par le matériel du poste de travail et l'émulateur. Certaines configurations réseau peuvent définir des limites de transmission inférieures à ces valeurs. Dans ces cas, la valeur de remplacement de la taille du tampon de transfert de données dans le profil de configuration de l'émulateur sera utilisée pour la prise en charge de champ structuré. La **taille du tampon de communication de requête** reflétera toutes les valeurs de remplacement de taille de tampon saisies dans le profil de configuration de l'émulateur.

Query Communication Event (81)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Communication Event** permet au programme EHLLAPI de déterminer si des événements de communication se sont produits.

Appels prérequis

Start Communication Notification (80)

Paramètres d'appel

	Interface améliorée
Numéro de fonction	Doit être 81
Chaîne de données	Nom court à 1 caractère de l'espace de présentation hôte ou un espace vide ou Null indiquant une demande de mise à jour de l'espace de présentation connecté à l'hôte
Longueur	4 est implicite
Position PS	NA

La structure de données appelante contient ces éléments :

Octet	**Définition**
1	Un nom court d'espace de présentation (PSID) à 1 caractère
2-4	Réservé

Paramètres de retour

Code retour	Définition
0	La fonction a réussi
1	Un PSID incorrect a été spécifié
8	Aucun appel préalable à la fonction **Start Communication Notification** (80) n'a été appelé pour le PSID
9	Une erreur système a été rencontrée
21	Le PSID indiqué a été connecté
22	Le PSID indiqué a été déconnecté

Query Cursor Location (7)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Cursor Location** indique la position du curseur dans l'espace de présentation connecté à l'hôte en renvoyant la position du curseur.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 7	
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Position du curseur dans l'espace de présentation hôte.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query Cursor Location** a réussi.
1	Votre programme n'est actuellement pas connecté à une session hôte.
9	Une erreur système a été rencontrée.

Query Field Attribute (14)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Field Attribute** renvoie l'octet d'attribut du champ contenant la position de l'espace de présentation hôte d'entrée. Cette information est renvoyée dans le paramètre de longueur renvoyé.

Pour le PC/3270, notez également que :

- Le paramètre de longueur renvoyé est défini sur 0 si l'écran n'est pas formaté.
- Les octets d'attribut sont égaux ou supérieurs à l'hexagone C0.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 14.	
Chaîne de données	NA.	
Longueur	NA.	
Position PS	Identifie la cible. Il peut s'agir de la position PS de n'importe quel octet dans le champ cible.	

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

La valeur de l'attribut si l'écran est formaté, ou 0 si l'écran n'est pas formaté.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query Field Attribute** a réussi.

Code retour	Explication
1	Votre programme n'est actuellement pas connecté à une session hôte.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	Octet d'attribut introuvable ou espace de présentation hôte non formaté.

Notes sur l'utilisation de cette fonction

Les attributs de champs renvoyés sont définis dans les tableaux suivants. Les positions des bits sont au format IBM, le bit 0 étant le bit le plus à gauche de l'octet.

- Attribut de champ 3270 :

Position des bits	Signification
0–1	Les deux = 1, octet d'attribut de champ
2	Non protégé/protégé 0 = Champ de données non protégé 1 = Champ protégé
3	A/N 0 = Données alphanumériques 1 = Données numériques uniquement
4–5	I/SPD 00 = Intensité normale, stylet non détectable 01 = Intensité normale, stylet détectable 10 = Haute intensité, stylet détectable 11 = Non affiché, stylet non détectable
6	Réservé
7	MDT 0 = Le champ n'a pas été modifié 1 = Le champ a été modifié

- Attributs de champ 5250 :

Position des bits	Signification
0	Indicateur d'attribut de champ 0 = Indicateur d'attribut hors champ 1 = Indicateur d'attribut de champ
1	Visibilité

Position des bits	Signification
	0 = Non-affichage 1 = Affichage
2	Non protégé/protégé 0 = Champ de données non protégé 1 = Champ protégé
3	Intensité 0 = Intensité normale 1 = Haute intensité
4–6	Type de zone 000 = Données alphanumériques : tous les caractères sont disponibles 001 = Alphabet uniquement : majuscules et minuscules, virgule, point, trait d'union, espace ou clé Dup disponibles 010 = Décalage numérique : décalage automatique du numéro 011 = Données numériques uniquement : 0 à 9, virgule, point, plus, moins, espace ou clé Dup disponibles 101 = Données numériques uniquement : 0 à 9 ou clé Dup disponibles 110 = Données de l'appareil de lecture de bande magnétique uniquement 111 = Données numériques signées : 0 à 9, plus, moins ou clé Dup disponibles
7	MDT 0 = Le champ n'a pas été modifié 1 = Le champ a été modifié

Query Host Update (24)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Host Update** permet à l'opérateur programmé de déterminer si l'hôte a mis à jour l'espace de présentation hôte ou l'OIA pour les raisons suivantes :

- La fonction **Start Host Notification** (23) a été appelée (au premier appel à la fonction **Query Host Update** uniquement)
- Appel précédent à la fonction **Query Host Update** (pour tous les appels à la fonction **Query Host Update** sauf le premier).

Appels prérequis

Start Host Notification (23)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 24	
Chaîne de données	Nom court à 1 caractère de l'espace de présentation hôte, ou un espace vide ou Null indiquant une demande de mise à jour de l'espace de présentation connecté à l'hôte	
Longueur	1 est implicite	4 est implicite
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

Code retour	Définition
0	Aucune mise à jour n'a été effectuée depuis le dernier appel.
1	Un espace de présentation hôte incorrect a été spécifié.
8	Aucune fonction **Start Host Notification** (23) antérieure n'a été appelée pour l'ID d'espace de présentation hôte.
9	Une erreur système a été rencontrée.
21	L'OIA a été mis à jour.
22	L'espace de présentation a été mis à jour.
23	L'OIA et l'espace de présentation hôte ont été mis à jour.
44	L'impression est terminée dans la session d'impression.

Notes sur l'utilisation de cette fonction

L'espace de présentation cible doit être spécifié dans la chaîne de données, même si une connexion à l'espace de présentation hôte n'est pas nécessaire pour vérifier les mises à jour.

Query Session Status (22)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Session Status** est utilisée pour obtenir des informations spécifiques à la session.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	16 bits	**32 bits**
Numéro de fonction	Doit être 22.	
Chaîne de données	Chaîne de 18/20 octets composée d'un nom court de 1 octet de l'espace de présentation cible plus 17 octets pour les données renvoyées. La position 1 peut être pourvue par : 1. Un espace vide ou Null pour indiquer une demande pour l'espace de présentation host_connected. 2. Un * (astérisque) pour indiquer une demande d'espace de présentation du propriétaire du clavier.	
Longueur	Doit être 18	Doit être 20
Position PS	NA	

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2–9	5–12	Nom long de la session (identique au nom du profil ; ou, si le profil n'est pas défini, identique au nom court)
10	13	**Session Type** **D** Affichage 3270 **E** 3270 Imprimante

Octet		Définition
		F Affichage 5250 **G** 5250 Imprimante **H** ASCII VT
11	14	Caractéristiques de session exprimées par un nombre binaire comprenant les bits de caractéristiques de session suivants **Bit 0** EAB 0 : la session possède l'attribut de base. 1 : la session a l'attribut étendu **Bit 1** PSS 0 : la session ne prend pas en charge les symboles programmés 1 : la session prend en charge les symboles programmés **Bits 2 à 7** Réservé
12–13	15–16	Nombre de lignes dans l'espace de présentation hôte, exprimé sous forme de nombre binaire
14–15	17–18	Nombre de colonnes dans l'espace de présentation hôte, exprimé sous forme de nombre binaire
16–17	19–20	Page de codes hôte exprimée sous forme de nombre binaire
18		Réservé

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query Session Status** a réussi.
1	Un espace de présentation hôte incorrect a été spécifié.
2	Une longueur de chaîne incorrecte a été créée.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Les instructions requises pour préallouer cette mémoire varient en fonction du langage dans lequel votre application est écrite. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Query Sessions (10)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Sessions** renvoie une chaîne de données de 16 octets (12 octets pour l'interface standard) décrivant chaque session hôte.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

Fonctionnalité	**La description**	
	Interface standard	Interface améliorée
Numéro de fonction	Doit être 10	
Chaîne de données	Chaîne pré-attribuée de 16*n* octets de long (12*n* pour 16 bits) (*n* =nombre de sessions) ou plus	
Longueur	12*n* octets	16*n* octets
Position PS	NA	

Note: Lorsque la durée ne correspond pas au nombre de sessions, le code de retour est 2.

Paramètres de retour

Cette fonction renvoie une chaîne de données, une longueur et un code retour.

Chaîne de données :

La chaîne de données renvoyée a une longueur de 16*n* octets (12*n* pour l'interface standard), où *n* est le nombre de sessions hôtes. Les descripteurs sont concaténés dans la chaîne de données et dans chaque type de session et dans la taille de l'espace de présentation d'une session hôte.

Le format de chaque descripteur de session de 16 octets (12 octets pour l'interface standard) est le suivant :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2–9	5–12	Nom long de la session (identique au nom du profil ; ou, si le profil n'est pas défini, identique au nom court)
10	13	Type de connexion H=hôte
	14	Réservé
11–12	15–16	Taille de l'espace de présentation de l'hôte (il s'agit d'un nombre binaire et qui n'est pas au format d'affichage). Si le type de session est une session d'impression, la valeur est 0.

Longueur :

Nombre de sessions hôtes démarrées.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query Sessions** a réussi.
2	Une longueur de chaîne incorrecte a été créée.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Si un programme d'application reçoit RC=2 ou RC=0, le nombre de sessions actives est renvoyé dans le champ longueur. Le programme d'application peut reconnaître la longueur minimale de la chaîne grâce à ce numéro.
2. La fonction **Query Sessions** est affectée par l'option de session CFGSIZE/NOCFGZISE (voir l'élément 16 on page 149 pour plus d'informations) et par l'option EXTEND_PS/NOEXTEND_PS (voir l'élément 20 on page 150 pour plus d'informations).

Note:

1. Lorsque NOCFGSIZE est défini dans **Set Session Parameters** (9) pour une session 5250, la valeur de la taille de l'espace de présentation renvoyée en position d'octet 11 et 12 depuis **Query Sessions** (10) sera modifiée conformément à la sélection de EXTEND_PS ou NOEXTEND_PS.
2. Lorsque EXTEND_PS est défini dans **Set Session Parameters** (9), la taille de l'espace de présentation renvoyée par **Query Sessions** (10) inclura la taille de la ligne de message, si elle existe.
3. Lorsque NOEXTEND_PS est défini, la valeur ne changera pas quelle que soit l'existence d'une ligne de message. Dans le cas d'un espace de présentation de 25 lignes et 80 colonnes, la valeur peut être 1920 ou 2000.

Query System (20)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query System** peut être utilisée par un programme d'application EHLLAPI pour déterminer le niveau de prise en charge de Z and I Emulator for Windows et autres valeurs liées au système. Cette fonction renvoie une chaîne contenant les données système appropriées. La plupart de ces informations sont destinées à être utilisées par un coordinateur de service lorsque vous appelez le centre de support IBM après avoir reçu un code de retour 9 (une erreur système a été rencontrée).

Les octets de cette chaîne renvoyée sont définis dans Paramètres de retour on page 113.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 20	
Chaîne de données	Chaîne pré-allouée de 35 octets	36 octets
Longueur	Doit être 35	Doit être 36
Position PS	NA	

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Une chaîne de données de 35 octets (pour 16 bits) ou 36 octets (pour 32 bits) est renvoyée. Les octets sont définis comme suit :

Octet		Définition
Standard	Etendu	
1	1	EHLLAPI numéro de version
2–3	2–3	EHLLAPI numéro de niveau
4–9	4–9	Réservé
10–12	10–12	Réservé
13	13	Base matérielle, U=Impossible de déterminer
14	14	Type de programme, où P=HCLZ and I Emulator for Windows
15–16	15–16	Réservé
17–18	17–18	Z and I Emulator for Windows version/niveau sous forme de valeur ASCII sur 2 octets
19	19	Réservé
20–23	20–23	Réservé
24–27	24–27	Réservé
28–29	28–29	Réservé
	30	Réservé
30–31	31–32	Type NLS exprimé sous forme de nombre binaire sur 2 octets
33–35	34–36	Réservé

Code retour

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query System** a réussi ; la chaîne de données a été renvoyée.
1	EHLLAPI n'est pas chargé. (PC/3270 seulement)
2	Une longueur de chaîne incorrecte a été spécifiée. (PC/3270 seulement)
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

Pour utiliser cette fonction, préallouez de la mémoire pour recevoir le paramètre de chaîne de données renvoyé. Consultez Allocation de mémoire on page 11 pour plus d'informations.

Query Window Coordinates (103)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Query Window Coordinates** demande les coordonnées de la fenêtre d'un espace de présentation. Les coordonnées de la fenêtre sont renvoyées en pixels.

Note: (0,0) indique le coin supérieur gauche de la fenêtre.

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 103	
Chaîne de données	ID de session court à 1 caractère de l'espace de présentation hôte	
Longueur	17 est implicite	20 est implicite
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Un nom court d'espace de présentation (PSID) à 1 caractère • Un espace vide ou Null indiquant un appel de fonction pour l'espace de présentation de connexion actuel
	2–4	Réservé
2-17	5–20	Réservé

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Un identifiant de session court d'espace de présentation à 1 caractère • Un espace vide ou Null indiquant un appel de fonction pour l'espace de présentation de connexion actuel
	2–4	Réservé
2–17	5–20	Quatre entiers non signés de 32 bits qui renvoient :
2–5	5–8	XLeft Entier long en pixels de la coordonnée X gauche de la fenêtre rectangulaire par rapport à la fenêtre du bureau

Octet		Définition
6–9	9–12	YBottom Entier long en pixels de la coordonnée Y inférieure de la fenêtre rectangulaire par rapport à la fenêtre du bureau
10–13	13–15	XRight Entier long en pixels de la coordonnée X droite de la fenêtre rectangulaire par rapport à la fenêtre du bureau
14–17	16–20	YTop Entier long en pixels de la coordonnée Y supérieure de la fenêtre rectangulaire par rapport à la fenêtre du bureau

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Query Window Coordinates** a réussi.
1	Votre programme n'était actuellement pas connecté à la session hôte.
9	Une erreur système s'est produite.
12	La session s'est arrêtée.

Read Structured Fields (126)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Read Structured Field** permet à une application de lire les données de champs structurés à partir de l'application hôte. Si l'appel spécifie S (pour Synchrone), l'application ne reçoit pas le contrôle tant que la fonction **Read Structured Field** n'est pas terminée. Si l'appel spécifie A (pour Asynchrone), l'application reçoit le contrôle immédiatement après l'appel. Si l'appel spécifie M (pour Asynchrone, mode message), l'application reçoit le contrôle immédiatement après l'appel. L'application peut attendre le message. Dans tous les cas (S, A ou M), l'application fournit l'adresse du tampon dans laquelle les données de l'hôte doivent être placées.

Pour une exécution asynchrone réussie de cette fonction, les instructions suivantes s'appliquent :

Le champ du code retour dans la liste des paramètres peut ne pas contenir les résultats des E/S demandées. Si le code retour n'est pas 0, la demande a échoué. L'application doit prendre l'action appropriée en fonction du code retour.

Si le code retour de cette demande est 0, l'application doit utiliser l'ID de demande renvoyé avec cet appel de fonction pour émettre l'appel de fonction **Get Request Completion** afin de déterminer les résultats d'achèvement de la fonction associée à l'ID de demande. L'appel de fonction **Get Request Completion** renvoie les informations suivantes :

1. ID de demande de fonction
2. Adresse de la chaîne de données de la requête asynchrone
3. Longueur de la chaîne de données
4. Code retour de la fonction terminée

Appels prérequis

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

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 126	
Chaîne de données	Voir le tableau suivant	
Longueur	8, 10 ou 14	20
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
2	5	S ou A ou M **S =** Synchrone. Le contrôle n'est pas renvoyé à l'application tant que la lecture n'est pas satisfaite. **A =** Asynchrone. Le contrôle est renvoyé immédiatement à l'application, peut attendre l'objet événement. **M =** Asynchrone. Le contrôle est renvoyé immédiatement à l'application, peut attendre le message.
	6	Réservé.
3–4	7–8	ID de destination/origine sur 2 octets.
5–8	9–12	Adresse sur 4 octets du tampon dans lequel les données doivent être lues. Le tampon doit être obtenu à l'aide de la fonction **Allocate Communications Buffer** (123).
9–10	13–16	Réservé.
11–12	17–20	Lorsque M est spécifié en position 2, le descripteur de fenêtre de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (« PCSHLL ») (différente de 0).
13–14		Les données dans ces positions sont ignorées par EHLLAPI. Cependant, aucune erreur n'est générée si le programme de migration contient des

Octet		Définition
		données à ces positions. Ces données sont acceptées pour assurer la compatibilité avec les applications en migration.

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Si A (asynchrone) est spécifié en position 5, (2 pour l'interface standard) et que la fonction s'exécute avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
Standard	Etendu	
9–10	13–14	ID de demande de fonction sur 2 octets. Il est utilisé par la fonction **Get Request Completion** (125) pour déterminer l'achèvement de cet appel de fonction.
	15–16	Réservé.
	17–20	Valeur de 4 octets dans laquelle l'adresse de l'objet événement est renvoyée par EHLLAPI. L'application peut attendre cet objet événement. Lorsque l'objet événement est effacé, l'application doit émettre l'appel de fonction **Get Request Completion** (125) (32 bits uniquement).

Note: Une adresse d'objet d'événement est renvoyée pour chaque demande asynchrone réussie. L'objet événement ne doit plus être utilisé. Un nouvel objet événement est renvoyé pour chaque demande et n'est valide que pour la durée de cette demande.

Chaîne de données :

Si « M » (mode de message asynchrone) est spécifié en position 5 (2 pour les applications 16 bits) et que la fonction s'exécute avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
9–10	13–14	Un ID de demande de fonction sur 2 octets. Il est utilisé par la fonction **Get Request Completion (125)** pour déterminer l'achèvement de cet appel de fonction.
	15–16	Réservé.
11–12	17–18	ID de tâche du mode de message asynchrone.
	19–20	Réservé.

Note: Si la fonction est terminée avec succès, une fenêtre d'application reçoit un message. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL). Le paramètre wParam contient l'ID de tâche

renvoyé par l'appel de fonction. Le HIWORD du paramètre lParam contient le code retour 0, qui indique que la fonction a réussi, et LOWORD du paramètre lParam contient le numéro de fonction 126.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Read Structured Fields** a réussi.
1	Un ID de session court de l'espace de présentation hôte spécifié n'était pas valide ou n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
11	Ressource non disponible (mémoire non disponible).
35	Demande rejetée. Une transmission sortante de l'hôte a été annulée.
36	Demande rejetée. Perte de contact avec l'hôte.
37	La fonction a réussi, mais l'hôte est désactivé pour les appels entrants.

Notes sur l'utilisation de cette fonction

1. Le code retour 35 sera renvoyé lorsque la fonction **Read Structured Fields** ou **Write Structured Fields** est demandée pour la première fois après l'annulation d'une transmission sortante de l'hôte. Les mesures correctives relèvent de la responsabilité de l'application.
2. Le code retour 36 nécessite que l'application se déconnecte du programme d'émulation, puis se reconnecte pour rétablir la communication avec l'hôte. Les mesures correctives relèvent de la responsabilité de l'application.
3. Le code retour 37 sera renvoyé si l'hôte est désactivé pour les appels entrants. La fonction **Read Structured Fields** a été demandée avec succès.
4. Le EHLLAPI permet qu'un maximum de 20 requêtes asynchrones par application soient en attente. Un code retour pour les ressources indisponibles (RC=11) est renvoyé si plus de 20 requêtes asynchrones sont tentées.

Les données de champ structuré contiennent les champs structurés d'application reçus de l'hôte. Les en-têtes de champs structurés sont supprimés par le EHLLAPI avant que les données de champ structurées n'atteignent l'application.

Le format des données de champ structuré est le suivant :

Décalage	Longueur	Contenus
0	1 mot	X'0000'.
2	1 mot	m (longueur du message : le nombre d'octets de données dans le message, le nombre n'inclut pas le préfixe d'en-tête du tampon, qui contient 8 octets). Cette valeur est renvoyée par EHLLAPI.

Décalage	Longueur	Contenus
4	1 mot	n (taille du tampon : la longueur fournie du tampon de données qui inclut l'en-tête du message de 8 octets). Cette valeur doit être définie par l'application.
6	1 mot	X'C000'.
8	8 octets	Longueur du premier (ou du seul) message de champ structuré.
10	1 octet	Premier octet hors longueur du message de champ structuré.
		⋮
m+7	1 octet	Dernier octet du message de champ structuré.

Les octets 0 à 7 sont l'en-tête du tampon. Ces 8 premiers octets sont utilisés par le programme d'émulation. La section utilisateur du tampon commence par le décalage 8. Les octets 8 et 9 contiennent le nombre d'octets dans le premier champ structuré (un message à champ structuré peut contenir plusieurs champs structurés), dont 2 octets pour les octets 8 et 9. Octets 8 à *m*+7 sont utilisés pour le message de champ structuré reçu de l'hôte (qui peut contenir plusieurs champs structurés).

L'application utilisatrice doit fournir le tampon complet avec le mot au décalage 0 mis à zéro. La longueur du tampon doit être dans le mot au décalage 4. Le mot au décalage 6 doit être X'C000'. Le programme d'émulation placera le message de données commençant au décalage 8 et placera la longueur du message dans le mot au décalage 2. La longueur du tampon n'est pas perturbée par EHLLAPI.

Demandes synchrones

Lorsque **Read Structured Fields** est demandée de manière synchrone (option S dans la chaîne de données), le contrôle est renvoyé à l'application uniquement une fois la demande satisfaite. L'application peut supposer que :

- Le code retour est correct.
- Les données dans le tampon de communication (tampon de lecture) sont correctes.
- L'hôte ne traite plus la demande **Read Structured Fields**.

Demandes asynchrones

Lorsque **Read Structured Fields** est demandée de manière asynchrone (option A dans la chaîne de données), l'application *ne peut pas* supposer que :

- Le code retour est correct.
- Les données dans le tampon de communication (tampon de lecture) sont correctes.
- L'hôte ne traite plus la demande **Read Structured Fields**.

Lorsqu'il est demandé de manière asynchrone, EHLLAPI renvoie les valeurs suivantes :

- Un ID de demande de 16 bits aux positions 13 à 14 (9 à 10 pour l'interface standard) de la chaîne de données
- L'adresse d'un objet événement aux positions 17 à 20 de la chaîne de données

Ceux-ci sont utilisés pour terminer l’appel asynchrone **Read Structured Fields**.

Les étapes suivantes doivent être effectuées pour déterminer le résultat d'un appel de fonction asynchrone **Read Structured Fields** :

- Si le code retour EHLLAPI n'est pas zéro, la demande a échoué. Aucune demande asynchrone n'a été effectuée. L'application doit prendre les mesures appropriées avant de tenter à nouveau l'appel.
- Si le code retour est zéro, l'application doit attendre que l'objet événement soit dans l'état signalé à l'aide de la fonction **Get Request Completion** (125) ou **Wait For Single Object**. L'objet événement ne doit pas être réutilisé. L'objet événement est valide uniquement pendant la durée de l'appel de fonction **Read Structured Fields** jusqu'à la fin de l'appel de fonction **Get Request Completion** (125).
- Une fois que l'objet événement est dans l'état signalé, utilisez l'ID de demande de 16 bits renvoyé comme paramètre ID de demande dans un appel à la fonction **Get Request Completion** (125). La chaîne de données renvoyée par l'appel de fonction **Get Request Completion** (125) contient le code retour final de l'appel de fonction **Read Structured Fields**.

Lorsque **Read Structured Fields** est demandée de manière asynchrone (l'option M dans la chaîne de données), l'application *ne peut pas* supposer que :

- Le code retour est correct.
- Les données dans le tampon de communication (tampon de lecture) sont correctes.
- L'hôte ne traite plus la demande **Read Structured Fields**.

Lorsqu'il est demandé de manière asynchrone avec l'option M, EHLLAPI renvoie les valeurs suivantes :

- Un ID de demande de 16 bits aux positions 13 à 14 (9 à 10 pour l'interface standard) de la chaîne de données
- ID de tâche du mode de message asynchrone aux positions 17 à 18 (11 à 12 pour l'interface standard) de la chaîne de données.

Ceux-ci sont utilisés pour terminer l'appel asynchrone **Read Structured Fields**.

Receive File (91)

3270	***5250***	***VT***
Oui	Oui	Non

La fonction **Receive File** permet de transférer un fichier de la session hôte vers la session du poste de travail. Elle est utilisée de la même manière que la commande RECEIVE dans le PC/3270. La fonction **Receive File** peut être appelée par un programme d'application EHLLAPI.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 91.	
Chaîne de données	Référez-vous aux exemples.	
Longueur	Longueur, en nombre d'octets, de la chaîne de données. Annulé si en mode EOT.	

Voici des exemples de chaînes de données pour un jeu de caractères sur un octet (SBSC) :

Session 3270

- Pour recevoir le fichier du système hôte VM/CMS :

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

- Pour recevoir le fichier du système hôte MVS™/TSO :

  ```
  pc_filename[ID :]dataset[(membre)] [/mot de passe] [option]
  ```

- Pour recevoir le fichier du système hôte CICS® :

  ```
  pc_filename [ID :]hôte_filename [(option]
  ```

Session 5250

- Pour recevoir le fichier du système hôte iSeries™, eServer™ i5 ou System i5™ :

  ```
  pc_filename [ID :]membre du fichier de bibliothèque [option]
  ```

Paramètres de retour

Code retour	Explication
2	Erreur de paramètre ou vous avez spécifié une longueur trop longue (plus de 255 octets) pour le tampon EHLLAPI. Le transfert de fichiers a échoué.
3	Transfert de fichiers terminé.
4	Transfert de fichiers terminé avec enregistrements segmentés.
9	Une erreur système a été rencontrée.
27	Le transfert de fichiers s'est terminé en raison de l'utilisation d'un bouton Cancel ou du délai d'attente défini par la fonction **Set Session Parameter**(9).
101	Le transfert de fichiers a réussi (transfert vers/depuis CICS®).

Si vous recevez le code retour 2 ou 9, il y a un problème avec le système ou avec la façon dont vous avez spécifié votre chaîne de données.

D'autres codes retour peuvent également être reçus, liés aux numéros de message générés par le programme de transfert hôte. Pour les transferts vers un programme de transfert d'hôte CICS®, soustrayez 100 du code retour pour obtenir la partie numérique du message. Par exemple, un code retour de 101 signifierait que le numéro de message INW0001 a été émis par l'hôte. Pour les autres programmes de transfert d'hôte, utilisez simplement le code retour

comme partie numérique du message. Par exemple, un retour de 34 signifierait que le message TRANS34 a été émis par le programme de transfert hôte. La documentation de votre programme de transfert d'hôte devrait donner plus d'informations sur la signification des messages spécifiques.

Les codes d'erreur du système d'exploitation signalés par EHLLAPI sont supérieurs à 300. Pour déterminer le code d'erreur, soustrayez 300 et reportez-vous à la documentation du système d'exploitation pour connaître les codes de retour.

Notes sur l'utilisation de cette fonction

1. Quatre ensembles de paramètres sous la fonction **Set Session Parameters** (9) sont liés à cette fonction. Il s'agit des options de session STRLEN/STREOT, EOT=c, QUIET/NOQUIET et TIMEOUT=c/TIMEOUT=0. Voir les éléments1 on page 144 et 2 on page 144 et les éléments 7 on page 145 et 8 on page 145 pour plus d'informations.
2. Si aucun chemin n'est spécifié lors de l'exécution de la fonction **Receive File**, le fichier reçu est stocké dans le sous-répertoire actuel, qui est le répertoire dans lequel votre application s'exécute.

Release (12)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Release** déverrouille le clavier associé à l'espace de présentation hôte réservé à l'aide de la fonction **Reserve** (11).

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 12	
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Code retour	**Explication**
0	La fonction **Release** a réussi.
1	Votre programme n'est pas connecté à une session hôte.

Code retour	Explication
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

Si vous ne **libérez** pas un espace de présentation hôte réservé à l'aide de la fonction **Reserve** (11), vous serez exclu de cette session jusqu'à ce que vous appeliez la fonction **Reset System** (21), que vous appeliez la fonction **Disconnect Presentation Space** (2) ou que vous mettiez fin au programme d'application EHLLAPI.

Reserve (11)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Reserve** verrouille le clavier associé à l'espace de présentation connecté à l'hôte pour bloquer les entrées de l'opérateur du terminal.

L'espace de présentation hôte réservé reste verrouillé jusqu'à ce que l'un des événements suivants se produise :

- La fonction **Connect** (1) est exécutée sur une nouvelle session.
- La fonction **Disconnect Presentation Space** (2) est exécutée.
- La fonction **Release** (12) est exécutée.
- La fonction **Reset System** (21) est exécutée.
- La fonction **Start Keystroke Intercept** (50) est exécutée.
- EHLLAPI le programme d'application est terminé.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 11	
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Code retour	Explication
0	La fonction **Reserve** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
5	L'espace de présentation ne peut pas être utilisé.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Si votre programme d'application EHLLAPI envoie une série de transactions à l'hôte, vous devrez peut-être empêcher l'utilisateur d'accéder à cette session jusqu'à ce que le traitement de votre demande soit terminé.
2. La saisie au clavier effectuée par un utilisateur lorsque le clavier est verrouillé par cette fonction est mise en file d'attente et traitée une fois la session terminée.
3. Cette fonction verrouille à la fois la saisie de la souris et celle du clavier. Le programme d'application doit déverrouiller l'espace de présentation pour permettre la saisie à la souris ou au clavier.

Reset System (21)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Reset System** réinitialise EHLLAPI à son état de départ. Les options des paramètres de session sont réinitialisées à leurs valeurs par défaut. La notification d'événement est arrêtée. La session hôte réservée est libérée. L'espace de présentation hôte est déconnecté. L'interception des frappes est désactivée.

Vous pouvez utiliser la fonction **Reset System** pendant l'initialisation ou à la fin du programme pour réinitialiser le système à un état initial connu.

Appels prérequis

Il n’y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 21	
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Code retour	Définition
0	La fonction **Reset System** a réussi.
1	EHLLAPI n'est pas chargé.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

Pour le PC/3270, cette fonction peut être utilisée pour vérifier si EHLLAPI est chargé. Appelez cette fonction au début de votre application et recherchez un code de retour de 1.

Search Field (30)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Search Field** examine un champ dans l'espace de présentation hôte connecté pour détecter l'occurrence d'une chaîne spécifiée. Si la chaîne cible est trouvée, cette fonction renvoie la position décimale de la chaîne numérotée à partir du début de l'espace de présentation hôte. (Par exemple, dans un espace de présentation de 24 lignes sur 80 colonnes, la position de la ligne 1, colonne 1 est numérotée *1* et la position de la ligne 5, colonne 1 est numérotée *321*.)

Cette fonction peut être utilisée pour rechercher des champs protégés ou non, mais uniquement dans un espace de présentation hôte *formaté par champ*.

Note: Si le champ à la fin de l'espace de présentation hôte est renvoyé à la ligne, celui-ci se produit lorsque la fin de l'espace de présentation est atteinte.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 30.	
Chaîne de données	Chaîne cible pour la recherche.	
Longueur	Longueur de la chaîne de données cible. Remplacé en mode EOT.	

	Interface standard	Interface améliorée
Position PS	Identifie le champ cible. Pour `SRCHALL`, cela peut être la position PS de n'importe quel octet dans le champ cible. Pour `SRCHFROM`, c'est le point de départ de la recherche de `SRCHFRWD` ou le point final de la recherche de `SRCHBKWD`. Voir la note 3 on page 127.	

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Les codes suivants sont définis :

Longueur	Explication
= 0	La chaîne n'a pas été trouvée.
> 0	La chaîne a été trouvée à la position indiquée dans l'espace de présentation hôte.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Search Field** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Erreur de paramètre. Soit la longueur de la chaîne était nulle, soit le mode EOT a été spécifié mais aucun caractère EOT n'a été trouvé dans la chaîne de données appelante.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	La chaîne de recherche n'a pas été trouvée ou l'espace de présentation hôte n'était pas formaté.

Notes sur l'utilisation de cette fonction

1. Quatre ensembles de paramètres sous la fonction **Set Session Parameters** (9) sont liés à cette fonction. Il s'agit des options de session SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD et EOT=c. Voir les éléments1 on page 144 à 4 on page 145 pour plus d'informations.
2. Vous pouvez utiliser la fonction **Set Session Parameters** (9) pour déterminer si vos recherches se déroulent en avant (SRCHFRWD) ou en arrière (SRCHBKWD) dans un champ.
3. La fonction **Search Field** vérifie normalement l'intégralité du champ (mode par défaut SRCHALL). Cependant, vous pouvez utiliser la fonction 9 pour spécifier SRCHFROM. Dans ce mode, le paramètre de position PS appelant fait plus que simplement identifier le champ cible. Il fournit également un point de début ou de fin pour la recherche.

- Si l'option SRCHFRWD est activée, la recherche de la chaîne désignée commence à la position PS spécifiée et se poursuit vers la fin du champ.
- Si l'option SRCHBKWD est activée, la recherche de la chaîne désignée commence à la fin du champ et continue vers la position PS spécifiée. Si la chaîne cible n'est pas trouvée, la recherche se termine à la position PS spécifiée dans le paramètre de position PS appelant.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la ligne 25 sur la ligne 24 ou sur la barre d'état. Pour que les informations soient affichées sur la barre d'état, la barre d'état doit être configurée. Consultez *Guide d'initiation* pour plus d'informations sur la configuration de la barre d'état. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Search Presentation Space (6)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Search Presentation Space** permet à votre programme EHLLAPI d'examiner l'espace de présentation hôte pour détecter l'occurrence d'une chaîne spécifiée.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 6.	
Chaîne de données	Chaîne cible pour la recherche.	
Longueur	Longueur de la chaîne de données cible. Remplacé en mode EOT.	
Position PS	Position dans l'espace de présentation hôte où la recherche doit commencer (option SRCHFRWD) ou se terminer (option SRCHBKWD). Remplacé en mode SRCHALL (par défaut).	

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Les codes suivants sont définis :

Longueur	Explication
= 0	La chaîne n'a pas été trouvée.
> 0	La chaîne a été trouvée à la position indiquée dans l'espace de présentation hôte.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Search Presentation Space** a réussi.
1	Votre programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification des paramètres.
7	La position de l'espace de présentation hôte n'est pas valide.
9	Une erreur système a été rencontrée.
24	La chaîne de recherche n'a pas été trouvée.

Notes sur l'utilisation de cette fonction

1. Quatre ensembles de paramètres sous la fonction **Set Session Parameters** (9) sont liés à cette fonction. Il s'agit des options de session SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD et EOT=c. Voir les éléments1 on page 144 à 4 on page 145à pour plus d'informations.
2. Vous pouvez utiliser la fonction **Set Session Parameters** (9) pour spécifier SRCHBKWD. Lorsque cette option est activée, l'opération de recherche localise la *dernière* occurrence de la chaîne.
3. La fonction **Search Presentation Space** vérifie normalement l'intégralité de l'espace de présentation hôte. Cependant, vous pouvez utiliser la fonction **Set Session Parameters** (9) pour spécifier SRCHFROM. Dans ce mode, le paramètre de position PS appelant spécifie un point de début ou de fin pour la recherche.
    - Si l'option SRCHFRWD est activée, la recherche de la chaîne désignée commence à la position PS spécifiée et se poursuit vers la fin de l'espace de présentation hôte.
    - Si l'option SRCHBKWD est activée, la recherche de la chaîne désignée commence à la fin du PS et recule vers la position PS spécifiée. Si la chaîne cible n'est pas trouvée, la recherche se termine à la position PS spécifiée dans le paramètre de position PS appelant.
4. L'option SRCHFROM est également utile si vous recherchez un mot clé susceptible d'apparaître plusieurs fois dans l'espace de présentation hôte.
5. La fonction **Search Presentation Space** est utile pour déterminer quand l'espace de présentation hôte est disponible. Si votre application EHLLAPI attend une invite ou un message spécifique avant d'envoyer des données, la fonction **Search Presentation Space** vous permet de rechercher un message d'invite avant de continuer.

Note: L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq. Z and I Emulator for Windows affiche les informations de la ligne 25 sur la ligne 24 ou sur la barre d'état. Pour que les informations soient affichées sur la barre d'état, la barre d'état doit être configurée. Consultez *Guide d'initiation* pour plus d'informations sur la configuration de la barre d'état. Avec l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication ManagerEHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.

Envoyer le fichier (90)

3270	*5250*	*VT*
Oui	Oui	Non

La fonctionnalité **Envoyer un fichier** est utilisée pour transférer un fichier de la session de poste de travail où EHLLAPI s'exécute vers une session hôte.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 90.	
Chaîne de données	Référez-vous aux exemples.	
Longueur	Longueur de la chaîne de données cible. Remplacé en mode EOT.	
Position PS	Doit être 0.	

Voici des exemples de chaînes de données pour SBCS

Session 3270

- Pour envoyer le fichier au système hôte VM/CMS :

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

- Pour envoyer le fichier au système hôte MVS/TSO :

  ```
  pc_filename [ID :]dataset[(membre)] [/mot de passe] [option]
  ```

- Pour envoyer le fichier au système hôte CICS :

  ```
  pc_filename [ID :]hôte_filename [(option]
  ```

Session 5250

- Pour envoyer le fichier au système hôte iSeries™, eServer™ i5 ou System i5™ :

```
pc_filename [ID :]membre du fichier de bibliothèque [option]
```

Paramètres de retour

Code retour	Explication
2	Erreur de paramètre ou vous avez spécifié une longueur trop longue (plus de 255 octets) pour le tampon EHLLAPI. Le transfert de fichiers a échoué.
3	Transfert de fichiers terminé.
4	Transfert de fichiers terminé avec enregistrements segmentés.
5	Le nom du fichier du poste de travail n'est pas valide ou est introuvable. Le transfert de fichiers a été annulé.
9	Une erreur système a été rencontrée.
27	Le transfert de fichiers s'est terminé en raison de l'utilisation d'un bouton Cancel ou du délai d'attente défini par la fonction **Set Session Parameter**(9).
101	Le transfert de fichiers a réussi (transfert vers/depuis CICS).

Si vous recevez le code retour 2 ou 9, il y a un problème avec le système ou avec la façon dont vous avez spécifié votre chaîne de données.

D'autres codes de retour peuvent également être reçus, liés aux numéros de message générés par le programme de transfert hôte. Pour les transferts vers un programme de transfert d'hôte CICS, soustrayez 100 du code retour pour obtenir la partie numérique du message. Par exemple, un code retour de 101 signifierait que le numéro de message INW0001 a été émis par l'hôte. Pour les autres programmes de transfert d'hôte, utilisez simplement le code retour comme partie numérique du message. Par exemple, un retour de 34 signifierait que le message TRANS34 a été émis par le programme de transfert hôte. La documentation de votre programme de transfert d'hôte devrait donner plus d'informations sur la signification des messages spécifiques.

Les codes d'erreur du système d'exploitation signalés par EHLLAPI sont supérieurs à 300. Pour déterminer le code d'erreur, soustrayez 300 et reportez-vous à la documentation du système d'exploitation pour connaître les codes de retour.

Notes sur l'utilisation de cette fonction

1. Quatre ensembles de paramètres sous la fonction **Set Session Parameters** (9) sont liés à cette fonction. Il s'agit des options de session QUIET/NOQUIET, STRLEN/STREOT, TIMEOUT=c/TIMEOUT=0 et EOT=c. Voir les éléments1 on page 144 et 2 on page 144, ainsi que les éléments 7 on page 145 et 8 on page 145 pour plus d'informations.

Send Key (3)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Send Key** est utilisée pour envoyer une frappe ou une chaîne de frappes à l'espace de présentation hôte.

Vous définissez la chaîne de frappes à envoyer avec le paramètre de chaîne de données appelante. Les frappes apparaissent à la session cible comme si elles avaient été saisies par l'opérateur du terminal. Vous pouvez également envoyer toutes les clés d'identification d'attention (AID) telles que Entrée, etc. Tous les champs hôtes dont la saisie est protégée ou qui sont uniquement numériques doivent être traité en conséquence.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 3.	
Chaîne de données	Une chaîne de frappes, maximum 255. Les caractères ASCII majuscules et minuscules sont représentés littéralement. Les touches de fonction et les touches de fonction décalées sont représentées par des mnémoniques. Voir Mnémoniques du clavier on page 134.	
Longueur	Longueur de la chaîne de données source. Annulé si en mode EOT.	
Position PS	NA	

Paramètres de retour

Code retour	**Explication**
0	Les frappes au clavier ont été envoyées ; l'état est normal.
1	Votre programme n'est pas connecté à une session hôte.
2	Un paramètre incorrect a été transmis à EHLLAPI.
4	La session hôte était occupée ; toutes les frappes n'ont pas pu être envoyées.
5	L'entrée dans la session cible a été inhibée ou rejetée ; toutes les frappes n'ont pas pu être envoyées.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Les paramètres sous la fonction **Set Session Parameters** (9) sont liés à cette fonction. Il s'agit des options de session AUTORESET/NORESET, STRLEN/STREOT, EOT=c, ESC=c et RETRY/NORETRY. Voir les éléments1 on page 144 et 2 on page 144, 9 on page 147 et 10 on page 147, et 19 on page 150 pour plus d'informations.
2. Les frappes ne peuvent pas être envoyées à la session hôte lorsque le clavier est verrouillé ou occupé. Vous pouvez vérifier cette condition avec la fonction **Wait** (4).
3. Si l'hôte est occupé, la saisie peut être rejetée.
4. La longueur de la chaîne de données doit être explicitement définie par le paramètre de longueur par défaut, mais elle peut être définie implicitement par l'option EOT=c de la fonction **Set Session Parameters** (9).

   Lors de la définition explicite de la longueur (voir élément 1), la valeur du paramètre de longueur transmis par l'application doit être calculée. Pour ce calcul, prévoyez 2 octets pour les frappes composées telles que `@E` et 4 octets pour les frappes composées telles que `@A@C`.
5. Pour envoyer des clés de contrôle spéciales, un schéma de codage de caractères composé est utilisé. Dans ce schéma de codage, une frappe est représentée par une séquence de deux à quatre caractères ASCII. Le premier et le troisième caractère sont toujours le caractère d'échappement. Les deuxième et quatrième caractères sont toujours un code clé.

   Pour envoyer la séquence `LOGON ABCDE` suivie de la touche Entrée, vous coderiez la chaîne `LOGON ABCDE@E`. Une liste complète de ces codes clés est représentée dans Mnémoniques du clavier on page 134.

   Cette technique de codage composé permet une représentation sous forme de chaîne ASCII de tous les codes de frappe nécessaires sans nécessiter l'utilisation de codes de touches hexadécimaux complexes.

   Le caractère d'échappement par défaut est `@`. La valeur du caractère d'échappement peut être remplacée par n'importe quel autre caractère avec l'option ESC=c de la fonction **Set Session Parameters** (9).
6. Les utilisateurs ayant besoin de niveaux de performances plus élevés doivent utiliser la fonction **Copy String to Field** (33) ou **Copy String to Presentation Space** (15) plutôt que d'envoyer des frappes avec la fonction **Send Key** (3). Mais n'oubliez pas que seule la fonction **Send Key** (3) peut envoyer les touches de contrôle spéciales.
7. Reportez-vous à l'option de session **Set Session Parameters (9) on page 142** 10 on page 147 (option NORESET) pour améliorer les performances de cette fonction.

   A moins que NORESET ne soit requis, le mnémonique de réinitialisation est ajouté aux chaînes de touches en tant que préfixe. Par conséquent, tous les statuts réinitialisables, à l'exception du blocage d'entrée, sont réinitialisés.

   L'option NORESET n'est pas la même que la fonction **Reset System** (21).
8. Les chaînes de frappe, y compris la clé AID, sont envoyées à l'hôte via plusieurs chemins. Chaque chemin envoie les chaînes avant la première clé AID (ou incluant la clé AID). EHLLAPI ajuste la longueur de la chaîne et la position de départ de chaque chemin. Pour un programme d'application hôte, toute frappe sur une touche peut être perdue par le processus de clé AID. Par conséquent, vous ne devez pas envoyer une liste de frappes comprenant plusieurs touches AID.

9. Lors du processus `@P` (Imprimer) ou `@A@T` (Imprimer l'espace de présentation), toutes les demandes mettant à jour l'espace de présentation sont rejetées. Si l'espace de présentation est occupé ou que la demande d'interruption survient pendant la demande d'impression, le mnémonique @A@R (Actualisation de l'appareil – Annuler pour imprimer l'espace de présentation) annule la demande et réinitialise le statut.

Mnémoniques du clavier

Les mnémoniques du clavier fournissent les caractères ASCII représentant les clés de fonction spéciales du clavier du poste de travail. Les codes abrégés facilitent la mémorisation des mnémoniques des clés spéciales. Un code de clé alphabétique est utilisé pour les clés les plus courantes. Par exemple, la touche **Clear** est *C* et la touche **Tab** est *T*.

Table 7: Mnémoniques avec caractères alphabétiques majuscules on page 134 montre les mnémoniques en utilisant des caractères alphabétiques majuscules :

Table 7. Mnémoniques avec caractères alphabétiques majuscules

Mnémonique	Signification	3270	5250	VT
@B	Tabulation gauche	Oui	Oui	Non
@C	Effacer	Oui	Oui	Non
@D	Supprimer	Oui	Oui	Non
@E	Entrée	Oui	Oui	Non
@F	Effacer EOF	Oui	Oui	Non
@H	Aide	Non	Oui	Non
@I	Insérer	Oui	Oui	Non
@J	Sauter (définir la mise au point)	Oui	Oui	Non
@L	Curseur vers la gauche	Oui	Oui	Oui
@N	Nouvelle ligne	Oui	Oui	Oui
@O	Espace	Oui	Oui	Oui
@P	Imprimer	Oui	Oui	Oui
@R	Réinitialiser	Oui	Oui	Non
@T	Tabulation droite	Oui	Oui	Oui
@U	Curseur vers le haute	Oui	Oui	Oui
@V	Curseur vers le bas	Oui	Oui	Oui
@Z	Curseur vers la droite	Oui	Oui	Oui

Table 8: Mnémoniques avec des chiffres ou des caractères minuscules on page 135 affiche les mnémoniques en utilisant un nombre ou des caractères alphabétiques minuscules.

Table 8. Mnémoniques avec des chiffres ou des caractères minuscules

Mnémonique	Signification	3270	5250	VT
@0	Domicile	Oui	Oui	Non
@1	PF1/F1	Oui	Oui	Non
@2	PF2/F2	Oui	Oui	Non
@3	PF3/F3	Oui	Oui	Non
@4	PF4/F4	Oui	Oui	Non
@5	PF5/F5	Oui	Oui	Non
@6	PF6/F6	Oui	Oui	Oui
@7	PF7/F7	Oui	Oui	Oui
@8	PF8/F8	Oui	Oui	Oui
@9	PF9/F9	Oui	Oui	Oui
@a	PF10/F10	Oui	Oui	Oui
@b	PF11/F11	Oui	Oui	Oui
@c	PF12/F12	Oui	Oui	Oui
@d	PF13	Oui	Oui	Oui
@e	PF14	Oui	Oui	Oui
@f	PF15	Oui	Oui	Oui
@g	PF16	Oui	Oui	Oui
@h	PF17	Oui	Oui	Oui
@i	PF18	Oui	Oui	Oui
@j	PF19	Oui	Oui	Oui
@k	PF20	Oui	Oui	Oui
@l	PF21	Oui	Oui	Non
@m	PF22	Oui	Oui	Non
@n	PF23	Oui	Oui	Non
@o	PF24	Oui	Oui	Non
@q	Fin	Oui	Oui	Non
@u	Touche Page précé-dente	Non	Oui	Non
@v	Touche Page suivante	Non	Oui	Non
@x	PA1	Oui	Oui	Non
@y	PA2	Oui	Oui	Non
@z	PA3	Oui	Oui	Non

Table 9: Mnémoniques avec les caractères @A et @ alphabétiques majuscules on page 136 affiche les mnémoniques en utilisant la combinaison de clés @A et @alphabetic uppercase (A – Z).

Table 9. Mnémoniques avec les caractères @A et @ alphabétiques majuscules

Mnémonique	Signification	3270	5250	VT
@A@C	Tester	Non	Oui	Non
@A@D	Supprimer un mot	Oui	Oui	Non
@A@E	Zone Quitter	Oui	Oui	Non
@A@F	Effacer l'entrée	Oui	Oui	Non
@A@H	Demande du système	Oui	Oui	Non
@A@I	Insérer une bascule	Oui	Oui	Non
@A@J	Sélection à l'aide du curseur	Oui	Oui	Non
@A@L	Curseur vers la gauche rapidement	Oui	Oui	Non
@A@Q	Attention	Oui	Oui	Non
@A@R	Annulation de l'appareil (annule l'impression de l'espace de présentation)	Oui	Oui	Non
@A@T	Imprimer l'espace de présentation	Oui	Oui	Oui
@A@U	Curseur vers le haut rapidement	Oui	Oui	Non
@A@V	Curseur vers le bas rapidement	Oui	Oui	Non
@A@Z	Curseur vers la droite rapidement	Oui	Oui	Non

Table 10: Mnémoniques avec les caractères @A et @ alphabétiques minuscules on page 136 affiche les mnémoniques en utilisant la combinaison de clés @A et @number ou @A et @alphabetic lowercase (a–z).

Table 10. Mnémoniques avec les caractères @A et @ alphabétiques minuscules

Mnémonique	Signification	3270	5250	VT
@A@9	Vidéo inversée	Oui	Oui	Non
@A@b	Tiret de soulignement	Oui	Non	Non
@A@C	Réinitialiser la vidéo inversée	Oui	Non	Non
@A@D	Rouge	Oui	Non	Non
@A@E	Rose	Oui	Non	Non
@A@F	Vert	Oui	Non	Non
@A@g	Jaune	Oui	Non	Non
@A@H	Bleu	Oui	Non	Non
@A@I	Turquoise	Oui	Non	Non

Table 10. Mnémoniques avec les caractères @A et @ alphabétiques minuscules (continued)

Mnémonique	Signification	3270	5250	VT
@A@J	Blanc	Oui	Non	Non
@A@L	Réinitialiser les couleurs de l'hôte	Oui	Non	Non
@A@T	Imprimer (ordinateur personnel)	Oui	Oui	Non
@A@Y	Tabulation mot en avant	Oui	Oui	Non
@A@Z	Tabulation mot en arrière	Oui	Oui	Non

Table 11: Mnémoniques avec caractères @A et @ alphanumériques (spéciaux) on page 137 affiche les mnémoniques en utilisant la combinaison @A et @caractère spécial.

Table 11. Mnémoniques avec caractères @A et @ alphanumériques (spéciaux)

Mnémonique	Signification	3270	5250	VT
@A@-	Champ -	Non	Oui	Non
@A@+	Champ +	Non	Oui	Non
@A@<	Enregistrer le retour arrière	Non	Oui	Non

Table 12: Mnémoniques avec @S (Shift), @W (Edit) et @ caractères alphabétiques on page 137 montre les mnémoniques en utilisant la combinaison @S , @W et @alphabetic lowercase.

Table 12. Mnémoniques avec @S (Shift), @W (Edit) et @ caractères alphabétiques

Mnémonique	Signification	3270	5250	VT
@S@E	Imprimer l'espace de présentation sur l'hôte	Non	Oui	Non
@S@x	Dup	Oui	Oui	Non
@S@y	Zone Marquer	Oui	Oui	Non
@W@C	Modifier Copier	Oui	Oui	Oui
@W@D	Modifier Effacer	Oui	Oui	Oui
@W@E	Modifier Copier Ajouter	Oui	Oui	Oui
@W@L	Modifier Copier Liaison	Oui	Oui	Oui
@W@N	Modifier Coller Suivant	Oui	Oui	Oui
@W@V	Modifier Coller	Oui	Oui	Oui

Table 12. Mnémoniques avec @S (Shift), @W (Edit) et @ caractères alphabétiques (continued)

Mnémonique	Signification	3270	5250	VT
@W@X	Modifier Couper	Oui	Oui	Oui
@W@Z	Modifier Annuler	Oui	Oui	Oui

Note: Les mnémoniques d'édition @W sont pris en charge uniquement dans les fonctions EHLLAPI en mode amélioré. Voir la fonction Start Keystroke Intercept sous Résumé des fonctions EHLLAPI on page 32.

VT uniquement : Table 13: Mnémoniques utilisant @M, @Q et @Alphabetic Lowercase (pour VT uniquement) on page 138 affiche les mnémoniques en utilisant la combinaison @M et @number ou @alphabetic lowercase (a-z)

Table 13. Mnémoniques utilisant @M, @Q et @Alphabetic Lowercase (pour VT uniquement)

Mnémonique	Signification	3270	5250	VT
@M@0	Pavé numérique VT 0	Non	Non	Oui
@M@1	VT Numeric Pad 1	Non	Non	Oui
@M@2	Pavé numérique VT 2	Non	Non	Oui
@M@3	Pavé numérique VT 3	Non	Non	Oui
@M@4	Pavé numérique VT 4	Non	Non	Oui
@M@5	Pavé numérique VT 5	Non	Non	Oui
@M@6	Pavé numérique VT 6	Non	Non	Oui
@M@7	Pavé numérique VT 7	Non	Non	Oui
@M@8	Pavé numérique VT 8	Non	Non	Oui
@M@9	Pavé numérique VT 9	Non	Non	Oui
@M@-	Pavé numérique VT -	Non	Non	Oui
@M@,	Pavé numérique VT,	Non	Non	Oui
@M@.	Pavé numérique VT .	Non	Non	Oui
@M@e	Pavé numérique VT Entrée	Non	Non	Oui
@M@f	VT Modifier Rechercher	Non	Non	Oui
@M@i	VT Modifier Insérer	Non	Non	Oui
@M@r	VT Modifier Sup-primer	Non	Non	Oui
@M@s	VT Modifier Sélection-ner	Non	Non	Oui
@M@p	VT Modifier l'écran précédent	Non	Non	Oui
@M@n	VT Modifier l'écran suivant	Non	Non	Oui
@M@a	VT PF1	Non	Non	Oui

Table 13. Mnémoniques utilisant @M, @Q et @Alphabetic Lowercase (pour VT uniquement) (continued)

Mnémonique	Signification	3270	5250	VT
@M@b	VT PF2	Non	Non	Oui
@M@c	VT PF3	Non	Non	Oui
@M@d	VT PF4	Non	Non	Oui
@M@h	VT Conserver l'écran	Non	Non	Oui
@M@(espace)	Code de contrôle NUL	Non	Non	Oui
@M@A	Code de contrôle SOH	Non	Non	Oui
@M@B	Code de contrôle STX	Non	Non	Oui
@M@C	Code de contrôle ETX	Non	Non	Oui
@M@D	Code de contrôle EOT	Non	Non	Oui
@M@E	Code de contrôle ENQ	Non	Non	Oui
@M@F	Code de contrôle ACK	Non	Non	Oui
@M@G	Code de contrôle BEL	Non	Non	Oui
@M@H	Code de contrôle BS	Non	Non	Oui
@M@I	Code de contrôle HT	Non	Non	Oui
@M@J	Code de contrôle LF	Non	Non	Oui
@M@K	Code de contrôle VT	Non	Non	Oui
@M@L	Code de contrôle FF	Non	Non	Oui
@M@M	Code de contrôle CR	Non	Non	Oui
@M@N	Code de contrôle SO	Non	Non	Oui
@M@O	Code de contrôle SI	Non	Non	Oui
@M@P	Code de contrôle DLE	Non	Non	Oui
@M@Q	Code de contrôle DC1	Non	Non	Oui
@M@R	Code de contrôle DC2	Non	Non	Oui
@M@S	Code de contrôle DC3	Non	Non	Oui
@M@T	Code de contrôle DC4	Non	Non	Oui
@M@U	Code de contrôle NAK	Non	Non	Oui
@M@V	Code de contrôle SYN	Non	Non	Oui
@M@W	Code de contrôle ETB	Non	Non	Oui
@M@X	Code de contrôle CAN	Non	Non	Oui
@M@Y	Code de contrôle EM	Non	Non	Oui
@M@Z	Code de contrôle SUB	Non	Non	Oui
@M@U	Code de contrôle ESC	Non	Non	Oui
@M@V	Code de contrôle FS	Non	Non	Oui
@M@W	Code de contrôle GS	Non	Non	Oui
@M@X	Code de contrôle RS	Non	Non	Oui
@M@Y	Code de contrôle US	Non	Non	Oui
@M@Z	Code de contrôle DEL	Non	Non	Oui

Table 13. Mnémoniques utilisant @M, @Q et @Alphabetic Lowercase (pour VT uniquement) (continued)

Mnémonique	Signification	3270	5250	VT
@Q@A	Clé définie par l'utilisateur VT 6	Non	Non	Oui
@Q@B	Clé définie par l'utilisateur VT 7	Non	Non	Oui
@Q@C	Clé définie par l'utilisateur VT 8	Non	Non	Oui
@Q@D	Clé définie par l'utilisateur VT 9	Non	Non	Oui
@Q@E	Clé définie par l'utilisateur VT 10	Non	Non	Oui
@Q@F	Clé définie par l'utilisateur VT 11	Non	Non	Oui
@Q@G	Clé définie par l'utilisateur VT 12	Non	Non	Oui
@Q@H	Clé définie par l'utilisateur VT 13	Non	Non	Oui
@Q@I	Clé définie par l'utilisateur VT 14	Non	Non	Oui
@Q@J	Clé définie par l'utilisateur VT 15	Non	Non	Oui
@Q@K	Clé définie par l'utilisateur VT 16	Non	Non	Oui
@Q@L	Clé définie par l'utilisateur VT 17	Non	Non	Oui
@Q@M	Clé définie par l'utilisateur VT 18	Non	Non	Oui
@Q@N	Clé définie par l'utilisateur VT 19	Non	Non	Oui
@Q@0	Clé définie par l'utilisateur VT 20	Non	Non	Oui
@Q@A	VT Retour arrière	Non	Non	Oui
@Q@r	VT Effacer la page	Non	Non	Oui
@Q@s	VT Modifier	Non	Non	Oui

Le tableau suivant présente les mnémoniques utilisant un caractère spécial.

Table 14. Mnémoniques avec clés de caractères spéciaux

Mnémonique	Signification	3270	5250	VT
@@	@	Oui	Oui	Oui

Table 14. Mnémoniques avec clés de caractères spéciaux (continued)

Mnémonique	Signification	3270	5250	VT
@$	Curseur alternatif (interface Presentation Manager® uniquement)	Oui	Oui	Oui
@<	Retour arrière	Oui	Oui	Oui

Les clés de caractères suivantes sont interprétées telles quelles.

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

Set Cursor (40)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Set Cursor** est utilisée pour définir la position du curseur dans l'espace de présentation hôte. Avant d'utiliser la fonction **Set Cursor**, une application du poste de travail doit être connectée à l'espace de présentation hôte.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 40	
Chaîne de données	NA	
Longueur	NA	
Position PS	Position souhaitée du curseur dans l'espace de présentation de l'hôte connecté	

Paramètres de retour

Code retour	Explication
0	Le curseur a été localisé avec succès à la position spécifiée.

Code retour	Explication
1	Votre programme n'est pas connecté à une session hôte.
4	La session est occupée.
7	Un emplacement de curseur inférieur à 1 ou supérieur à la taille de l'espace de présentation de l'hôte connecté a été spécifié.
9	Une erreur système s'est produite.

Set Session Parameters (9)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Set Session Parameter** vous permet de modifier certaines options de session par défaut dans EHLLAPIpour toutes les sessions. Quand EHLLAPI est chargé, les paramètres par défaut des options de session sont ceux indiqués par les entrées soulignées dans les tableaux qui apparaissent dans Options de session on page 144. N'importe lequel, certains ou tous ces paramètres peuvent être modifiés en incluant l'option souhaitée dans la chaîne de données d'appel, comme expliqué ci-dessous. Les paramètres spécifiés restent en vigueur jusqu'à ce que :

- Ils soient modifiés par une fonction ultérieure **Set Session Parameters** (9) qui spécifie une nouvelle valeur.
- La fonction **Reset System** (21) est exécutée.
- Le programme d'application EHLLAPI est terminé.

Le tableau suivant répertorie les fonctions EHLLAPI affectées par les options de session. Les fonctions non répertoriées dans le tableau ne sont affectées par aucune des options de session. Les options de session qui affectent chaque fonction sont indiquées par les entrées correspondantes dans la colonne *"Voir les éléments"*. Ces entrées sont indexées dans la liste qui suit Paramètres d'appel on page 143.

Numéro de fonction	Nom de fonction	Voir les éléments
1	**Connect Presentation Space**	**11 on page 147**, **21 on page 151**, **22 on page 151**
3	**Send Key**	**1 on page 144**, **2 on page 144**, **9 on page 147**, **10 on page 147**, **19 on page 150**
4	**Wait**	**12 on page 147**
5	**Copy Presentation Space**	**5 on page 145**, **13 on page 148**, **14 on page 149**, **15 on page 149**, **17 on page 149**, **20 on page 150**
6	**Search Presentation Space**	**1 on page 144**, **2 on page 144**, **3 on page 144**, **4 on page 145**

Numéro de fonction	Nom de fonction	Voir les éléments
8	Copy Presentation Space to String	5 on page 145, 13 on page 148, 14 on page 149, 15 on page 149, 17 on page 149, 20 on page 150
10	Query Sessions	16 on page 149, 20 on page 150
15	Copy String to Presentation Space	1 on page 144, 2 on page 144, 13 on page 148, 14 on page 149, 18 on page 150, 20 on page 150
18	Pause	6 on page 145
30	Search Field	1 on page 144, 2 on page 144, 3 on page 144, 4 on page 145, 20 on page 150
33	Copy String to Field	1 on page 144, 2 on page 144, 13 on page 148, 14 on page 149, 18 on page 150, 20 on page 150
34	Copy Field to String	5 on page 145, 13 on page 148, 14 on page 149, 17 on page 149, 20 on page 150
35	Copy Presentation Space to Clipboard	5 on page 145, 13 on page 148, 14 on page 149, 17 on page 149, 20 on page 150
36	Paste Clipboard to Presentation Space	1 on page 144, 2 on page 144, 13 on page 148, 14 on page 149, 18 on page 150, 20 on page 150
51	Get Key	9 on page 147, 12 on page 147
90	Send File	1 on page 144, 2 on page 144, 7 on page 145, 8 on page 145
91	Receive File	1 on page 144, 2 on page 144, 7 on page 145, 8 on page 145
101	Connect Window Services	21 on page 151, 22 on page 151

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 9.	
Chaîne de données	Chaîne contenant les valeurs souhaitées des options de session qui doivent être modifiées. La chaîne de données peut contenir n'importe	

	Interface standard	Interface améliorée
	quelle valeur dans les tables deOptions de session on page 144. Les valeurs doivent être placées sur la ligne de chaîne de données, séparées par des virgules ou des espaces. Les ensembles de paramètres sont expliqués en fonction des fonctions qu'ils affectent.	
Longueur	Longueur explicite de la chaîne de données source (l'option `STREOT` n'est pas autorisée).	
Position PS	NA.	

Options de session

Les tableaux suivants présentent les options de session. La valeur par défaut est soulignée.

1. Les valeurs du tableau suivant déterminent la façon dont la longueur de la chaîne de données est définie pour les fonctions **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90) et **Receive File** (91).

Valeur	Explication
STRLEN	Une longueur explicite est transmise pour toutes les chaînes.
STREOT	Les longueurs ne sont pas explicitement codées. Les chaînes de données appelantes (source) se terminent par un caractère EOT.

2. L'instruction du tableau suivant est utilisée pour spécifier le caractère utilisé comme délimiteur de fin de texte (EOT) dans la chaîne de données appelante (source) pour les fonctions EHLLAPI **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90) et **Receive File** (91).

Valeur	Explication
EOT=c	Permet de spécifier le caractère EOT pour les terminateurs de chaîne (en mode STREOT). Le zéro binaire est la valeur par défaut. Ne laissez pas de blanc après le symbole égal.

   Pour être valide, `c` doit être saisi sous la forme d'un caractère littéral de chaîne de 1 octet sans espaces précédents. Le caractère EOT spécifié par cette instruction est utilisé pour déterminer la longueur d'une chaîne de données appelante uniquement lorsque l'option STREOT (voir élément 1) est en vigueur.
3. Les valeurs du tableau suivant affectent les fonctions de recherche **Search Presentation Space** (6) et **Search Field** (30).

Valeur	Explication
SRCHALL	La fonction **Search Presentation Space** (6) et la fonction **Search Field** (30) balayent l'intégralité de l'espace ou du champ de présentation hôte.

Valeur	Explication
SRCHFROM	La fonction **Search Presentation Space** (6) et la fonction **Search Field** (30) commencent à partir d'une position PS spécifiée (pour `SRCHFRWD`) ou se terminent à une position PS spécifiée (pour `SRCHBKWD`).

4. Les valeurs du tableau suivant affectent les fonctions de recherche **Search Presentation Space** (6) et **Search Field** (30). Elles déterminent la direction de la recherche.

Valeur	Explication
SRCHFRWD	La fonction **Search Presentation Space** (6) et la fonction **Search Field** (30) fonctionnent dans une direction ascendante.
SRCHBKWD	La fonction **Search Presentation Space** (6) et la fonction **Search Field** (30) fonctionnent dans une direction descendante. Une recherche est satisfaite si le premier caractère de la chaîne demandée commence dans les limites spécifiées pour la recherche.

5. Les valeurs du tableau suivant déterminent la manière dont les octets d'attribut sont traités pour les fonctions **Copy Presentation Space** (5), **Copy Presentation Space to String** (8) et **Copy Field to String** (34).

Valeur	Explication
NOATTRB	Convertissez toutes les valeurs inconnues en blancs.
ATTRB	Renvoyez tous les codes qui n'ont pas d'équivalent ASCII comme valeurs d'origine.
NULLATTRB	Convertissez tous les attributs de champ en caractères nuls.

6. Les valeurs du tableau suivant affectent la fonction **Pause** (18).

Valeur	Explication
FPAUSE	Une pause complète dure la durée que vous avez spécifiée dans la fonction **Pause** (18).
IPAUSE	Pause interrompue. Après l'exécution de la fonction **Start Host Notification** (23), un événement hôte satisfait à une pause.

7. Les valeurs du tableau suivant déterminent si les messages générés par les fonctions de transfert de fichiers **Send File** (90) et **Receive File** (91) sont affichés.

Valeur	Explication
NOQUIET	Les messages SEND et RECEIVE s’affichent.
QUIET	Les messages SEND et RECEIVE ne sont pas affichés.

8. Les instructions du tableau suivant déterminent combien de temps Z and I Emulator for WindowsEHLLAPI attend avant d'émettre automatiquement une annulation lors de l'exécution des fonctions de transfert de fichiers **Send File** (90) et **Receive File** (91). Pour être valide, `c` doit être une lettre majuscule J à N et ne doit pas être précédé d'un espace.

Valeur	Explication
TIMEOUT=0	Une annulation est automatiquement émise après un délai de 20 secondes (environ).
TIMEOUT=c	Une annulation est automatiquement émise après un délai spécifié. Un indicateur à 1 caractère du tableau ci-dessous indique à Z and I Emulator for Windows combien de cycles de 30 secondes il doit accepter avant d'émettre lui-même une annulation. **Caractère** **Valeur (en minutes)** **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 **J** 5.0 **K** 5.5 **L** 6.0

Valeur	Explication
	M 6.5 **N** 7.0

9. L'instruction du tableau suivant est utilisée pour définir le caractère d'échappement pour les mnémoniques de frappe. Cette option de session affecte les fonctions **Send Key** (3) et **Get Key** (51). La valeur de `c` doit être saisie sous la forme d'une chaîne de caractères littéraux de 1 octet sans espaces précédents.

Valeur	Explication
ESC=c	Spécifie le caractère d'échappement pour les mnémoniques de frappe (@ est la valeur par défaut). Ne laissez pas de blanc après le symbole égal. Un espace n'est pas un caractère d'échappement valide.

10. Les valeurs du tableau suivant déterminent si EHLLAPI précède automatiquement les chaînes envoyées à l'aide de la fonction **Send Key** (3) d'une réinitialisation.

Valeur	Explication
AUTORESET	EHLLAPI tente de réinitialiser toutes les conditions inhibées en préfixant toutes les chaînes de clés envoyées à l'aide de la fonction **Send Key** (3) avec une réinitialisation.
NORESET	Ne pas exécuter AUTORESET.

11. Les valeurs du tableau suivant affectent la manière dont la commande **Connect Presentation Space** (1) fonctionne.

Valeur	Explication
CONLOG	Etablit une connexion logique entre la session du poste de travail et une session hôte. Pendant la connexion, ne passe pas à l'espace de présentation demandé.
CONPHYS	Etablit une connexion physique entre la session du poste de travail et une session hôte. Pendant la connexion, passe à l'espace de présentation demandé.

12. Les valeurs du tableau suivant affectent la fonction **Wait** (4) et la fonction **Get Key** (51). Pour chaque valeur, il existe deux effets différents, un pour chaque fonction.

Valeur	Explication
TWAIT	Pour la fonction **Wait** (4), attend jusqu'à une minute avant l'expiration du délai sur XCLOCK (X []) ou XSYSTEM.

Valeur	Explication
	Pour la fonction **Get Key** (51), ne rend pas le contrôle à votre programme d'application EHLLAPI tant qu'il n'a pas intercepté une clé (clé normale ou AID en fonction de l'option spécifiée sous la fonction **Start Keystroke Intercept** (50)).
LWAIT	Pour la fonction **Wait** (4), attend que XCLOCK (X [])/XSYSTEM s'efface. Cette option n'est pas recommandée, car le contrôle ne revient pas à votre application tant que l'hôte n'est pas disponible. Pour la fonction **Get Key** (51), ne rend pas le contrôle à votre programme d'application EHLLAPI tant qu'il n'a pas intercepté une clé (clé normale ou AID en fonction de l'option spécifiée sous la fonction **Start Keystroke Intercept** (50)).
NWAIT	Pour la fonction **Wait** (4), vérifie le statut et revient immédiatement (pas d'attente). Pour la fonction **Get Key** (51), renvoie le code retour 25 (frappes de touches non disponibles) dans le quatrième paramètre si rien n'est mis en file d'attente correspondant à l'option spécifiée sous la fonction **Start Keystroke Intercept** (50).

Note: L'utilisation de NWAIT est recommandée.

13. Les valeurs du tableau suivant affectent **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33) et **Copy Field to String** (34). Les octets d'attributs étendus (EAB) incluent les attributs de caractères étendus et les attributs de champ étendus.

Valeur	Explication
NOEAB	Transmettez uniquement les données, pas les EAB.
EAB	Transmettez les données de l'espace de présentation avec des octets d'attributs étendus. Pour chaque caractère qui apparaît à l'écran, 2 octets de données sont transmis. Par conséquent, un tampon deux fois plus grand que l'espace de présentation doit être pré-alloué ; par exemple 2 x 1920 = 3840 pour un espace de présentation de 24 lignes sur 80 colonnes. Les attributs étendus pour une chaîne de caractères peuvent être signalés comme attributs de l'octet de champ, plutôt que comme attributs de chaque caractère individuel du champ. Dans ce cas, pour savoir si un caractère ou un ensemble de caractères particulier sur un écran est souligné, effectuez un CopyPStoString spécifiant la position de l'octet d'attribut de champ (l'octet avant le champ affiché à l'écran) pour obtenir les informations des EAB qui s'appliquent à tous les caractères de ce champ.

Note: Lors de l'utilisation de la fonction **EHLLAPI Copy PS to String**, le texte est copié et doit être invisible pour l'opérateur. Utilisez la fonction EHLLAPI Set Session Parameters pour définir l'option NODISPLAY afin de déterminer s'il existe des données masquées. Cela amène EHLLAPI à renvoyer les

champs non affichés comme Null. Une autre procédure courante pour masquer des données consiste à définir les mêmes couleurs de premier plan et d'arrière-plan (NOIR, par exemple) afin que le texte soit affiché, mais non visible par l'opérateur humain. La seule façon pour votre application de détecter cela est d'utiliser les paramètres de session EAB et XLATE, puis de copier le PS. La couleur de premier plan/arrière-plan de chaque position est renvoyée et vous pouvez déterminer quels caractères sont invisibles.

14. Les valeurs du tableau suivant affectent Copy Presentation Space (5), Copy Presentation Space to String (8), Copy String to Presentation Space (15), Copy String to Field (33) et Copy Field to String (34).

Valeur	Explication
NOXLATE	Les EAB ne sont pas traduits.
XLATE	Les EAB sont traduits au format de l'adaptateur graphique couleur (CGA) pour PC.

15. Les valeurs du tableau suivant affectent **Copy Presentation Space** (5), **Copy Presentation Space to String** (8) et **Copy Presentation Space to Clipboard** (35) si NOATTRB et NOEAB sont spécifiés.

Valeur	Explication
VIERGE	Convertissez toutes les valeurs inconnues en X'20'.
NOBLANK	Convertissez toutes les valeurs inconnues en X'00'.

La valeur par défaut est BLANK. Si vous souhaitez modifier la valeur par défaut en NOBLANK, ajoutez l'instruction suivante dans le fichier PCSWIN.INI situé dans le répertoire de données d'application de classe utilisateur Z and I Emulator for Windows :

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

16. Les valeurs du tableau suivant affectent la taille de l'espace de présentation renvoyée par les sessions **Query Sessions** (10).

Valeur	Explication
CFGSIZE	Renvoie la taille configurée de l'espace de présentation connecté. Cette option ignore tout remplacement de la taille configurée par l'hôte.
NOCFGSIZE	Renvoie la taille actuelle de l'espace de présentation connecté.

17. Les valeurs du tableau suivant affectent **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy Field to String** (34) et **Copy Presentation Space to Clipboard** (35).

Valeur	Explication
DISPLAY	Copiez les champs non affichés de l'espace de présentation vers la zone tampon cible de la même manière que les champs d'affichage. Les applications actuelles fonctionnent normalement.
NODISPLAY	Ne copiez pas les champs non affichés de l'espace de présentation vers la zone tampon cible. Copiez les champs non affichés dans le tampon cible sous la forme d'une chaîne de caractères Null. Cela permet aux applications d'af-

Valeur	Explication
	ficher les tampons copiés dans la fenêtre de présentation sans afficher d'informations confidentielles, telles que les mots de passe.

18. Les valeurs du tableau suivant affectent **Copy String to Presentation Space** (15), **Copy String to Field** (33) et **Paste Clipboard to Presentation Space** (36).

Valeur	Explication
NOPUTEAB	EAB n'est pas contenu dans la chaîne de données de **Copy String to Presentation Space** ou **Copy String to Field**.
PUTEAB	EAB est contenu avec des données de caractères dans la chaîne de données de **Copy String to Presentation Space** ou **Copy String to Field**.

    Cette option est utilisée pour la compatibilité avec Communication Manager/2. Pour Communication Manager/2, la chaîne de données, qui est spécifiée dans **Copy String to Presentation Space** ou **Copy String to Field**, doit contenir EAB (ou EAD) avec des données de caractères lorsque EAB (ou EAD) est valide dans **Set Session Parameters**. A l'inverse, pour le précédent Z and I Emulator for Windows, la chaîne de données spécifiée dans ces fonctions doit être constituée de données de caractères uniquement même si EAB (ou EAD) est valide. Cependant, Z and I Emulator for Windows permet que la chaîne de données contienne EAB (ou EAD) en définissant PUTEAB pour assurer la compatibilité avec Communication Manager/2.
19. Les valeurs du tableau suivant affectent la fonction **Send Key** (3). Les frappes ne sont pas traitées si le clavier est bloqué ou en cours d'utilisation. Les options déterminent si la fonction tente de renvoyer les frappes jusqu'à ce qu'un délai d'attente de 4 minutes se produise ou si la fonction revient immédiatement après avoir déterminé que le clavier est bloqué ou en cours d'utilisation.

Valeur	Explication
RETRY	Continue de tenter d'envoyer des frappes jusqu'à ce qu'elles soient envoyées ou jusqu'à ce qu'un délai d'attente de 4 minutes se produise.
NORETRY	Renvoie immédiatement après avoir déterminé que le clavier est bloqué ou en cours d'utilisation.

20. Les valeurs du tableau suivant affectent **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) et **Paste Clipboard to Presentation Space** (36).

Valeur	Explication
EXTEND_PS	L'émulation 5250 prend en charge un espace de présentation de 24 lignes sur 80 colonnes. Dans certains cas, l'émulation Communication Manager 5250 affiche une 25ème ligne. Cela se produit lorsqu'un message d'erreur de l'hôte s'affiche ou lorsque l'opérateur sélectionne la clé SysReq.Z and I Emulator for Windows affiche les informations de la 25e ligne sur la ligne 24, mais EHLLAPI voit normalement la *vraie* 24ème rangée. Par l'option **EXTEND_PS**, une application EHLLAPI peut utiliser la même interface avec Communication Manager-

Valeur	Explication
	EHLLAPI et l'espace de présentation valide est étendu lorsque cette condition se produit.
NOEXTEND_PS	L'espace de présentation n'est pas étendu lorsque la condition ci-dessus se produit. Il s'agit de la valeur par défaut.

21. Les valeurs du tableau suivant affectent les fonctions **Connect Presentation Space** (1) et **Connect Window Services** (101). Les options précisent si une application peut ou va partager l'espace de présentation auquel elle est connectée avec une autre application. Une seule des valeurs suivantes peut être spécifiée avec chaque appel **Set Session Parameter**.

Valeur	Explication
SUPER_WRITE	L'application permet à d'autres applications autorisant le partage et disposant d'autorisations d'accès en écriture de se connecter simultanément au même espace de présentation. L'application d'origine exécute des fonctions de type supervision mais ne crée pas d'erreurs pour les autres applications partageant l'espace de présentation.
WRITE_SUPER	L'application nécessite un accès en écriture et permet uniquement à l'application de supervision de se connecter simultanément à son espace de présentation. Il s'agit de la valeur par défaut.
WRITE_WRITE	L'application nécessite un accès en écriture et permet à des applications partenaires ou à d'autres applications ayant un comportement prévisible de partager l'espace de présentation.
WRITE_READ	L'application nécessite un accès en écriture et permet à d'autres applications qui exécutent des fonctions en lecture seule de partager l'espace de présentation. L'application est également autorisée à copier l'espace de présentation et à effectuer d'autres opérations en lecture seule comme d'habitude.
WRITE_NONE	L'application a l'usage exclusif de l'espace de présentation. Aucune autre application n'est autorisée à partager l'espace de présentation, y compris les applications de supervision. L'application est autorisée à copier l'espace de présentation et à effectuer des opérations en lecture seule comme d'habitude.
READ_WRITE	L'application nécessite uniquement un accès en lecture pour surveiller l'espace de présentation et permet à d'autres applications effectuant des fonctions de lecture ou d'écriture, ou les deux, de partager l'espace de présentation. L'application est également autorisée à copier l'espace de présentation et à effectuer d'autres opérations en lecture seule comme d'habitude.

22. Les valeurs du tableau suivant permettent aux applications qui ont des exigences de partage d'espace de présentation de limiter le partage à une application partenaire (une application qui a été développée pour fonctionner avec elle).

Valeur	Explication
NOKEY	Permet à l'application d'être compatible avec les applications existantes qui ne spécifient pas le paramètre **KEY**.
KEY$*nnnnnnnn*	Utilise un mot clé pour restreindre l'accès au partage à l'espace de présentation qu'il prend en charge. Le mot clé doit avoir une longueur exacte de 8 octets.

Paramètres de retour

Cette fonction renvoie une longueur et un code retour.

Longueur :

Nombre de paramètres de session valides définis.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	Les paramètres de la session ont été définis.
2	Un ou plusieurs paramètres n'étaient pas valides.
9	Une erreur système a été rencontrée.

Start Close Intercept (41)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Start Close Intercept** permet à l'application d'intercepter les demandes de fermeture générées lorsqu'un utilisateur sélectionne l'option de fermeture dans la fenêtre de session de l'émulateur. Cette fonction intercepte la demande de fermeture et l'ignore jusqu'à ce qu'une fonction **Start Close Intercept** (43) soit demandée.

Après avoir utilisé cette fonction, votre programme d'application peut utiliser la fonction **Query Close Intercept** (42) pour déterminer quand une demande de fermeture s'est produite.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

Octet	**Définition**	
	Interface standard	Interface améliorée
Numéro de fonction	Doit être 41	
Chaîne de données	Voir le tableau suivant	
Longueur	5 ou 6	Doit être 12
Position PS	NA	

La chaîne de données contient les éléments suivants.

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
4–5		Les données dans ces positions sont ignorées par EHLLAPI. Cependant, aucune erreur n'est générée si le programme de migration contient des données à ces positions. Ces données sont acceptées pour assurer la compatibilité avec les applications en migration.
6	5	Spécifiez M pour demander le mode de message asynchrone (Windows uniquement).
	6–8	Réservé.
2–3	9–12	Lorsque M est spécifié en position 5 (6 pour 16 bits), le descripteur de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL) (différente de 0).

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Si le mode de message asynchrone n'est pas spécifié en position 5 (6 pour l'interface standard) et que la fonction s'exécute avec succès, la chaîne de données suivante est renvoyée.

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–8	Réservé.
	9–12	Valeur de 4 octets dans laquelle l'adresse de l'objet événement est renvoyée par EHLLAPI. L'application peut attendre cet objet événement. (32 bits uniquement).

Chaîne de données :

Si M (mode de message asynchrone) est spécifié en position 5 (6 pour l'interface standard) et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée.

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–8	Réservé
2–3	9–10	ID de tâche du mode de message asynchrone

Note: Si un utilisateur sélectionne l'option de fermeture, une fenêtre d'application reçoit un message. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL). Le paramètre wParam contiendra l'ID de tâche renvoyé par cet appel de fonction. Le HIWORD du paramètre lParam contiendra le code retour 26, qui indique qu'une interception de fermeture s'est produite, et le LOWORD du paramètre lParam contiendra le numéro de fonction 41.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Start Close Intercept** a réussi.
1	Un espace de présentation hôte incorrect a été spécifié.
2	Une erreur de paramètre s'est produite.
9	Une erreur système s'est produite.
10	La fonction n'est pas prise en charge par le programme d'émulation.

Notes sur l'utilisation de cette fonction

1. L'objet événement ou le sémaphore renvoyé est dans un état non signalé lorsque la fonction de demande de démarrage revient. L'objet événement est dans l'état signalé à chaque fois qu'une demande de fermeture se produit. Pour recevoir une notification de plusieurs événements de demande de fermeture, placez l'objet événement dans l'état signalé à chaque fois à l'aide de **SetEvent** ou de la fonction **Query Close Intercept** (42).
2. Après avoir utilisé cette fonction, votre programme d'application peut utiliser la fonction **Query Close Intercept** (42) pour déterminer quand une demande de fermeture s'est produite. L'application peut attendre l'objet événement renvoyé pour déterminer quand l'événement s'est produit.
3. Il ne s'agit pas d'un appel exclusif. Plusieurs applications peuvent demander cette fonction pour le même ID de session courte.
4. Si aucune application n'intercepte les demandes de fermeture pour une session, toutes les demandes de fermeture ultérieures sélectionnées par l'utilisateur dans la boîte de dialogue des opérations de l'émulateur entraînent un arrêt normal demandé pour cette session.

Start Communication Notification (80)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Start Communication Notification** démarre le processus par lequel votre application EHLLAPI peut déterminer si la session spécifiée est connectée à un hôte.

Après avoir utilisé cette fonction, l'application peut utiliser **Query Communication Event** (81) pour déterminer si la session est connectée ou déconnectée.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface améliorée
Numéro de fonction	Doit être 80
Chaîne de données	Structure préallouée ; voir le tableau suivant
Longueur	16
PSPosition	NA

La structure de données appelante contient ces éléments

Octet	**Définition**
1	Un nom court d'espace de présentation (PSID) à 1 caractère.
2-4	Réservé
5	Correspond à l'une des valeurs suivantes : • Le caractère C demande une notification lorsque la session se déconnecte ou se connecte à l'hôte. • Le caractère A demande le mode de notification asynchrone. Lorsque A est spécifié, les positions 9 à 12 renvoient l'adresse d'un objet événement (Windows). Le caractère C doit être placé en position 13. • Le caractère M demande le mode message asynchrone de la notification. Lorsque M est spécifié, le caractère de sélection d'événement C doit être placé en position 13.
6-8	Réservé
9-12	Lorsque M est spécifié en position 5, le descripteur de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL) — (non nulle).
13	Celui-ci doit contenir le caractère C si la position 5 est A ou M.

14-16	Réservé

Chaîne de données

Si A (mode asynchrone) est spécifié en position 5 de la structure de données appelante et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet	Définition
1	Un nom court d'espace de présentation (PSID) à 1 caractère
2-8	Réservé
9-12	Valeur binaire de 4 octets dans laquelle le descripteur de l'objet d'événement est renvoyé par EHLLAPI. L'application peut attendre cet objet événement.

Si M (mode de message asynchrone) est spécifié en position 5 de la structure de données appelante et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet	Définition
1	Un nom court d'espace de présentation (PSID) à 1 caractère
2-8	Réservé
9-10	ID de tâche du mode de message asynchrone

Lorsque la session se connecte ou se déconnecte, une fenêtre d'application reçoit un message. Le message est la valeur de retour du message RegisterWindow (PCSHLL). Le wParam contient l'ID de tâche renvoyé par l'appel de fonction. HIWORD de lParam contient un `21` si la session est connectée à l'hôte ou un 22 si la session est déconnectée. Le LOWORD de lParam contient le numéro de fonction `80`.

Paramètres de retour

Code retour	Définition
0	La fonction a réussi
1	Un PSID incorrect a été spécifié
2	Une erreur a été commise lors de la désignation des paramètres
9	Une erreur système a été rencontrée

Notes sur l'utilisation de cette fonction

1. Un programme d'application peut émettre cette fonction pour plusieurs sessions hôte. La fonction **Query Communication Event** (81) peut être utilisée pour déterminer l'état de communication de la session.
2. Si l'application choisit l'option asynchrone, elle peut utiliser l'appel du SDK Windows **WaitForSingleObject** pour attendre que l'état de communication des sessions change.
3. L'objet événement est initialement dans un état non signalé. Il est signalé à chaque fois qu'un événement se produit. Pour recevoir une notification pour plusieurs événements, l'application doit placer l'objet événement

dans l'état non signalé chaque fois qu'il est signalé, à l'aide de l'appel du SDK Windows **ResetEvent** ou à l'aide de la fonction 81 **Query Communications Event**.

4. Plusieurs appels à cette fonction avec les mêmes options depuis la même application seront ignorés.
5. Ceci n'est pas exclusif à une seule application. Plusieurs applications peuvent demander cette fonction pour le même ID de session.

Start Host Notification (23)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Start Host Notification** démarre le processus par lequel votre programme d'application EHLLAPIdétermine si l'espace de présentation hôte ou l'OIA ont été mis à jour.

Après avoir utilisé cette fonction, votre programme d'application peut utiliser la fonction **Query Host Update** (24) pour déterminer quand un événement hôte s'est produit.

Appels prérequis

Il n'y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 23	
Chaîne de données	Chaîne préallouée ; voir le tableau suivant	
Longueur	6 ou 7 implicite	16
Position PS	NA	

La chaîne de données appelante contient ces éléments :

Octet		**Définition**
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Un nom court d'espace de présentation (PSID) à 1 caractère • Un espace vide ou Null indiquant une demande d'espace de présentation hôte connecté à l'hôte
	2–4	Réservé.
2	5	Correspond à l'une des valeurs suivantes :

Octet		Définition
		• Le caractère B demandant une notification de l'espace de présentation hôte et des mises à jour OIA. • Le caractère O demandant une notification uniquement pour les mises à jour OIA. • Le caractère P demandant une notification uniquement pour les mises à jour de l'espace de présentation hôte. • Le caractère A demandant le mode asynchrone de la notification. Lorsque A est spécifié, la position 9 à 12 renvoie l'adresse d'un objet événement. Le caractère de sélection d'événement B, O ou P doit être placé en position 13. • Le caractère M demandant le mode message asynchrone de la notification. Lorsque M est spécifié, le caractère de sélection d'événement B, O ou P doit être placé en position 13 (7 pour 16 bits). • E Le caractère E demandant une notification de fin lors d'une session d'impression.
	6–8	Réservé.
3–4	9–12	Lorsque M est spécifié en position 5 (2 pour 16 bits), le descripteur de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL) (différente de 0).
7	13	L'une des valeurs suivantes si la position 5 (2 pour 16 bits) est A ou M : • Le caractère B demandant une notification à la fois de l'espace de présentation hôte et des mises à jour OIA • Le caractère O demandant uniquement la notification des mises à jour OIA • Le caractère P demandant uniquement la notification de la mise à jour de la présentation hôte.
	14–16	Réservé.

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Si A (mode de notification asynchrone) est spécifié en position 5 et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.

Octet		Définition
	2–8	Réservé.
	9–12	Valeur de 4 octets dans laquelle l'adresse de l'objet événement est renvoyée par EHLLAPI. L'application peut attendre cet objet événement (32 bits uniquement).

Chaîne de données :

Si M (mode de message asynchrone) est spécifié en position 5 (2 pour l'interface standard) et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–8	Réservé
3–4	9–10	ID de tâche du mode de message asynchrone

Note: Si l'OIA ou l'espace de présentation est mis à jour, une fenêtre d'application reçoit un message. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL). Le paramètre wParam contient l'ID de tâche renvoyé par l'appel de fonction. HIWORD de lParam contient le code retour 21 (indique que l'OIA est mis à jour), 22 (indique que l'espace de présentation hôte est mis à jour) ou 23 (indique que l'OIA et l'espace de présentation hôte sont mis à jour), et le paramètre LOWORD du paramètre lParam contient le numéro de fonction 23.

Code retour :

Les codes suivants sont définis :

Code retour	Définition
0	La fonction **Start Host Notification** a réussi.
1	Un espace de présentation hôte incorrect a été spécifié.
2	Une erreur a été commise lors de la désignation des paramètres.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. Un programme d'application peut émettre cette fonction pour plusieurs sessions hôte. La fonction **Pause** (18) peut avertir l'application lorsqu'une ou plusieurs sessions hôte (PS, OIA ou les deux) sont mises à jour. La fonction **Query Host Update** (24) peut être utilisée pour déterminer si un PS, un OIA ou les deux ont été mis à jour.
2. Si l'application choisit l'option asynchrone, elle peut attendre l'objet événement ou le sémaphore renvoyé pour déterminer quand un événement hôte s'est produit.

3. L'objet événement ou sémaphore est initialement dans un état non signalé et est signalé chaque fois qu'un événement approprié se produit. Pour recevoir une notification pour plusieurs événements, l'application doit placer l'objet événement dans l'état non signalé chaque fois qu'il a été signalé à l'aide de la fonction **ResetEvent** ou de la fonction **Query Host Update** (24).
4. Une application ne peut pas demander plusieurs fois la notification de démarrage de l'hôte avec les mêmes options.
5. Il ne s'agit pas d'un appel exclusif. Plusieurs applications peuvent demander cette fonction pour le même ID de session courte.

Start Keystroke Intercept (50)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Start Keystroke Intercept** permet à une application de poste de travail de filtrer toutes les frappes envoyées à une session par un opérateur de terminal. Après un appel à cette fonction, les frappes sont interceptées et enregistrées jusqu'à ce que la file d'attente des frappes déborde ou jusqu'à ce que la fonction **Stop Keystroke Intercept** (53) ou la fonction **Reset System** (21) soit appelée. Les frappes interceptées peuvent être :

- Reçu via la fonction **Get Key** (51) et envoyé à la même session ou à une autre session avec la fonction **Send Key** (3)
- Accepté ou rejeté via la fonction **Post Intercept Status** (52)
- Remplacé par d'autres frappes avec la fonction **Send Key** (3)
- Utilisé pour déclencher d'autres processus

Appels prérequis

Il n’y a aucun appel prérequis pour cette fonction.

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 50	
Chaîne de données	Voir le tableau suivant	
Longueur	Taille du tampon de frappe EHLLAPI alloue 32 octets minimum pour ce tampon.	
Position PS	NA	

La chaîne de données appelante contient :

Octet		Définition
Standard	Etendu	
1	1	Correspond à l'une des valeurs suivantes : • Un nom court d'espace de présentation hôte spécifique (PSID) • Un espace vide ou Null indiquant une demande d'espace de présentation hôte connecté à l'hôte
	2–4	Réservé.
2	5	Un caractère de **code d'option** : • D pour les frappes AID uniquement. • L pour toutes les frappes. • E pour les clés d'édition et toutes les frappes (disponible en mode Enhanced uniquement) • M pour demander le mode message asynchrone de la notification (Windows uniquement). Lorsque M est spécifié, un caractère de code D, L ou E (Mode Enhanced) doit être placé en position 13 (7 pour 16 bits). Condition préalable : les touches du clavier doivent être mappées pour éditer des fonctions, par exemple Ctrl+C mappées pour la la fonction d'édition de la copie. Voir Table 12: Mnémoniques avec @S (Shift), @W (Edit) et @ caractères alphabétiques on page 137 pour les fonctions d'édition prises en charge.
	6–8	Réservé.
3–4	9–12	Lorsque M est spécifié en position 5 (2 pour 16 bits), le descripteur de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL) (différente de 0).
7	13	L'une des valeurs suivantes si la position 5 (2 pour 16 bits) est M : • D pour les frappes AID uniquement. • L pour toutes les frappes. • E pour modifier les clés et toutes les frappes. (Disponible en mode Enhanced uniquement.)
	14–16	Réservé.

Chaîne de données :

Si M (mode de message asynchrone) est spécifié en position 5 (2 pour l'interface standard) et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
Standard	Etendu	

Octet		Définition
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–8	Réservé
3–4	9–10	ID de tâche du mode de message asynchrone

Note: Si un utilisateur envoie des frappes au clavier à une session, une fenêtre d'application reçoit un message. Le message est une valeur de retour de RegisterWindowMessge (PCSHLL). Le paramètre wParam contient l'ID de tâche renvoyé par l'appel de fonction. HIWORD du paramètre lParam contient le code retour 0, qui indique que la fonction a réussi, et LOWORD du paramètre lParam contient le numéro de fonction 50.

Paramètres de retour

Code retour	Explication
0	La fonction **Start Keystroke Intercept** a réussi.
1	Un espace de présentation incorrect a été spécifié.
2	Une option incorrecte a été spécifiée.
4	L'exécution de la fonction a été inhibée, car l'espace de présentation cible était occupé.
9	Une erreur système a été rencontrée. La fonction Release est utilisée.

Notes sur l'utilisation de cette fonction

1. Si un code retour de 31 apparaît pour la fonction **Get Key** (51), soit :
   - Augmentez la valeur du paramètre de longueur d'appel pour cette fonction, ou
   - Exécutez la fonction **Get Key** (51) plus fréquemment.

   Une frappe interceptée occupe 3 octets dans le tampon. La prochaine frappe interceptée est placée dans les 3 octets adjacents. Lorsque la fonction **Get Key** (51) récupère une frappe (première entrée, première sortie ou FIFO), les 3 octets qu'elle occupait sont rendus disponibles pour une autre frappe. En augmentant la taille du tampon ou la vitesse à laquelle les frappes sont récupérées du tampon, vous pouvez éliminer le dépassement de la mémoire tampon.

   Dans le PC/3270, une autre façon d'éliminer le code retour 31 est d'utiliser l'émulateur PC/3270 en mode reprise.
2. Si le code d'option D est fourni, EHLLAPI écrit les clés non-AID interceptées dans l'espace de présentation auquel elles étaient initialement destinées et renvoie uniquement les clés AID à l'application.
3. Appelez la fonction **Stop Keystroke Intercept** (53) avant de quitter votre application EHLLAPI. Sinon, l'interception des frappes reste activée avec des résultats imprévisibles.

Start Playing Macro (110)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Start Playing Macro** appelle une macro. La macro sera exécutée dans la session connectée.

Note: Cette macro doit exister dans le répertoire de données d'application de classe utilisateur Z and I Emulator for Windows et aucune extension ne doivent être spécifiés dans l'appel de fonction pour le nom de la macro.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard
Numéro de fonction	Doit être 110
Chaîne de données	Voir le tableau suivant
Longueur	Longueur du nom de la macro, plus 3
Position PS	NA

Octet		**Définition**
Standard	Etendu	
1-2		Réservé
3-n		Nom de macro terminé par un caractère Null

Paramètres de retour

Code retour	Explication
0	La fonction **Start Playing Macro** a réussi.
1	Le programme n'est pas connecté à une session hôte.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système a été rencontrée.

Stop Close Intercept (43)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Stop Close Intercept** permet à l'application de désactiver la fonction **Start Close Intercept** (41). Une fois que l'application a émis la fonction **Stop Close Intercept**, les demandes de fermeture ultérieures entraînent un arrêt normal envoyé à la session du terminal logique.

Appels prérequis

Start Close Intercept (41)

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 43	
Chaîne de données	ID de session court à 1 caractère de l'espace de présentation hôte	
Longueur	1	Doit être 4
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

Code retour	**Explication**
0	La fonction **Stop Close Intercept** a réussi.
1	Un espace de présentation hôte incorrect a été spécifié.
2	Une erreur a été commise lors de la spécification des paramètres.
8	Aucune fonction **Start Close Intercept** (41) précédente n'a été émise.
9	Une erreur système s'est produite.
12	La session s'est arrêtée.

Arrêter la notification de communication (82)

3270	***5250***	***VT***
Oui	Oui	Oui

La fonctionnalité **Arrêter la notification de communication** désactive la fonctionnalité **Événement de communication de requête** (81) permettant de déterminer si des événements de communication se sont produits dans la session spécifiée.

Appels prérequis

Start Communication Notification (80)

Paramètres d'appel

	Interface améliorée
Numéro de fonction	Doit être 82
Chaîne de données	Nom court à 1 caractère de l'espace de présentation hôte, ou un espace vide ou Null indiquant une demande de mise à jour de l'espace de présentation connecté à l'hôte.
Longueur	4 est implicite
PSPosition	NA

La structure de données appelante contient ces éléments :

Octet	**Définition**
1	Un nom court d'espace de présentation (PSID) à 1 caractère
2-4	Réservé

Paramètres de retour

Code retour	**Définition**
0	La fonction a réussi
1	Un PSID incorrect a été spécifié
8	Aucun appel préalable à la fonction **Start Communication Notification** (80) n'a été appelé pour le PSID
9	Une erreur système a été rencontrée

Stop Host Notification (25)

3270	***5250***	***VT***
Oui	Oui	Oui

La fonction **Stop Host Notification** désactive la capacité de la fonction **Query Host Update** (24) pour déterminer si l'espace de présentation hôte ou OIA a été mis à jour. Cette fonction empêche également les événements de l'hôte d'affecter la fonction **Pause** (18).

Appels prérequis

Start Host Notification (23)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 121	
Chaîne de données	Voir la note suivante	
Longueur	1 est implicite	Doit être 4
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Note: Nom court à 1 caractère de l'ID de l'espace de présentation cible, ou un espace ou une valeur Null pour indiquer une demande pour l'espace de présentation connecté à l'hôte.

Paramètres de retour

Code retour	Définition
0	La fonction **Stop Host Notification** a réussi.
1	Un espace de présentation hôte incorrect a été spécifié.
8	Aucune précédente fonction **Start Host Notification** (23) n'a été émise.
9	Une erreur système a été rencontrée.

Arrêter l'interception de frappe (53)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonctionnalité **Arrêter l'interception de frappe** met fin à la capacité de votre programme d'application à intercepter les frappes.

Appels prérequis

Start Keystroke Intercept (50)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 53	
Chaîne de données	Nom court de l'espace de présentation cible (PSID)	
Longueur	1 est implicite	Doit être 4
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé

Paramètres de retour

Code retour	Explication
0	La fonction **Stop Keystroke Intercept** a réussi.
1	Un espace de présentation incorrect a été spécifié.
8	Aucune fonction **Start Keystroke Intercept** (50) antérieure n'a été appelée pour cet espace de présentation.
9	Une erreur système a été rencontrée.

Wait (4)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Wait** vérifie l'état de l'espace de présentation connecté à l'hôte. Si la session attend une réponse de l'hôte (indiquée par XCLOCK (X []) ou XSYSTEM), la fonction **Wait** force EHLLAPI à attendre jusqu'à 1 minute pour voir si la condition disparaît.

Appels prérequis

Connect Presentation Space (1)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 4	

	Interface standard	Interface améliorée
Chaîne de données	NA	
Longueur	NA	
Position PS	NA	

Paramètres de retour

Code retour	Définition
0	Le clavier est déverrouillé et prêt pour la saisie.
1	Votre programme d'application n'est pas connecté à une session valide.
4	Expiration du délai d'attente alors que vous êtes encore dans XCLOCK (X []) ou XSYSTEM.
5	Le clavier est verrouillé.
9	Une erreur système a été rencontrée.

Notes sur l'utilisation de cette fonction

1. La fonction **Wait** est utilisée pour donner aux demandes d'hôte comme celles faites par la fonction **Send Key** (3) le temps nécessaire pour être complétées. A l'aide de la fonction **Set Session Parameters** (9), vous pouvez demander l'option `TWAIT`, `LWAIT` ou `NWAIT`. Voir l'élément 12 on page 147.
2. Vous pouvez utiliser cette fonction pour voir si l'OIA hôte est inhibé.
3. La fonction **Wait** est satisfaite par le déverrouillage du clavier par l'hôte. Par conséquent, un code retour de 0 ne signifie pas nécessairement que la transaction est terminée. Pour vérifier l'achèvement de la transaction, vous devez utiliser la fonction **Search Field** (30) ou la fonction **Search Presentation Space** (6) combinée à la fonction **Wait** pour rechercher les invites de mots clés attendues.

Window Status (104)

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **Window Status** permet à l'application d'interroger ou de modifier la taille, l'emplacement ou l'état visible de l'espace de présentation d'une fenêtre.

Appels prérequis

Connect Window Services (101)

Paramètres d'appel

	Interface standard	Interface améliorée
Numéro de fonction	Doit être 104	
Chaîne de données	Voir le tableau suivant	
Longueur	16 ou 20	24 ou 28
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		Définition
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère
	2–4	Réservé
2	5	Une valeur d'option de demande, sélectionnez l'une des valeurs suivantes : • X'01' pour le statut défini **Note:** Lorsque la session est incorporée sur place dans un document OLE composé, la forme définie de cette fonction (octet 5 = X'01') renvoie toujours 0 mais n'a aucun effet. • X'02' pour demander le statut • X'03' pour demander le statut étendu
	6	Réservé

Si la valeur de l'option de demande est X'01' (définir le statut) :

Octet		Définition
Standard	Etendu	
3–4	7–8	Un mot de 16 ou 32 bits contenant les bits de statut défini si l'option de demande est 1 (définir le statut). Les codes suivants sont des valeurs de retour valides si l'option de demande est définie sur le statut : **X'0001'** Changez la taille de la fenêtre. (Non valide avec réduire, agrandir, restaurer ou déplacer.) **X'0002'** Déplacez la fenêtre. (Non valide avec réduire, agrandir, dimensionner ou restaurer.)

Octet		Définition
		X'0004' Remplacement de fenêtre ZORDER. **X'0008'** Définissez la fenêtre sur visible. **X'0010'** Définissez la fenêtre sur invisible. **X'0080'** Activez la fenêtre. (Définit le focus sur la fenêtre et la place au premier plan sauf si ZORDER est spécifié. Dans ce cas, le placement ZORDER est utilisé.) **X'0100'** Désactivez la fenêtre. (Désactive la fenêtre et fait de la fenêtre la fenêtre du bas à moins que ZORDER ne soit également spécifié. Dans ce cas, le placement ZORDER est utilisé.) **X'0400'** Réglez la fenêtre sur réduite. (Non valide avec agrandir, restaurer, dimensionner ou déplacer.) **X'0800'** Réglez la fenêtre sur agrandie. (Non valide avec réduire, restaurer, dimensionner ou déplacer.) **X'1000'** Restaurez la fenêtre. (Non valide avec réduire, agrandir, dimensionner ou déplacer.)
5–6	9–12	Un mot de 16 ou 32 bits contenant la coordonnée de position de la fenêtre X. (Ignoré si l'option de déplacement n'est pas définie.)
7–8	13–16	Un mot de 16 ou 32 bits contenant la coordonnée de position de la fenêtre Y. (Ignoré si l'option de déplacement n'est pas définie.)
9–10	17–20	Un mot de 16 ou 32 bits contenant la taille de la fenêtre X en unités d'appareil. (Ignoré si l'option de taille n'est pas définie.)
11–12	21–24	Mot de 16 ou 32 bits contenant la taille de la fenêtre Y en unités d'appareil. (Ignoré si l'option de taille n'est pas définie.)
13–16	25–28	Un mot de 16 ou 32 bits contenant un descripteur de la fenêtre pour le placement relatif de la fenêtre. Ces deux mots concernent uniquement l'option définie. (Ignoré si l'option ZORDER n'est pas définie.) Les valeurs valides sont les suivantes :

Octet		Définition
		X'00000003' Placer devant toutes les fenêtres sœurs. X'00000004' Placer derrière toutes les fenêtres sœurs.

Si la valeur de l'option de demande est X'02' (interroger le statut) :

Octet		Définition
Standard	Etendu	
3–4	7–8	Un mot de 16 ou 32 bits contenant X'0000' si l'option de demande est 2 (interroger le statut). Les codes suivants sont des valeurs de retour possibles si l'option de demande est une demande de statut. Plus d'un état est possible. **X'0008'** La fenêtre est visible. **X'0010'** La fenêtre est invisible. **X'0080'** La fenêtre est activée. **X'0100'** La fenêtre est désactivée. **X'0400'** La fenêtre est réduite. **X'0800'** La fenêtre est agrandie.
5–6	9–12	Un mot de 16 ou 32 bits contenant la coordonnée de position de la fenêtre X. (Ignoré si l'option de déplacement n'est pas définie.)
7–8	13–16	Un mot de 16 ou 32 bits contenant la coordonnée de position de la fenêtre Y. (Ignoré si l'option de déplacement n'est pas définie.)
9–10	17–20	Un mot de 16 ou 32 bits contenant la taille de la fenêtre X en unités d'appareil. (Ignoré si l'option de taille n'est pas définie.)
11–12	21–24	Mot de 16 ou 32 bits contenant la taille de la fenêtre Y en unités d'appareil. (Ignoré si l'option de taille n'est pas définie.)
13–16	25–28	Un mot de 16 ou 32 bits contenant un descripteur de la fenêtre pour le placement relatif de la fenêtre. Ces deux mots concernent uniquement l'option définie. (Ignoré si l'option ZORDER n'est pas définie.) Les valeurs valides sont les suivantes : X'00000003' Placer devant toutes les fenêtres sœurs. X'00000004' Placer derrière toutes les fenêtres sœurs.

Si la valeur de l'option de demande est X'03' (interroger le statut étendu) :

Octet		**Définition**
Standard	Etendu	
3–4	7–8	Un mot de 16 ou 32 bits contenant X'0000' si l'option de demande est 3 (interroger le statut étendu). Les codes suivants sont des valeurs de retour possibles si l'option de demande est une requête pour l'état étendu. Plus d'un état est possible. **X'0008'** La fenêtre est visible. **X'0010'** La fenêtre est invisible. **X'0080'** La fenêtre est activée. **X'0100'** La fenêtre est désactivée. **X'0400'** La fenêtre est réduite. **X'0800'** La fenêtre est agrandie.
5–6	9–10	Un mot de 16 ou 32 bits contenant la taille de police actuelle dans la dimension X. La valeur est en pixels d'écran.
7–8	11–12	Un mot de 16 ou 32 bits contenant la taille de police actuelle dans la dimension Y. La valeur est en pixels d'écran.
9–12	13–16	Réservé. Cette valeur est toujours nulle.
13–14	17–18	Mot de 16 ou 32 bits contenant le numéro de ligne du premier caractère visible de l'espace de présentation. Cette valeur est généralement une, à moins que l'option de police de taille fixe soit activée et que la fenêtre ait été redimensionnée de telle sorte qu'une partie de l'espace de présentation soit masquée.
15–16	19–20	Un mot de 16 ou 32 bits contenant le numéro de colonne du premier caractère visible de l'espace de présentation.
17–20	21–24	Mot de 16 ou 32 bits contenant le descripteur de la fenêtre de l'espace de présentation de la session.

Paramètres de retour

Code retour	Explication
0	La fonction **Window Status** a réussi.
1	L'espace de présentation n'était pas valide ou n'était pas connecté.
2	Une option incorrecte a été spécifiée.
9	Une erreur système s'est produite.
12	La session s'est arrêtée.

Notes sur l'utilisation de cette fonction

Les fenêtres du terminal logique (LT) utilisent des cellules de caractères. Lors du redimensionnement des fenêtres LT, le LT arrondit le nombre pour éviter la troncature des cellules de caractères. La taille et la position demandées peuvent être légèrement différentes de celles demandées. Suivez l'option définie avec une option de demande pour déterminer la position et la taille finales de la fenêtre Presentation Manager®. Toutes les positions et tailles des coordonnées x et y sont en pixels.

Write Structured Fields (127)

3270	*5250*	*VT*
Oui	Non	Non

La fonction **Write Structured Field** permet à une application d'écrire des données de champs structurés dans l'application hôte. Si l'appel spécifie S (pour Synchrone), l'application ne reçoit pas le contrôle tant que la fonction **Write Structured Fields** n'est pas terminée. Si l'appel spécifie A (pour Asynchrone), l'application reçoit le contrôle immédiatement après l'appel. Si l'appel spécifie M, l'application reçoit le contrôle immédiatement après l'appel. L'application peut attendre le message. Dans tous les cas (S, A ou M), l'application fournit l'adresse du tampon dans laquelle les données destinées à l'hôte doivent être placées.

Pour une exécution asynchrone réussie de cette fonction, les instructions suivantes s'appliquent :

Le champ du code retour dans la liste des paramètres peut ne pas contenir les résultats des E/S demandées. Si le code retour n'est pas 0, la requête a échoué. L'application doit prendre l'action appropriée en fonction du code retour.

Si le code retour de cette demande est 0, l'application doit utiliser l'ID de demande renvoyé avec cet appel de fonction pour émettre l'appel de fonction **Get Request Completion** afin de déterminer les résultats d'achèvement de la fonction associée à l'ID de demande. L'appel de fonction **Get Request Completion** renvoie les informations suivantes :

1. ID de demande de fonction
2. Adresse de la chaîne de données de la requête asynchrone
3. Longueur de la chaîne de données
4. Code retour de la fonction terminée

Appels prérequis

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

Paramètres d'appel

	Interface standard	**Interface améliorée**
Numéro de fonction	Doit être 127	
Chaîne de données	Voir le tableau suivant	
Longueur	8, 10 ou 14	Doit être 20
Position PS	NA	

La chaîne de données appelante peut contenir :

Octet		**Définition**
Standard	Etendu	
1	1	Un nom court d'espace de présentation (PSID) à 1 caractère.
	2–4	Réservé.
2	5	S ou A ou M **S =** Synchrone. Le contrôle n'est pas renvoyé à l'application tant que la lecture n'est pas satisfaite. **A =** Asynchrone. Le contrôle est renvoyé immédiatement à l'application, peut attendre l'objet événement. **M =** Asynchrone. Le contrôle est renvoyé immédiatement à l'application, peut attendre le message.
	6	Réservé.
3–4	7–8	ID de destination/origine sur 2 octets.
5–8	9–12	Adresse sur 4 octets du tampon à partir duquel les données doivent être écrites. Le tampon doit être obtenu à l'aide de la fonction **Allocate Communications Buffer** (123).
9–10	13–16	Réservé.
11–12	17–20	Lorsque « M » est spécifié en position 5 (2 pour 16 bits), le descripteur de la fenêtre qui reçoit le message doit être défini. Le message est une valeur de retour de RegisterWindowMessage (« PCSHLL ») (différente de 0).
13–14		Les données dans ces positions sont ignorées par EHLLAPI Cependant, aucune erreur n'est générée si le programme de migration contient des

Octet		Définition
		données à ces positions. Ces données sont acceptées pour assurer la compatibilité avec les applications en migration.

Paramètres de retour

Cette fonction renvoie une chaîne de données et un code retour.

Chaîne de données :

Si A (asynchrone) est spécifié en position 5 (2 pour l'interface standard) et que la fonction s'exécute avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
9–10	13–14	ID de demande de fonction sur 2 octets. Il est utilisé par la fonction **Get Request Completion** (125) pour déterminer l'achèvement de cet appel de fonction.
	15–16	Réservé.
	17–20	Valeur de 4 octets dans laquelle l'adresse de l'objet événement est renvoyée par EHLLAPI. L'application peut attendre cet objet événement. Lorsque l'objet événement est effacé, l'application doit émettre l'appel de fonction **Get Request Completion** (125) pour obtenir les résultats de la demande **Write Structured Fields**. (32 bits uniquement).

Note: Un objet événement est renvoyé pour chaque demande asynchrone réussie. L'objet événement ne doit plus être utilisé. Un nouvel objet événement est renvoyé pour chaque demande et n'est valide que pour la durée de cette demande.

Chaîne de données :

Si M (mode de message asynchrone) est spécifié en position 5 (2 pour l'interface standard) et que la fonction est terminée avec succès, la chaîne de données suivante est renvoyée :

Octet		Définition
9–10	13–14	ID de demande de fonction sur 2 octets. Il est utilisé par la fonction **Get Request Completion** (125) pour déterminer l'achèvement de cet appel de fonction.
	15–16	Réservé.
11–12	17–18	ID de tâche du mode de message asynchrone.
	19–20	Réservé.

Note: Si la fonction est terminée avec succès, une fenêtre d'application reçoit un message. Le message est une valeur de retour de RegisterWindowMessage (PCSHLL). Le paramètre wParam contient l'ID de tâche

renvoyé par l'appel de fonction. HIWORD du paramètre lParam contient le code retour 0, qui indique que la fonction a réussi, et LOWORD du paramètre lParam contient le numéro de fonction 127.

Code retour :

Les codes suivants sont définis :

Code retour	Explication
0	La fonction **Write Structured Fields** a réussi.
1	Un ID de session court de l'espace de présentation hôte spécifié n'était pas valide ou n'était pas connecté.
2	Une erreur a été commise lors de la spécification des paramètres.
9	Une erreur système s'est produite.
11	Ressource non disponible (mémoire non disponible).
34	Le message envoyé à l'hôte a été annulé.
35	Une transmission sortante de l'hôte a été annulée.
36	Demande rejetée. Perte de contact avec l'hôte.
37	Echec. L'hôte est désactivé pour les communications entrantes.

Notes sur l'utilisation de cette fonction

1. Le code retour 35 sera renvoyé lorsque la fonction **Read Structured Fields** ou **Write Structured Fields** est demandée pour la première fois après l'annulation d'une transmission sortante de l'hôte. Les mesures correctives relèvent de la responsabilité de l'application.
2. Le code retour 36 nécessite que l'application se déconnecte du programme d'émulation, puis se reconnecte pour rétablir les communications avec l'hôte. Les mesures correctives relèvent de la responsabilité de l'application.
3. Le code retour 37 sera renvoyé si l'hôte est désactivé pour les appels entrants.
4. Le EHLLAPI permet qu'un maximum de 20 requêtes asynchrones par application soient en attente. Un code retour pour les ressources indisponibles (RC=11) est renvoyé si plus de 20 requêtes asynchrones sont tentées.

Le format des données de champ structuré est le suivant :

Décalage	Longueur	Contenus
0	1 mot	X'0000'
2	1 mot	m (longueur du message : le nombre d'octets de données dans le message, le nombre n'inclut pas le préfixe d'en-tête du tampon, qui contient 8 octets) Cette valeur doit être définie par l'application.
4	1 mot	X'0000'
6	1 mot	X'0000'
8	8 octets	Longueur du premier (ou du seul) message de champ structuré.

Décalage	Longueur	Contenus
10	1 octet	Premier octet hors longueur du message de champ structuré.
		⋮
m+7	1 octet	Dernier octet du message de champ structuré.

Les octets 0 à 7 sont l'en-tête du tampon. Ces 8 premiers octets sont utilisés par le programme d'émulation. La section utilisateur du tampon commence par le décalage 8. Les octets 8 et 9 contiennent le nombre d'octets dans le premier champ structuré (un message de champ structuré peut contenir plusieurs champs structurés) dont 2 octets pour les octets 8 et 9. Octets 8 à *m*+7 sont utilisés pour le message de champ structuré envoyé à l'hôte.

Demandes synchrones

Lorsque la fonction **Write Structured Fields** est demandée de manière synchrone (option S dans la chaîne de données), le contrôle est renvoyé à l'application uniquement une fois la demande satisfaite. L'application peut supposer que :

- Le code retour est correct.
- Les données dans le tampon de communication (tampon de lecture) sont correctes.
- L'hôte ne traite plus la demande **Write Structured Fields**.

Demandes asynchrones

Lorsque **Write Structured Fields** est demandée de manière asynchrone (option A dans la chaîne de données), l'application *ne peut pas* supposer que :

- Le code retour est correct.
- Les données dans le tampon de communication (tampon d'écriture) sont correctes.
- L'hôte ne traite plus la demande **Write Structured Fields**.

Lorsqu'il est demandé de manière asynchrone, EHLLAPI renvoie les valeurs suivantes :

- Un ID de demande de 16 bits aux positions 13 à 14 (9 à 10 pour l'interface standard) de la chaîne de données
- L'adresse d'un objet événement aux positions 17 à 20 de la chaîne de données.

Ceux-ci sont utilisés pour terminer l’appel asynchrone **Write Structured Fields**.

Les étapes suivantes doivent être effectuées pour déterminer le résultat d’un appel de fonction asynchrone **Write Structured Fields** :

- Si le code retour EHLLAPI n'est pas zéro, la demande a échoué. Aucune demande asynchrone n'a été effectuée. L'application doit prendre les mesures appropriées avant de tenter à nouveau l'appel.
- Si le code retour est zéro, l'application doit attendre que l'objet événement soit dans l'état signalé à l'aide de la fonction **Get Request Completion** (125). L'objet événement (fonction **Get Request Completion** (125)) ne

doit pas être réutilisé. L'objet événement est valide uniquement pendant la durée de l'appel de fonction **Write Structured Fields** jusqu'à la fin de l'appel de fonction **Get Request Completion** (125).
- Une fois que l'objet événement est dans l'état signalé, utilisez l'ID de demande de 16 bits renvoyé comme paramètre ID de demande dans un appel à la fonction **Get Request Completion** (125). La chaîne de données renvoyée par l'appel de fonction **Get Request Completion** (125) contient le code retour final de l'appel de fonction **Write Structured Fields**.

Demandes asynchrones

Lorsque la fonction **Write Structured Fields** est demandée de manière asynchrone (l'option M dans la chaîne de données), l'application ne peut pas supposer que :

- Le code de retour est correct
- Les données dans le tampon de communication (tampon d'écriture) sont correctes
- L'hôte ne traite plus la demande **Write Structured Fields**

Lorsqu'il est demandé de manière asynchrone avec l'option M, EHLLAPI renvoie les valeurs suivantes :

- Un ID de demande de 16 bits aux positions 13 à 14 (9 à 10 pour l'interface standard) de la chaîne de données
- ID de tâche du mode de message asynchrone aux positions 17 à 18 (11 à 12 pour l'interface standard)

Ceux-ci sont utilisés pour terminer l'appel asynchrone **Write Structured Fields**.

Chapter 4. Fonctions d'extension WinHLLAPI

Ce chapitre décrit les fonctions d'extension fournies lors de l'utilisation de la prise en charge de la programmation WinHLLAPI.

Résumé des fonctions WinHLLAPI

Les fonctions WinHLLAPI suivantes sont disponibles pour 3270, 5250 et VT :

Fonctions asynchrones WinHLLAPI

Les sections suivantes décrivent les fonctions asynchrones WinHLLAPI.

WinHLLAPIAsync

Ce point d'entrée est utilisé pour six fonctions WinHLLAPI dont l'exécution prend souvent beaucoup de temps. Avec WinHLLAPIAsync, la fonction sera lancée de manière asynchrone et n'interférera pas avec la progression continue de l'application appelante. Ces fonctions sont : **Wait** (04), **Start Host Notify** (23), **Start Close Intercept** (41), **Start Keystroke Intercept** (50), **Send File** (90) et **Receive File** (91), et sont décrites dans Fonctions d'extension WinHLLAPI on page 179.

HANDLE WinHLLAPIAsync (HWIND hWnd, LPWORD *lpnFunction*, LPBYTE *lpData*, LPWORD *lpnLength*, LPWORD *lpnRetC*)*

La liste des paramètres est la même que celle de WinHLLAPI, sauf qu'un descripteur de la fenêtre est requis avant le numéro de fonction. La fonction fonctionnant de manière asynchrone, son achèvement est signalé par un message enregistré. Le descripteur de la fenêtre est requis comme cible du message.

Deux messages doivent être enregistrés par l'application WinHLLAPI via des appels à **RegisterWindowsMessage()** avec les chaînes **WinHLLAPIAsync** (pour toutes les fonctions sauf 90 et 91) et **WinHLLAPIAsyncFileTransfer** (pour les fonctions 90 et 91). Le format standard est le suivant :

WPARAM

contient le descripteur de la tâche renvoyé par l'appel de fonction d'origine.

LPARAM

le mot haut contient le code d'erreur et le mot bas contient le numéro de fonction d'origine.

Wait (4)

Cette fonction détermine si la session hôte est dans un état inhibé. Si, pour une raison quelconque, la session est dans un état inhibé, cette fonction signalera à votre application avec un message l'expiration de l'état inhibé ou de votre période d'attente. Le temps d'attente est défini avec la fonction **Set Session Parameters** (9).

Fonctions prérequises

Connect Presentation Space (1)

WinHLLAPIAsync(*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	Description
Chaîne de données	NA
Longueur des données	NA
Position PS	NA

Codes retour

Code	Description
WHLLOK	Le PS est réactivé et prêt à être saisi.
WHLLNOTCONNECTED	Votre application WinHLLAPI n'est pas connectée à une session hôte valide.
WHLLPSBUSY	La fonction a expiré alors qu'elle était encore inhibée.
WHLLNHIBITED	Le PS est inhibé.
SHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLCANCEL	La fonction asynchrone a été annulée.

Remarques

L'attente asynchrone est utilisée pour notifier l'application appelante lorsque l'état inhibé du service PS a expiré. Lorsque l'état inhibé a expiré, cette version de **Wait** publiera un message **WinHLLAPIAsync** dans la fenêtre spécifiée par *hWnd*. Les options de session **TWAIT**, **LWAIT** et **NWAIT** affectent la durée d'attente de cette fonction. Consultez Set Session Parameters (9) on page 142 pour plus de détails sur ces options de session.

Note: Si **NWAIT** est spécifié dans les paramètres de session et que l'application s'enregistre à l'aide de la révision 1.1 de l'implémentation WinHLLAPI, l'appel **WINHLLAPIAsync** fonctionnera de la même manière que

l'appel **WinHLLAPI** et n'enverra pas de message. Si la révision 1.0 est utilisée, **Wait** renverra immédiatement un message avec l'état inhibé du PS.

Start Host Notification (23)

Cette fonction vous permet d'informer votre application WinHLLAPI des modifications apportées à l'espace de présentation de la session hôte (PS) ou à la zone d'informations sur les opérations (OIA).

Fonctions prérequises

Il n'y a aucune fonction prérequise pour cette fonction.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	Description
Chaîne de données	Une chaîne de 7 octets au format suivant : **Octet 1** Nom court de l'ID de session de la session hôte souhaitée, ou espace ou valeur Null pour la session hôte actuelle. **Octet 2** Mode notifications. « P » pour la mise à jour de l'espace de présentation uniquement, « O » pour la mise à jour OIA uniquement, « B » pour les mises à jour de l'espace de présentation et de l'OIA. Lors de l'appel de WinHLLAPIAsync, cette position peut être « A ». **Octet 3-6** Non utilisé. Fourni pour la compatibilité avec les anciennes applications. **Octet 7** Réservé ou remplacé par l'un des éléments suivants si vous utilisez WinHLLAPIAsync et A dans l'octet 2 : P pour la mise à jour de l'espace de présentation uniquement, O pour la mise à jour OIA uniquement ; et B pour l'espace de présentation et les mises à jour de l'OIA.
Longueur des données	Longueur du tampon d'événements hôte (256 recommandés).
Position PS	NA

Paramètres de retour

Paramètre	Description
Chaîne de données	Identique à *Data String* lors de l'appel.

Codes retour

Code	Description
WHLLOK	Notification de l'hôte activée.
WHLLNOTCONNECTED	La session hôte spécifiée n'est pas valide.
WHLLPARAMETERERROR	Un ou plusieurs paramètres ne sont pas valides.
WHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLCANCEL	La fonction asynchrone a été annulée.

Remarques

Une fois activée, la notification de l'hôte est activée jusqu'à ce que vous appeliez **Stop Host Notification** (25) ou **WinHLLAPICancelAsyncRequest()**. La fonction lance la notification de l'hôte et rend immédiatement le contrôle à votre application Windows HLLAPI. Cela libère votre application pour qu'elle puisse effectuer d'autres tâches en attendant les mises à jour de l'hôte. Lorsqu'une mise à jour se produit, la fonction notifiera la fenêtre spécifiée par *hWnd* avec le message enregistré **WinHLLAPIAsync**.

Start Close Intercept (41)

Cette fonction intercepte les demandes des utilisateurs pour fermer Z and I Emulator for Windows.

Fonctions prérequises

Il n'y a aucune fonction prérequise pour cette fonction.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	Description
Chaîne de données	Une chaîne de 5 octets pour l'adresse du sémaphore renvoyée. Le premier octet est le nom court de la session à interroger, ou un espace ou une valeur Null pour la session en cours.
Longueur des données	Doit être spécifié.
Position PS	NA

Paramètres de retour

Paramètre	Description
Chaîne de données	Une chaîne de 5 octets au format suivant : **Octet 1** Nom court de la session, ou espace ou valeur Null pour la session en cours **Octets 2 à 5** Adresse du sémaphore.

Code retour

Code	Description
WHLLOK	La fonction a réussi.
WHLLNOTCONNECTED	Un espace de présentation non valide a été spécifié.
WHLLPARAMETERERROR	Une option non valide a été spécifiée.
WHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLCANCEL	La fonction asynchrone a été annulée.

Remarques

Une fois activée, la notification de l'hôte reste activée jusqu'à ce que vous appeliez **Stop Close Intercept** (43)ou **WinHLLAPICancelAsyncRequest ()**. Initialement, le sémaphore est défini. Après avoir utilisé cette fonction, les demandes de fermeture de l'utilisateur sont rejetées et le sémaphore est effacé.

La fonction lance une interception de fermeture et rend immédiatement le contrôle à votre application Windows HLLAPI. Cela libère votre application pour qu'elle puisse effectuer d'autres tâches en attendant les demandes de fermeture. Lorsqu'une demande de fermeture se produit, la fonction notifiera la fenêtre spécifiée par *hWnd* avec le message enregistré **WinHLLAPIAsync**.

Start Keystroke Intercept (50)

Cette fonction intercepte les frappes envoyées à une session par l'utilisateur.

Fonctions prérequises

Il n'y a aucune fonction prérequise pour cette fonction.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	Description
Chaîne de données	Une chaîne de 6 octets au format suivant : **Octet 1** Nom court de la session, ou espace ou valeur Null pour la session hôte en cours. **Octet 2** Code d'interception de frappe. « D » provoque l'interception des frappes au clavier AID uniquement ; « L » provoque l'interception de toutes les frappes. **Octets 3 à 6** Réservé
Longueur des données	Variable (256 est recommandé)
Position PS	NA

Code retour

Code	Description
WHLLOK	L'interception de frappe a été lancée.
WHLLNOTCONNECTED	L'espace de présentation de la session hôte n'est pas valide.
WHLLPARAMETERERROR	Un ou plusieurs paramètres ne sont pas valides.
WHLLPSBUSY	La session est occupée.
WHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLCANCEL	La fonction asynchrone a été annulée.

Remarques

La fonction lance l'interception des frappes au clavier et rend immédiatement le contrôle à votre application Windows HLLAPI. Cela libère votre application pour qu'elle puisse effectuer d'autres tâches en attendant les frappes au clavier. Une fois lancée, la fonction publiera un message **WinHLLAPIAsync** dans la fenêtre spécifiée par *hWnd* chaque fois que l'utilisateur enverra une clé au PS. Après notification, les frappes interceptées peuvent être traitées de n'importe quelle manière autorisée par une application EHLLAPI normale. Notez que le tampon de frappe est de taille limitée, chaque frappe doit donc être gérée et supprimée du tampon.

Send File (90)

Cette fonction transfère un fichier du PC vers l'hôte.

Fonctions prérequises

Il n'y a aucune fonction prérequise pour cette fonction.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	Description
Chaîne de données	Paramètres de la commande SEND.
Longueur des données	Longueur de la *chaîne de données*. NA si l'option de session EOT est spécifiée.
Position PS	NA

Codes retour

Code	Description
WHLLOK	Le transfert de fichiers a démarré avec succès.
WHLLPARAMETERERROR	L'erreur de paramètre ou la valeur de *Data Length* est nulle ou supérieure à 255.
WHLLFTXCOMPLETE	Transfert de fichiers terminé.
WHLLFTXSEGMENTED	Le transfert est terminé avec des enregistrements segmentés.
WHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLTRANSABORTED	Le transfert de fichiers a été interrompu, soit parce que l'utilisateur a cliqué sur le bouton Cancel, soit parce que le délai d'attente est écoulé.
WHLLFILENOTFOUND	Fichier PC introuvable.
WHLLFTXCOMPLETECICS	Le transfert de fichiers a réussi (transfert vers CICS).
WHLLACCESSDENIED	Accès refusé au fichier PC.
WHLLMEMORY	Mémoire insuffisante.
WHLLINVALIDENVIRONMENT	Environnement invalide.

Remarques

Une seule opération de transfert de fichiers est prise en charge par session hôte connectée.

La fonction lance le transfert de fichiers et rend immédiatement le contrôle à votre application Windows HLLAPI. Cela libère votre application pour qu'elle puisse effectuer d'autres tâches pendant le transfert de fichiers. Une fois lancée, la fonction publiera régulièrement des messages **WinHLLAPIAsyncFileTransfer** dans la fenêtre spécifiée par *hWnd*.

Ces messages informeront l'application WinHLLAPI de l'état du transfert et enverront un message final une fois le transfert terminé.

wParm

Est l'indicateur d'état : l'octet de poids fort contient l'ID de session, l'octet de poids faible contient l'état. Si l'octet de poids faible est nul, le transfert de fichier est toujours en cours. Si l'octet de poids faible est un, le transfert de fichier est terminé.

lParm

Si l'octet de poids faible de *wParm* est nul (en cours), *lParm* est le nombre d'octets transférés. Si l'octet de poids faible *wParm* est un (complété), *lParm* est le code de fin d'exécution.

Receive File (91)

Cette fonction transfère un fichier du PC vers l'hôte.

Fonctions prérequises

Il n'y a aucune fonction prérequise pour cette fonction.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode* **)**

Paramètres d'appel

Paramètre	**Description**
Chaîne de données	Paramètres de la commande RECEIVE.
Longueur des données	Longueur de la *chaîne de données*. NA si l'option de session EOT est spécifiée.
Position PS	NA

Codes retour

Code	**Description**
WHLLOK	Le transfert de fichiers a démarré avec succès.
WHLLPARAMETERERROR	L'erreur de paramètre ou la valeur de *Data Length* est nulle ou supérieure à 255.
WHLLFTXCOMPLETE	Transfert de fichiers terminé.
WHLLFTXSEGMENTED	Le transfert est terminé avec des enregistrements segmentés.
WHLLSYSERROR	La fonction a échoué en raison d'une erreur système.
WHLLTRANSABORTED	Le transfert de fichiers a été interrompu, soit parce que l'utilisateur a cliqué sur le bouton Cancel, soit parce que le délai d'attente est écoulé.

Code	Description
WHLLFILENOTFOUND	Fichier PC introuvable.
WHLLFTXCOMPLETECICS	Le transfert de fichiers a réussi (transfert vers CICS).
WHLLACCESSDENIED	Accès refusé au fichier PC.
WHLLMEMORY	Mémoire insuffisante.
WHLLINVALIDENVIRONMENT	Environnement invalide.

Remarques

Une seule opération de transfert de fichiers est prise en charge par session hôte connectée.

La fonction lance le transfert de fichiers et rend immédiatement le contrôle à votre application Windows HLLAPI. Cela libère votre application pour qu'elle puisse effectuer d'autres tâches pendant le transfert de fichiers. Une fois lancée, la fonction publiera régulièrement des messages **WinHLLAPIAsyncFileTransfer** dans la fenêtre spécifiée par *hWnd*. Ces messages informeront l'application WinHLLAPI de l'état du transfert et enverront un message final une fois le transfert terminé.

wParm

Est l'indicateur d'état : l'octet de poids fort contient l'ID de session, l'octet de poids faible contient l'état. Si l'octet de poids faible est nul, le transfert de fichier est toujours en cours. Si l'octet de poids faible est un, le transfert de fichier est terminé.

lParm

Si l'octet de poids faible de *wParm* est nul (en cours), *lParm* est le nombre d'octets transférés. Si l'octet de poids faible *wParm* est un (complété), *lParm* est le code de fin d'exécution.

WinHLLAPICancelAsyncRequest

Cette fonction annule une fonction asynchrone exceptionnelle lancée par un appel à **WinHLLAPIAsync()**.

Syntaxe

int WinHLLAPICancelAsyncRequest (HANDLE *hAsyncTask*, WORD *wFunction***)**

Paramètres

hAsyncTask

Le descripteur renvoyé par WinHLLAPIAsync() lorsque la fonction a été lancée.

wFunction

Le numéro de fonction de la tâche asynchrone à annuler. Ce paramètre étant obligatoire pour la révision 1.1, mais pas pour la version 1.0, il est facultatif.

Avec cette fonction, toute tâche asynchrone précédemment initiée par un appel à WinHLLAPIAsync() peut être annulée alors qu'elle est encore en cours.

Renvoie

La valeur de retour indique si la fonction spécifiée a effectivement été annulée. Si la fonction a été annulée, la valeur de retour est WHLLOK (0). Si la fonction asynchrone en cours n'a pas été annulée, l'un des codes suivants sera renvoyé.

WHLLINVALID

hAsyncTask n'est pas un descripteur de tâche valide.

WHLLALREADY

La tâche asynchrone spécifiée par hAsyncTask est déjà terminée.

Fonctions d'initialisation et de terminaison

La section suivante décrit les fonctions d'initialisation et de terminaison de la prise en charge de la programmation WinHLLAPI.

Démarrage WinHLLAPI

Cette fonction est utilisée pour enregistrer l'application auprès de l'implémentation WinHLLAPI et doit être appelée avant tout autre appel à l'implémentation WinHLLAPI. Cette implémentation prend en charge les versions 1.0 et 1.1 de la spécification WinHLLAPI. L'application WinHLLAPI doit négocier la compatibilité des versions avec cette fonction.

Syntaxe

int WinHLLAPIStartup(WORD *wVersionRequired,* LPWHLLAPIDATA *lpData***)**

Paramètres

wVersionRequired

Il s'agit de la version requise par l'application WinHLLAPI. L'octet de poids faible contient le numéro de version majeure et l'octet de poids fort contient le numéro de version mineure (ou de révision).

lpData

Il s'agit d'un pointeur vers une structure WHLLAPIDATA qui recevra le numéro de version de l'implémentation et une chaîne décrivant le fournisseur d'implémentation WinHLLAPI. La structure WHLLAPIDATA est définie comme :

```
#define WHLLDESCRIPTION_LEN 127 typedef struct tagWHLLAPIDATA { WORD wVersion; Char
 szDescription[WHLLDESCRIPTION_LEN + 1]; }WHLLAPIDATA, * PWHLLAPIDATA, FAR *LPWHLLAPIDATA;
```

Renvoie

La valeur de retour indique la réussite ou l'échec de l'enregistrement de l'application WinHLLAPI auprès de l'implémentation. Si l'enregistrement a réussi, la valeur de retour est WHLLOK (zéro). Sinon, il s'agit de l'un des éléments suivants :

WHLLSYSNOTREADY

Indique que le sous-système réseau sous-jacent n'est pas disponible.

WHLLVERNOTSUPPORTED

Indique que la version demandée n'est pas fournie par cette implémentation. Cette implémentation prend en charge uniquement les versions 1.0 et 1.1.

Nettoyage WinHLLAPI

La spécification WinHLLAPI recommande que cette fonction soit utilisée par l'application WinHLLAPI pour se désinscrire de l'implémentation WinHLLAPI.

Syntaxe

BOOL WinHLLAPICleanup()

Renvoie

Renvoie TRUE si la désinscription a réussi. Sinon, renvoie FALSE.

Routines de blocage

Les sections suivantes décrivent les routines de blocage prises en charge par la programmation WinHLLAPI.

Note: Bien que les routines de blocage soient prises en charge pour la conformité WinHLLAPI, leur utilisation n'est pas recommandée. L'utilisation des fonctions WinHLLAPIAsync est la méthode recommandée pour le traitement asynchrone.

WinHLLAPIIsBlocking

Cette fonction indique à l'unité d'exécution d'application WinHLLAPI appelant si elle est en train d'exécuter un appel bloquant. Un appel bloquant est toute fonction synchrone qui prend beaucoup de temps à s'exécuter et qui ne revient qu'une fois terminée. Il existe cinq appels bloquants dans cette implémentation de WinHLLAPI. Les appels bloquants sont : **Get Key (51)**, **Wait (4)**, **Pause (18)**, **Send File (90)** et **Receive File (91)**.

Syntaxe

BOOL WinHLLAPIIsBlocking()

Renvoie

Si l'unité d'exécution d'application WinHLLAPI est au milieu d'un appel bloquant, la fonction renvoie TRUE, sinon elle renvoie FALSE.

Remarques

Etant donné que le crochet de blocage par défaut permet de traiter les messages pendant les appels bloquants, il est possible de rappeler l'appel bloquant.

WinHLLAPISetBlockingHook

Cette fonction définit une procédure définie par l'application à exécuter en attendant la fin d'un appel bloquant. Un appel bloquant est toute fonction synchrone qui prend beaucoup de temps à s'exécuter et qui ne revient qu'une fois terminée. Il existe cinq appels bloquants dans cette implémentation de WinHLLAPI. Les appels bloquants sont : **Get Key (51)**, **Wait (4)**, **Pause (18)**, **Send File (90)** et **Receive File (91)**.

Syntaxe

FARPROC WinHLLAPISetBlockingHook(FARPROC *lpfnBlockingHook***)**

Paramètres

lpfnBlockingHook

Ceci est un pointeur vers la nouvelle procédure de blocage.

Description

L'implémentation WinHLLAPI dispose d'une procédure de blocage par défaut qui ne consiste en rien d'autre qu'un gestionnaire de messages. Ce mécanisme par défaut est illustré dans l'exemple suivant :

```
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; }
```

Le crochet de blocage est implémenté pour chaque unité d'exécution. Un crochet de blocage défini par cette fonction restera en vigueur pour l'unité d'exécution jusqu'à ce qu'elle soit remplacée par un autre appel à **WinHLLAPISetBlockingHook()** ou jusqu'à ce que la valeur par défaut soit restaurée par un appel à **WinHLLAPIUnhookBlockingHook()**.

La fonction de blocage doit renvoyer **FALSE** si elle reçoit un message **WM_QUIT** afin que WinHLLAPI puisse rendre le contrôle à l'application pour traiter le message et se terminer correctement. Sinon, la fonction devrait renvoyer **TRUE**.

Renvoie

Cette fonction renvoie un pointeur vers la fonction de blocage en cours de remplacement.

WinHLLAPIUnhookBlockingHook

Cette fonction restaure le crochet de blocage par défaut pour l'unité d'exécution appelante.

Syntaxe

BOOL WinHLLAPIUnhookBlockingHook()

Renvoie

Cette fonction renvoie TRUE si le mécanisme de blocage par défaut a été restauré avec succès, sinon elle renvoie FALSE.

WinHLLAPICancelBlockingCall

Cette fonction annule un appel bloquant en cours d'exécution dans l'*unité d'exécution actuelle*. Un appel bloquant est toute fonction synchrone qui prend beaucoup de temps à s'exécuter et qui ne revient qu'une fois terminée. Il existe cinq appels bloquants dans cette implémentation de WinHLLAPI. Les appels bloquants sont : **Get Key** (51), **Wait** (4), **Pause** (18), **Send File** (90) et **Receive File** (91). Si l'un de ces appels bloquants est annulé, la fonction annulée renverra WHLLCANCEL.

Syntaxe

int WinHLLAPICancelBlockingCall()

Renvoie

La valeur de retour indique si la fonction spécifiée a effectivement été annulée. Si la fonction a été annulée, la valeur de retour est WHLLOK (0). S'il n'y a aucune fonction de blocage en attente, le code retour suivant sera renvoyé :

WHLLINVALID

Indique qu'aucun appel bloquant n'est en cours d'exécution.

Chapter 5. Fonctions PCSAPI

Z and I Emulator for Windows fournit un ensemble d'API, défini ici et appelé *PCSAPI*. Alors que EHLLAPI est utilisé pour gérer l'interaction entre un programme d'application de poste de travail et les systèmes hôtes une fois la session établie, le PCSAPI peut être utilisé pour contrôler la session Z and I Emulator for Windows elle-même.

Comment utiliser PCSAPI

Vous pouvez écrire des programmes d'application à l'aide de PCSAPI en C ou C++. Pour développer une application PCSAPI, procédez comme suit :

1. Préparez le code source et ajoutez les appels PCSAPI appropriés.
2. Incluez le fichier d'en-tête PCSAPI.H dans le programme d'application.
3. Compilez le code source.
4. Liez les fichiers .OBJ résultants avec le fichier objet ou les bibliothèques appropriées.

   Vous devez également le lier à la bibliothèque d'importation PCSAPI, PCSCALLS.LIB pour 16 bits et PCSCAL32.LIB pour 32 bits.

Conventions de mise en page

Tous les appels de fonction PCSAPI sont présentés dans le même format afin que vous puissiez récupérer rapidement les informations dont vous avez besoin. Son format est le suivant :

- Nom de fonction
  - Type de fonction
  - Type et description du paramètre
  - Code retour

Type de fonction

« Type de fonction » affiche le type de fonction dans le format suivant :

TYPE FunctionName(*TYPE Parameter1, ...* **)**

Type et description du paramètre

« Type et description du paramètre » répertorie le type et décrit chacun des paramètres à spécifier dans l'appel de fonction PCSAPI.

Code retour

« Code retour » liste les codes qui doivent être reçus par votre programme après un appel à la fonction PCSAPI.

pcsConnectSession

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsConnectSession** démarre les communications avec une session hôte spécifiée par l'ID de session court. La session doit déjà être démarrée. Cet appel est équivalent à l'élément de menu **Communications → Connect** sur le panneau de session de l'émulateur.

Type de fonction

BOOL WINAPI pcsConnectSession(*char cShortSessionID* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

Code retour

Code retour	**Signification**
TRUE	La fonction s'est terminée avec succès.
FALSE	Cela signifie l'une des choses suivantes : • La session n'a pas commencé. • Un ID de session incorrect a été spécifié. • Echec de l'appel.

pcsDisconnectSession

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsDisconnectSession** arrête la liaison de communication avec une session hôte spécifiée par l'ID de session court. Cela ne fait que déconnecter la liaison ; cela n'arrête pas la session. Cet appel est équivalent à l'élément de menu **Communications → Disconnect** sur le panneau de session de l'émulateur.

Type de fonction

BOOL WINAPI pcsDisconnectSession(*char cShortSessionID* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l’espace de présentation.

Code retour

Code retour	Signification
TRUE	La fonction s'est terminée avec succès.
FALSE	Cela signifie l'une des choses suivantes : • La session n'a pas commencé. • Un ID de session incorrect a été spécifié. • Echec de l'appel.

pcsQueryConnectionInfo

3270	*5250*	*VT*
Oui	Non	Non

La fonction **pcsQueryConnectionInfo** renvoie des informations sur la connexion Telnet de la session hôte spécifiée. Les informations résultantes sont renvoyées dans le tampon fourni par l'application.

Type de fonction

BOOL WINAPI pcsQueryConnectionInfo(*char cShortSessionID, CONNECTIONINFO *ConnectionInfo* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l’espace de présentation.

CONNECTIONINFO *ConnectionInfo

Pointeur vers une structure CONNECTIONINFO où les données d’informations de connexion seront renvoyées.

Code retour

Code retour	Signification
TRUE	La fonction s'est terminée avec succès.
FALSE	Cela signifie l'une des choses suivantes : • La session n'a pas commencé. • Un ID de session incorrect a été spécifié. • La session spécifiée n'était pas un type de connexion pris en charge pour cette API (pas Telnet).

ConnectionInfo

La structure CONNECTIONINFO sera remplie avec les informations sur la connexion hôte, composées des informations suivantes :

Structure	Information
Nom d'hôte	Indique le nom de l'hôte Telnet actuellement connecté.
Nom de la LU	Indique le nom de la LU actuellement attribuée.
Numéro de port	Indique le numéro de port hôte utilisé pour la connexion.
Indicateur SSL	Indique une connexion sécurisée (1 = sécurisée ; Ø = non sécurisée).

Note: Cette API est valide uniquement avec la version 32 bits de PCSAPI et ne fonctionne que pour les connexions Telnet.

Exemple

```
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*
Oui	Oui	Oui

La fonction **pcsQueryEmulatorStatus** renvoie l'état de la session hôte spécifié par l'ID de session court.

Type de fonction

ULONG WINAPI pcsQueryEmulatorStatus(*char cShortSessionID* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l’espace de présentation.

Code retour

La valeur du code retour doit être traitée de manière significative, c'est-à-dire par l'une des valeurs suivantes ou par une valeur OR parmi les valeurs suivantes :

Code retour	Valeur	Signification
PCS_SESSION_STARTED	0x00000001	La session spécifiée a démarré. Lorsque ce bit est désactivé, la session spécifiée n'a pas démarré ou un ID de session incorrect a été spécifié.
PCS_SESSION_ONLINE	0x00000002	La session spécifiée est en ligne (connectée). Lorsque ce bit est désactivé, la session spécifiée est hors ligne (déconnectée).
PCS_SESSION_API_ENABLED	0x00000004	L'API (EHLLAPI) est activée sur la session spécifiée. Si ce bit est désactivé, l'API est désactivée sur cette session.

pcsQuerySessionList

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsQuerySessionList** renvoie une liste de toutes les sessions hôte en cours. L'application doit fournir un tableau de structures SESSINFO tel que défini dans le fichier PCSAPI.H, ainsi qu'un décompte du nombre d'éléments dans le tableau. Cette fonction remplit les structures avec des informations sur chaque session et renvoie le nombre de sessions trouvées.

Si le tableau contient moins d'éléments qu'il n'y a de sessions hôtes, alors seuls les éléments fournis du tableau sont renseignés. La fonction renvoie toujours le nombre réel de sessions, même si le tableau est trop petit.

Une application peut appeler cette fonction avec zéro élément de tableau pour déterminer le nombre de sessions existantes. Un deuxième appel peut alors être effectué pour obtenir les informations de session.

Type de fonction

ULONG WINAPI pcsQuerySessionList(*ULONG Count, SESSINFO *SessionList* **)**

Type et description du paramètre

Compte ULONG

Nombre d'éléments dans le tableau SessionList.

SESSINFO *SessionList

Pointeur vers un tableau de structures SESSINFO telles que définies dans PCSAPI.H.

Paramètres de retour

Code retour

Nombre total des sessions Z and I Emulator for Windows. Ce nombre peut être supérieur ou inférieur au paramètre Count.

SessionList

La matrice des structures SESSINFO est rempli d'informations sur les sessions hôtes. Les sessions peuvent être placées dans la liste dans n'importe quel ordre. Chaque structure SESSINFO contient les champs suivants (définis dans PCSAPI32.H)

Nom

Une combinaison de char et ULONG qui contient l'ID de session (A-Z). Dans l'implémentation actuelle de Z and I Emulator for Windows, seul l'octet de poids faible (char) est utilisé, les autres octets sont renvoyés à zéro.

Statut

Une combinaison d'indicateurs binaires qui indiquent l'état actuel de la session. Les indicateurs (PCS_SESSION_*) sont définis dans le tableau suivant.

La valeur d'état doit être traitée de manière significative, c'est-à-dire par l'une des valeurs suivantes ou par une valeur OR parmi les valeurs suivantes :

Code retour	Signification
PCS_SESSION_STARTED	La session est en cours. Si cet indicateur n'est pas défini, tous les autres ne sont pas définis.
PCS_SESSION_ONLINE	La session a établi un lien de communication avec l'hôte (c'est-à-dire que la session est connectée).
PCS_SESSION_API_ENABLED	La session est activée pour la programmation des API. Si cet indicateur n’est pas défini, les API EHLLAPI et Bib-

Code retour	Signification
	liothèque de classes Host Access ne peuvent pas être utilisées sur cette session.

Exemple

```
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*
Oui	Oui	Oui

La fonction **pcsQueryWorkstationProfile** renvoie le nom du profil de poste de travail qui a été utilisé pour appeler la session hôte. Pour spécifier la session hôte, l'ID de session court doit être utilisé. Le nom du profil du poste de travail est copié dans le tampon de travail fourni par l'application.

Type de fonction

BOOL WINAPI pcsQueryWorkstationProfile(*char cShortSessionID, PSZ lpBuffer* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l’espace de présentation.

PSZ lpBuffer

Tampon de travail pour copier un nom de profil de poste de travail terminé par un caractère Null. Le tampon doit être suffisamment grand pour contenir un nom de fichier complet.

Code retour

Code retour	Signification
TRUE	La fonction s'est terminée avec succès.
FALSE	Cela signifie l'une des choses suivantes : • La session n'a pas commencé. • Un ID de session incorrect a été spécifié.

pcsSetLinkTimeout

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsSetLinkTimeout** définit le délai d'inactivité d'une liaison Telnet appartenant à SSCP. Cette fonction n'a aucun effet sur les connexions non-TN ou sur les connexions qui ne sont pas dans un état appartenant à SSCP. Si la valeur du délai d'attente est définie sur zéro, la liaison n'expirera pas. Sinon, la liaison expirera (se déconnectera) après avoir été inactive dans l'état SSCP pendant le nombre de minutes spécifié.

Prototype de fonction

ULONG WINAPI pcsSetLinkTimeout(*char cShortSessionID, USHORT Timeout* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l’espace de présentation.

Délai d'attente USHORT

Valeur du délai d'attente en minutes. Une valeur de zéro désactive le délai d'attente.

Code retour

Code retour	Signification
PCS_SUCCESSFUL	La fonction s'est terminée avec succès.
PCS_SYSTEM_ERROR	Une erreur système s'est produite.

pcsStartSession

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsStartSession** démarre une session hôte en utilisant un profil de poste de travail spécifié. Un identifiant de session court peut également être spécifié.

Type de fonction

ULONG WINAPI pcsStartSession(*PSZ lpProfile, char cShortSessionID, USHORT fuCmdShow* **)**

Type et description du paramètre

PSZ lpProfile

Chemin et nom de fichier complet du profil à charger. Le chemin est facultatif mais le nom de fichier complet doit être spécifié (l'extension .ws n'est pas supposée).

char cShortSessionID

ID de session court de l’espace de présentation. L'espace ou la valeur NULL indique le prochain ID de session disponible.

USHORT fuCmdShow

Spécifie comment la fenêtre doit être affichée. L'une des valeurs suivantes de PCSAPI.H :

- PCS_HIDE
- PCS_SHOW
- PCS_MINIMIZE
- PCS_MAXIMIZE

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_USED_ID	2	L'ID de session court spécifié est déjà utilisé.
PCS_INVALID_PROFILE	3	Une erreur s'est produite lors de la spécification du profil du poste de travail ou le paramètre de fenêtre n'était pas valide.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

pcsStopSession

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsStopSession** arrête une session hôte spécifiée par l'ID de session court.

Type de fonction

BOOL WINAPI pcsStopSession(*char cShortSessionID, USHORT fuSaveProfile* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

USHORT fuSaveProfile

Ce paramètre peut prendre l'une des valeurs suivantes :

fuSaveProfile	Valeur	Signification
PCS_SAVE_AS_PROFILE	0	Enregistrez le profil comme spécifié dans le profil actuel.
PCS_SAVE_ON_EXIT	1	Enregistrez le profil à la sortie.
PCS_NOSAVE_ON_EXIT	2	Ne sauvegardez pas le profil à la sortie.

Code retour

Code retour	Signification
TRUE	La fonction s'est terminée avec succès.
FALSE	Cela signifie l'une des choses suivantes : • La session n'a pas commencé. • Un ID de session incorrect a été spécifié.

Fonctions Page Setup

Les fonctions PCSAPI répertoriées dans cette section vous permettent de contrôler et de récupérer les paramètres **Page Setup** de la session d'émulateur Z and I Emulator for Windows.

Restrictions

Si les restrictions suivantes ne sont pas respectées, l'API échouera. Le code retour indique la raison de l'échec.

- La session hôte spécifiée dans l'argument `cShortSessionID` ne doit pas être en mode PDT.
- La session hôte ne doit pas imprimer lorsque l'API est invoquée.
- La boîte de dialogue **File → Page Setup** ne doit pas être utilisée.

Certains membres de la structure PAGEINFO peuvent être valides ou pris en charge uniquement pour des types de session spécifiques. Si aucune restriction n'est spécifiée, ce membre est valide ou pris en charge pour les types de session suivants :

- Affichage 3270
- 3270 Imprimante

- Affichage 5250
- ASCII VT

Les sessions d'imprimante 5250 ne sont pas prises en charge.

pcsGetPageSettings

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsGetPageSettings** récupère les valeurs des paramètres de page de la session hôte (similaires aux paramètres de la boîte de dialogue **File → Page Setup**). Seuls les paramètres de l'onglet **Text** de la boîte de dialogue sont pris en charge.

Type de fonction

ULONG WINAPI pcsGetPageSettings(*char cShortSessionID, PAGEINFO * const pPageInfo, ULONG * const pErrorInfo* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

PAGEINFO * const pPageInfo

Pointeur vers la structure PAGEINFO, où les paramètres de page sont renvoyés.

nFlags

Combinaison d'indicateurs binaires indiquant quels membres de la structure sont valides. Ces indicateurs peuvent être utilisés indépendamment ou en les associant ensemble pour restaurer la page de propriétés (définie dans PCSAPI32.H). Les indicateurs, ainsi que les membres valides correspondants dans la structure, sont les suivants :

Marquer

Membres valides dans la structure

PCS_PAGE_CPI

nCPI

PCS_PAGE_LPI

nLPI

PCS_PAGE_FACE_NAME

szFaceName

PCS_PAGE_MPL

nMPL

PCS_PAGE_MPP

nMPP

nCPI

Le nombre de caractères imprimés par pouce.

LOWORD est la valeur réelle de la CPI.

Si la fonction Font CPI (CPI de police) est configurée dans la session, HIWORD est 1. Si la fonction Font CPI (CPI de police) n'est pas configurée, HIWORD est 0.

nLPI

Le nombre de lignes imprimées par pouce.

LOWORD est la valeur LPI réelle.

Si la fonction Font LPI (LPI de police) est configurée dans la session, HIWORD est 1. Si la fonction Font LPI (LPI de police) n'est pas configurée, HIWORD est 0.

szFaceName

Nom de la police de l'imprimante. Il doit s'agir d'une chaîne terminée par un caractère Null.

nMPL

Nombre maximum de lignes pouvant être imprimées par page.

Ceci est également appelé MPL (Maximum Print Lines). La plage prise en charge est de 1 à 255.

nMPP

Nombre maximum de caractères pouvant être imprimés par ligne.

Ceci est également appelé MPP (Maximum Print Position). La plage prise en charge est de 1 à 255.

ULONG * const pErrorInfo

Non utilisé. Celui-ci doit être défini sur NULL par l'appelant.

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_INVALID_SESS_TYPE	2	Non pris en charge pour le type de session hôte.
PCS_DIALOG_IN_USE	3	Echec, car la boîte de dialogue Page Setup ou Printer Setup de la session hôte était en cours d'utilisation.

Code retour	Valeur	Signification
PCS_PRINTING	4	Les paramètres de page ne peuvent pas être obtenus, car la session hôte était en cours d'impression.
PCS_PDT_MODE	5	Les paramètres de page ne peuvent pas être obtenus, car la session hôte est en mode PDT.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

Exemple

```
{ 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*
Oui	Oui	Oui

La fonction **pcsRestorePageDefaults** restaure les valeurs système par défaut des pages de propriétés de mise en page définies dans le champ nFlags. Cela équivaut à cliquer sur **Default** dans les pages de propriétés de la boîte de dialogue **File → Page Setup**. Seuls les paramètres de l'onglet **Text** sont pris en charge.

Type de fonction

ULONG WINAPI pcsRestorePageDefaults(*char cShortSessionID, ULONG nFlags* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

ULONG nFlags

L'indicateur suivant décrit le nom de la page de propriétés de la boîte de dialogue **Page Setup** spécifiée. Cet indicateur peut être effectué avec un ORed au niveau du bit pour restaurer la page de propriétés (définie dans PCSAPI32.H).

PCS_PAGE_TEXT

Cet indicateur décrit la page de propriétés Text. Il s'agit de la seule page de propriétés actuellement prise en charge.

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_INVALID_SESS_TYPE	2	Le paramètre nFlags possède une ou plusieurs options qui ne sont pas valides pour le type de session hôte. Aucun paramètre n'a été restauré.
PCS_DIALOG_IN_USE	3	Echec, car la boîte de dialogue Page Setup ou Printer Setup de la session hôte était en cours d'utilisation.
PCS_PRINTING	4	Les paramètres de page ne peuvent pas être modifiés, car la session hôte était en cours d'impression.
PCS_PDT_MODE	5	Les paramètres de page ne peuvent pas être modifiés, car la session hôte est en mode PDT.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

Exemple

```
{ ULONG Rc = 0; Rc = pcsRestorePageDefaults('A', PCS_PAGE_TEXT); if (Rc != PCS_SUCCESSFUL)
 printf("Failure. Return code = %d\n", Rc); }
```

pcsSetPageSettings

3270	*5250*	*VT*
Oui	Oui	Oui

La fonction **pcsSetPageSettings** définit les paramètres de la page de la session hôte. Ceci est similaire à la configuration des paramètres de la boîte de dialogue **File → Page Setup**. Seuls les paramètres de l'onglet **Text** sont pris en charge.

Note:

1. CPI, LPI et FontSize dépendent du FaceName configuré dans la session hôte. Si cette API est utilisée pour définir ensemble CPI, LPI, FontSize et FaceName, FaceName est défini en premier, puis les propriétés dépendantes.
2. Si cette API est utilisée pour définir FaceName et les propriétés dépendantes dans des appels distincts, définissez d'abord FaceName, puis définissez CPI, LPI et FontSize. Sinon, chaque fois que FaceName est défini, interrogez CPI, LPI et FontSize et assurez-vous qu'ils ont les valeurs souhaitées.
3. Si CPI, LPI ou FontSize sont définis avant FaceName, différentes valeurs pour CPI, LPI ou FontSize peuvent être configurées dans la session hôte. Cela peut se produire si les valeurs CPI, LPI ou FontSize actuelles ne sont pas valides pour le nouvel ensemble FaceName.

Type de fonction

ULONG WINAPI pcsSetPageSettings(*char cShortSessionID, const PAGEINFO * const pPageInfo, ULONG * const pErrorInfo* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

const PAGEINFO * const pPageInfo

Pointeur vers la structure PAGEINFO, où les paramètres de la page sont mentionnés.

nFlags

Combinaison d'indicateurs binaires indiquant quels membres de la structure sont valides. Ces indicateurs peuvent être utilisés indépendamment ou en les associant ensemble pour restaurer la page de propriétés (définie dans PCSAPI32.H). Les indicateurs, ainsi que les membres valides correspondants dans la structure, sont les suivants :

Marquer

Membres valides dans la structure

PCS_PAGE_CPI

nCPI

PCS_PAGE_LPI

nLPI

PCS_PAGE_FACE_NAME

szFaceName

PCS_PAGE_MPL

nMPL

PCS_PAGE_MPP

nMPP

nCPI

Le nombre de caractères imprimés par pouce.

Pour sélectionner Font CPI (CPI de police), définissez le HIWORD de nCPI sur 1. LOWORD de nCPI sera ignoré.

Pour sélectionner une valeur CPI particulière, procédez comme suit :

1. Définissez le HIWORD de nCPI sur 0.
2. Définissez le LOWORD de nCPI sur la valeur CPI réelle.

nLPI

Le nombre de lignes imprimées par pouce.

Pour sélectionner Font LPI (LPI de police), définissez le HIWORD de nLPI sur 1. LOWORD de nLPI sera ignoré

Pour sélectionner une valeur LPI particulière, procédez comme suit :

1. Définissez le HIWORD de nLPI sur 0.
2. Réglez le LOWORD de nLPI sur la valeur LPI réelle.

szFaceName

Nom de la police de l'imprimante. Il doit s'agir d'une chaîne terminée par un caractère Null.

nMPL

Nombre maximum de lignes pouvant être imprimées par page.

Ceci est également appelé MPL (Maximum Print Lines). La plage prise en charge est de 1 à 255.

nMPP

Nombre maximum de caractères pouvant être imprimés par ligne.

Ceci est également appelé MPP (Maximum Print Position). La plage prise en charge est de 1 à 255.

ULONG * const pErrorInfo

Contient les informations d'erreur étendues lorsque l'API échoue avec le code retour PCS_FAILURE. Si les informations détaillées sur l'erreur ne sont pas nécessaires, cet indicateur doit être défini sur NULL par l'appelant.

Il s'agit d'une combinaison d'indicateurs binaires qui décrivent quels membres de la structure PAGEINFO n'ont pas pu être définis avec succès. Les indicateurs définis dans PCSAPI32.H sont les suivants :

Marquer

Membres valides dans la structure

PCS_PAGE_CPI

Seul le nCPI n'est pas valide.

PCS_PAGE_LPI

Seul nLPI n'est pas valide.

PCS_PAGE_FACE_NAME

Seul szFaceName n'est pas valide.

PCS_PAGE_MPL

Seul nMPL n'est pas valide.

PCS_PAGE_MPP

Seul nMPP n'est pas valide.

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_INVALID_SESS_TYPE	2	Non pris en charge pour le type de session hôte.
PCS_DIALOG_IN_USE	3	Echec, car la boîte de dialogue Page Setup ou Printer Setup de la session hôte était en cours d'utilisation.
PCS_PRINTING	4	Les paramètres de page ne peuvent pas être modifiés, car la session hôte était en cours d'impression.
PCS_PDT_MODE	5	Les paramètres de page ne peuvent pas être modifiés, car la session hôte est en mode PDT.
PCS_FAILURE	6	Les paramètres de la page de session hôte ne sont pas entièrement appliqués. Cela peut être dû au fait que des données non valides ont été fournies pour certains ou tous les champs de la structure PAGEINFO. Examinez pErrorInfo pour plus de détails sur les paramètres qui ne sont pas appliqués.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

Exemple

```
{ 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); }
```

Fonctions Printer Setup

Les fonctions PCSAPI répertoriées dans cette section vous permettent de contrôler et de récupérer les paramètres **Printer Setup** de la session d'émulateur Z and I Emulator for Windows.

Restrictions

Si les restrictions suivantes ne sont pas respectées, l'API échouera. Le code retour indique la raison de l'échec.

- La session hôte ne doit pas imprimer lorsque l'API est invoquée.
- La boîte de dialogue **File → Printer Setup** ne doit pas être utilisée.

pcsGetPrinterSettings

3270	***5250***	***VT***
Oui	Oui	Oui

La fonction **pcsGetPrinterSettings** récupère les paramètres d'imprimante de la session hôte (similaires aux paramètres de la boîte de dialogue **File → Printer Setup**).

Type de fonction

ULONG WINAPI pcsGetPrinterSettings(*char cShortSessionID, PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

PRINTINFO * const pPrintInfo

Pointeur vers la structure PRINTINFO, où les paramètres de l'imprimante sont spécifiés.

nFlags

Doit être défini sur 0. Ceci est ignoré.

nBufSize

Taille du buffer alloué pour les champs suivants :

- lpPDTFile
- lpPrtToDskAppFile

- lpPrtToDskSepFile
- lpPrinterName

Si plusieurs de ces membres sont récupérés dans un seul appel d'API, l'appelant doit alors allouer la même taille à tous les tampons et transmettre cette taille dans ce membre.

Si ce membre est défini sur 0, les champs sont ignorés. La taille maximale requise pour les buffers des champs est renvoyée dans nSizeNeeded.

nSizeNeeded

La valeur de ce membre est déterminée par des conditions liées aux champs suivants :

- lpPDTFile
- lpPrtToDskAppFile
- lpPrtToDskSepFile
- lpPrinterName

Les conditions sont les suivantes :

- La valeur est le nombre d'octets nécessaires, si la taille du tampon alloué par l'appelant n'est pas suffisamment grande pour renvoyer les champs répertoriés ci-dessus.
- La valeur correspond à la taille maximale du tampon requis, si plusieurs des champs répertoriés ci-dessus sont obtenus par l'appelant.
- Si nBufSize est défini sur 0 par l'appelant, ce membre contient la taille maximale requise pour les tampons des champs répertoriés ci-dessus.

bPromptDialog

Les valeurs possibles sont les suivantes :

- Si TRUE, la boîte de dialogue Printer Setup s'affiche avant l'impression.
- Si FALSE, la boîte de dialogue Printer Setup ne s'affiche pas avant l'impression.

bPDTMode

Les valeurs possibles sont les suivantes :

- Si TRUE, la session hôte est en mode PDT.
- Si FALSE, la session hôte est en mode non-PDT (mode GDI).

lpPDTFile

Doit être défini sur NULL si l’appelant n’est pas intéressé à obtenir ce membre. Le fichier PDT est renvoyé s'il ne s'agit pas d'un pointeur nul. Celui-ci doit pointer vers le tampon de taille nBufSize alloué par l'appelant.
Lorsque l'API est renvoyée, ce membre contient l'un des éléments suivants :

- Nom de chemin complet du fichier PDT de session.
- Une chaîne vide (« ») si aucun fichier PDT n'est configuré dans la session.
- Un nom de fichier tronqué si la taille du tampon n'est pas suffisante. Le membre nSizeNeeded contient la taille du tampon nécessaire.

nPrtMode

Il s'agit d'une valeur énumérée qui indique le PrintMode de la connexion. Le type de données enum PRINTMODE est défini dans PCSAPI32.H. Le paramètre nPrtMode doit être l'un des suivants :

- **PrtToDskAppend** (mode **Print to Disk-Append**)

  Cela équivaut à sélectionner l'option **Append** dans la boîte de dialogue **Printer Setup → Printer → Print to Disk**de la session hôte.
- **PrtToDskSeparate** (mode **Print to Disk-Separate**)

  Cela équivaut à sélectionner l'option **Separate** dans la boîte de dialogue **Printer Setup → Printer → Print to Disk** de la session hôte.
- **WinDefaultPrinter** (mode **Windows Default Printer**)

  Cela équivaut à sélectionner l'option **Use Windows Default Printer** dans la boîte de dialogue **Printer Setup** de la session hôte.
- **SpecificPrinter** (mode **Specific Printer**)

  Cela équivaut à sélectionner une imprimante dans la boîte de dialogue **Printer Setup** de la session hôte, tout en laissant la case **Use Windows Default Printer** décochée.

lpPrtToDskAppFile

Doit être défini sur NULL si l'appelant n'est pas intéressé à obtenir ce membre. Le fichier **Print to Disk-Append** est renvoyé s'il ne s'agit pas d'un pointeur Null. Celui-ci doit pointer vers le tampon de taille nBufSize alloué par l'appelant.
Lorsque l'API est renvoyée, ce membre contient l'un des éléments suivants :

- Le nom de chemin complet du fichier **Print to Disk-Append** de la session.
- Une chaîne vide (« ») si aucun fichier **Print to Disk-Append** n'est configuré pour la session.
- Un nom de fichier tronqué si la taille du tampon n'est pas suffisante. Le membre nSizeNeeded contient la taille du tampon nécessaire.

lpPrtToDskSepFile

Doit être défini sur NULL si l'appelant n'est pas intéressé à obtenir ce membre. Le fichier **Print to Disk-Separate** est renvoyé s'il ne s'agit pas d'un pointeur nul. Celui-ci doit pointer vers le tampon de taille nBufSize alloué par l'appelant.
Lorsque l'API est renvoyée, ce membre contient l'un des éléments suivants :

- Nom de chemin complet du fichier **Print to Disk-Separate** de la session.
- Une chaîne vide (« ») si aucun fichier **Print to Disk-Separate** n'est configuré pour la session.
- Un nom de fichier tronqué si la taille du tampon n'est pas suffisante. Le membre nSizeNeeded contient la taille du tampon nécessaire.

lpPrinterName

Doit être défini sur NULL si l'appelant n'est pas intéressé à obtenir ce membre. Le nom de l'imprimante est renvoyé s'il ne s'agit pas d'un pointeur Null. Celui-ci doit pointer vers le tampon de taille nBufSize alloué par l'appelant.
Lorsque l'API est renvoyée, ce membre possède l'un des éléments suivants :

- Le nom de l'imprimante spécifique configurée dans la session, si la session hôte nPrtMode est SpecificPrinter.
- Le nom de l'imprimante par défaut Windows configurée dans la session, si la session hôte nPrtMode est WinDefaultPrinter.
- Une chaîne vide (« »), si la session hôte nPrtMode est PrtToDskAppend ou PrtToDskSeparate.
- Un nom d'imprimante tronqué, si la taille du tampon n'est pas suffisante. nSizeNeeded a la taille du tampon nécessaire.

PrinterName doit avoir le format suivant :

```
<Nom de l'imprimante> sur <Nom du port>
```

Par exemple :

- `IBM InfoPrint 40 PS sur port réseau`
- `HP LaserJet série 4050 PCL 6 sur LPT1`

ULONG * const pErrorInfo

Ceci est rempli avec les informations d'erreur étendues lorsque l'API échoue avec le code retour PCS_FAILURE. pErrorInfo doit être défini sur NULL par l'appelant, si les détails des erreurs ne sont pas nécessaires.

La section suivante décrit les indicateurs définis dans PCSAPI32.H.

Indicateurs pour le membre pErrorInfo de la structure PRINTINFO

PCS_PRINT_PRINTMODE_ERROR

PrintMode n'est pas configuré dans la session hôte.

PCS_PRINT_PDTFILE_SIZEERR

La taille du tampon n'est pas suffisante pour lpPDTFile, le nom du fichier est donc tronqué. Le membre nSizeNeeded contient la taille réelle du tampon requis pour renvoyer le fichier PDT.

PCS_PRINT_DSKAPPFILE_SIZEERR

La taille du tampon n'est pas suffisante pour lpPrtToDskAppFile, le nom du fichier est donc tronqué. Le membre nSizeNeeded contient la taille réelle du tampon requis pour renvoyer le fichier **Print to Disk-Append**.

PCS_PRINT_DSKSEPFILE_SIZEERR

La taille du tampon n'est pas suffisante pour lpPrtToDskSepFile, le nom du fichier est donc tronqué. Le membre nSizeNeeded contient la taille réelle de la mémoire tampon requise pour renvoyer le fichier **Print to Disk-Separate**.

PCS_PRINT_PRINTERNAME_SIZEERR

La taille du tampon n'est pas suffisante pour lpPrinterName, le nom de l'imprimante est donc tronqué. Le membre nSizeNeeded contient la taille réelle du tampon requis pour renvoyer le nom de l'imprimante.

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_DIALOG_IN_USE	3	Echec, car la boîte de dialogue Page Setup ou Printer Set-up de la session hôte était en cours d'utilisation.
PCS_PRINTING	4	Les paramètres de l'imprimante n'ont pas pu être modifiés, car la session hôte était en cours d'impression. L'application doit réessayer plus tard
PCS_FAILURE	6	Certains paramètres de l'imprimante n'ont pas pu être récupérés avec succès. pErrorInfo contient des informations détaillées sur les erreurs sur lesquelles les paramètres n'ont pas pu être récupérés.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

Exemple

```
{ 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*
Oui	Oui	Oui

La fonction **pcsSetPrinterSettings** contrôle les paramètres d'imprimante de la session hôte (similaires aux paramètres de la boîte de dialogue **File → Printer Setup**).

Type de fonction

ULONG WINAPI pcsSetPrinterSettings(*char cShortSessionID, const PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo* **)**

Type et description du paramètre

char cShortSessionID

ID de session court de l'espace de présentation.

const PRINTINFO * const pPrintInfo

Pointeur vers la structure PRINTINFO, où les paramètres de l'imprimante sont mentionnés.

nFlags

Combinaison d'indicateurs binaires indiquant quels membres de la structure sont valides. Ces indicateurs peuvent être utilisés indépendamment ou en les associant ensemble pour restaurer la page de propriétés (définie dans PCSAPI32.H). Les indicateurs, ainsi que les membres valides correspondants dans la structure, sont les suivants :

Marquer

Membres valides dans la structure

PCS_PRINT_PDT

bPDTMode, lpPDTFile

PCS_PRINT_PRINTMODE

nPrtMode, lpPrtToDskAppFile, lpPrtToDskSepFile, lpPrinterName

PCS_PRINT_PROMPT_DIALOG

bPromptDialog

nBufSize

Doit être défini sur 0. Ceci est ignoré.

nSizeNeeded

Doit être défini sur 0. Ceci est ignoré.

bPromptDialog

Les valeurs possibles sont les suivantes :

- Si TRUE, la boîte de dialogue Printer Setup s'affiche avant l'impression.
- Si FALSE, la boîte de dialogue Printer Setup ne s'affiche pas avant l'impression.

bPDTMode

Les valeurs possibles sont les suivantes :

- Si TRUE, la connexion est définie en mode PDT.
- Si FALSE, la connexion est définie en mode non-PDT (mode GDI).

lpPDTFile

Utilisé uniquement si bPDTMode est défini sur TRUE. Ceci est ignoré si bPDTMode est défini sur FALSE.
Il s'agit d'une chaîne terminée par un caractère Null contenant le nom du fichier PDT et doit être l'un des éléments suivants :

- NULL

  Le fichier PDT actuellement configuré dans la connexion est utilisé. Si aucun fichier PDT n'est déjà configuré dans la connexion, l'API échoue avec une exception.
- Nom du fichier, sans le chemin

  lpPDTFile dans le sous-dossier PDFPDT du chemin d'installation de Z and I Emulator for Windows est utilisé.
- Nom de chemin complet du fichier

  Si lpPDTFile n'existe pas, l'API échoue.

nPrtMode

Il s'agit d'une valeur énumérée qui indique le PrintMode de la connexion. Le type de données enum PRINTMODE est défini dans PCSAPI32.H. Le paramètre nPrtMode doit être l’un des suivants :

- **PrtToDskAppend** (mode **Print to Disk-Append**)

  Cela équivaut à sélectionner l'option **Append** dans la boîte de dialogue **Printer Setup → Printer → Print to Disk**de la session hôte.

- **PrtToDskSeparate** (mode **Print to Disk-Separate**)

  Cela équivaut à sélectionner l'option **Separate** dans la boîte de dialogue **Printer Setup → Printer → Print to Disk** de la session hôte.
- **WinDefaultPrinter** (mode **Windows Default Printer**)

  Cela équivaut à sélectionner l'option **Use Windows Default Printer** dans la boîte de dialogue **Printer Setup** de la session hôte.
- **SpecificPrinter** (mode **Specific Printer**)

  Cela équivaut à sélectionner une imprimante dans la boîte de dialogue **Printer Setup** de la session hôte, tout en laissant l'option **Use Windows Default Printer** décochée.

lpPrtToDskAppFile

Ceci est utilisé uniquement si nPrtMode est défini sur PrtToDskAppend.
Il s'agit d'une chaîne terminée par un caractère Null contenant le nom du fichier **Print to Disk-Append** et doit être l'un des éléments suivants :

- NULL

  Le fichier actuellement configuré pour le mode PrtToDskAppend dans la connexion est utilisé. Si aucun fichier PDT n'est déjà configuré dans la connexion, l'API échouera.
- Nom du fichier, sans le chemin

  Le chemin du répertoire des données d'application de classe utilisateur est utilisé pour localiser le fichier. Si le fichier existe, il est utilisé. Sinon, il sera créé une fois l’impression terminée.
- Nom de chemin complet du fichier

  Le répertoire doit exister dans le chemin, sinon l'API échouera. Il n'est pas nécessaire que le fichier existe dans le chemin.

lpPrtToDskSepFile

Les valeurs possibles sont les suivantes :

- Nom de chemin complet du fichier **Print to Disk-Separate** pour la session.
- Une chaîne vide (« ») si aucun fichier **Print to Disk-Separate** n'est configuré pour la session.
- Un nom de fichier tronqué si la taille du tampon n'est pas suffisante. Le membre nSizeNeeded contient la taille du tampon nécessaire.

lpPrinterName

Ceci est utilisé uniquement si nPrtMode est défini sur SpecificPrinter. Autrement, il est ignoré. Il s'agit d'une chaîne terminée par un caractère Null contenant le nom de l'imprimante. Si l'imprimante n'existe pas, ce membre échoue.
PrinterName doit avoir le format suivant :

```
<Nom de l'imprimante> sur <Nom du port>
```

Par exemple :

- `IBM InfoPrint 40 PS sur port réseau`
- `HP LaserJet série 4050 PCL 6 sur LPT1`

ULONG * const pErrorInfo

Ceci est rempli avec les informations d'erreur étendues lorsque l'API échoue avec le code retour PCS_FAILURE. pErrorInfo doit être défini sur NULL par l'appelant, si les détails des erreurs ne sont pas nécessaires.

La section suivante décrit les indicateurs définis dans PCSAPI32.H.

Indicateurs pour le membre pErrorInfo de la structure PRINTINFO

PCS_PRINT_PDTMODE_ERROR

Cela peut se produire pour l'une des raisons suivantes :

- bPDTMode est défini sur TRUE, lpPDTFile est défini sur NULL et aucun fichier PDT n'est déjà configuré pour la session hôte.
- nPrtMode est défini sur PrtToDskAppend ou PrtToDskSeparate, PCS_PRINT_PDT n'est pas défini dans nFlags et la session hôte n'est pas déjà en mode PDT.
- nPrtMode est défini sur PrtToDskAppend ou PrtToDskSeparate et bPDTMode est défini sur FALSE.

PCS_PRINT_PDTFILE_ERROR

Le fichier ou le chemin spécifié dans lpPDTFile est introuvable.

PCS_PRINT_PRTTODSK_FILE_ERROR

Cela peut se produire pour l'une des raisons suivantes :

- Le dossier spécifié dans le champ lpPrtToDskAppFile ou lpPrtToDskSepFile n'existe pas ou n'a pas d'accès en écriture.
- Une extension est spécifiée dans le champ lpPrtToDskSepFile.

PCS_PRINT_PRINTMODE_ERROR

nPrtMode ne peut pas être défini avec succès. Cela peut se produire pour l'une des raisons suivantes :

- La valeur de nPrtMode ne fait pas partie des constantes énumérées du type de données enum PRINTMODE.
- nPrtMode est défini sur PrtToDskAppend, lpPrtToDskAppFile est défini sur NULL et aucun fichier **Print to Disk-Append** n'est déjà configuré dans la session hôte.
- nPrtMode est défini sur PrtToDskSeparate, lpPrtToDskSepFile est défini sur NULL et aucun fichier **Print to Disk-Separate** n'est déjà configuré dans la session hôte.
- nPrtMode est défini sur SpecificPrinter et l'imprimante indiquée dans le champ lpPrinterName est introuvable.
- nPrtMode est défini sur WinDefaultPrinter et aucune imprimante Windows® par défaut n'est configurée dans le système.
- bPDTMode est défini sur FALSE et PCS_PRINT_PRINTMODE n'est pas défini dans nFlags, mais la session hôte PrintMode est PrtToDskAppend ou PrtToDskSeparate.

Code retour

Code retour	Valeur	Signification
PCS_SUCCESSFUL	0	La fonction s'est terminée avec succès.
PCS_INVALID_ID	1	Un ID de session incorrect a été spécifié.
PCS_DIALOG_IN_USE	3	Echec, car la boîte de dialogue Page Setup ou Printer Setup de la session hôte était en cours d'utilisation.
PCS_PRINTING	4	Les paramètres de l'imprimante n'ont pas pu être modifiés, car la session hôte était en cours d'impression. L'application doit réessayer plus tard.
PCS_FAILURE	6	Aucun paramètre d'imprimante de session hôte n'a été appliqué. Cela peut se produire parce que des données non valides ont été fournies pour certains ou tous les champs de la structure PRINTINFO. pErrorInfo contient des détails sur les erreurs.
PCS_SYSTEM_ERROR	9	Une erreur système s'est produite.

Exemple

```
{ 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); }
```

Chapter 6. Dépannage pour la programmation de l'émulateur

Vous pouvez utiliser les ressources et outils d'informations d'auto-assistance suivants pour vous aider à résoudre les problèmes :

- consulter les informations relatives à l'édition afin de prendre connaissance des incidents connus, des solutions palliatives et des informations d'identification et de résolution des problèmes pour votre produit ;
- vérifier si un téléchargement ou un correctif est disponible afin de résoudre votre problème ;
- consulter les bases de connaissances disponibles pour voir si la résolution de votre problème y est déjà documentée.
- Si vous avez encore besoin d'aide, contactez le service de support logiciel HCL et signalez votre problème.

Entrée EHLLAPI partielle sur l'écran hôte Z and I Emulator for Windows

Incident

Un texte de commande tronqué a été envoyé à un hôte lors de l'utilisation de HCLZ and I Emulator for Windows.

Cause

Si une application EHLLAPI envoie une clé SYSREQ à l'hôte puis tente de saisir une commande sur l'écran de l'hôte, parfois seule une partie tronquée de la commande est envoyée à l'hôte. Ce problème se produit en raison d'un manque de synchronisation entre le traitement SYSREQ au niveau du côté hôte de Z and I Emulator for Windows et la saisie des commandes de l'application EHLLAPI.

Lorsque l'application envoie une commande SYSREQ à l'hôte, les situations suivantes se produisent :

- L'OIA est mis à jour pour indiquer que vous êtes dans une session SSCP-LU.
- La session Z and I Emulator for Windows envoie la commande AO (SYSREQ) à l'hôte 3270.

Dès que l'hôte reçoit la commande SYSREQ, il répond à Z and I Emulator for Windows avec le code 0x15 ou NL (NewLine). Quand Z and I Emulator for Windows traite cette commande NL en remplissant le reste de la ligne avec des valeurs NULL et en déplaçant le curseur au début de la ligne suivante.

Un problème survient lorsque l'application EHLLAPI continue de saisir diverses commandes dans l'écran hôte (via la fonction SendKeys), avant même que la session Z and I Emulator for Windows a reçu la commande NL de l'hôte et l'a traitée. En conséquence, une partie de la commande d'entrée est d'abord saisie à l'écran, tandis que la commande NL est traitée et que le curseur est déplacé vers la ligne suivante. Ensuite, la partie restante de la commande est saisie sur la ligne suivante. Ainsi, seule la deuxième partie tronquée de la commande est envoyée à l'hôte, provoquant des résultats erronés.

Résolution

La solution à ce problème consiste à forcer l'application EHLLAPI à attendre que la commande NL soit reçue et traitée, avant de continuer à saisir les commandes sur l'écran hôte. Une fois que la session a notifié l'application EHLLAPI que la réponse de l'hôte pour SYSREQ a été traitée, l'application EHLLAPI peut alors continuer sa saisie (car la session est désormais dans le bon état pour accepter une nouvelle saisie). Pour ce faire, utilisez les appels de fonction EHLLAPI suivants :

```
Start_Host_Notification (23) Pause (18) Set_Session_Parameters (9) Query_Host_Update (24).
```

Le code possible dans l'application EHLLAPI est le suivant :

- Appel Sendkeys(@A@H). Cela envoie la commande SYSREQ à la session.
- Appel StartHostNotify avec l'entrée B, où B indique la notification d'OIA et de PS. Cela indique à la session de notifier l'application EHLLAPI lorsque l'OIA et/ou le PS de la session sont mis à jour par l'hôte.
- Appel Pause, en spécifiant un délai d'attente suffisant. Cela amène l'application EHLLAPI à attendre que la session la notifie d'une mise à jour de l'hôte vers l'OIA et/ou le PS de la session. Cela se produit lorsque la session reçoit la réponse de l'hôte la plus attendue pour la commande SYSREQ. Notez que si la valeur du délai d'attente a été dépassée et qu'aucune notification de l'hôte n'a été reçue, l'appel de la fonction Pause est toujours renvoyé.

De plus, pour que cet appel Pause fonctionne, vous devez utiliser l'appel de fonction Set_Session_Parameters (9) pour activer l'option IPAUSE. Ceci est nécessaire, car il indique à l'appel d'API Pause de revenir lorsque l'hôte notifie la session d'une mise à jour OIA et/ou PS.

Si Pause est renvoyée en raison d'une mise à jour OIA/PS (notification de l'hôte), sa valeur de retour est de 26. Si tel est le cas, vous êtes prêt à envoyer la commande de l'hôte. Sinon, vous devez attendre à nouveau la réponse de l'hôte.

L'application EHLLAPI peut poursuivre la commande une fois qu'elle sait que l'OIA ou l'espace de présentation (ou les deux) ont été mis à jour par l'hôte. QueryHostUpdate est utilisé pour vérifier ce qui a été mis à jour : c'est-à-dire si l'OIA seul a été mis à jour (code retour 21), ou si le PS seul a été mis à jour (code retour 22) ou si l'OIA et le PS ont été mis à jour (code retour 23).

Par exemple, le code EHLLAPI peut ressembler à la partie suivante :

```
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
 */
```

Il s'agit de la solution la plus appropriée à ce problème, car l'application EHLLAPI attend le temps minimum exact requis pour permettre à la session de recevoir et de traiter la réponse de l'hôte SYSREQ, avant d'envoyer sa commande.

Une autre solution consiste à ajouter un délai [par exemple, Sleep(1000)] dans l'application EHLLAPI entre la commande SYSREQ et la commande suivante, afin que la session dispose de suffisamment de temps pour recevoir et traiter la réponse de l'hôte. Cependant, cette solution n'est pas la meilleure, car le délai peut être trop faible ou excessif.

Reportez-vous à la RFC 2355 (améliorations TN3270) pour plus d'informations sur la fonctionnalité 3270 SYSREQ.

HCL Z and I Emulator for Windows L'exemple VBHLLAPI ne s'exécute pas dans le FDCC de Windows Vista

Incident

L'exemple HCLZ and I Emulator for Windows VBHLLAPI utilise les contrôles fournis par comdlg32.ocx, qui n'est pas installé dans la configuration fédérale de base du bureau (FDCC) de Microsoft Windows Vista.

Cause

VBHLLAPI utilise les contrôles ActiveX et Common Dialog fournis par le module Microsoft comdlg32.ocx. Pour des raisons de sécurité, le FDCC de Windows Vista ne contient pas ce module particulier.

Résolution

La version FDCC de Windows Vista est personnalisée et les modifications ne sont pas recommandées.

Si des exemples HLLAPI contenant VBHLLAPI doivent être exécutés, le module comdlg32.ocx doit être copié depuis une machine Windows Vista standard vers le répertoire `\Windows\System32\` de l'installation FDCC de Windows Vista.

Redémarrez ensuite le système pour que la modification prenne effet.

Appendix A. Structures de données de réponse à la requête prises en charge par EHLLAPI

Cette annexe répertorie et définit les structures de réponse à la requêtes prises en charge par l'interface de champ structuré EHLLAPI pour PC/3270. Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* ou, dans le cas d'un programme sous licence IBM, à la documentation du programme sous licence spécifique.

Note:

1. EHLLAPI doit analyser les tampons de réponse à la requête pour localiser le paramètre auto-définissant (SDP) de l'ID de destination/origine (DOID) pour que la prise en charge des champs structurés fonctionne et soit fiable. Le champ DOID est ensuite rempli avec l'ID attribué.
2. L'application doit créer les structures de données de réponse à la requête dans la mémoire privée de l'application.
3. Seule une vérification superficielle est effectuée sur les données de réponse à la requête. Seule la validité de l'ID et de la longueur de la structure est vérifiée.
4. L'ordre des octets du champ de longueur de 2 octets au début de chaque réponse à la requête **n'est pas inversé**.
5. Une seule connexion de type de base de gestion des données distribuées (DDM) est autorisée par session hôte. Si la connexion DDM prend en charge le SDP pour le DOID, plusieurs connexions sont autorisées.
6. Si un code retour différent de zéro est reçu indiquant qu'une application est déjà connectée à la session sélectionnée (RC 32 ou 39), utilisez cet espace de présentation avec prudence. Des conflits avec le transfert de fichier et d'autres applications EHLLAPI peuvent en résulter.

La réponse à la requête DDM

Plusieurs formats de réponse aux requêtes DDM sont pris en charge. En voici quelques uns :

Table 15. Format de base de réponse à la requête DDM

Décalage	Longueur	Contenu	Signification
0	1 mot	Longueur	Longueur de la structure
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'95'	Type de réponse à la requête
4–5	2 octets	INDICATEURS	Réservé
6–7	2 octets	LIMIN	Nombre maximal d'octets DDM autorisés dans les transmissions entrantes

Table 15. Format de base de réponse à la requête DDM (continued)

Décalage	Longueur	Contenu	Signification
8–9	2 octets	LIMOUT	Nombre maximal d'octets DDM autorisés dans les transmissions sortantes
10	1 octet	NSS	Identifiant du nombre de sous-ensembles
11	1 octet	DDMSS	Identifiant de sous-ensemble DDM

Paramètre autodéfinissant le nom de l'application DDM

Le paramètre autodéfini du nom de l'application DDM fournit à l'application hôte le nom de l'application contenant le contrôle de l'appareil auxiliaire DDM. L'application de contrôle est identifiée par le DOID dans le paramètre autodéfinissant l'accès direct.

Ce paramètre autodéfini est facultatif, mais il est nécessaire si une application hôte doit identifier un appareil auxiliaire DDM distinct lorsque plusieurs applications existent sur un poste de travail distant.

Table 16. Paramètre autodéfinissant le nom de l'application DDM

Décalage	Longueur	Contenu	Signification
0	1 octet	Longueur	Longueur du paramètre
1	1 octet	X'02'	Nom de l'application DDM
2–n	n-2 octets	NAME	Nom du programme d'application distant

NAME

Le nom comprend 8 caractères ou moins et constitue le moyen par lequel une application hôte peut se rapporter à une application sur un poste de travail distant. Il est de la responsabilité des utilisateurs de l'application hôte et distante de s'assurer que le nom est compris par l'application à chaque extrémité.

Le protocole PCLK contrôle les paramètres autodéfinissant

Le paramètre autodéfinissant des contrôles de protocole PCLK indique que le champ structuré des contrôles de protocole PCLK, ID = X'1013', peut être utilisé à la fois pour les flux de données entrants et sortants destinés à ou depuis le processeur de l'appareil auxiliaire DDM.

Table 17. Paramètre autodéfinissant de l'appareil auxiliaire DDM PCLK

Décalage	Longueur	Contenu	Signification
0	1 octet	X'04'	Longueur du paramètre
1	1 octet	X'03'	Contrôles du protocole PCLK
2–3	2 octets	VERS	Version du protocole

VERS

La valeur donnée dans VERS est utilisée pour indiquer les versions de PCLK installées dans le terminal au moment où la réponse à la requête est renvoyée. Par exemple, X'0001' indique PCLK version 1.1.

Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour connaître les définitions de champ pour cette réponse à la requête.

Formats de réponse aux requêtes DDM de base

Les formats de réponse aux requêtes suivants sont des *exemples* de certaines des combinaisons Base + SDP (paramètre autodéfinissant) possibles. Les combinaisons ne sont pas toutes affichées.

Table 18. Format de réponse à la requête DDM de base avec nom et paramètres autodéfinissant à accès direct

Décalage	Longueur	Contenu	Signification
0	1 mot	Longueur	Longueur de la structure (inclut les paramètres autodéfinissant)
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'95'	Type de réponse à la requête
4–5	2 octets	INDICATEURS	Réservé
6–7	2 octets	LIMIN	Nombre maximal d'octets DDM autorisés dans les transmissions entrantes
8–9	2 octets	LIMOUT	Nombre maximal d'octets DDM autorisés dans les transmissions sortantes
10	1 octet	NSS	Nombre de sous-ensembles pris en charge
11	1 octet	DDMSS	Identifiant de sous-ensemble DDM
12	1 octet	Longueur (n+2)	Longueur du paramètre
13	1 octet	X'02'	Nom de l'application DDM
14– (13+n)	n octets	Nom	Nom du programme d'application distant
14+n	1 octet	X'04'	Longueur du paramètre
15+n	1 octet	X'01'	Identifiant d'accès direct
16+n – 17+n	2 octets	DOID	ID de destination/origine attribué par le sous-système

Les paramètres autodéfinissant commencent aux décalages 12 et (14 + *n*) où *n* est la longueur du nom de l'application fourni au décalage 14.

Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour connaître les définitions de champ pour cette réponse à la requête.

Table 19. Format de réponse aux requêtes DDM de base avec accès direct et paramètres autodéfinissant du nom

Décalage	Longueur	Contenu	Signification
0	1 mot	Longueur	Longueur de la structure (inclut les paramètres autodéfinissant)
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'95'	Type de réponse à la requête
4–5	2 octets	INDICATEURS	Réservé
6–7	2 octets	LIMIN	Nombre maximal d'octets DDM autorisés dans les transmissions entrantes
8–9	2 octets	LIMOUT	Nombre maximal d'octets DDM autorisés dans les transmissions sortantes
10	1 octet	NSS	Nombre de sous-ensembles pris en charge
11	1 octet	DDMSS	Identifiant de sous-ensemble DDM
12	1 octet	X'04'	Longueur du paramètre
13	1 octet	X'01'	Identifiant d'accès direct
14–15	2 octets	DOID	ID de destination/origine attribué par le sous-système
16	1 octet	Longueur (n+2)	Longueur du paramètre
17	1 octet	X'02'	Nom de l'application DDM
16+*n* – 17+*n*	*n* octets	Nom	Nom du programme d'application distant

Les paramètres autodéfinissant commencent aux décalages 12 et 16.

Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour connaître les définitions de champ pour cette réponse à la requête.

Réponse à la requête de l'appareil auxiliaire IBM

La réponse à la requête de l'appareil auxiliaire est utilisée pour indiquer à l'application hôte la prise en charge d'un appareil auxiliaire IBM qui utilise un flux de données défini par IBM. Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour plus de détails.

Lorsque la fonction est prise en charge, la réponse à la requête est transmise de manière entrante en réponse à un champ structuré de partition de lecture spécifiant Query ou Query List (Liste QCODE = X'9E', Equivalent ou Tous).

Lorsqu'un poste de travail prend en charge plusieurs appareils auxiliaires, la réponse à la requête des appareils auxiliaires IBM doit être envoyée pour chaque appareil.

Paramètres optionnels

Tous les paramètres affichés dans la partie de base de la réponse à la requête doivent être présents. Les paramètres non utilisés sont définis sur X'00'.

Au moins un paramètre autodéfinissant doit être présent.

Table 20. Format de base de l'appareil auxiliaire IBM avec paramètre autodéfinissant l'accès direct

Décalage	Longueur	Contenu	Signification
0–1	1 mot	Longueur	Longueur de la structure (inclut les paramètres autodéfinissant)
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'9E'	Réponse de l'appareil auxiliaire IBM
4	1 octet BIT 0 1–7	FLAGS QUERY B'1' RES	Réservé Partie de lecture (requête, liste de requêtes) L'appareil auxiliaire prend en charge les requêtes réservées, doit être B'0's
5	1 octet	INDICATEURS	Réservé
6–7	2 octets	LIMIN	Nombre maximal d'octets DDM autorisés dans les transmissions entrantes
8–9	2 octets	LIMOUT	Nombre maximal d'octets DDM autorisés dans les transmissions sortantes
10	1 octet	TYPE X'01'X'02' Autres	Type d'appareil auxiliaire pris en charge Affichage de l'appareil auxiliaire IBM Imprimante de l'appareil auxiliaire IBM Réservé
11	1 octet	X'04'	Longueur du paramètre
12	1 octet	X'01'	Accès direct
13–14	1 mot	DOID	ID de destination/origine attribué par le sous-système

QUERY	Ce bit doit être défini sur B'1' pour tous les appareils auxiliaires IBM afin d'indiquer qu'ils prennent en charge la réception d'une partition en lecture (requête, liste de requêtes). Les applications hôtes peuvent alors utiliser une partition de lecture dirigée vers l'appareil auxiliaire pour

	déterminer ses caractéristiques. Le champ structuré destination/origine est utilisé pour diriger le champ structuré Read Partition vers l'appareil auxiliaire. Le niveau de prise en charge minimum pour l'appareil auxiliaire IBM consiste à renvoyer la réponse à la requête Null en réponse au champ Read Partition.
LIMIN	Indique le nombre maximum d'octets pouvant être envoyés dans une transmission entrante. Une valeur LIMIN de X'0000' indique qu'il n'y a aucune limite d'implémentation sur le nombre d'octets transmis en entrée.
LIMOUT	Indique le nombre maximum d'octets pouvant être envoyés à un appareil auxiliaire IBM lors d'une transmission sortante. Une valeur LIMOUT de X'0000' indique qu'il n'y a aucune limite d'implémentation sur le nombre d'octets transmis en sortie.
TYPE	Identifie l'appareil auxiliaire pris en charge. Deux valeurs sont valides. L'un identifie un affichage auxiliaire et l'autre identifie une imprimante auxiliaire. Toutes les autres valeurs sont réservées.

Le processeur d'appareil auxiliaire IBM prend en charge deux paramètres autodéfinissant, 01 et 03. Ils sont définis dans Table 21: Paramètre autodéfinissant l'accès direct à l'appareil auxiliaire IBM on page 227.

Paramètre autodéfinissant l'accès direct

Le paramètre autodéfinissant l'accès direct fournit l'ID à utiliser dans le champ structuré destination/origine dans l'accès direct de l'appareil auxiliaire IBM.

Ce SDP est toujours requis pour accompagner la réponse à la requête de base.

Table 21. Paramètre autodéfinissant l'accès direct à l'appareil auxiliaire IBM

Décalage	Longueur	Contenu	Signification
0	1 octet	X'04'	Longueur du paramètre
1	1 octet	X'01'	Identifiant d'accès direct
2–3	2 octets	DOID	ID de destination/origine

DOID

La valeur de ces octets est utilisée dans le champ ID du champ structuré destination/origine pour identifier l'appareil auxiliaire comme destination ou origine des données qui suivent.

Le protocole PCLK contrôle les paramètres autodéfinissant

La présence du paramètre autodéfinissant des contrôles de protocole PCLK indique que le champ structuré de contrôle de protocole PCLK, ID = X'1013', peut être utilisé à la fois pour les flux de données entrants et sortants destinés à ou depuis le processeur d'appareil auxiliaire IBM.

Table 22. Paramètre autodéfinissant PCLK de l'appareil auxiliaire IBM

Décalage	Longueur	Contenu	Signification
0	1 octet	X'04'	Longueur du paramètre

Table 22. Paramètre autodéfinissant PCLK de l'appareil auxiliaire IBM (continued)

Décalage	Longueur	Contenu	Signification
1	1 octet	X'03'	Contrôles du protocole PCLK
2–3	2 octets	VERS	Version du protocole

VERS

La valeur donnée dans VERS est utilisée pour indiquer les versions de PCLK installées dans le terminal au moment où la réponse à la requête est renvoyée. Par exemple, X'0001' indique PCLK version 1.1.

Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour connaître les définitions de champ pour cette réponse à la requête.

La réponse à la requête définie par le produit

Cette réponse à la requête est utilisée par les produits IBM à l'aide de sous-identifiants enregistrés dans la structure de données X'9C'. La réponse à la requête de flux de données défini par le produit indique la prise en charge d'un appareil auxiliaire de poste de travail 3270DS qui utilise un flux de données défini par le produit IBM. Le flux de données *n'est pas* défini par un document d'architecture de format ayant un point de contrôle identifiable tel qu'un comité de révision d'architecture.

Lorsqu'un appareil auxiliaire prend en charge un flux de données défini par un produit IBM, cette réponse à la requête est transmise de manière entrante en réponse à une liste de requêtes (Liste QCODE = X'9C' ou All).

Paramètres optionnels

Tous les paramètres affichés dans la partie de base de la réponse à la requête et le paramètre autodéfinissant l'accès direct doivent être présents.

Le format de la réponse à la requête définie par le produit est le suivant :

Table 23. Format de base de réponse aux requêtes définies par le produit IBM

Décalage	Longueur	Contenu	Signification
0–1	1 mot	Longueur	Longueur de la structure (inclut les paramètres autodéfinissant)
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'9C'	Flux de données défini par le produit IBM
4–5	2 octets	INDICATEURS	Réservé
6	1 octet	REFID	Identifiant de référence
7	1 octet	SSID	Identifiant de sous-ensemble
8	1 octet	X'04'	Longueur du paramètre

Table 23. Format de base de réponse aux requêtes définies par le produit IBM (continued)

Décalage	Longueur	Contenu	Signification
9	1 octet	X'01'	Accès direct
10–11	1 mot	DOID	ID de destination/origine attribué par le sous-système

Les valeurs valides pour REFID (décalage 6) et SSID (décalage 7) de la réponse à la requête définie par le produit sont les suivantes :

Table 24. Valeurs REFID et SSID valides pour la réponse à la requête définie par le produit IBM

REFID	SSID	Documentation sur les produits et les flux de données
X'01'		Système graphique 5080 : Cet ID de référence indique que le flux de données du système graphique 5080 est pris en charge par l'appareil auxiliaire. Les descriptions de l'architecture graphique 5080, du champ structuré, de l'ID de sous-ensemble, du DOID et des ensembles de fonctions associés sont définies dans le document *IBM 5080 Graphics System Principles of Operation*
	X'01' X'02'	Sous-ensemble graphique 5080 HGFD Sous-ensemble de ports RS232 5080
X'02'		API WHIP (remplacée par le nom de la SRL une fois écrit) Cet ID de référence indique que le flux de données de l'API WHIP est pris en charge par l'appareil auxiliaire. Une description de l'architecture de l'API WHIP est définie dans le document *IBM RT PC Workstation Host Interface Program Version 1.1 User's Guide and Reference Manual*
	X'01'	WHIP Sous-ensemble 1
X'03' à X'FF'		Toutes les autres valeurs sont réservées.

Le processeur défini par le produit IBM prend en charge uniquement le paramètre autodéfinissant l'accès direct. Il est défini dans Table 25: Paramètre autodéfinissant l'accès direct défini par le produit IBM on page 229.

Paramètre autodéfinissant l'accès direct

La présence du paramètre autodéfinissant l'ID de l'accès direct indique que l'appareil auxiliaire est accessible directement en utilisant le champ structuré destination/origine. Lorsque plusieurs appareils auxiliaires utilisant un flux de données défini par le produit sont pris en charge, des réponses distinctes aux requêtes de flux de données définies par le produit doivent être fournies, chacune d'entre elles ayant un DOID unique.

Table 25. Paramètre autodéfinissant l'accès direct défini par le produit IBM

Décalage	Longueur	Contenu	Signification
0	1 octet	X'04'	Longueur du paramètre

Table 25. Paramètre autodéfinissant l'accès direct défini par le produit IBM (continued)

Décalage	Longueur	Contenu	Signification
1	1 octet	X'01'	Identifiant d'accès direct
2–3	2 octets	DOID	ID de destination/origine

DOID

La valeur de ces octets est utilisée dans le champ ID du champ structuré destination/origine pour identifier l'appareil auxiliaire comme destination ou origine des données qui suivent.

La réponse à la requête Document Interchange Architecture

Cette réponse à la requête indique l'ensemble de fonctions DIA (Document Interchange Architecture) pris en charge. Le format de la réponse à la requête DIA est le suivant :

Table 26. Format de base IBM DIA

Décalage	Longueur	Contenu	Signification
0	1 mot	Longueur	Longueur de la structure (inclut les paramètres autodéfinissant)
2	1 octet	X'81'	ID de réponse à la requête
3	1 octet	X'97'	IBM DIA
4–5	2 octets	INDICATEURS	Réservé
6–7	2 octets	LIMIN	Nombre maximal d'octets DDM autorisés dans les transmissions entrantes
8–9	2 octets	LIMOUT	Nombre maximal d'octets DDM autorisés dans les transmissions sortantes
10	1 octet	NFS	Nombre d'ID d'ensemble de fonctions sur 3 octets qui suivent
11–13	3 octets	DIAFS	Identifiant de l'ensemble de fonctions DIA
14– (13+(N*3))	N*3 octets	DIAFS	ID d'ensemble de fonctions DIA supplémentaires
14+(N*3)	1 octet	X'04'	Longueur du paramètre
15+(N*3)	1 octet	X'01'	Accès direct
16+(N*3)	1 mot	DOID	ID de destination/origine attribué par le sous-système

Le processeur du dispositif auxiliaire DIA prend uniquement en charge le paramètre auto-défini à accès direct. Il est défini dans Table 27: Paramètre autodéfinissant l'accès direct défini par le produit IBM on page 231.

La présence du paramètre autodéfinissant l'ID de l'accès direct indique que l'appareil auxiliaire est accessible directement en utilisant le champ structuré destination/origine.

Table 27. Paramètre autodéfinissant l'accès direct défini par le produit IBM

Décalage	Longueur	Contenu	Signification
0	1 octet	X'04'	Longueur du paramètre
1	1 octet	X'01'	Identifiant d'accès direct
2–3	2 octets	DOID	ID de destination/origine

DOID

La valeur de ces octets est utilisée dans le champ ID du champ structuré destination/origine pour identifier l'appareil auxiliaire comme destination ou origine des données qui suivent.

Reportez-vous au document *IBM 3270 Information Display System Data Stream Programmer's Reference* pour connaître les définitions de champ pour cette réponse à la requête.

Appendix B. Différences avec Communication Manager/2 EHLLAPI

Cette annexe décrit les différences entre EHLLAPI de Z and I Emulator for Windows et EHLLAPI pour Communication Manager/2.

Les fonctions EHLLAPI suivantes sont différentes de celles portant les mêmes noms dans Communication Manager/2. Vous devez comprendre les différences lorsque vous utilisez ces fonctions :

- **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)
- Mnémoniques ASCII

Set Session Parameter (9)

Définir les options

Z and I Emulator for Windows ne fournit pas les options suivantes fournies par Communication Manager :

OLDOIA, NEWOIA
COMPCASE, COMPICASE
OLD5250OIA, NEW5250OIA

Paramètres de retour

Lorsque la fonction **Set Session Parameter** (9) est terminée, Communication Manager renvoie une longueur de la chaîne de données valide comme troisième paramètre, la longueur de la chaîne de données. Cependant, Z and I Emulator for Windows renvoie un certain nombre d'options d'ensemble valides comme longueur de chaîne de données.

Option EAB

Dans Communication Manager/2, un remappage de couleurs affecte la valeur de la couleur des caractères dans l'attribut EAB copié par la fonction **Copy PS** (5) ou **Copy PS to String** (8) lorsque l'option EAB est spécifiée dans la fonction **Set Session Parameter** (9).

Dans Z and I Emulator for Windows, cependant, la valeur de la couleur des caractères dans l'attribut EAB dépend du contenu de l'espace de présentation, quel que soit le remappage des couleurs, et elle n'est pas affectée par un remappage des couleurs.

Copy OIA (13)

La fonction **Copy OIA** (13) présente les différences suivantes entre Communication Manager/2 et Z and I Emulator for Windows. Pour plus d'informations sur les positions des groupes et des colonnes, reportez-vous à Copy OIA (13) on page 50.

- Position d'octet 21
  - Z and I Emulator for Windows renvoie X'F6'.
  - Communication Manager/2 renvoie X'20'.
- Positions d'octet 61 à 63
  - Z and I Emulator for Windows ne renvoie pas les informations sur l'imprimante.
  - Communication Manager/2 renvoie les informations sur l'imprimante.
- Groupe 3 : changement d'état

  Communication Manager/2 ne renvoie pas la valeur du bit 2. Le bit 2 est réservé et le bit 0 contient à la fois le décalage supérieur et le verrouillage des majuscules.
- Groupe 8 Octet 1 : entrée interdite
  - Z and I Emulator for Windows ne renvoie pas le bit 6 (l'appareil ne fonctionne pas).
  - Communication Manager/2 peut renvoyer le bit 6.
- Groupe 8 Octet 3 : entrée interdite
  - Z and I Emulator for Windows ne renvoie pas le bit 1 (opérateur non autorisé) ni le bit 2 (opérateur non autorisé -f).
  - Communication Manager/2 peut renvoyer les bits 1 et 2.
- Groupe 8 Octet 4 : entrée interdite
  - Z and I Emulator for Windows ne renvoie pas le bit 2 (attente système).
  - Communication Manager/2 peut renvoyer le bit 2.
- Groupe 10 : groupe 2 mis en évidence
  - Z and I Emulator for Windows ne renvoie pas le bit 0 (sélectionné).
  - Communication Manager/2 peut renvoyer le bit 0.
- Groupe 11 : groupe de couleurs 2
  - Z and I Emulator for Windows ne renvoie pas le bit 0 (sélectionné).
  - Communication Manager/2 peut renvoyer le bit 0.
- Groupe 13 : statut de l'imprimante
  - Dans Z and I Emulator for Windows, ce groupe est réservé.
  - Communication Manager/2 peut renvoyer ce groupe.
- Groupe 14 : graphiques

  Communication Manager/2 ne renvoie pas le bit 0 (curseur graphique).

Copy String to PS (15)

Dans Communication Manager/2, l'option EAB de la fonction **Set Session Parameter** (9) affecte la fonction **Copy String to PS**. Lorsque vous spécifiez l'option EAB, transmettez les données d'attribut qui ont la même taille que les données texte à la fonction avec les données texte.

Dans Z and I Emulator for Windows, cependant, les données à transmettre sont uniquement des données texte, quelle que soit l'option EAB. Si vous souhaitez utiliser la même interface avec Communication Manager/2, utilisez l'option `PUTEAB` de **Set Session Parameter** (9).

Storage Manager (17)

La fonction **Storage Manager** (17) fournie par Communication Manager/2 n'est pas prise en charge par Z and I Emulator for Windows. Utilisez les API fournies par Windows® pour allouer de la mémoire aux applications.

Copy String to Field (33)

Dans Communication Manager/2, lorsque l'option EAB de la fonction **Set Session Parameter** (9) est spécifiée, les données d'attribut sont transmises à la fonction en tant que partie des données. Par conséquent, lorsque vous spécifiez l'option EAB, transmettez les données d'attribut qui ont la même taille que les données texte à la fonction avec les données texte.

Dans Z and I Emulator for Windows, cependant, l'option EAB n'affecte pas le contenu des données de la fonction **Copy String to Field** (33). Les données à transmettre ne sont pas les données d'attribut, mais uniquement les données de texte. Si vous souhaitez utiliser la même interface avec Communication Manager/2, utilisez l'option `PUTEAB` de **Set Session Parameter** (9).

Get Key (51)

Communication Manager/2 renvoie l'état de décalage en utilisant @A, @S ou @r, si l'état de décalage d'une clé transmise n'est pas une clé ou une fonction reconnue par la session d'émulateur. Z and I Emulator for Windows ne prend pas en charge ces mnémoniques ASCII.

Window Status (104)

La fonction EHLLAPI 104 (PM_WINDOW_STATUS) commande 'query extend status' (0x03) renverra le descripteur de la fenêtre de l'espace de présentation de l'émulateur. Ceci est cohérent avec la définition de la fonction et l'implémentation de Communication Manager/2. Cependant, Z and I Emulator for Windows EHLLAPI renvoie le descripteur de la fenêtre de cadre. Les applications EHLLAPI écrites pour Z and I Emulator for Windows utilisant cette fonction doivent utiliser le parent du descripteur de la fenêtre renvoyé.

Query Sessions (10)

Dans Communication Manager/2, le descripteur de l'ordinateur personnel est renvoyé. Cependant, le descripteur n'est pas renvoyé dans Z and I Emulator for Windows.

Connect for Structured Fields (120)

L'objet événement pour l'état de la connexion de communication fourni par Communication Manager/2 n'est pas dans Z and I Emulator for Windows.

Allocate Communications Buffer (123)

Dans Communication Manager/2, la valeur maximale de la taille de tampon demandée est de 64 Ko moins 8 octets (X'FFF8').

Dans Z and I Emulator for Windows, cependant, il s'agit de 64 Ko moins 256 octets (X'FF00').

Mnémoniques ASCII

Les mnémoniques ASCII suivants ne sont pas pris en charge dans Z and I Emulator for Windows :

Mnémoniques	**Signification**
@A@N	Obtenir le curseur
@A@O	Localiser le curseur
@A@X	Hexadécimal
@A@Y	Touche Cmd (Fonction)
@A@a	Retour arrière destructeur
@S@A	Effacer EOL
@S@B	Avancement sur le terrain
@S@C	Retour arrière dans le champ
@S@D	Retour arrière de caractère valide
@S@P	POR (Pour envoi uniquement)
@S@T	Aller à Task Manager
@/	Dépassement de file d'attente (Uniquement dans la fonction **Get Key**)

Get Request Completion (125)

Z and I Emulator for Windows ne prend pas en charge un ID de session vide ou Null.

Appendix C. Avis

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services HCL non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial HCL. Toute référence à un produit, logiciel ou service HCL n'implique pas que seul ce produit, logiciel ou service HCL puisse être utilisé. Tout autre élément, programme ou service fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit de propriété intellectuelle d'HCL. Cependant, il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même le fonctionnement des produits, logiciels ou services non HCL.

HCL peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
Etats-Unis
A l'attention de : Office of the General Counsel

HCL TECHNOLOGIES LTD. FOURNIT LE PRESENT DOCUMENT "EN L'ETAT" ET DECLINE TOUTE RESPONSABILITE, EXPLICITE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE VALEUR MARCHANDE OU D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. HCL peut, à tout moment et sans préavis, apporter des améliorations et/ou modifier les produits et/ou logiciels décrits dans ce document.

Les références à des documents non HCL ou sites Web non HCL sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces documents ou sites Web ne font pas partie des éléments du présent produit HCL et l'utilisation de ces documents ou sites relève de votre seule responsabilité.

HCL pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les détenteurs de licence souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

HCL
330 Potrero Ave.
Sunnyvale, CA 94085

Etats-Unis
A l'attention de : Office of the General Counsel

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par HCL conformément aux dispositions du document HCL Customer Agreement, des Conditions internationales d'utilisation des logiciels HCL ou de tout autre accord équivalent.

Les données de performance présentées ici ont été obtenues dans des conditions de fonctionnement spécifiques. Les résultats peuvent donc varier.

Les informations concernant des produits non HCL ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. HCL n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non HCL. Toute question concernant les performances de produits non HCL doit être adressée aux fournisseurs de ces produits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

Marques

HCL, le logo HCL et hcl.com sont des marques d'HCL Technologies Ltd. dans de nombreux pays. Les autres noms de produit et service peuvent être des marques d'IBM® ou d'autres sociétés.