HCL Z and I Emulator for Windows (ENGLISH)

Contents

Chapter 1. Release Notes

README

HCL Z and I Emulator for Windows Version 3.0 Readme

Read Me - Please!

This document contains information supplementary to the online help and the publications, it includes newly added functions, hints, tips, restrictions, and corrections. Refer the Z and I Emulator for Windows Information Center for other considerations when using Z and I Emulator for Windows Version 3.0, and for complete product documentation.

For information on installing HCL Z and I Emulator for Windows, for information about the new features added in this version, refer to the Installation Guide and Quick Beginnings guide.

Names and license terms for third-party components are referenced in *license.txt*, which is located in the ZIE for Windows installation directory or in the product installation image.

Thank you for choosing Z and I Emulator for Windows Version 3.0.

Table of Contents

What's New in ZIEWIN 3.0
Version 3.0

Return to Top on page 1

New Code Signer Certificate

The code signer certificate used to sign the previous versions of HCL Z and I Emulator for Windows software will expire on April 20, 2024. In the current version, the HCL Z and I Emulator for Windows software is signed with a new code signer certificate that is valid up to January 27, 2027.

This certificate is issued to "HCL America Inc." by "DigiCert SHA2 Assured ID Code Signing CA".

Return to Top on page 1

Support for French Locale

In this release, support for French locale is included in the MLS package.

Return to Top on page 1

Removal of HCL Licensing Server Pre-requisite

From this release onwards, configuration of the HCL Z and I Emulator for Windows with HCL Licensing Server (CLLS) is not required to use the product.

For more information on configuring the Z and I Emulator for Windows, refer to HCL ZIE License Manager.

Return to Top on page 1

Fixes for APARs and Internal defects

This release contains fixes for APARs and internal defects.

Return to Top on page 1

Z and I Emulator for Windows Information Center

Find documentation and links to other resources at the Z and I Emulator for Windows Information Center.

Return to Top on page 1

System Requirements

For information about hardware and software requirements for installing Z and I Emulator for Windows, refer to the System Requirements for Z and I Emulator for Windows.

Return to Top on page 1

Installation Instructions

For instructions on how to install Z and I Emulator for Windows, refer to the Installation Guide for ZIEWin.

Return to Top on page 1

HCL Software Support

For support information and managing product cases, refer to the HCL Software Customer Support Portal..

Return to Top on page 1

ZIE for Windows Blogs

For additional information on Z and I Emulator for Windows, read through the blogs and articles on 'Z and I Emulator for Windows' available on the ZIE for Windows Blogs.

Return to Top on page 1

Notices

This information was developed for products and services offered in the U.S.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

- HCL
- 330 Potrero Ave.
- Sunnyvale, CA 94085
- USA
- Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL product and use of those websites is at your own risk.

HCL may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

- HCL
- 330 Potrero Ave.
- Sunnyvale, CA 94085
- USA
- Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademark Acknowledgments

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Return to Top on page 1

More information

Known Issues in HCL Z and I Emulator for Windows

This document provides the details about some of the known issues in HCL Z and I Emulator for Windows.

Problem

- In the French Operating System for ZIEWin version 3.0, a local user cannot perform Start/Stop/Restart services.

Issue

- After installing Z and I Emulator for Windows version 3.0, the ZIEWinService options (Start/Stop/Restart) are disabled for a local user account in the French Operating System.

Cause

- The local user does not have access to the ZIEWinService.

Resolving the Problem

- Not Applicable.

Workaround

- Administrator user has permission to perform Start/Stop/Restart ZIEWinService.

Frequently Asked Questions in ZIEWin (FAQs)

This document answers some of the FAQs in Z and I Emulator for Windows.

Section-1: FAQs in ZIEWin v1 Refresh Pack update installer

1. **QUESTION:** What is the difference between a ‘Refresh Pack’ and ‘Refresh Pack Update Installer’?

   ANSWER: Starting with version v1.1.1.0, HCL Z and I Emulator for Windows (ZIEWin) is shipped in two forms, as ‘Refresh Pack’ and ‘Refresh Pack Update Installer’.

   - **Refresh Pack** is a complete installer image of the ZIEWin product, which can be installed independently. This does not require any previous version of the product.
   - **Refresh Pack Update Installer** contains only the fixes developed after the release of ZIEWin v1.1.0.0 base version. Each update installer is cumulative in nature; in other words, a new update installer will also contain fixes from the previous update installers. This requires ZIEWin v1.1.0.0 base version or Refresh Pack installed.
2. **QUESTION:** Can I install ZIEWin v1.1.0.0 (Base version) over ZIEWin Refresh Pack Update Installer v1.1.1.0?

   ANSWER: No, we advise not to do this. The v1.1.1.0 Refresh Pack Update Installer allows this downgrade installation though.

   If you have done this accidentally, uninstall both the v1.1.1.0 Refresh Pack Update Installer and ZIEWin v1.1.0.0 (base version). Then proceed to install ZIEWin v1.1.0.0.
3. **QUESTION:** After installing ZIEWin v1.1.0.0, I have upgraded to ZIEWin Refresh Pack Update Installer v1.1.1.0. Can I directly uninstall the base version?

   ANSWER: No, the base version should not be uninstalled without uninstalling the Refresh Pack Update Installer. If done accidentally, please uninstall the Refresh Pack Update Installer.
4. **QUESTION:** What happens when I install a newer ZIEWin Refresh Pack Update Installer over a previous ZIEWin Refresh Pack Update Installer?

   ANSWER: The ZIEWin Refresh Pack Update Installer will be upgraded to the newer version. Both, the ZIEWin base version and the latest Refresh Pack Update Installer version are shown in **Add or Remove Programs** in the Control Panel.

-

Chapter 2. Product Documentation

Licensing

HCL ZIE License Manager

Table of Contents

I. Introduction:

The **License manager** is a tool that facilitates effective software management between end users and software vendors, thereby enabling organizations to track and document the usage of the company's software products. *HCL ZIE License Manager* is a tool used to track the license information for Mainframe Terminal emulator products like *HCL ZIE for Windows*.

The HCL ZIE License Manager can be configured for *HCL ZIE for Windows* version 1.0 & above.

The HCL ZIE License Manager (LM) tracks the license usage for ZIE for Windows, when it is installed and configured with WebSphere Application Server (WAS), ZIEWin Embedded Server or Tomcat.

II. Installation and Configuration of HCL ZIE License Manager Server

Prerequisites for the Installation of License Manager:

- Application Server

Installation of HCL ZIE License Manager:

Follow the below procedure to install License Manager:

1. Download the zip file from *Flexnet Operations.*
2. Extract the zip file into a folder.
3. Install the .ear or .war file on the application server by following the deployment instructions for the respective application server.
4. Enter the URL to access the License Manager Web application:

For example:

http://<appserver-address>:<port-num>/<context-root>/<License Logger>

where,

<appserver-address> : is the hostname or IP address of the server on which the license manager is installed,

<port-num> : is the port that is specified during the deployment of the application server.

<context-root> : is the location name that the Administrator can configure.

Steps to configure the License Manager:

1. Client-based configuration:

   For ZIE for Web:

   For all the client types (ZIEWeb Lite Client Launcher and Java Webstart client) add the following parameters to the configuration file (*config.properties)*, that is located in the ZIEWEB server publish directory.

   - **licenseserverurl =** License Logger URL – <appserver>:<port>/<context-root>/LicenseLogger
   - **timeout=** Logging request interval (in mins) after which the server marks the client as timed out if the request is not sent. Minimum value is 5 and maximum value is 30.
   - **enableMacAddress =** set true to enable mac address logging, default value is false.
   - **enableMachineName** = set false to disable machine name logging, default value is true.

   For example:

   ```
   licenseserverurl=http://127.0.0.1:9080/LicenseManager/LicenseLogger
   ```

   ```
   timeout=5
   ```

   ```
   enableMacAddress=true
   ```

   For ZIE for Windows:

   For ZIE for Windows, the License Manager can be configured using the following methods:

   - InstallShield Wizard / License Manager Settings
   - Updating the pcswin.ini file

   A. Configuring the License Manager through InstallShield Wizard / License Manager Settings

   The License Manager settings can either be configured by providing the required server details in the 'InstallShield Wizard' at the time of ZIEWin installation or can be added/updated in the 'License Manager Settings' section of the Advanced tab within the Preferences.

   - **License Manager Settings**

     License URL: Specifies the HTTP URL of the License Manager Server to which the HCL Z and I Emulator for Windows session sends license parameters.

     Example: " http://<Application Server Hostname or IP>:<Application Server Port>/LicenseLogger"

     where,

     - ***<Application Server Hostname or IP>***: is the hostname or IP of the system where the application and the License Manager are installed.
     - ***<Application Server Port>***: is the system's port where the application and the License Manager are installed.

     Interval: Specifies the time period in minutes at which the HCL Z and I Emulator for Windows session sends license parameters. It is the request interval after which the server marks the client as timeout if the request is not sent. Default and the minimum value is 5 minutes and the maximum value is 30 minutes.

Note: The License Manager Settings set by "Preferences" utility takes precedence over the settings set through installation. If the installation is "user installation" where the application data location is %appdata% in user directory the setting that are set in "Preferences" utility is applicable only for the current user.

B. Configuring the License Manager by Updating the pscwin.ini file

Add the following parameter values to the pcswin.ini file, typically located in the License section of the file as follows:

```
C:\Users\\AppData\Roaming\IBM\PersonalCommunications
Name: licenseserverurl
Value =  http://<Application Server Hostname or IP>:<Application Server Port>/LicenseLogger
Interval = Logging request interval (in mins) after which the server marks the client as
timeout if the request is not sent. The minimum value is 5.
```

For Example:

```
[License]
URL=http://127.0.0.1:9080/LicenseManager/LicenseLogger
Interval=5
```

In an intranet environment, when a License Manager Server is configured with local Certificate Authorities, if an error occurs while verifying the certificates received from the license manager, the HTTPS connectivity from the ZIEWin client to the license manager may also fail.

The following keywords have been introduced into the **pcswin.ini** file to handle this:

```
[License]
IgnoreUnknownCA=Y
IgnoreInvalidCertCN=Y
IgnoreCertRevCheck=Y
```

- When `IgnoreUnknownCA is set to Y`, it allows an invalid certificate authority. This allows ZIEWin to send License information even when the License Manager Server sends an untrusted CA. The setting is recommended only under test environments. The default value of the keyword is set to N.
- When `IgnoreInvalidCertCN is set to Y`, it allows an invalid common name in a certificate; the server's name specified by the application does not match the common name in the certificate. The setting is recommended only under test environments. The default value of the keyword is set to N.
- When `IgnoreCertRevCheck is set to Y`, it ignores certificate revocation problems. This allows ZIEWin to send License information even when it cannot verify whether the host certificate is valid or revoked. The setting is recommended only under test environments. The default value of the keyword is set to N.

Note: The keywords are recommended only under test environments.

Also, the following keywords in the *.ini* file are provided to enable or disable HCL ZIE for Windows emulator sessions from sending the MAC Address and machine name to the HCL License Manager Server. The keywords are part of the License section of the *pcswin.ini* file.

```
[License]
enableMacAddress = N
enableMachineName = Y
```

When `enableMacAddress is set to N`, the license manager server log shows `MAC_ADDRESS_DISABLED` under the `MAC ADDRESS column`. The default value of the keyword is set to N. When, `enableMachineName is set to N`, the license manager server log shows `MACHINE_NAME_DISABLED` under the `MACHINE NAME column`. The default value of the keyword is set to Y.

For Host Access Client Library (HACL):

For HACL applications, developers can set the License server URL and timeout values with the below session parameters and statements:

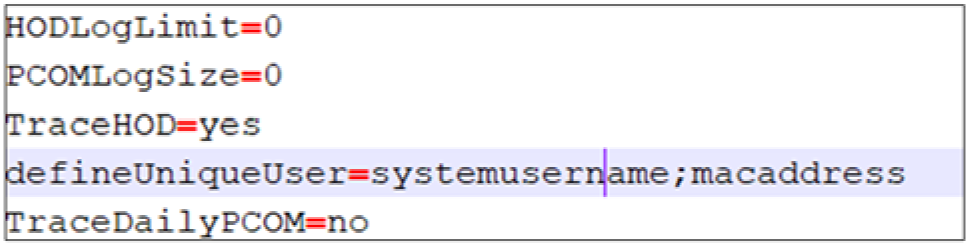
```
p.put(Session.LICENSE_SERVER_URL,"http://<server-address>:<server-port>/<context-root>/LicenseLog
ger");

        p.put(Session.LICENSE_SERVER_TIME_OUT,"5");
 p.put(Session.
        LICENSE_SERVER_MAC_ADDRESS,"true");
 p.put(Session.
        LICENSE_SERVER_MACHINE_NAME,"false");
```

2. Configuration of Unique License Count

   Administrators can mandate license uniqueness, based on different values, by modifying the *adminConfig.properties* file in the License Manager installed directory.

   For Example:

   `defineUniqueUser=systemusername;macaddress`



```
HODLogLimit=0
PCOMLogSize=0
TraceHOD=yes
defineUniqueUser=systemusername;macaddress
TraceDailyPCOM=no
```

Below are the available parameters that need to be modified to define uniqueness.

IP address: ip

System user name: systemusername

MAC address: macaddress

Machine name: machinename

Any combination of the above attributes can be used to define license uniqueness. If more than one parameter value has to be configured, then each parameter should be separated by a semi colon (;).

Note:

- While re-defining a unique user with existing logs, provide a different installation location for License Manager in the servlet parameters (or delete the existing License Manager folder) and restart, to avoid errors during license count calculation.
- Restart the HCL ZIE License Manager, after re-defining the unique user.

3. Configuring the ZIE License Manager Servlet Parameters

Following is a list of parameters that can be used to configure the License Manager:

Property	Value	Description
ZIE_WIN_Enabled	true/false	Enable/ disable ZIE for Win
Directory_Location	C:\\dir_location	Directory Location for logs

Depending on the application server used to deploy the License Manager application, the path from which the above properties can be initialized may change.

For WebSphere Application Server (WAS):

- Login to *WebSphere Application Server*.
- Go to *Applications*.
- Click the *WebSphere enterprise applications* under the Application Type.
- Click the License Manager .war file.
- Click the *Initialize parameters for servlets link* under the Web Module Properties section.
- Enter the required values.
- Save the changes.

For Embedded Server:

- Users can override the License Manager configuration by modifying the properties of “lm_overrides.xml” in the *conf* directory under the *lib* directory of the product.
- Save the changes.

For Tomcat Server:

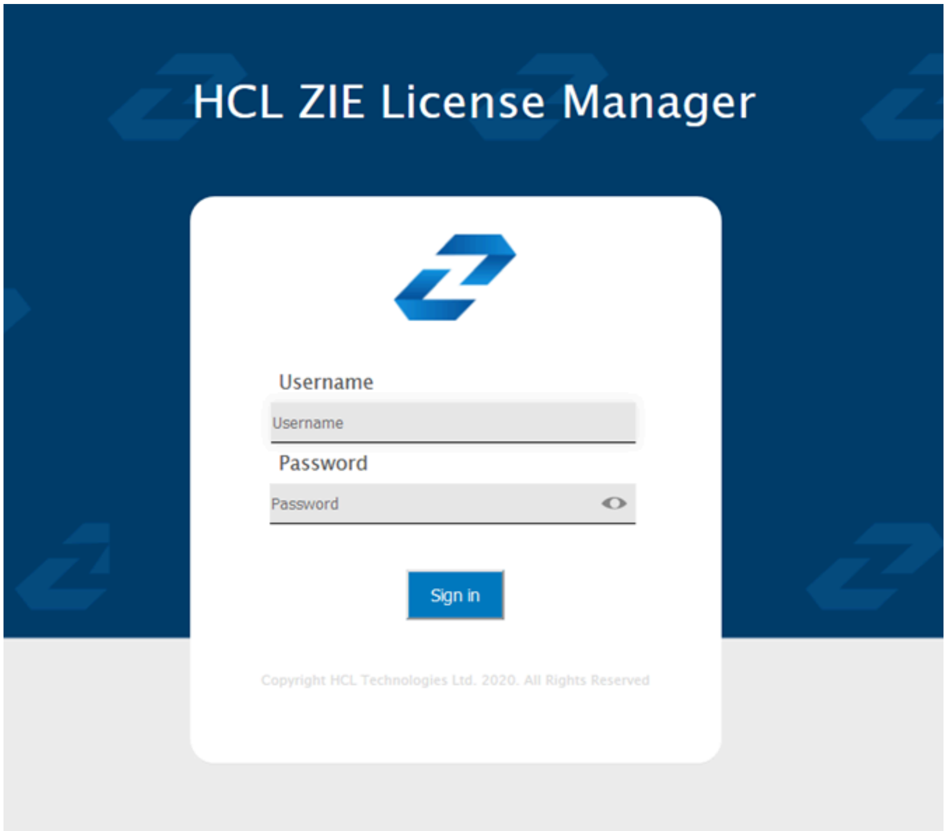
- Navigate to the *application* folder under the *webapps* directory of tomcat.
- Edit the *web.xml* file of the application.
- Save the changes.

III. Using the HCL ZIE License Manager

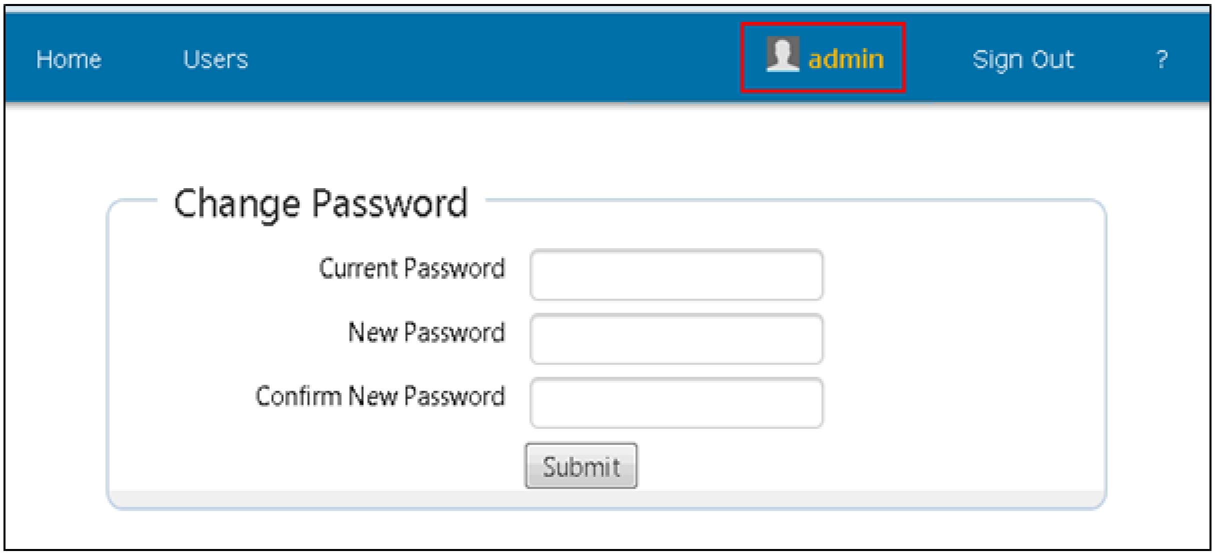
After the successful installation of the server module, Administrators can use the License Logger to monitor the client login from the License Manager Admin console.

The console can be accessed by navigating to: http://appserver-address:port-num/<context-root>/

When prompted for a username and password, users may log in with the default username as **admin** and password as **password**.

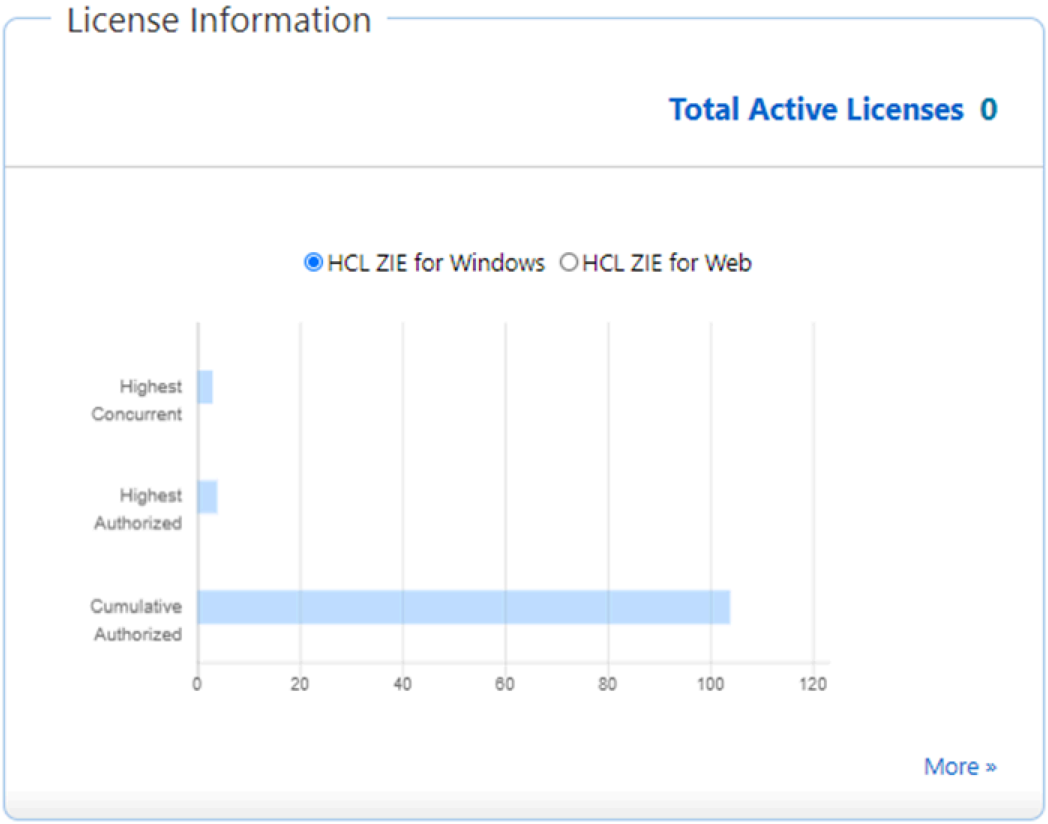


After logging in as a License Manager Admin, you can change the password by clicking on ***admin*** on the menu bar. Refer to the **Create New User** and **Change User Password** sections below, for more information.



As a License Manager Administrator, you can perform the following tasks:

- Manage Users who can access the administrator console
- Configure the number of licenses
- Monitor the number of Users currently active
- Configure the log settings
- View or download log files



License Information:

- **Total active licenses**

  Number of active concurrent licenses for ZIE for Web & ZIE for Windows combined.

License Count Statistics:

- **Highest Concurrent License Count**

  The highest concurrent license count gives the maximum number of distinct users who have accessed the product (ZIE for Web and ZIE for Windows) simultaneously, since the installation of License Manager.

- **Highest Authorized License Count**

  The highest authorized license count gives the maximum number of authorized users who have accessed either of the products (ZIE for Web and ZIE for Windows) on a day.

- **Cumulative Authorized License Count**

  Cumulative Authorized License Count gives the total number of distinct authorized users who have accessed either of the products (ZIE for Web and ZIE for Windows) till date, since the installation of License Manager.

- **More**

  Clicking on the **More** option takes us to the License Summary Report for both the products combined.

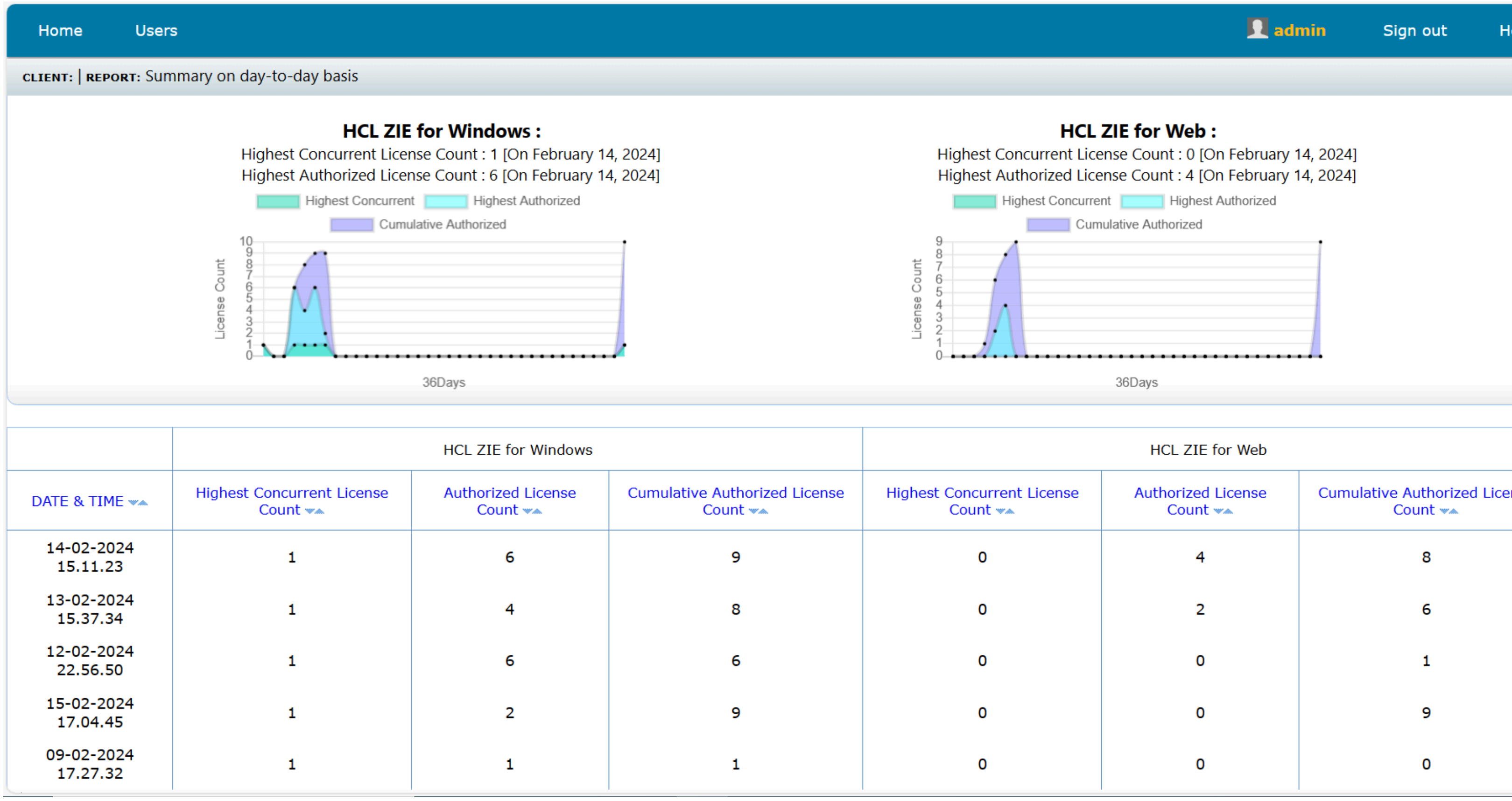
  By default, the report shows a daily summary of licenses for the individual products, ZIE for Web and ZIE for Windows. The filter feature can be used to view the license usage for a specified date interval. It includes the following information for each product:

  - Highest Concurrent License Count (MM-DD-YYYY)
  - Highest Authorized License Count (MM-DD-YYYY)

  The tabular view shows the following license details for ZIE for Web and ZIE for Windows with time stamp.

  - Highest Concurrent License Count
  - Highest Authorized License Count
  - Cumulative Authorized License Count

  The admin can sort specific columns as required.



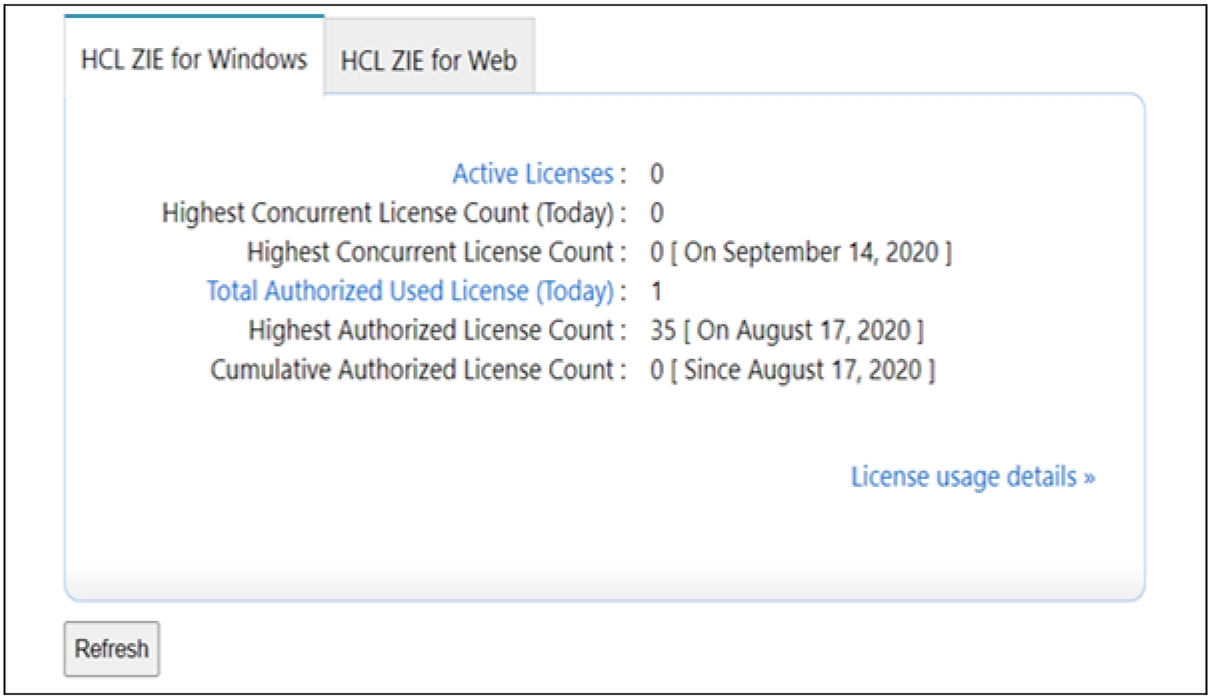
CLIENT: | **REPORT:** Summary on day-to-day basis

HCL ZIE for Windows :
Highest Concurrent License Count : 1 [On February 14, 2024]
Highest Authorized License Count : 6 [On February 14, 2024]

HCL ZIE for Web :
Highest Concurrent License Count : 0 [On February 14, 2024]
Highest Authorized License Count : 4 [On February 14, 2024]

	HCL ZIE for Windows			HCL ZIE for Web		
DATE & TIME	Highest Concurrent License Count	Authorized License Count	Cumulative Authorized License Count	Highest Concurrent License Count	Authorized License Count	Cumulative Authorized Lice
14-02-2024 15.11.23	1	6	9	0	4	8
13-02-2024 15.37.34	1	4	8	0	2	6
12-02-2024 22.56.50	1	6	6	0	0	1
15-02-2024 17.04.45	1	2	9	0	0	9
09-02-2024 17.27.32	1	1	1	0	0	0

License Count Statistics specific for *ZIE for Web* and *ZIE for Windows*:



Active Licenses : 0
Highest Concurrent License Count (Today) : 0
Highest Concurrent License Count : 0 [On September 14, 2020]
Total Authorized Used License (Today) : 1
Highest Authorized License Count : 35 [On August 17, 2020]
Cumulative Authorized License Count : 0 [Since August 17, 2020]

Each section/tab shows the following information for the corresponding product.

- Active Licenses
- Highest Concurrent License Count (Today)
- Highest Concurrent License Count
- Total Authorized Used License (Today)
- Highest Authorized License Count
- Cumulative Authorized License Count

Active Licenses

Active concurrent licenses for ZIE for Web / ZIE for Windows (for the active sessions). Admin can view the active client details by clicking the highlighted Active Licenses link. Active clients report shows the below information about all active clients at that time.

- IP Address
- System Username
- Machine Name
- MAC Address
- Sub-Client Type
- Check-In-Time

The Administrator can sort specific columns as required.

Highest Authorized License Count

Highest Authorized license count gives the maximum number of authorized license users accessing the product (ZIE for Web / ZIE for Windows) on any day since the installation of License Manager.

Cumulative Authorized License Count

Cumulative Authorized license count of distinct authorized users since the installation of license Manager for the product (ZIE for Web / ZIE for Windows).

License usage details:

License details report for the current day (ZIE for Web / ZIE for Windows).

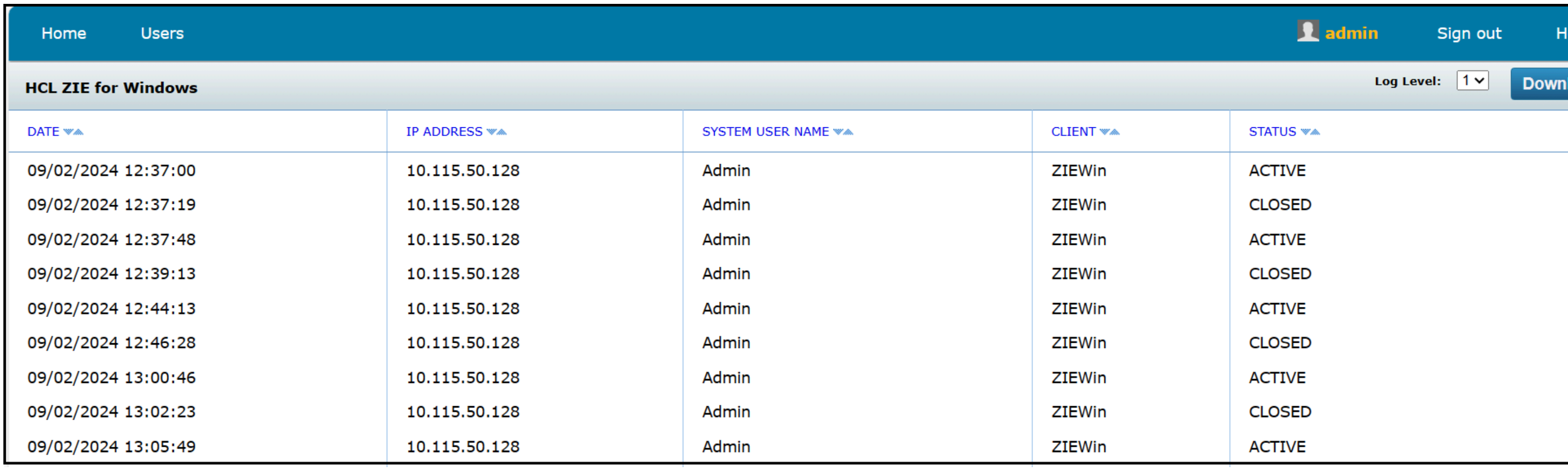
Log level

This option is available on "**License usage details**" page.

Select the log level (from 1-LOW to 3-HIGH) to filter the number of entries to be logged. The default log level, which is Level 3 (HIGH), logs all the entry parameters, including connection closed entry along with the periodic client check-ins. All the other log levels (1 and 2) will log only the first client check-in and the connection closed entry.

Following are the parameters which are logged:

- IP Address of the client
- System User Name
- Machine Name
- MAC Address
- Client Type
- Status of connection
- Timestamp



DATE	IP ADDRESS	SYSTEM USER NAME	CLIENT	STATUS
09/02/2024 12:37:00	10.115.50.128	Admin	ZIEWin	ACTIVE
09/02/2024 12:37:19	10.115.50.128	Admin	ZIEWin	CLOSED
09/02/2024 12:37:48	10.115.50.128	Admin	ZIEWin	ACTIVE
09/02/2024 12:39:13	10.115.50.128	Admin	ZIEWin	CLOSED
09/02/2024 12:44:13	10.115.50.128	Admin	ZIEWin	ACTIVE
09/02/2024 12:46:28	10.115.50.128	Admin	ZIEWin	CLOSED
09/02/2024 13:00:46	10.115.50.128	Admin	ZIEWin	ACTIVE
09/02/2024 13:02:23	10.115.50.128	Admin	ZIEWin	CLOSED
09/02/2024 13:05:49	10.115.50.128	Admin	ZIEWin	ACTIVE

Level 3: All Parameters

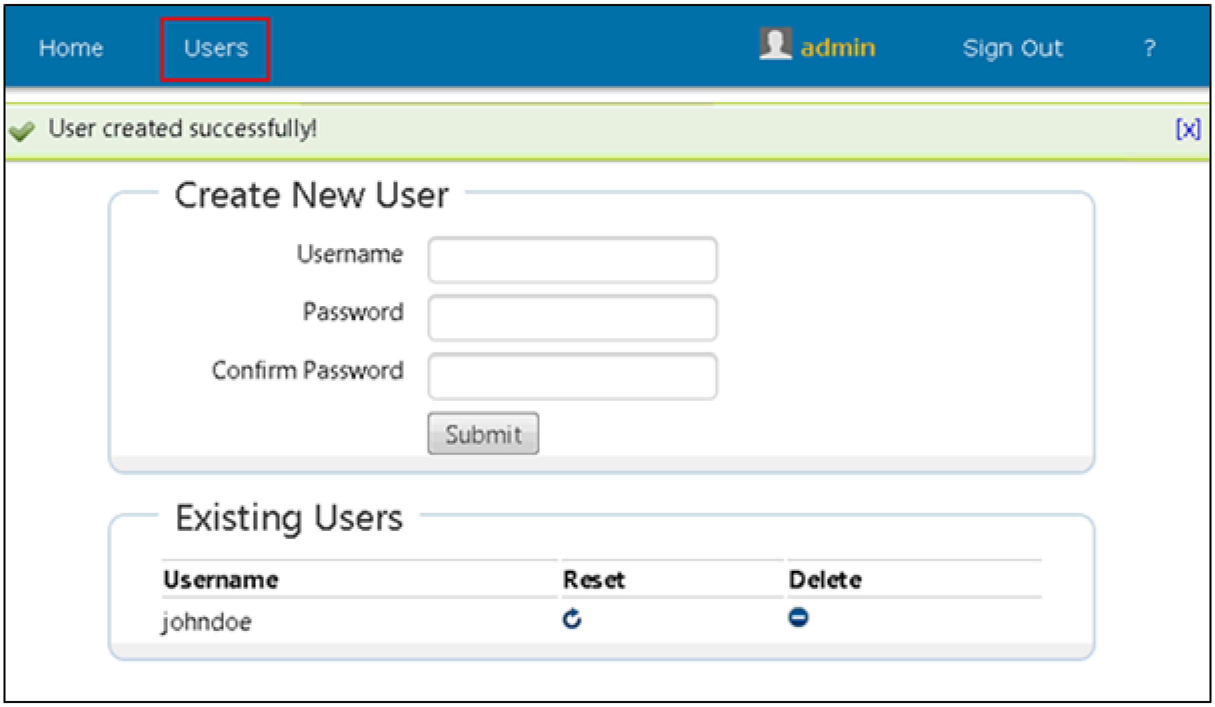
Level 2: Timestamp, IP, System User Name, Client Type, Sub-Client Type, Machine Name, Connection Status

Level 1: Timestamp, IP, System User Name, Client Type, Sub-Client Type, Connection Status

The license usage report containing Log level 3 (HIGH) with license usage information can be extracted by the Admin into a .csv file (that can be opened as an Excel Workbook).

User Management:

To manage Users who can access the License Manager Admin console, click on **Users** on the menu bar.



User created successfully!

Create New User

Username	
Password	
Confirm Password	

Existing Users

Username	Reset	Delete
johndoe	↻	⊖

Create a New User

To create a new user, follow the below steps:

- Go to the **Users** tab.
- Specify the **Username** (username is not case sensitive).
- Specify the required **password** (password length must be between 5 to 20 characters).
- Re-enter the password in the **Confirm Password** box.
- Click **Submit** .

Note: The Admin can create new users for the license manager.

Delete an Existing User

Under the Existing Users section, there is an option to delete the existing user by selecting the icon ⊖ next to the username.

To Reset the Password for an Existing User

If required, Admin can reset the password for a selected user to the default password (**password**), by clicking the icon ↻ next to the username in the existing users list.

To Change User Password

Users can change the current password, by hovering the mouse pointer over the **Username** field displayed on top of the screen. After specifying the current and new password, click **Submit**.

IV. Limitations of HCL ZIE License Manager

National Translation is not available. Help files are currently available only in English.

Quick Beginnings

About This Book

This book describes how to install, configure, and start HCL Z and I Emulator for Windows. After you get Z and I Emulator for Windows up and running and begin to perform various tasks, use the online help whenever you need additional information. See Where to Find More Information on page 19 for information about online help, the Z and I Emulator for Windows library, and related publications. .

This book is for users of:

- *HCL Z and I Emulator for Windows, Version 3.0*
- *HCL Z and I Emulator for Windows iSeries, Version 3.0*

See What's in the Package on page 18 for information regarding what is in the product package.

In this book, *Windows*® refers to Microsoft® Windows® 7, Windows® 8/8.1, Windows® 10, Windows® Server 2008, and Windows® Server 2012. When information is applicable only to a specific operating system, this will be indicated in the text.

What's in the Package

The Z and I Emulator for Windows Version 3.0 package contains *HCL Z and I Emulator for Windows, Version 3.0*, which provides 5250, 3270, and VT emulation and connections to z/OS™, z/VM™, eServer™ i5, System i5™, iSeries™, zSeries™ and ASCII host systems.

In addition to the products previously mentioned, the Z and I Emulator for Windows installation image contains the following:

- Adobe Acrobat Reader, to enable you to read softcopy books available on the installation image
- Book files in PDF format

Note:

1. For each of the two basic packaging options, there are also separate installation images, depending on:

   Security Levels

   Z and I Emulator for Windows is shipped at the 168-bit encryption level.

Where to Find More Information

The following sections discuss getting help when you are installing, configuring, or using Z and I Emulator for Windows.

The *Quick Beginnings* book is also available online, in HTML form. You can access the book from the **Help** menu in the Session Manager.

Information Center

You can find documentation and links to other resources at the Z and I Emulator for Windows Information Center, at the following address:

https://help.hcltechsw.com/zie/ziewin/3.0/index.html

The Information Center contains reference material that is not found in this book, such as keyboard layouts and host code page tables.

Online Help

The help facility describes how to install, configure, and use Z and I Emulator for Windows. Online help is very extensive and includes information about every aspect of configuring and using Z and I Emulator for Windows. You can use Z and I Emulator for Windows online help just as you use the online help for Windows®.

Use help to obtain information about:

- Menu choices
- Operation procedures
- Operations in windows
- Meanings of the terms displayed in windows
- Causes of errors and the corresponding actions to take
- Mouse-based operations
- Operation without a mouse
- Detailed explanations of specific terms
- Further technical information about Z and I Emulator for Windows
- Detailed explanations of operator information area (OIA) messages

Z and I Emulator for Windows Library

The Z and I Emulator for Windows library includes the following publications:

- *Installation Guide*
- *Quick Beginnings* (this document)
- *Emulator User's Reference*
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Host Access Class Library*
- *Host Access Class Library*
- *Configuration File Reference*

In addition to the PDF documents, there are HTML documents provided with Z and I Emulator for Windows:

Quick Beginnings

The HTML form of *Quick Beginnings* contains the same information as the PDF version. The HTML files are installed automatically and can be accessed from the Help menus in the Session Manager and .WS session panels.

Contacting HCL

This section lists ways you can reach HCL in case you encounter a problem or concern with Z and I Emulator for Windows. Depending on the nature of your problem or concern, we ask that you be prepared to provide the following information to allow us to serve you better.

- The environment in which the problem occurs:
  - Z and I Emulator for Windows configuration
    - Z and I Emulator for Windows version and manufacturing refresh level
    - The name of the workstation profile
  - Workstation configuration
    - The machine type and model, the system memory, the video adapter
    - The communication adapter you are using
    - Other adapters (especially communication adapters) installed
    - The printer type and model
    - Other devices installed, such as sound cards, modems, or fax machines
  - Software configuration
    - Windows® version and level
    - Communication and device-driver version and level
    - Other communication programs (such as Microsoft® or Microsoft® Data Link Control) that are running and using resources
    - Printer driver version and level
  - Host configuration
    - The upstream host connection and configuration

- Problem analysis information
    - Symptoms
    - Type of problem
    - OIA messages or error messages (if any)
    - Key factors related to the problem

If you have a technical problem, take the time to review and carry out the actions suggested here. Use your local support personnel before contacting HCL. You can also check the Hints and Tips at the Z and I Emulator for Windows support Web page for more information. Only persons with in-depth knowledge of the problem should contact HCL; therefore, support personnel should act as the interface with HCL.

For information about problem analysis tools, refer to *Emulators User's Reference* for Z and I Emulator for Windows Version 3.0. This reference also provides detailed, specific emulator information about printing, file and data transfer, node operations, and other topics.

Managed ZIEWIN and Interoperablity

This section provides detailed information about Managed ZIEWin and Interoperabilty between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients.

HCL Z and I Emulator for Windows uses Session Manager Online dialog to provide easy access to workstation profiles and batch files on the ZIE Server. With Session Manager Online users can create or start a single or multiple sessions and or batch files. Users can create their own profile to the ZIE server and migrate existing files such as the workstation profiles (*.WS) and batch files (*.BCH) that were stored on the ZIE Server.

This "How To" document aims to supplement additional detailed information in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN) as referenced below.

Steps to Install Using Managed:

Refer to Planning to Install Z and I Emulator for Windows on page 42

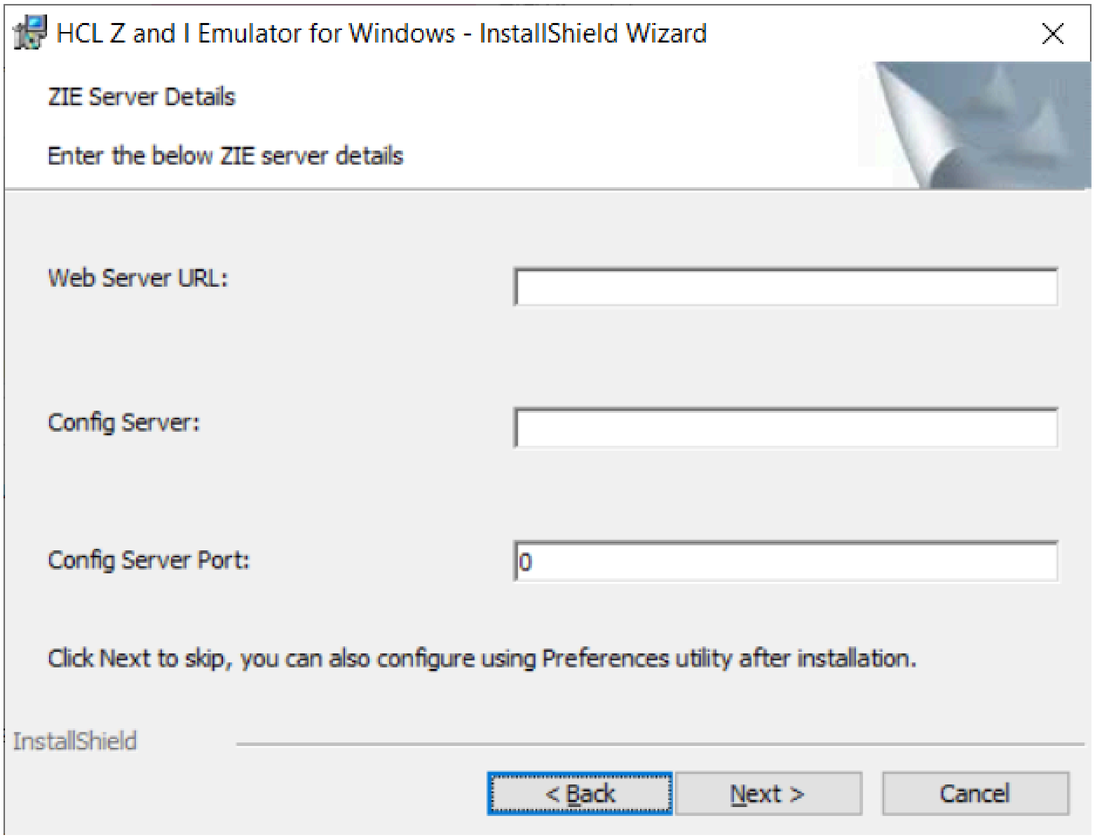
The steps provided in this document are applicable to all Windows 10 versions which are 64-bit OS level.

Prerequisites :

1. Download a copy of HCL Z and I Emulator for Windows 64-bit base package and HCL Z and I Emulator for Windows RP1.zip
2. A HCL Z and I Emulator for Web server is required for the Session Manager Online to work.
3. Create a folder (e.g. MPZiewin) in the ZIEWEB published directory.
4. Unzip and dump the HCL Z and I Emulator for Windows RP1 contents in *MPZiewin* folder.
5. Right click the *MPZiewin folder > Properties > Sharing > Advance Sharing > Put a check on Share this Folder*.
6. Click **OK** then **Close**.
7. Repeat steps 5-6 with ZIEWEB folder.
8. Obtain the IP address of the ZIEWEB server and use it at step 2 below.

Follow the steps below in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN):

1. There are two ways to input the ZIE Server configuration details. Choosing to go either way will have the same results.
   - During the ZIEWIN installation a new panel has been added.



HCL Z and I Emulator for Windows - InstallShield Wizard

ZIE Server Details

Enter the below ZIE server details

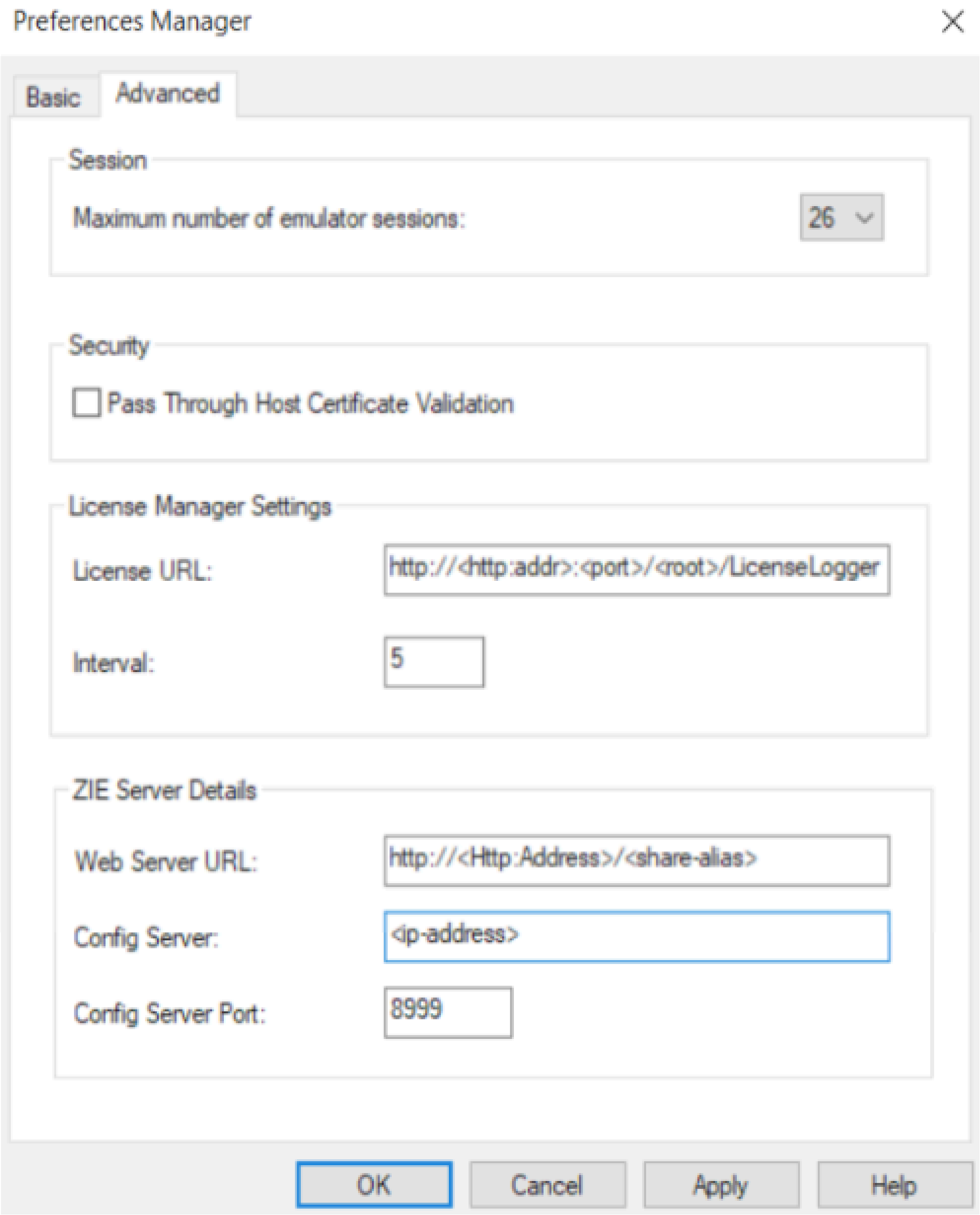
Web Server URL:

Config Server:

Config Server Port: 0

Click Next to skip, you can also configure using Preferences utility after installation.

   - Preferences Manager - Click on Start > HCL Z and I Emulator for Windows > Preferences > Advanced



Preferences Manager

Basic | Advanced

Session

Maximum number of emulator sessions: 26

Security

□ Pass Through Host Certificate Validation

License Manager Settings

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

ZIE Server Details

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

2. Enter the configuration parameters based on the information below:
   - **Web Server URL** : The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file will be downloaded for installation. Installer or fix pack will be installed on the system by "Start or Configure Sessions - Online" program.>
   - **Config Server** : URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the HOD Embedded Server or on any configured Application Server.

     Example: http://< Application Server IP >/<Configured context root of the application>

     For more details on WAR file deployment, refer to the technote <hyper-link>.

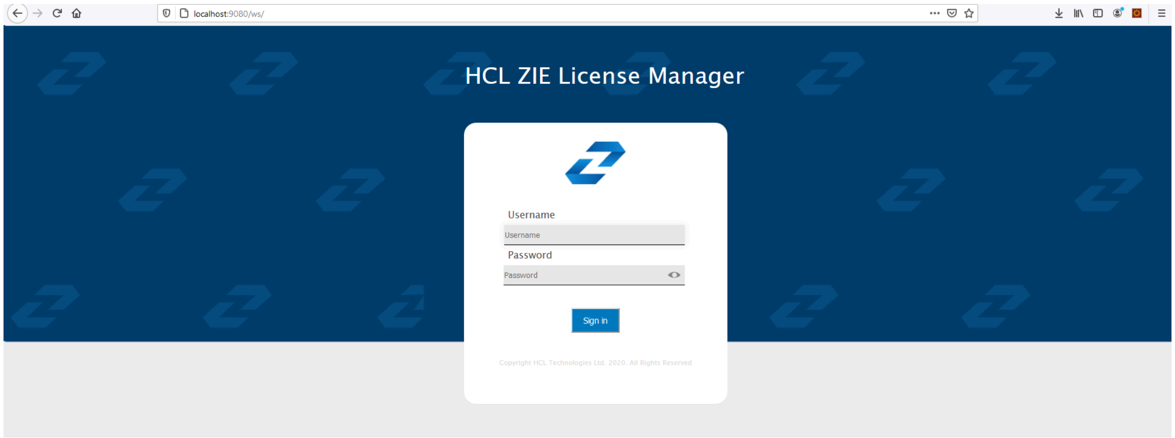
- **Config Server Port** : Port number of Application Server where interoperability module (.war file) is deployed.

  Example: 9080

3. Click **OK**.
4. *Open File Explorer > This PC > Map network drive.* Use the IP address of the Web Server along with the folder where the HCL Z and I Emulator for Windows RP1.msi is located. E.g. \\192.168.56.102\MPZiewin
5. Click **Save**.
6. Repeat steps 4-5 with ZIEWEB folder. E.g. \\192.168.56.102\ZIEWEB

Note: This completes configuring Managed HCL Z and I Emulator for Windows (ZIEWIN).

7. To validate that the configuration is correct, create a new Username or use an Existing User in the Session Manager Online. *Click on Start > HCL Z and I Emulator for Windows > Start or Configure sessions - Online*



Points to consider:

- When mapping the network drive ensure that client machine and ZIEWEB server is within the same network.
- The Session Manager Online checks for updates at startup. It is essential that HCL Z and I Emulator for Windows RP1 is in the ZIEWEB published directory.

Interoperability between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients

The interoperability feature allows the ZIEWin users to use the ZIEWin sessions from other HCL terminal emulator clients, such as ZIEWeb and ZIEWeb Client. ZIEWin users can use the “Session Manager Online” utility to store the new sessions and migrate the existing sessions to ZIE server, these sessions are then converted to ZIEWeb Session formats for the ZIEWeb and ZIEWeb Client usage.

Note: Interoperability feature is introduced in ZIEWin 2.1 version.

ZIEWin client communicates with the ZIE server over HTTP/HTTPS connectivity using JSON data format.

The interoperability feature is supported from ZIEWeb v2.1.0.0 & ZIEWeb Client v2.1.0.0 onwards and is applicable for 3270 Display, 5250 Display, 3270 Printer, 5250 Printer, and VT sessions.

Note: The session conversion happens only for ZIEWin to ZIEWeb sessions and not vice versa.

When the user stores the ZIEWin sessions using the “Session Manager Online” utility, they are converted to ZIEWeb sessions before saving them to the ZIE server . After storing to the ZIE server , users can log in from ZIEWin, ZIEWeb, or ZIEWeb Client to work with the stored ZIEWin sessions.

Using ZIEWin Sessions from ZIEWeb and ZIEWeb Client:

After the ZIEWin sessions are stored in the ZIE server , if any changes are made to the session definition from any of the clients, it is saved in the ZIE server . These session changes will be available to ZIEWin users after the next login.

Below is the list of supported parameters as part of the Interoperability feature.

Table 1. List of Supported Parameters for Interoperability

ZIEWIN Parameter	ZIEWEB Parameter
Primary Host Name or IP Address	Destination Address
Primary Port Number	Destination Port
Primary LU or Pool Name	LU or Pool Name
Screen Size	Screen Size
Host Code-Page	Host Code-Page
Auto-reconnect	Auto-reconnect
Backup 1 Host Name or IP Address	Backup 1 Destination Address
Backup 2 Host Name or IP Address	Backup 2 Destination Address
Backup 1 Port Number	Backup 1 Destination Port
Backup 2 Port Number	Backup 2 Destination Port
Backup 1 LU or Pool Name	Backup 1 LU or Pool Name
Backup 2 LU or Pool Name	Backup 2 LU or Pool Name
Enable Security	Protocol
Workstation ID	Workstation ID
Server Authentication	Server Authentication
Message Queue	Message Queue
Message Library	Queue Library
Send Personal Certificate to Server if it is Requested	Send a Certificate
Send Personal Certificate Trusted by Server	Certificate Source
Send Personal Certificate Based on Key Usage	Enable Key Usage
Machine Mode	Terminal Type (VT session)

Table 1. List of Supported Parameters for Interoperability (continued)

ZIEWIN Parameter	ZIEWEB Parameter
AutoWrap	AutoWrap (VT session)

Note: Only the listed parameters will be modified from ZIEWeb / ZIEWeb Client for a ZIEWin profile. If any other parameters are updated from ZIEWeb / ZIEWeb Client, there will not be any changes to the ZIEWin session. Users should modify the ZIEWin sessions either from ZIEWin or ZIEWeb / ZIEWeb Client at a time and should avoid simultaneous modifications from different clients.

Interoperability 2.1.0.0 Configuration Introduction:

ZIEWeb v3.0 (from v2.1.0.0 onwards) introduced interoperability between ZIEWin and ZIEWeb. This allowed ZIEWin sessions to be accessed through ZIEWeb and ZIEWeb Client after the session definitions were uploaded to the ZIE server .

Password provided during the user creation will be encrypted using AES 128-bit algorithm and will be sent to the server through the HTTP/HTTPS protocol as Json object. UID is added to the WS and BCH profile files for unique identification. Only Connection parameters are considered for the interoperability between ZIEWin and ZIEWeb Clients and vice-versa.

After the ZIEWin sessions are converted and stored in the ZIE server , any changes made to the common parameters from any of the clients will be saved on the ZIE server . These parameter changes will be available to ZIEWin users after the next login.

Steps to install:

1. Install the ZIEWeb v3.0.
2. Install the ZIEWin v3.0

WAR File Configuration:

The interoperability executable (**ZIEWeb_Interoperability.war**) is available under the lib directory of the product.

For Embedded Web Server:

If the Embedded Web Server is used, by default Interoperability application is running on context root "interop”. If the user needs to change the context root, add the following parameter to the configuration file **(*config.properties*)**, located in the ZIE server publish directory.

Example: InterOpContextPath=interop

The default ZIE server IP is 127.0.0.1 and the ZIE server port is 8999. If the user needs to connect to the ZIE server located on a different machine, then override the interoperability configuration by modifying the properties of ***“interop_overrides.xml”*** in the conf directory under the lib directory of the product.

Table 2. List of properties that can be used to configure the Interoperability

Property	Value	Description
ZIEWEB_SERVER_IP	127.0.0.1	ZIE server address

Table 2. List of properties that can be used to configure the Interoperability (continued)

Property	Value	Description
ZIEWEB_SERVER_PORT	8999	ZIEWEB Config Server port
Directory_Location	C:\\dir_location	Directory Location for logs

The user can utilize **ZIEWeb_Interoperability.war** file (available under the lib directory of the product) to deploy to different application servers such as WAS/Tomcat.

For WebSphere Application Server (WAS):

1. Log in to **WebSphere Application Server**.
2. Go to **Applications**.
3. Click WebSphere enterprise applications under **Application Type**.
4. Select **ZIEWeb_Interoperability.war file**.
5. Click on Initialize parameters for servlets link under **Web Module Properties** section.
6. Enter the required values.

Supported Application Servers: *Apache Tomcat and WAS*.

Limitations

1. Only connection parameters are considered for interoperability between ZIEWin and ZIEWeb and vice-versa.
2. Session creation from ZIEWeb / ZIEWeb Client will not be converted to a ZIEWin session.

Known Issues

1. For stored ZIEWin sessions, changes to any session parameters (not only the listed parameters) from ZIEWeb / ZIEWeb Client will be overridden or set to default when there is an update from **"Session Manager Online"** (ZIEWin Client).
2. If there are simultaneous profile updates from any of the two clients, the most recent update will be saved as the final copy in the ZIE server .
3. Modifications done in multiple sessions (add, delete sessions, or rename) from the ZIEWeb Clients do not reflect in the ZIEWin Client.
4. Saving/renaming profiles with special characters (Ex: \ / : * ? " < > |.) in ZIEWeb/ ZIEWeb Clients will result in unexpected behavior in the ZIEWin Client.

   How to setup Managed HCL Z and I Emulator for Windows (ZIEWIN)

   HCL Z and I Emulator for Windows uses Session Manager Online dialog to provide easy access to workstation profiles and batch files on the ZIE Server. With Session Manager Online users can create or start a single or multiple sessions and or batch files. Users can create their own profile to the ZIE server and migrate existing files such as the workstation profiles (*.WS) and batch files (*.BCH) that were stored on the ZIE Server.

   This "How To" document aims to supplement additional detailed information in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN) as referenced below.

   Steps to Install Using Managed: Refer to Planning to Install Z and I Emulator for Windows on page 42

The steps provided in this document are applicable to all Windows 10 versions which are 64-bit OS level.

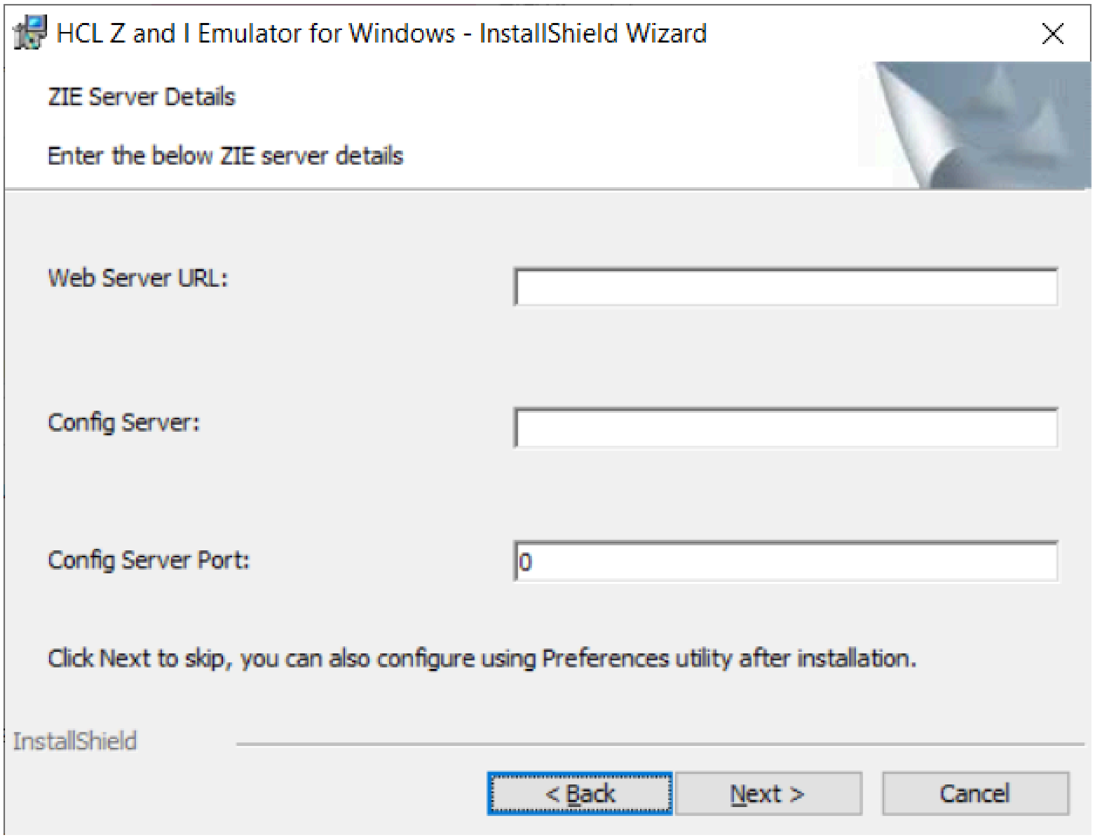
Prerequisites :

a. Download a copy of HCL Z and I Emulator for Windows 64-bit base package and HCL Z and I Emulator for Windows RP1.zip
a. A HCL Z and I Emulator for Web server is required for the Session Manager Online to work.
b. Create a folder (e.g. MPZiewin) in the ZIEWEB published directory.
c. Unzip and dump the HCL Z and I Emulator for Windows RP1 contents in *MPZiewin* folder.
d. Right click the *MPZiewin folder > Properties > Sharing > Advance Sharing > Put a check on Share this Folder*.
e. Click **OK** then **Close**.
f. Repeat steps 5-6 with ZIEWEB folder.
g. Obtain the IP address of the ZIEWEB server and use it at step 2 below.

Follow the steps below in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN):

There are two ways to input the ZIE Server configuration details. Choosing to go either way will have the same results.

- During the ZIEWIN installation a new panel has been added.


HCL Z and I Emulator for Windows - InstallShield Wizard

ZIE Server Details

Enter the below ZIE server details

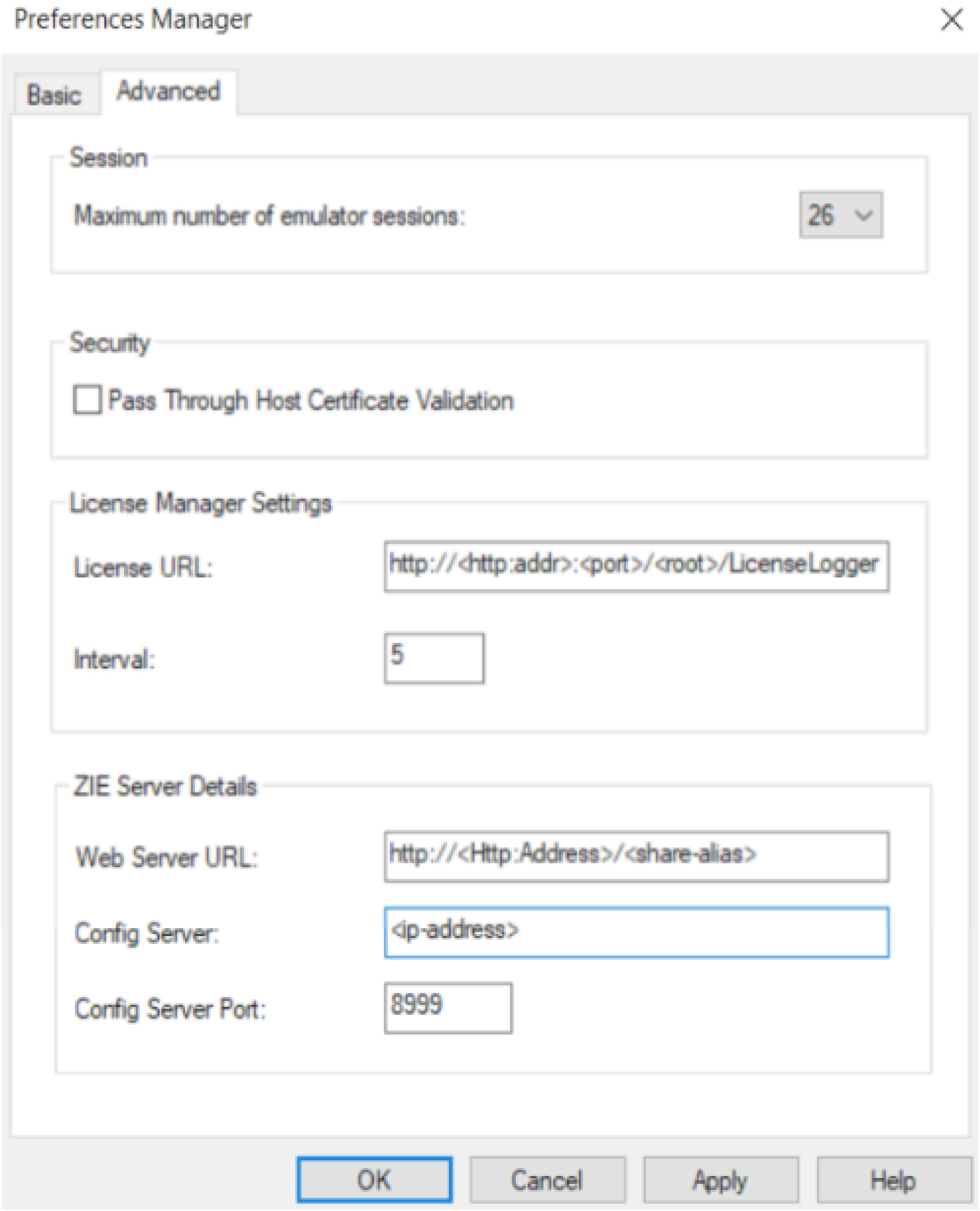
Web Server URL:

Config Server:

Config Server Port: 0

Click Next to skip, you can also configure using Preferences utility after installation.

- Preferences Manager - Click on Start > HCL Z and I Emulator for Windows > Preferences > Advanced



Maximum number of emulator sessions: 26

□ Pass Through Host Certificate Validation

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

a. Enter the configuration parameters based on the information below:
    - **Web Server URL** : The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file will be downloaded for installation. Installer or fix pack will be installed on the system by "Start or Configure Sessions - Online" program.>
    - **Config Server** : URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the HOD Embedded Server or on any configured Application Server.

      Example: http://< Application Server IP >/<Configured context root of the application>

      For more details on WAR file deployment, refer to the technote <hyper-link>.

- **Config Server Port** : Port number of Application Server where interoperability module (.war file) is deployed.

  Example: 9080

b. Click **OK**.

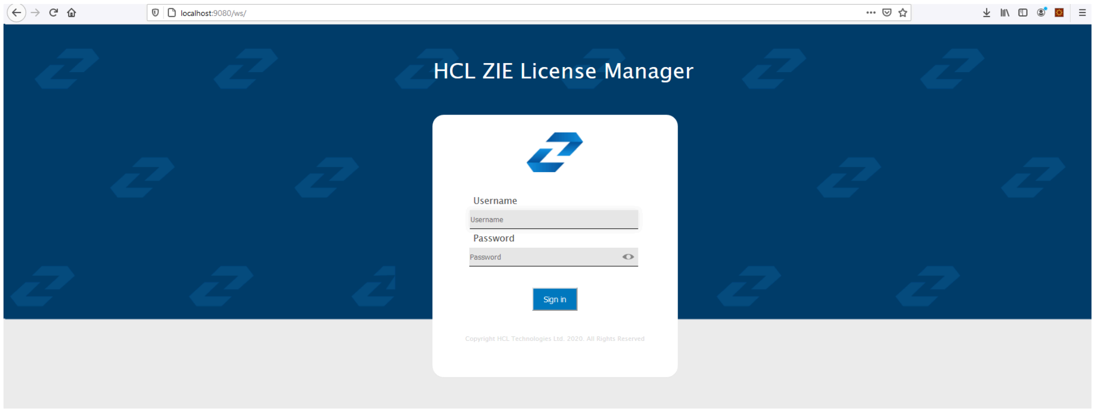
c. *Open File Explorer > This PC > Map network drive.* Use the IP address of the Web Server along with the folder where the HCL Z and I Emulator for Windows RP1.msi is located. E.g. \\192.168.56.102\MPZiewin

d. Click **Save**.

e. Repeat steps 4-5 with ZIEWEB folder. E.g. \\192.168.56.102\ZIEWEB

Note: This completes configuring Managed HCL Z and I Emulator for Windows (ZIEWIN).

f. To validate that the configuration is correct, create a new Username or use an Existing User in the Session Manager Online. *Click on Start > HCL Z and I Emulator for Windows > Start or Configure sessions - Online*



Points to consider:

- When mapping the network drive ensure that client machine and ZIEWEB server is within the same network.
- The Session Manager Online checks for updates at startup. It is essential that HCL Z and I Emulator for Windows RP1 is in the ZIEWEB published directory.

Interoperability between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients

The interoperability feature allows the ZIEWin users to use the ZIEWin sessions from other HCL terminal emulator clients, such as ZIEWeb and ZIEWeb Client. ZIEWin users can use the “Session Manager Online” utility to store the new sessions and migrate the existing sessions to ZIE server, these sessions are then converted to ZIEWeb Session formats for the ZIEWeb and ZIEWeb Client usage.

Note: Interoperability feature is introduced in ZIEWin 2.1 version.

ZIEWin client communicates with the ZIE server over HTTP/HTTPS connectivity using JSON data format.

The interoperability feature is supported from ZIEWeb v2.1.0.0 & ZIEWeb Client v2.1.0.0 onwards and is applicable for 3270 Display, 5250 Display, 3270 Printer, 5250 Printer, and VT sessions.

Note: The session conversion happens only for ZIEWin to ZIEWeb sessions and not vice versa.

When the user stores the ZIEWin sessions using the "Session Manager Online" utility, they are converted to ZIEWeb sessions before saving them to the ZIE server . After storing to the ZIE server , users can log in from ZIEWin, ZIEWeb, or ZIEWeb Client to work with the stored ZIEWin sessions.

Using ZIEWin Sessions from ZIEWeb and ZIEWeb Client:

After the ZIEWin sessions are stored in the ZIE server , if any changes are made to the session definition from any of the clients, it is saved in the ZIE server . These session changes will be available to ZIEWin users after the next login.

Below is the list of supported parameters as part of the Interoperability feature.

Table 3. List of Supported Parameters for Interoperability

ZIEWIN Parameter	ZIEWEB Parameter
Primary Host Name or IP Address	Destination Address
Primary Port Number	Destination Port
Primary LU or Pool Name	LU or Pool Name
Screen Size	Screen Size
Host Code-Page	Host Code-Page
Auto-reconnect	Auto-reconnect
Backup 1 Host Name or IP Address	Backup 1 Destination Address
Backup 2 Host Name or IP Address	Backup 2 Destination Address
Backup 1 Port Number	Backup 1 Destination Port
Backup 2 Port Number	Backup 2 Destination Port
Backup 1 LU or Pool Name	Backup 1 LU or Pool Name
Backup 2 LU or Pool Name	Backup 2 LU or Pool Name
Enable Security	Protocol
Workstation ID	Workstation ID
Server Authentication	Server Authentication
Message Queue	Message Queue
Message Library	Queue Library
Send Personal Certificate to Server if it is Requested	Send a Certificate
Send Personal Certificate Trusted by Server	Certificate Source
Send Personal Certificate Based on Key Usage	Enable Key Usage
Machine Mode	Terminal Type (VT session)

Table 3. List of Supported Parameters for Interoperability (continued)

ZIEWIN Parameter	ZIEWEB Parameter
AutoWrap	AutoWrap (VT session)

Note: Only the listed parameters will be modified from ZIEWeb / ZIEWeb Client for a ZIEWin profile. If any other parameters are updated from ZIEWeb / ZIEWeb Client, there will not be any changes to the ZIEWin session. Users should modify the ZIEWin sessions either from ZIEWin or ZIEWeb / ZIEWeb Client at a time and should avoid simultaneous modifications from different clients.

Interoperability 3.0 Configuration Introduction:

ZIEWeb v3.0 (from v2.1.0.0 onwards) introduced interoperability between ZIEWin and ZIEWeb. This allowed ZIEWin sessions to be accessed through ZIEWeb and ZIEWeb Client after the session definitions were uploaded to the ZIE server .

Password provided during the user creation will be encrypted using AES 128-bit algorithm and will be sent to the server through the HTTP/HTTPS protocol as Json object. UID is added to the WS and BCH profile files for unique identification. Only Connection parameters are considered for the interoperability between ZIEWin and ZIEWeb Clients and vice-versa.

After the ZIEWin sessions are converted and stored in the ZIE server , any changes made to the common parameters from any of the clients will be saved on the ZIE server . These parameter changes will be available to ZIEWin users after the next login.

Steps to install:

1. Install the ZIEWeb v3.0.
2. Install the ZIEWin v3.0.

WAR File Configuration:

The interoperability executable (**ZIEWeb_Interoperability.war**) is available under the lib directory of the product.

For Embedded Web Server:

If the Embedded Web Server is used, by default Interoperability application is running on context root "interop”. If the user needs to change the context root, add the following parameter to the configuration file **(*config.properties*)**, located in the ZIE server publish directory.

Example: InterOpContextPath=interop

The default ZIE server IP is 127.0.0.1 and the ZIE server port is 8999. If the user needs to connect to the ZIE server located on a different machine, then override the interoperability configuration by modifying the properties of ***“interop_overrides.xml”*** in the conf directory under the lib directory of the product.

Table 4. List of properties that can be used to configure the Interoperability

Property	Value	Description
ZIEWEB_SERVER_IP	127.0.0.1	ZIE server address

Table 4. List of properties that can be used to configure the Interoperability (continued)

Property	Value	Description
ZIEWEB_SERVER_PORT	8999	ZIEWEB Config Server port
Directory_Location	C:\\dir_location	Directory Location for logs

The user can utilize **ZIEWeb_Interoperability.war** file (available under the lib directory of the product) to deploy to different application servers such as WAS/Tomcat.

For WebSphere Application Server (WAS):

1. Log in to **WebSphere Application Server**.
2. Go to **Applications**.
3. Click WebSphere enterprise applications under **Application Type**.
4. Select **ZIEWeb_Interoperability.war file**.
5. Click on Initialize parameters for servlets link under **Web Module Properties** section.
6. Enter the required values.

Supported Application Servers: *Apache Tomcat and WAS*.

Limitations

1. Only connection parameters are considered for interoperability between ZIEWin and ZIEWeb and vice-versa.
2. Session creation from ZIEWeb / ZIEWeb Client will not be converted to a ZIEWin session.

Known Issues

1. For stored ZIEWin sessions, changes to any session parameters (not only the listed parameters) from ZIEWeb / ZIEWeb Client will be overridden or set to default when there is an update from **“Session Manager Online”** (ZIEWin Client).
2. If there are simultaneous profile updates from any of the two clients, the most recent update will be saved as the final copy in the ZIE server .
3. Modifications done in multiple sessions (add, delete sessions, or rename) from the ZIEWeb Clients do not reflect in the ZIEWin Client.
4. Saving/renaming profiles with special characters (Ex: \ / : * ? " < > |.) in ZIEWeb/ ZIEWeb Clients will result in unexpected behavior in the ZIEWin Client.

How to setup Managed HCL Z and I Emulator for Windows (ZIEWIN)

HCL Z and I Emulator for Windows uses Session Manager Online dialog to provide easy access to workstation profiles and batch files on the ZIE Server. With Session Manager Online users can create or start a single or multiple sessions and or batch files. Users can create their own profile to the ZIE server and migrate existing files such as the workstation profiles (*.WS) and batch files (*.BCH) that were stored on the ZIE Server.

This "How To" document aims to supplement additional detailed information in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN) as referenced below.

Steps to Install Using Managed:

Refer to Planning to Install Z and I Emulator for Windows on page 42

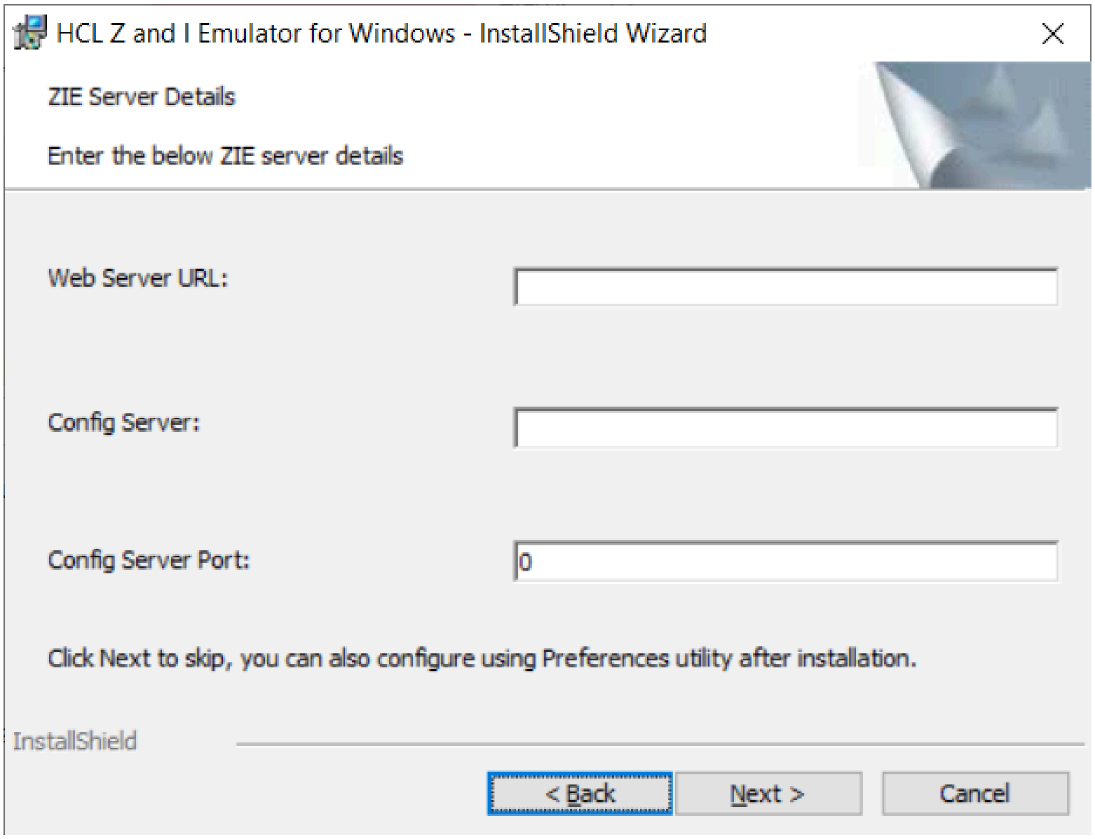
The steps provided in this document are applicable to all Windows 10 versions which are 64-bit OS level.

Prerequisites :

1. Download a copy of HCL Z and I Emulator for Windows 64-bit base package and HCL Z and I Emulator for Windows RP1.zip
2. A HCL Z and I Emulator for Web server is required for the Session Manager Online to work.
3. Create a folder (e.g. MPZiewin) in the ZIEWEB published directory.
4. Unzip and dump the HCL Z and I Emulator for Windows RP1 contents in *MPZiewin* folder.
5. Right click the *MPZiewin folder* > *Properties* > *Sharing* > *Advance Sharing* > *Put a check on Share this Folder*.
6. Click **OK** then **Close**.
7. Repeat steps 5-6 with ZIEWEB folder.
8. Obtain the IP address of the ZIEWEB server and use it at step 2 below.

Follow the steps below in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN):

1. There are two ways to input the ZIE Server configuration details. Choosing to go either way will have the same results.
   - During the ZIEWIN installation a new panel has been added.



HCL Z and I Emulator for Windows - InstallShield Wizard

ZIE Server Details

Enter the below ZIE server details

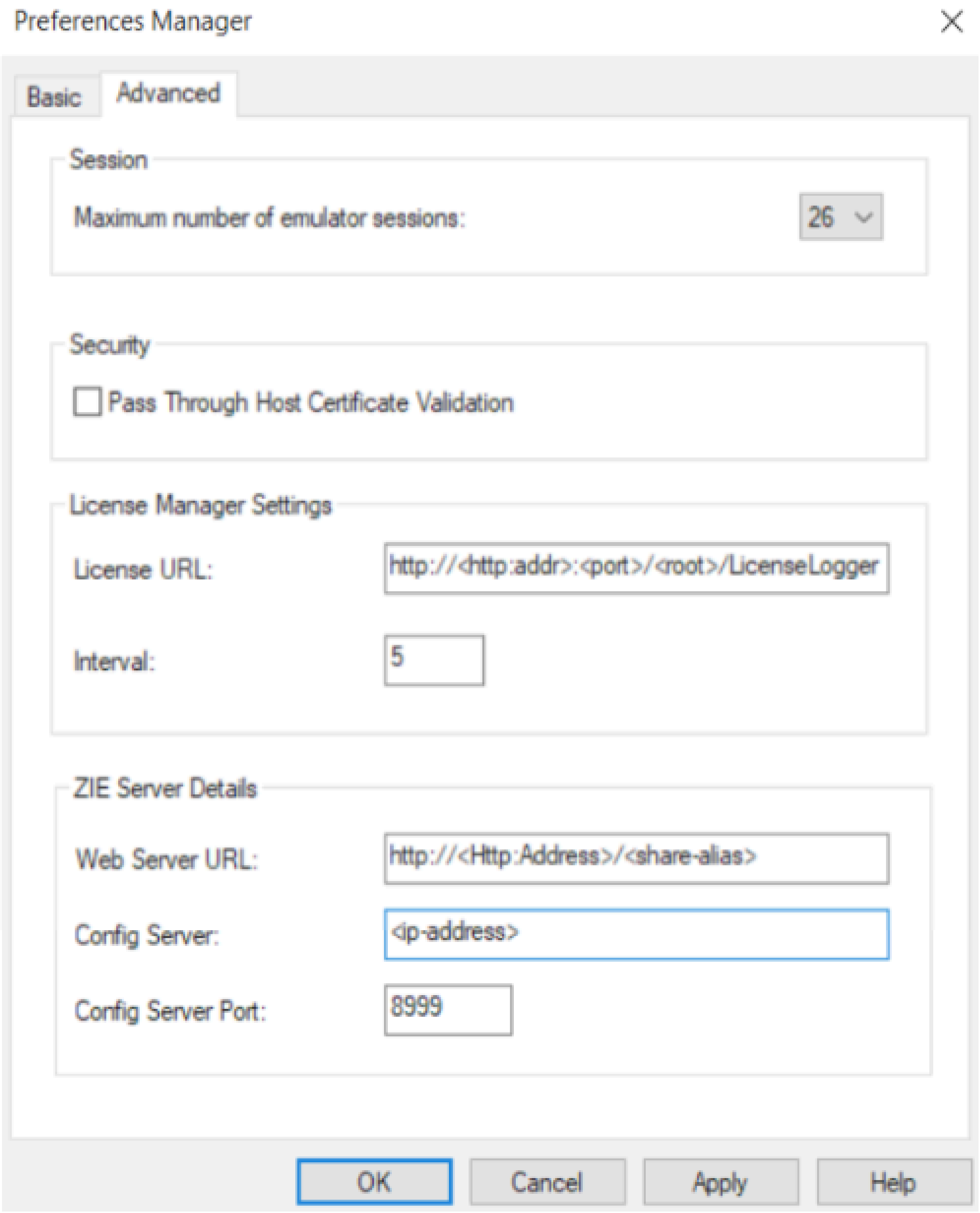
Web Server URL:

Config Server:

Config Server Port: 0

Click Next to skip, you can also configure using Preferences utility after installation.

- Preferences Manager - Click on Start > HCL Z and I Emulator for Windows > Preferences > Advanced


Preferences Manager

Session

Maximum number of emulator sessions: 26

Security

□ Pass Through Host Certificate Validation

License Manager Settings

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

ZIE Server Details

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

2. Enter the configuration parameters based on the information below:
   - **Web Server URL** : The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file will be downloaded for installation. Installer or fix pack will be installed on the system by "Start or Configure Sessions - Online" program.>
   - **Config Server** : URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the HOD Embedded Server or on any configured Application Server.

     Example: http://< Application Server IP >/<Configured context root of the application>

For more details on WAR file deployment, refer to the technote <hyper-link>.

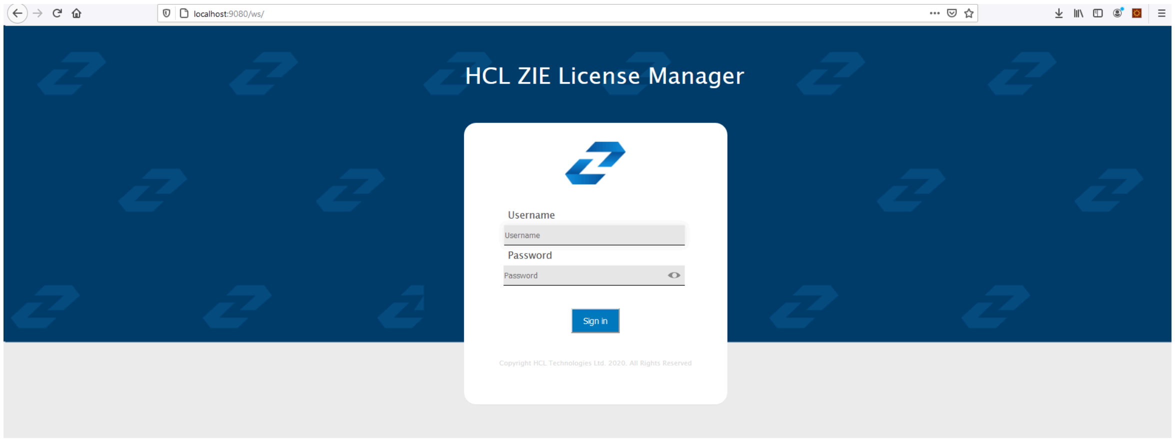
- **Config Server Port** : Port number of Application Server where interoperability module (.war file) is deployed.

  Example: 9080

3. Click **OK**.
4. *Open File Explorer* > *This PC* > *Map network drive.* Use the IP address of the Web Server along with the folder where the HCL Z and I Emulator for Windows RP1.msi is located. E.g. \\192.168.56.102\MPZiewin
5. Click **Save**.
6. Repeat steps 4-5 with ZIEWEB folder. E.g. \\192.168.56.102\ZIEWEB

Note: This completes configuring Managed HCL Z and I Emulator for Windows (ZIEWIN).

7. To validate that the configuration is correct, create a new Username or use an Existing User in the Session Manager Online. *Click on Start* > *HCL Z and I Emulator for Windows* > *Start or Configure sessions - Online*



Copyright HCL Technologies Ltd. 2020. All Rights Reserved

Points to consider:

- When mapping the network drive ensure that client machine and ZIEWEB server is within the same network.
- The Session Manager Online checks for updates at startup. It is essential that HCL Z and I Emulator for Windows RP1 is in the ZIEWEB published directory.

Introduction

Welcome to Z and I Emulator for Windows

Z and I Emulator for Windows brings the power of personal networking to your workstation by exploiting networking capabilities to provide a variety of connectivity options supporting local area network (LAN) and wide area network (WAN) environments. Whether it is for host terminal emulation, client/server applications, or connectivity, Z and I Emulator for Windows offers a robust set of communication, networking, and administrative features.

Z and I Emulator for Windows is a full-function emulator. In addition to host terminal emulation, it provides these useful features:

- File transfer
- Dynamic configuration
- An easy-to-use graphical interface
- Emulator APIs such as Emulator High-Level Language Programming Interface (EHLLAPI), Host Access Class Library (HACL), and PCSAPI. For example, EHLLAPI is often used for automated operator applications which read host screens and enter keystrokes without direct user intervention. Refer to *Emulator Programming* and *Host Access Class Library* for details.

A variety of application programming interfaces (APIs) are supported by Z and I Emulator for Windows. You can create applications that use the peer-to-peer client APIs, which are based on LU 6.2 and provided by Z and I Emulator for Windows. These APIs let you simultaneously access and process information on peer workstations.

What's New in Z and I Emulator for Windows

For information on new functions and enhancements in Z and I Emulator for Windows Version 3.0, refer to https://help.hcltechsw.com/zie/ziewin/3.0/doc/readme/readme.html.

Z and I Emulator for Windows Program Icons

When you have installed Z and I Emulator for Windows, the main functions that you can use are displayed as icons. Icons are grouped in sub-folders of the HCL Z and I Emulator for Windows program folder.

A brief explanation of each function follows:

Start or Configure Sessions

Use this icon to bring up the Session Manager. This dialog allows you to start or configure sessions. During configuration, you can specify the session type, screen size, LU number, graphics support, the type of communication link and its parameters, as well as other information. You can save all this information in a workstation profile. After saving, you can start the session by just clicking the session icon. Authorized users can also create new batch files from this dialog

Start or Configure Sessions Online

Use this icon to bring up the Session Manager Online. This dialog allows you to manage and use the online sessions available on ZIE Server. This provides option for auto install of Z and I Emulator for Windows available on ZIE Server.

Administrative and Problem Determination (PD) Aids

Information Bundler

Use this icon to gather system files and specific trace and log files, as well as registry information, such as the software installed or running on a machine. This can also be run from an active session by clicking **Actions → Launch → Information Bundler**.

Log Viewer

Use this icon to view, merge, and sort the Z and I Emulator for Windows message and trace logs. Z and I Emulator for Windows logs errors and informational messages during initialization and operation. This can also be run from an active session by selecting **Actions → Launch → Log Viewer**.

Migration Utility

Use this icon to migrate your user-class and system-class files and desktop icons to Z and I Emulator for Windows Version 3.0.

Trace Facility

Use this icon to turn trace functions on and off and to capture communication protocol information that passes between your workstation and other host systems. You can use traces to resolve communication problems. This can also be run from an active session by selecting **Actions → Launch → Trace Facility**.

Utilities

Note

These programs are provided on an as-is basis without any warranty of any kind, including the warranties of merchantability and fitness for a particular purpose which are expressly disclaimed.

iSeries Connection Configuration

Use this icon if you want to define connections to each iSeries™, eServer™ i5, or System i5™ host that will use the data transfer function.

Convert Macro

Use this icon to convert an existing Z and I Emulator for Windows macro to an XML or VBScript file.

Data Transfer (iSeries™ only)

Use this icon to transfer data between a workstation and an iSeries™, eServer™ i5, or System i5™ database.

Multiple Sessions

Use this icon to create batch files (.BCH), which can specify multiple emulator sessions (workstation profiles) or other supported Windows® programs that you want to start concurrently. You can create an icon for each batch file and start the programs just by clicking the icon.

Preferences

Use this icon if you want to set up or change the user preferences, such as changing the user interface language.

ZipPrint (3270 only)

Use this icon to start the ZipPrint program, which allows you to print host system files or screens, PROFS® notes, calendars, and documents, CMS files, and XEDIT workspaces. When started, ZipPrint adds an item to the menu bar of the session window.

FTP Client

Use this icon to start the Z and I Emulator for Windows FTP client application, which allows files and directories upload and download, and directory navigation of local and remote file systems running FTP servers.

Z and I Emulator for Windows Sessions

The sessions that Z and I Emulator for Windows provides are logical connections enabling communication between your workstation and a host system. The following session types are available:

Display session

Use your workstation as a display terminal connected to the host system.

Printer session

Use your workstation printer as a host system printer.

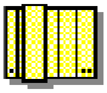
Client/server session

Establish connections that allow peer communications using CPI-C and APPC (LU 6.2).

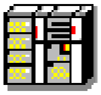
Z and I Emulator for Windows Connections

Z and I Emulator for Windows supports a variety of connections to the following host systems. Following are the icons you will encounter when you begin to configure an emulator session:

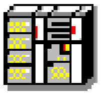
zSeries™



iSeries™

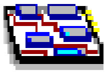
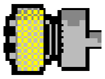


ASCII



zSeries Emulator Connections

Table 5. zSeries Emulator Connection Icons

Interface	Attachment
LAN 	Telnet3270 VT-over-Telnet (TCP/IP)
COM port 	Telnet 3270 VT over Telnet (TCP/IP)

iSeries Emulator Connections

Table 6. iSeries Emulator Connection Icons

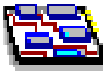
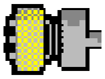
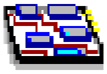
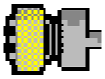
Interface	Attachment
LAN 	Telnet5250 over TCP/IP VT over Telnet
COM port	

Table 6. iSeries Emulator Connection Icons (continued)

Interface	Attachment
	

ASCII Emulator Connections (SBCS only)

Table 7. ASCII Emulator Connection Icons

Interface	Attachment
LAN 	VT over Telnet (TCP/IP)
COM port 	VT over Telnet (TCP/IP)

Planning to Install Z and I Emulator for Windows

Z and I Emulator for Windows supports a wide range of workstations. There are hardware and software requirements, as well as memory and storage requirements, to consider when planning the installation of Z and I Emulator for Windows.

The following sections describe and list support for monitors, adapters, and keyboards.

For detailed instructions on installing Z and I Emulator for Windows, refer to *Installation Guide*.

For instructions to install the HCL ZIE License Manager, refer to Installation of HCL ZIE License Manager, and see the topic Configuration of License Manager settings on page 121 to configure the ZIE License Manager for ZIEWin.

Workstation Hardware

Z and I Emulator for Windows supports workstations with the following hardware:

Table 8. Workstation Hardware Support

System units	The recommended system unit has an Intel Pentium® microprocessor and access to a DVD-ROM drive. A minimum of 180 MB of fixed drive space is required.
Display monitors	All VGA resolution or above display monitors supported by Windows®.
Video adapters	All VGA resolution or above video adapters supported by Windows®.
Keyboards	• Enhanced keyboard (101-key, 102-key, 104-key) • Space-saving keyboard • Microsoft® Natural keyboard
Printers	All printers supported by Windows® when a PDT file is not used. For more details about printers supported in PDT mode, refer to *Emulator User's Reference*.
Communication adapters	LAN, SDLC, COM Port, OEM, and Multiprotocol communication adapters.
Modems	All asynchronous modems that use the Hayes AT® command set and are supported by Windows®. Synchronous (SDLC) modems attached to a multiprotocol adapter (MPA) or SDLC adapter.

Workstation Memory Requirements

For Z and I Emulator for Windows the amount of memory you need depends on several factors, including the operating system you are running on, the attachment type, number of sessions, and the use of programming interfaces such as Emulator High-Level Language Application Programming Interface (EHLLAPI) and Dynamic Data Exchange (DDE).

Host Requirements

Refer to *Emulator User's Reference* for information about hardware requirements for host systems.

For all supported Windows operating systems, Java Runtime Environment 1.8 is installed.

Application Data

Application Data is generally defined to be files that contain user preferences or configuration information; an application may need some or all of these files to run properly. Z and I Emulator for Windows uses multiple configuration files: *User Class* files can be stored individually by user profile, while *System Class* files are stored in a common location.

Table 9: Application Data File Types on page 44 lists the classifications of some of the most common Z and I Emulator for Windows file extensions.

Table 9. Application Data File Types

User Class		System Class	
Extension	**File Type**	**Extension**	**File Type**
.ws	Workstation Profile	.mlg	Default Message Log
.bch	Multiple Sessions	.trc	Unformatted Trace
.ini	Session Size and Location	.tlg	Formatted Trace
.pmp	Popup Keypad Configuration	.cfg	FTP Client Configuration
.kmp	Keyboard Configuration	.pub	Client/Host public key
.srl	File Transfer List	.dat	FTP Client data file
.ndc	iSeries™ Connection Configuration		
.tto	iSeries™ Data Transfer Request (Receive)		
.tfr	iSeries™ Data Transfer Request (Send)		
.bar	Toolbar Setup		
.mac	Macro		
.mmp	Mouse Setup		
.xlt	Translation Table		
.cert	Certificate		
.der	Binary DER		

Application Data Locations

The location for Application Data is specified during installation of Z and I Emulator for Windows. The following tables list the default Application Data locations, based on operating system.

If the [UserProfile]\Application Data location was selected at installation, the following profile paths are used:

Operating System	User-Class Directory (Current User)[1]	System-Class Directory
Windows 7, Windows 8/8.1, Windows 10, Windows Server 2008, Windows Server 2012	C:\Users\%USERNAME%\AppData\Roaming\HCL\Z and I Emulator for Windows	C:\ProgramData\HCL\Z and I Emulator for Windows

[1] The FTP Client configuration files are stored in the profile path mentioned above, under the `FTP` folder.

If the All Users\Application Data location was selected at installation, the following profile paths are used:

Operating System	User-Class Directory (Current User)[1]	System-Class Directory
Windows 7, Windows 8/8.1, Windows 10, Windows Server 2008, Windows Server 2012	C:\ProgramData\HCL\Z and I Emulator for Windows	C:\ProgramData\HCL\Z and I Emulator for Windows

[1] The FTP Client configuration files are stored in the profile path mentioned above, under the `FTP` folder.

ZIEWin Trial Version

The ZIEWin Trial version package allows users to try and evaluate the product for up to 30 days. Trial version licenses can be obtained to use this version. Alternatively, users can also use the "Disable License" option to to try the ZIEWIN without licensing.

The ZIEWin trial version is supported in both English and Japanese languages.

To request a free trial version, users can visit the following link:

https://www.hcltechsw.com/mainframe-solutions/mainframe-solutions-free-trial?referrer=help.hcltechsw.com

Using Z and I Emulator for Windows

Configuring Sessions

Z and I Emulator for Windows saves emulator configuration information to a workstation profile (.WS). Depending on your Z and I Emulator for Windows configuration, you might have a workstation profile only or both a workstation profile and a configuration file. The workstation profile can be used later by other Z and I Emulator for Windows sessions, or to restart this session.

LDAP

You can have an icon created for each workstation profile. Then you can select the session icon to establish communication with the host system using the saved workstation profile.

Configuring for iSeries™, eServer™ i5, or System i5™

> To connect to an iSeries™, eServer™ i5, or System i5™, specific configuration information in the workstation profile must correspond to the information specified at the iSeries™, eServer™ i5, or System i5™ system. Refer to the iSeries™, eServer™ i5, or System i5™ configuration examples in *Emulator User's Reference* for more information about creating display, line, and controller descriptions on the iSeries™, eServer™ i5, or System i5™ system.

If you want to configure multiple links, refer to *Administrator's Guide and Reference*.

Creating a Configuration

To create a new session, use the following procedure:

1. From the Start menu, click **Programs → HCL Z and I Emulator for Windows → Start or Configure Sessions**.
2. From the Session Manager dialog, click **New Session**.

   The Customize Communication window appears.
3. Select the type of host from the **Type of Host** drop-down list box.
4. Select the interface you will use from the **Interface** drop-down list box.
5. Select the attachment type you want to use from the **Attachment** drop-down list box.
6. Click **Session Parameters** to modify the session type (display or printer), host code page, and display/graphics options.

   The Session Parameters – 3270, 5250, or ASCII – Host window appears (depending on the host you selected in step 3 on page 46). Click **OK**.
7. Click **Link Parameters**.

   Enter the appropriate information for each page and click **Next** to continue. Click **Finish** when you are done.

   Based on the attachment type that you have chosen, make your selections for the parameters in the window displayed. Click **Help** or press **F1** to display parameter details. Click **OK** when you are finished.

   Note: If your host is configured to support Secure Sockets Layer (SSL) or Transport Security Layer (TLS), then click the **Security Setup** tab. Refer to *Administrator's Guide and Reference* for details on configuring session security.
8. Click the **Host Definition** tab to configure the **Connection Options**.
   - Select **Auto-reconnect** to reestablish an interrupted connection.
   - The **Connection Timeout** value tells Z and I Emulator for Windows how long it should wait for connection to the host.
   - The **Try connecting to last configured host infinitely** option is enabled by default. Clear this box if you do not want Z and I Emulator for Windows to automatically and indefinitely wait for acknowledgement for a connect request from the last correctly configured server/host.
   - Select **Telnet keep alive** to send Telnet Keep Alive commands to the host.
   - There are two keep alive mechanisms supported: NOP and TIMING-MARK. The Keep Alive Timeout value specifies the interval between the keep-alive requests in seconds. The range of values is 30 to 99999 seconds.
   - Select **Bypass signon using Kerberos principal** to enable Kerberos authentication. A ticket is generated and passed to the iSeries™, eServer™ i5, or System i5™ host during TN5250 negotiation. This option is only available for 5250 sessions.

> **Note:** You must log into a Windows domain in order to use Kerberos authentication. Refer to the relevant Microsoft documentation for specific details. Refer to *Administrator's Guide and Reference* for details about express logon functions.

- Select **Bypass signon using Password substitute** to enable the user to bypass the iSeries login screen by sending a SHA1 password substitute.

9. To set up printer association, click the **Printer Association** tab and do the following:
    a. Select **Associated Printer Session**.
    b. Enter the .WS file for the printer that is to be associated with the specific terminal. You can also click **Browse** to locate the file.

    You can also set the following options:

    - Select the **Start Associated Printer Minimized** check box, if preferred. This option is unavailable until an associated printer is selected.
    - Select the **Automatically close Associated Printer Session with this session** check box, if preferred. This option is unavailable until an associated printer is selected.
    - Select **Associated device name** to associate the display session with any printer device that currently exists on an iSeries™, eServer™ i5, or System i5™ host. This option is only available for 5250 sessions.
10. After configuring the session options, click **OK** in the Telnet tab panel.
11. Click **OK** on the Customize Communication window. The session is displayed automatically.

Save the workstation profile as described in Saving a Workstation Profile on page 48.

Creating an FTP configuration

1. Open the FTP client from Start Menu by clicking **Start -> HCL Z and I Emulator for Windows -> Utilities -> FTP Client**.
2. Click **Communication -> Configuration** from the menu.
3. Choose the options in the **Connect** tab.
    - Enter the host name or IP address of the FTP server.
    - Set the required options from the **Connection** group box if the default values are to be changed.
4. Click the **Logon** tab.
    - Enter the user name and password.
    - Enter the Remote/Local Home Directory values so that once the connection is established, the client will list these directories you specified.
5. Click the **File Transfer** tab and select the appropriate choice for the Transfer mode from the drop-down list. Browse and select a Transfer List File if any.
6. Click the **SSL** tab to specify the security parameters for Secure FTP. SSL is optional and not enabled by default.
    - Check **Enable Security** to enable the SSL security.
    - Select the Security provider for the connection.

  - Enter the Channel Security parameters for the connection.
  - Select the Client Authentication method from the two options.
7. Click the **Runtime Preferences** tab.
  - Enter the action to be taken if target file exists during a file transfer.
  - Startup Commands enables you to provide a comma-separated list of FTP Command(s) to be executed after a successful connection.
  - Pass Through Host Certificate Validation allows the FTP client to finish a successful handshake bypassing the server certificate validation.
8. Click **OK** to close the configuration dialog.
9. Connect to the host by clicking **Communication->Connect** or the **Connect** button.

Environment variables in workstation profile

In the .WS profile, you can specify environment variables for the paths of the following Z and I Emulator for Windows files:

- Bar files (.bar)
- Popup keypad files (.pmp)
- Mouse customization files (.mmp)
- Keyboard map files (.kmp)

The syntax is as follows:

```
[Toolbar]
BarFile=C:\%USERDIR%\disp.bar
```

```
[Poppad]
DefaultPoppad=C:\%PROFILEDIR%\test.pmp
```

```
[Mouse]
DefaultMouse=%ZIEWinPROFILE%\vtpf.mmp
```

```
[Keyboard}
DefaultKeyboard=C:\%USERFOLDER%UserMap.kmp
```

In the above examples, USERDIR, PROFILEDIR, ZIEWinPROFILE, and USERFOLDER are environment variables that you specify. See the following example:

```
USERDIR = profile\toolbarfiles
```

Saving Configuration Information

This section describes how to save configuration information. The emulator workstation profile, and FTP Client configuration information is stored in a .WS, and .CFG file respectively.

Saving a Workstation Profile

If you save your emulator configuration information, the session will have the same characteristics the next time you start it. If you have an icon added to the Z and I Emulator for Windows folder, you can restart the session with the saved configuration information by clicking this icon from the Start menu. You are automatically given the opportunity to save your session information when you close a session. However, if you want to save the information at any time, use the following procedure:

1. Select **Save** from the File menu in the session window.

   The **Save WorkStation Profile As** window appears.
2. Type a file name (.WS) and then click **OK**. The name you enter will become the icon title unless you enter a description. Note that you can choose the directory where this file is saved, but the default directory is the application data directory specified during installation.
3. An icon associated with the profile will appear in the Session Manager.

Saving an FTP Client Configuration

You can save the FTP Client configuration in two ways:

- When exiting the FTP Client after you create or change a configuration in the FTP Client Session Configuration panel, you are prompted to select whether to save changes, exit, or continue to work in the application. If you select to save changes, type the file name and click **Save**. The default file type is .CFG and the default directory is the application data directory specified during installation.
- You can save configuration changes by selecting **File->Save or File->Save As**.

Changing Configuration Information

You can change all of the configuration parameters in the workstation profile.

Changing a Workstation Profile

To change a workstation profile, use the following procedure:

1. If your session window is not active, select the icon corresponding to the workstation profile to be changed.

   The session window appears.
2. Select **Configure** from the Communication menu.

   The subsequent steps are the same as for creating a new configuration, beginning with step 3 on page 46.
3. After you have made your changes, the following message appears:

```
Because you have changed the configuration,
communication will be terminated if you
proceed. Are you sure?
```

If you click **OK**, communication ends, but then you are reconnected using the new configuration information.

To save the changes in your workstation profile, click **Save** from the File menu in the session window and then click **Yes** to replace the existing file. Otherwise, click **No** to save this information in a new configuration file.

Tip

Unless you have deselected **Save on Exit** in the Exit Options dialog by selecting **Exit** from the **Settings** menu, changes are saved in the workstation profile automatically whenever you exit a session.

Starting and Stopping Emulator Sessions

This chapter describes how to start and stop single and multiple emulator sessions.

If you saved your emulator configuration information, as described in Saving Configuration Information on page 48, it is stored in a workstation profile (*.WS), which should be displayed in the Session Manager.

Note: If you are a first-time user of Z and I Emulator for Windows, or there are no session icons in the Session Manager, see Configuring Sessions on page 45 to create a configuration.

Session Manager

Z and I Emulator for Windows uses the Session Manager dialog to provide easy access to workstation profiles and batch files. You can use the Session Manager to start a single or multiple sessions, and create a new session or batch file.

Note: The Session Manager dialog only displays workstation profiles and batch files that are located in the Application Data location that was specified during Z and I Emulator for Windows installation. See Application Data on page 43 for more information about application data.

You can drag an icon from the Session Manager to the Windows® Start menu or to the desktop. Select one or more sessions and drag with the right mouse button. A contextual menu appears when the icons are dropped, which gives you the options for moving, copying, or creating a shortcut. If you drag an icon with the left mouse button held down, the icon is moved to that location. If you drag an icon while pressing the Ctrl key and with the left mouse button held down, the icon is copied to that location. If you drag an icon while pressing the Alt key and with the left mouse button held down, a shortcut is created for the icon.

You should use the shortcut option whenever possible. Moving and copying will affect the location (and therefore the function) of the profile. Specifically, when you copy a profile to the desktop instead of creating a shortcut, you have actually created another profile. Any changes you make to the desktop profile will not be reflected in the original profile (and vice versa). Also, the desktop profile is located in the desktop folder (not in the application data folder) and will not appear in the Session Manager—the original version of the profile remains in the Session Manager.

Session Manager Options

Various Session Manager options are available from the pull-down menus and the right-click menu. For example, you can customize the look of the displayed Session Manager information and import sessions or batch files into the Z and I Emulator for Windows Application Data directory.

Session Manager Menu

The following options are available from the Session Manager menu.

File

Change Directory

The user run files that are stored in a directory other than the Z and I Emulator for Windows Application Data directory.

Import

This option allows the user to copy sessions or batch files to the Z and I Emulator for Windows Application Data directory. Afterwards, the imported files is displayed in the Session Manager dialog.

View

Sessions

This option shows all valid workstation profiles that have the standard .WS extension and are located in the Application Data directory.

Multiple Sessions

This option shows all valid batch files that are have the standard .BCH extension and are located in the Application Data directory.

All File Extensions

This option shows all valid multiple sessions and workstation profiles that are located in the Application Data directory, regardless of extension.

Hidden

This option shows files that have been previously hidden using the right-click menu option. If this option is selected, hidden sessions are shown with black-and-white icons; otherwise, they will not be displayed.

Large Icons

This option shows large session icons in the Session Manager.

Small Icons

This option shows small session icons in the Session Manager.

Details

The following session detail information is displayed in the panel columns. The columns can be resized as needed.

- File Name
- File extension
- Type (session or batch file)
- Description (shows the information specified in the `Description=` field in the .WS file)
- The following session information is not shown for batch files.
  - Host Name
  - Host Type (shows the host type specified in the Customize Communication dialog during session configuration)
  - Interface (shows the interface specified in the Customize Communication dialog during session configuration)
  - Attachment (shows the attachment specified in the Customize Communication dialog during session configuration)
  - Session Type (printer or display)
- Modified (indicates the last modification date/time of the file)

Refresh

If the user manually copy a session or batch file into the Application Data directory, the user must refresh the Session Manager view in order to see the new files.

Package

Upgrade

The users can Upgrade to the latest RefreshPack via Session Manager (Online/Offline).

Rollback

The users can do the Rollback of installed RefreshPack via Session Manager (Online/Offline).

Detect and Repair

Detect and Repair might be automatically initiated if the Z and I Emulator for Windows installation is corrupted. This function uses Windows Installer to repair damage to the installed product. The user may be prompted for the installation source or image. User must be authorized in the System Policy to use this option. See Detect and Repair for more information about this feature.

Right-Click Menu (Contextual)

The following options are available by right-clicking on one or more session.

Start

Starts the selected sessions

Delete

Deletes the selected sessions. You must have permission in the System Policy to delete any sessions from the Session Manager.

Hide/Unhide

You can hide or unhide sessions using this option. To view hidden sessions, you must select **View → Hidden**. Hidden sessions have black-and-white icons when displayed.

Modify

This option is only available when selecting one or more batch files—the batch files are brought up in edit mode. You must have permission in the System Policy to modify batch files from the Session Manager.

Session Manager Online

ZIEWin uses the ZIE service manager (referred to, as 'ZIE Server' going forward) to provide auto-upgrade capability and centralized management of configuration files.

The Session Manager Online notifies users if any new version of Z and I Emulator for Windows is available online for update on the configured Web Server.

After confirmation from the User, Z and I Emulator for Windows will be auto-upgraded to newer versions. Z and I Emulator for Windows uses the Session Manager Online dialog to provide easy access to connect to the ZIE Server and work with workstation profiles and batch files online. Users can configure the ZIE Server and work with their User Profiles in the Online mode, but by default, are not allowed to create new user entries on the server, unless permitted to do so, by a system administrator. To enable users to be able to create user entries, a ZIEWeb (Z and I Emulator for Web) administrator must enable the *"Allow users to create accounts*" check box on the ZIE for Web Server, using the ZIEWeb (Z and I Emulator for Web) Administration Console. The new users that are created are added to the "ZIEWin" group on the ZIE Server.

The ZIE Server can either be configured by providing the required server details in the 'InstallShield Wizard' at the time of ZIEWin installation itself, or can be added/updated in the 'ZIE Server Detail' section of the Advanced tab within the Preferences. Users can migrate the Workstation profiles and batch files from Application Data location, upon login to the ZIE Server. If the User chooses to migrate, the Workstation profiles, batch files and their dependent files available on "Application Data" location get migrated

List of the supported files for online option are below:

- **.ws** (Workstation profile)
- **.bch -** (Multiple Sessions or Batch)
- **.pmp** (Popup-Keypad Configuration)
- **.kmp** (Keyboard Configuration)

- **.bar** (Toolbar Setup)
- **.mmp** (Mouse Setup)
- **.xlt** (Translation Table)
- **.cmp** (Color mapping configuration file)

Multiple instances of the Session Manager (offline) can be invoked at a time, but for Session Manager Online, only one instance can be invoked at a time. An instance of Session Manager Online and Session Manager (offline) can both be operated simultaneously.

First time configuration of the Online Session Manager :

When you select **Start or Configure Sessions - Online** for the first time from the Windows Start menu, the 'Z and I Emulator for Windows: Online' panel is displayed with the following tabs :

- **Create User :**

  The 'Create User' window containing the fields, *'User Name'*, *'Password'* and *'Confirm Password'* is displayed to a user who launches the 'Z and I Emulator for Windows: Online' for the first time, to create a new User entry on the Server. (The New User entry will be created only if the entered User name and Password matches the required criteria. For more information about the criteria for valid user credentials, see the 'Quick Beginnings' book by selecting it from the **Help** menu.) You can also click on 'Existing User' on the Create User page to use an existing ID on the server, instead of creating a new User entry.

  Note: After the successful creation of a user on the server and after migrating the user's profiles to the server successfully, the 'Create User' tab will not appear again and the user will be directly taken to the 'Login' page, upon launching the 'Z and I Emulator for Windows: Online' subsequently.

- **Login :**

  After creating a user, Login window appears with username and password fields for user to login. If user wants to create a new user, user can go back to 'Create User' window by clicking on create-user link. User can also change the password by checking the box of *'Change Password'*, while logging in. Once the login is successful, user is directed to profile migration window.

- **Profile Migration :**

  After a successful Login using an existing User Profile stored on the ZIE server, or upon the first time creation of a new User profile and connecting to the ZIE server, the user is taken to the 'Profile Migration' tab which gives the option to migrate (or upload) existing local user profiles (offline profiles) from the particular user's client machine to the ZIE server. The profiles that are migrated will be listed on the ZIE server as a server copy and can be retrieved for use thereafter, even if the offline copy of the profile is deleted from the client machine.

Session Manager - Online LOGIN Screen :

The **Login** Screen is the default page that opens whenever the Online Session Manager is launched, if a User is already created on the server and the Profile migration is also completed successfully. A user can login to the Online session manager using any of the User Profiles that were created on the ZIE server. The user can also change the password by clicking on the check box for *'Change Password'* on the Login page. If the user name or password entered on the Login page are incorrect, the user gets prompted to either 'Re-try with different credentials' to Login again, or, to 'Run offline', i.e, to close the Online Session Manager and launch the Offline Session Manager instead. After successful logon, profiles will be retrieved from the ZIE server, and can be viewed in the sessions list.

Interoperability between ZIEWin and ZIEWeb clients:

ZIEWin v2.1introduced interoperability between ZIEWin and ZIEWeb (Host On-Demand) clients. This allows ZIEWin sessions to be accessed through ZIEWeb and ZIEweb - Web Client products after the session definitions were migrated (uploaded) to the ZIE server.

From v2.1 onwards, ZIEWin client communicates with the ZIE server over HTTP/HTTPS connectivity using JSON data.

For Managed ZIEWin users configured with ZIE server of version lower than v2.0, a migration utility has been provided along with ZIEWeb v2.0 server component, to migrate the ZIEWin session definitions stored on the ZIE server to the new format. The migration is a pre-requisite for those who plan to migrate their ZIEWeb server to v2.0 and ZIEWin client to v2.0 or higher.

Password provided during the user creation will be encrypted using AES 128-bit algorithm and sent to the server through the HTTP/HTTPS protocol as a JSON object. UID is added to the WS and BCH profile files for the unique identification. Only connection parameters are considered for the interoperability between ZIEWin and ZIEWeb clients and vice-versa.

After the ZIEWin sessions are converted and stored in the ZIE server, any changes made to the common parameters from any of the clients will be saved on the ZIE server. These parameter changes will be available to ZIEWinusers after the next login.

For more information on ZIEWeb - ZIEWin interoperability and common parameters, refer to Interoperability between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients on page 30

Session Manager Online Options :

- **Start**

  Select one or more sessions or batch files in the Online mode, and click this button to start.
- **New Session**

  Click this button to bring up the **Customize Communication** panel in the Online mode.
- **New Multiple Sessions**

  Click this button to bring up the **Create/Modify Batch File** pane in the Online mode. You must be authorized in the System Policy to use this option.
- **Logout**

  Click on this button to Logout from the Online Session Manager. It will prompt for a confirmation. When User confirms the logout, all the active online sessions will be ended automatically without any more confirmation for exit or Save. Session Manager Online's login screen will be displayed.

File menu options

- **Upload**

  This option allows user to select and upload sessions or batch files to the ZIE server. Upload of user profile file will also upload the dependent configuration files (like a .kmp file when associated with a .ws file etc) available in the same directory.
- **Download**

  This option allows users to Download the profile files or batch files from the ZIE Server to the selected directory. Download of user profile file will also download the dependent configuration files (like a .kmp file when associated with a .ws file etc).
- **Exit**

  Clicking on Exit, will prompt for the confirmation, if user confirms the exit, the active online sessions will ended and the application is closed.

View menu options

- **Sessions**

  This option shows all valid workstation profiles that have the standard .WS extension and are available in the ZIE Server.
- **Multiple Sessions**

  This option shows all valid batch files that have the standard .BCH extension and are available in the ZIE Server.
- **Large Icons**

  This option shows large session icons in the Session Manager.
- **Small Icons**

  This option shows small session icons in the Session Manager.
- **Details**
  The following session detail information is displayed in the panel columns. The columns can be resized as needed.
  - **File Name**
  - **File Extension**
  - **Type**

    Session or Batch file
  - **Description**

    Shows the information specified in the **Description=** field in the .WS file
  - **Host Name** (not shown for batch files)

    Shows the host name for a Telnet session. If the session is not Telnet, other relevant information is shown (for example, the .ACG file name for SNA).
  - **Host Type** (not shown for batch files)

    Shows the host type specified in the Customize Communication dialog during session configuration
  - **Interface** (not shown for batch files)

    Shows the interface specified in the Customize Communication dialog during session configuration
    - **Attachment** (not shown for batch files)

      Shows the attachment specified in the Customize Communication dialog during session configuration
    - **Session Type** (not shown for batch files)

      Printer or Display
- **All File Extensions**

  This option shows all valid multiple sessions and workstation profiles that are available in the ZIE Server, regardless of extension.
- **Hidden**

  This option shows files that have been previously hidden using the right-click menu option. If this option is selected, hidden sessions are shown with black-and-white icons; otherwise, they will not be displayed.
- **Refresh**

  This option refreshes the Session Manager view.

Detect and Repair

Detect and Repair might be automatically initiated if the Z and I Emulator for Windows installation has been corrupted. This function uses Windows Installer to repair damage to the installed product; you may be prompted for the installation source or image.

You must be authorized in the System Policy to use this option. See Detect and Repair on page 104 for more information about this feature.

Right-click menu options (contextual)

- **Start**

  Starts the selected sessions
- **Delete**

  Deletes the selected sessions. You must have permission in the System Policy to delete any sessions from the Session Manager.
- **Hide/Unhide**

  You can hide or unhide sessions using this option. To view hidden sessions, you must select **View > Hidden**. Hidden sessions have black-and-white icons when displayed.
- **Modify**

  This option is only available when selecting one or more batch files; the batch files are brought up in edit mode. You must have permission in the System Policy to modify batch files from the Session Manager.

Icon drag-and-drop options

You can drag an icon from the Session Manager to the Windows Start menu or to the desktop. Select the session(s) and drag with the right mouse button. A contextual menu appears when the icons are dropped, which gives you the options for moving, copying, or creating a shortcut.

If you drag an icon with the left mouse button held down, the icon is moved to that location. If you drag an icon while pressing the CTRL key and with the left mouse button held down, the icon is copied to that location. If you drag an icon while pressing the ALT key and with the left mouse button held down, a shortcut is created for the icon.

You should use the shortcut option whenever possible. Moving and copying will affect the location (and therefore the function) of the profile. Specifically, when you copy a profile to the desktop instead of creating a shortcut, you have actually created another profile. Any changes you make to the desktop profile will not be reflected in the original profile (and vice versa). Also, the desktop profile is located in the desktop folder (not in the application data folder) and will not appear in the Session Manager (the original version of the profile remains in the Session Manager).

Installer Enhancements

ZIEWin Installer panel

As part of Managed HCL Z and I Emulator for Windows (MZIEWin) feature, a panel is available in HCL Z and I Emulator for Windows Installer, where a user provide the ZIE Server configuration details in the installation panel.

Here are the configuration parameters,

1. **Web Server Details :** The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file is downloaded for installation. Installer or fix pack is installed on the system by "Start or Configure Sessions - Online" program.
2. **Config Server :** URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the Embedded Server or on any configured Application Server.

   Example: http://< IP >/<Configured context root of the application>
3. **Config Server Port :** Port number of Application Server where interoperability module (.war file) is deployed.

   Example: 9080.

The ZIE server configuration is optional only, user can click **Next** to skip the configuration, and can configure this through "Preferences" utility post installation. *For more details regarding the Preferences, see* Preferences on page 119.

Auto-Update of HCL Z and I Emulator for Windows (ZIEWin):

ZIEWin supports automatic upgrade. ZIE administrators manage the upgradation of ZIEWin clients by placing the upgrade configuration file in the Web Server, which has the information of recommended fix-packs or refresh packs that are available on the Web Server. The Web Sever URL can be provided during installation or can be configured via the "Preferences" utility.

When a User invokes the "Start or Configure Sessions - Online", the application checks if the installed version of ZIEWin is lower than the recommended version. If the ZIEWin installed on the system is of a lower version, the User gets a notification of the latest available ZIEWin version. The User can either choose to upgrade or decline the upgrade option.

Note: The upgrade configuration file is unique for every refresh pack installer and is shipped along with the Fixpack package.

For details regarding the Managed ZIEWin Configuration parameters and changes in Preferences, see ZIE Server Details on page 123.

Starting Sessions

You can use the following methods to start sessions:

- Select a previously configured session icon from the Session Manager.
- Start from an existing session window.
- Specify a workstation profile name in the Run window.
- Enter the PCOMSTRT command in the Run window or MS-DOS prompt.
- Select an icon that has been previously dragged from the Session Manager.
- Start multiple sessions with a batch file.

Note: Connection status messages are displayed on a status bar at the bottom of your session window during connection to the host.

Starting from the Start or Configure Sessions Icon

Select **Programs → HCL Z and I Emulator for Windows → Start or Configure Sessions** from the Start menu. Select the desired session from the Session Manager dialog and click the **Start** button.

Starting from an Existing Session Window

Use the following methods to start from an existing session window:

Starting Another Session Using the Same Profile

Select **Run the Same** from the **File** menu. Another session starts, using the same profile.

Starting Another Session Using a Different Profile

1. Select **Run Other** from the File menu.

   The Open Other Workstation Profile window appears.
2. Double-click the desired workstation profile in the **File Name** list.
3. Select **OK**.

   Another session starts, using the profile specified in step 2 on page 59.

Starting a Different Type of Session from a Session Window

1. Select **Open** from the File menu.
2. Specify the desired workstation profile and then select **OK**.

   The current session ends and then another session starts, using the selected profile.

Starting Using a Command

To start a session, use the following procedure:

1. Start a DOS command prompt.
2. Enter the command

   ```
   PCOMSTRT /P=x:\AppData\my.WS
   ```

   where *my.WS* is the workstation profile stored in the Application Data directory specified during installation. This is the only required parameter.

   Note: If multiple /p parameters are given, PCOMSTRT only uses the last one to start a profile (.WS file).

   For a complete description of parameters, refer to *Administrator's Guide and Reference*.

Another method for invoking Z and I Emulator for Windows using a command is with the command for the PCSWS.EXE module (see Command line options for PCSWS.EXE on page 60).

Starting Multiple Sessions

If you installed the Multiple Sessions utility, you can use the batch program PCSWS.EXE, which runs batch files (*.BCH), to start two or more workstation profiles at the same time. Z and I Emulator for Windows batch files can also start other programs when you include their startup commands. This is especially useful if you always want to start an application when you start a session. For example, you might want to start an application, such as ZipPrint, that uses a Z and I Emulator for Windows API.

Note: You must have permission in the System Policy in order to create a new batch file.

If you created an icon for your batch file, double-click the icon in the Session Manager or select the icon and click the **Start** button.

Command line options for PCSWS.EXE

You can use the following options when creating or modifying a batch file.

- To specify which view should be used during a session, add the command **`/V=myview`**, where `myview` is the name of the previously saved view:

  ```
  C:\ZIEWin\PCSWS.EXE  C:\AppData\LAN1.WS  /V=myview
  ```

  If the specified view does not exist, the command is ignored. See Managing Emulator Sessions on page 102 for information on how to save a view.
- To suppress the HCL logo when you start one or more sessions, add the parameter **`/Q`** to the first command in the batch file:

  ```
  C:\ZIEWin\PCSWS.EXE C:\AppData\TCPIP1.WS  /Q
  ```

  where *`C:\ZIEWin\`* is the directory where you have Z and I Emulator for Windows installed, and *C:\AppData\* is the Application Data directory.
- To start a session as an icon, not as a window, add the parameter **`/I`** to the command in the batch file:

  ```
   C:\ZIEWin\PCSWS.EXE C:\AppData\LAN1.WS  /I
  ```

  where *`C:\ZIEWin\`* is the directory where you have Z and I Emulator for Windows installed, and *C:\AppData\* is the Application Data directory.
- To start a *hidden* session, not as an icon or a window, add the parameter **`/H`** to the command in the batch file:

  ```
   C:\ZIEWin\PCSWS.EXE C:\AppData\LAN1.WS  /H
  ```

  where *`C:\ZIEWin\`* is the directory where you have Z and I Emulator for Windows installed, and *C:\AppData\* is the Application Data directory.
- To start a session with a specific short session ID (session letter), insert the parameter **`/S=`***`m`* after PCSWS.EXE in the batch file:

  ```
   C:\ZIEWin\PCSWS.EXE  /S=m  C:\AppData\LAN1.WS
  ```

  where *`C:\ZIEWin\`* is the directory where you have Z and I Emulator for Windows installed, *`m`* is the short session ID, and *`C:\AppData\`* is the Application Data directory.
- To start a macro after the session start, add the parameter **`/M`** to the command in the batch file:

  ```
  C:\ZIEWin\PCSWS.EXE C:\AppData\LAN1.WS  /M=mymacro
  ```

  where *`C:\ZIEWin\`* is the directory where you have Z and I Emulator for Windows installed, *`C:\AppData\`* is the Application Data directory,

  ```
  LAN1.WS
  ```

  is the profile, and

  ```
  mymacro
  ```

  is the Z and I Emulator for Windows macro/script file name.

  If the specified macro/script does not exist, there will be a pop up with "PCSKBD400- The file: <macro name> is not a Z and I Emulator for Windows macro/script-file."

Note:

1. If you use the /S option to assign A as the short session ID, you should use this option for all of the sessions in the batch file. Otherwise, if another session starts first, it becomes the A Session and the session with the `/S=a` option will not start because of the conflicting short session IDs. Another way to prevent conflicts is to assign a character later in the alphabet for the short session ID.
2. Several parameters can be specified for controlling the particular characteristics for starting sessions; the switch values are designated by a single character.

Creating a Batch File

To create a batch file, use the following procedure:

1. From the Session Manager dialog, click **New Multiple Sessions**. You can also start a new batch file from the Windows® **Start** menu, using the **HCL Z and I Emulator for Windows → Utilities → Multiple Sessions** program.

   The Create/Modify Batch File panel appears.
2. There are several methods for including profiles or programs in a batch file:
   - Double-click the file names in the **File Name** list box.
   - Drag and drop the file names (using the right mouse button) from the **File Name** box to **Batch-File Entries**.
   - Select a file name from the **File Name** list box and then select **Add**.
   - Type the complete path and command file name in the batch files entries area.
   - You can also use the **Capture View** button to capture multiple session windows into a view.

   Z and I Emulator for Windows places the full path and command that is needed to run the workstation profile or other program above the cursor line in the edit area. If there is no cursor, the command is added to the last line.

   To see the contents of the profile you added to the batch file, click it in the **File Name** list box and then click **View File** or the magnifying glass.

   Note: Some brief instructions appear at the top of **Batch-File Entries**; you need not remove them, because they do not affect the running of the batch file.
3. Repeat step 2 on page 62 for each subsequent file to be added.
4. When you complete the edit, save the created batch file by selecting **Save** from the **File** menu.

   The Save Batch File As window appears.
5. Enter a name for the batch file (***.BCH**).

   The name you enter is used as the icon title, unless you enter a description as well.

The following example is a batch file that runs four workstation profiles located in the Application Data directory, and then runs **MYAPP.EXE**.

```
C:\dir\PCSWS.EXE C:\AppData\SLAN1.WS
C:\dir\PCSWS.EXE C:\AppData\SLAN2.WS
```

```
C:\dir\PCSWS.EXE C:\AppData\AS4Y1.WS
C:\dir\PCSWS.EXE C:\AppData\VT220.WS
C:\APPL\MYAPP.EXE
```

where `C:\AppData` is the Application Data directory specified during installation and `dir` is the installation directory.

Saving Multiple Session Views

You can use the Create/Modify Batch File panel to capture up to multiple session views. Simply size and position up to Maximum session windows, which is configured in the Preferences and click the **Capture View** button. Name the view and click **Save View** in the View Setup panel. You can save up to eight views. You may also delete previously saved views from the drop-down list in the View Setup panel.

If a view is already being used when you click Capture View, that view is automatically used and you are not prompted to save a new view.

Starting a Batch File

You can use one of the following methods to run a batch file:

- If you created an icon for your batch file, double-click the icon in the Session Manager or select the icon and click the **Start** button.
- Run the batch file from the Run command line:

  ```
   [drive]:\[path]\PCSBAT.EXE  [drive]:\[path]\xxxx.BCH /R
  ```

  Note: To run a batch file, specify the /R option.
- Start the **Multiple Sessions** Utility.
  1. Select **Open** from the **File** menu in the Create/Modify Batch File window.
  2. Select the desired batch file and then select **OK**.

     The contents of the batch file appears in the edit area.
  3. Select **Run** from the Run menu.

Editing an Existing Batch File

To edit an existing batch file, do one of the following:

- Right-click on the icon in the Session Manager and choose **Modify**.
  You can also use the following procedure:
  1. Start the **Multiple Sessions** Utility from the Start menu. The Create/Modify Batch File window appears.
  2. Select **Open** from the File menu. The Open Batch File window appears.
  3. Select the batch file you want to edit and then select **OK**. The contents of the batch file you selected appear in the edit area of the Create/Modify Batch File window.
- Edit the batch file. See step 2 on page 62 for more details.
- When you complete the edit, save your changes by selecting **Save** or **Save As** from the **File** menu.

    - Select **Save** to save your changes in the existing file.
    - Select **Save As** to save your changes in a new file and then continue with step 5 on page 62.
- Exit the Create/Modify window.

Starting Multiple Sessions without a Batch File

To start multiple sessions without a batch file, use the following procedure:

1. Start the Session Manager.
2. Select the icons for the sessions, then click the **Start** button. You can select icons using a drag selection box or holding down the Ctrl key while selecting icons with the mouse.
3. After it connects to the host, select one of the following choices from the **File** menu:
    - **Run the Same** to start another session with the same configuration.
    - **Run Other** to start a session with a different configuration.

      When the Open Other Workstation window appears, select the profile you want to start and then click **OK**.

Automatically Starting Sessions

To start one or more sessions automatically, use the following procedure:

1. From the **Start** menu, select **Settings → Taskbar**.
2. Click the **Start Menu Programs** tab and then click **Add**.
3. Click **Browse** and then open the Application Data directory specified during installation.
4. Change the file type to `All Files`.
5. Double-click the session icon or the batch icon.
6. Click **Next** and then double-click the Startup folder.
7. Accept the icon name or type a new one.
8. Click **Finish** and then **OK** when you are done.

You can also drag an icon from the Session Manager to the Startup folder as a shortcut.

Stopping Sessions

To stop a session, click the **X** in the upper right corner or double-click the upper left corner of the session window, or select **Exit** from the **File** menu.

To stop multiple sessions at the same time, select **Exit All** from the File menu. All emulator sessions end, and the associated session windows are closed.

Sessions can also be stopped using a command:

1. Select **Run** or **Programs → MS-DOS® Prompt** from the Start menu.
2. Enter one of the following commands:

   ```
   PCOMSTOP /S=x
   PCOMSTOP /ALL
   ```

   where *x* is the session letter of the particular session to be stopped; use `ALL` to stop all active sessions. There are other parameters; for a complete description, refer to *Emulator User's Reference*.

Note: Stopping a Telnet session automatically closes an associated printer session, if that option was selected when configuring the session. See Printer Session Association on page 68 for information on how to automatically close an associated printer session.

Option to suppress confirmation message for pcomstop

When invoking pcomstop.exe from the command line, the NCE option can be used to suppress the exit confirmation message, which is shown when one or all sessions.

Example:

```
PCOMSTOP /S=<session>|/ALL [/Q] [/C] [/NCE] [/?]
```

One of the following parameters must be specified:

- /S stops the session, while <*session*> is the letter of the session to be stopped
- /ALL stops all sessions

The following parameters are optional:

- /Q specifies quiet mode
- /C converts the output to Windows code page
- /NCE (No Confirm on Exit) stops one or all sessions (as defined by /S or /ALL) without confirmation, even if the Confirm on Exit or Exit All options are set.
- /? displays help information

Stopping an emulator session without access to the tool bar

This method can help you to stop a session when security restrictions do not allow tool bar access.

To stop an emulator session without access to the tool bar, you can use the mouse or a keyboard shortcut to launch pcomstop.exe. Use the following procedure to set up the pcomstop.exe shortcut:

1. Create a shortcut for pcomstop.exe on the desktop or wherever you need.
2. Right-click the shortcut to view the **Properties** window.
3. Click the **Shortcut** tab.

4. The executable name and path are in the **Target** input box. Append any required parameters to this path and click **OK**. These parameters will be used when the pcomstop.exe file is launched. For example, if you want to stop session A, modify the appended path:

   ```
   "E:\Program Files\HCL\Z and I Emulator for Windows\pcomstop.exe" /s=a
   ```

   Note: The /S or /ALL option is required to run pcomstop.exe. The /ALL option stops all sessions, while the /S=x option stops a particular session (where x is the session letter).

5. In the **Shortcut Key** input box, type the key that you want to use as a shortcut (for example, X), and click **OK**.

You can then launch pcomstop.exe by the following methods:

- Mouse
- Double-click the modified shortcut
- Keyboard

Windows® always adds the Ctrl+Alt sequence to the shortcut key. For example, Ctrl+Alt+X becomes the shortcut for invoking pcomstop.exe and closes the emulator session.

Using Emulator Sessions

This chapter describes how to use the printing, editing, and data transfer functions in an emulator session. It also describes some of the choices in the emulator session **Actions**, **Window**, and **Settings → Appearance** menus.

Accessibility

Z and I Emulator for Windows provides functionality with assistive technology such as screen readers. Following are some of the accessibility-related enhancements.

Sounds

Z and I Emulator for Windows supports the **ShowSounds** and **SoundsSentry** options available in the Windows® **Control Panel → Accessibility Options → Sound** dialog. The **ShowSounds** option displays a string representing the event that generated a sound in the status bar.

To mute all of the sounds generated by Z and I Emulator for Windows, select the **Mute** option in the **Settings → Appearance → Display Setup → Sound** panel.

Screen Reader Assist

Users can configure a toggle key to enable Z and I Emulator for Windows to replace blank and null characters in the input field with another character. This option enables screen readers to report the length of the field to visually impaired users. Data sent to and from the host is not changed—only the screen display and the screen reader's voicing of the display are affected. By default, this function is not enabled.

For 3270 and VT emulators, the default padding character is a blank. For 5250, the default replacement character is an underscore. You can choose another character if you prefer.

During the emulation session, you can turn the screen reader assist on or off, as needed. To map the screen reader toggle to a key, click **Settings → Keyboard**. Click **Customize** to access the keyboard setup dialog. Refer to the online help for a complete list of available keyboard functions.

Expanded OIA

For an accessible version of the Operator Information Area (bottom line of the session), you can display the expanded OIA window. Click **View → Expanded OIA** from the session menu bar. You can also select **Show Expanded OIA** from the session's system menu. You can change the number of lines displayed in the expanded OIA in the **Settings → Appearance → Window Setup** dialog.

To set focus to the expanded OIA so that a screen reader can read the values, you need to map a key to the function **OIA: Toggle focus to/from Expanded OIA**. This key enables you to toggle focus back and forth between the session window and the expanded OIA window. When you set focus to the expanded OIA window with a key, the focus in the expanded OIA is always set to the first line. When you return to the session window, the cursor should be where it was before you went into the expanded OIA window. Refer to the online help in the **Settings → Keyboard → Customize** dialog, for more information about custom key mapping.

Popup Keypad

Even though most users use the popup keypad with a mouse, it is possible to customize and use the poppads with the keyboard alone. To display (execute) a poppad without a mouse, you must map a few keys in the **Settings → Keyboard → Customize** dialog.

The **Display Poppad** function shows the last poppad and puts keyboard focus on it. The functions **Display Poppad Pad 1**, **Display Poppad Pad 2**, **Display Poppad Pad 3**, and **Display Poppad Pad 4** display a specific poppad and put keyboard focus on that pad. You can execute the button with current focus by pressing the space bar or the Enter key.

If you are using a sticky poppad, the poppad window remains open until you close it. A regular poppad exits when you push one of the buttons. To get focus to a sticky pad without a mouse, you must map the **Set Focus to Poppad** function to a key—this sets focus to the sticky poppad from the session window. Because you must use the Ctrl-Tab key combination to get focus from the sticky poppad back to the session, mapping the **Set Focus to Poppad** function to the Ctrl-Tab key combination is not advisable.

Quick Connect

You can connect a Telnet (3270/5250/ASCII) session quickly using the Quick Connect bar by configuring only Host, Port and LU Name (3270)/Workstation Id (5250). The LU Name and Workstation ID are optional.

The Quick Connect bar is enabled only for Telnet sessions (both display and Printer). For non-Telnet Sessions, the Quick Connect bar does not appear.

While using the Quick Connect bar, the other session parameters are taken from the active session, if any. If no session is active, then all session parameters will be of default.

You can activate or deactivate the Quick Connect bar by clicking **View -> Quick Connect Bar** in the session window menu.

Power Management

Z and I Emulator for Windows complies with Microsoft Windows Power Management requirements for handling sleep events (stand by and hibernate). This support minimizes session interruptions due to network disconnections caused by sleep on Windows 7 and later versions.

Refer to the *Administrator's Guide and Reference* for more information about Power Management.

Connected State

When Z and I Emulator for Windows is in the connected state and Windows 7 or later operating system indicates that the user is available for interaction, Z and I Emulator for Windows prompts the user to grant permission to sleep.

You can specify a setting in the Preferences Manager that allows the system to standby or hibernate without prompting. In default mode (unchecked), if there is at least one connected session, you will be prompted to allow the system to standby or hibernate. If there are no connected sessions, Z and I Emulator for Windows allows the system to standby or hibernate without prompting. See Standby/Hibernate on page 125.

Non-Connected State

When Z and I Emulator for Windows is not in the connected state, Windows 7 and later operating systems might automatically sleep, without prompting the user for permission.

Critical Sleep

When Windows 7 and later operating system's resumes after an emergency suspension, Z and I Emulator for Windows might display and log a warning message.

Printer Session Association

When you configure a 3270 or 5250 display session, Z and I Emulator for Windows lets you specify an associated printer session.

Advantages of this association are as follows:

- If sessions are associated, the person who configures the client workstation does not have to know any details about the printer session.
- When you start the display session, the associated printer session is started automatically.

When configuring a session, if you want the server to associate a printer with the session, do the following:

1. Click the **Printer Association** tab.
2. Select **Associated Printer Session**.
3. Enter the .WS file for the printer that is to be associated with the session. You can also click **Browse** to locate the file.

You can also set the following options:

- Select the **Start Associated Printer Minimized** check box, if preferred. This option is unavailable until an associated printer is selected.
- Select the **Automatically close Associated Printer Session with this session** check box, if preferred. This option is unavailable until an associated printer is selected.
- Select **Associated device name** to associate the display session with any printer device that currently exists on an iSeries™, eServer™ i5, or System i5™ host. This option is only available for 5250 sessions.

Note:

1. Stopping a Telnet session automatically closes an associated printer session, if that option was selected when configuring the session.
2. If a 5250 printer session is associated with multiple 5250 display sessions, then the printer session ends only when the last associated display session ends.
3. For a 5250 session, if the host name in the selected printer session profile differs from the values in the display session profile, then the display session profile host name is used instead. The display session values are not saved to the printer session profile.

Print Session Setup (3270 and 5250)

The Print Session Setup dialog enables you to customize the display options for a 3270 or 5250 printer session. This dialog can be accessed by clicking **Settings → Appearance → Print Session Setup**. You can also add the Print Session Setup dialog to the session tool bar.

The following customization options are available.

Show Text Information

You can specify the title and other information to be shown in the printer session display window. If this option is not selected, no text information about the session is displayed.

Configuration Details

The following items can be included in the text information.

Connection Details

The following display options are based on the session status and the settings of the **Session Parameters → Advanced** and **Customize Communications** dialog.

Connection Status	If this item is selected, **Connected** is displayed if the session is in connected state. **Disconnected** is displayed if the session is not connected.
Host Name	Host name or the IP address for the connection.
Host Type	Host system type to which the session is connected.
Interface	Interface type selected in the **Customize Communication** dialog.
Attachment	Physical and logical connection selected for the session.
WS Profile	If the session is started from a saved workstation (.WS) profile, the name of the profile is displayed. The field is blank if it is a newly configured session.
Host Codepage	Code page selected in the host Session Parameters configuration panel.

Host Device Details

The following display options are based on the selected device and the settings of the **Session Parameters – 5250 Host > Advanced** dialog. These options are available for 5250 sessions only.

Device Status	If this item is selected, **Started** is displayed if the device is in ready state. **Stopped** is displayed if the device is not in Ready state.
Workstation ID	Device name for the session.
Message Queue/Library	**Message Queue** and **Message Library** specified in the **Session Parameters → Advanced** dialog
Host Font	Host font selected in the **Session Parameters → Advanced** dialog
HPT	If this item is selected, **TRUE** is displayed if **Host Print Transform** is enabled. **FALSE** is displayed if HPT is not enabled.
HPT Printer Model	If this item is selected, the **Printer Model** is displayed. If HPT is not enabled, **Not configured** is displayed.
HPT Drawer 1	If this item is selected, the HPT **Drawer 1** paper size is displayed. If HPT is not enabled, **Not configured** is displayed.
HPT Drawer 2	If this item is selected, the HPT **Drawer 2** paper size is displayed. If HPT is not enabled, **Not configured** is displayed.
Envelope Hopper	If this item is selected, the **Envelope Hopper** is displayed. If HPT is not enabled, **Not configured** is displayed.
Customization Object/Library	If this item is selected, the **Customizing Object** and **Customizing Library** are displayed. If HPT is not enabled, **Not configured** is displayed.
ASCII code page 899	If this item is selected, **TRUE** or **FALSE** is displayed, depending on whether **ASCII Code Page 899** is enabled. If HPT is not enabled, **Not configured** is displayed.

- **Page Setup Details**

  The following display options are based on the workstation profile and the **Page Setup** dialog. For 3270 sessions, the listed options are on the **Text** and **Text Options** tabs. For 5250 sessions, the listed options are on the **Orientation** and **Advanced Options** tabs.

CPI/LPI (3270 sessions only)	Number of characters printed per inch and number of lines per inch.
MPL/MPP (3270 sessions only)	Maximum print line and maximum print position.
Font Name (3270 sessions only)	Device font of the printer device driver.
Margin – Left/Top	Left and top margin values.
Drawer1	Drawer 1 orientation.
Drawer2	Drawer 2 orientation.
Bestfit Scaling	If this item is selected, **TRUE** is displayed if Bestfit is enabled in the .WS profile. If Bestfit is not enabled, **FALSE** is displayed.
Suppress Null Lines (3270 sessions only)	If this item is selected, **TRUE** is displayed if **Suppress Null Lines** is enabled. If **Suppress Null Lines** is not enabled, **FALSE** is displayed.
Nulls as Spaces (3270 sessions only)	If this item is selected, **TRUE** is displayed if **Nulls as Spaces** is enabled. If **Nulls as Spaces** is not enabled, **FALSE** is displayed.
Ignore FF at First PP (3270 sessions only)	If this item is selected, **TRUE** is displayed if **Ignore FF at First PP** is enabled. If **Ignore FF at First PP** is not enabled, **FALSE** is displayed.
FF Takes PP if followed by Data (3270 sessions only)	If this item is selected, **TRUE** is displayed if **FF Takes PP if followed by Data** is enabled. If **FF Takes PP if followed by Data** is not enabled, **FALSE** is displayed.
CR at Max PP + 1 (3270 sessions only)	If this item is selected, **TRUE** is displayed if **CR at Max PP + 1** is enabled. If **CR at Max PP + 1** is not enabled, **FALSE** is displayed.
NL at Max PP + 1	If this item is selected, **TRUE** is displayed if **NL at Max PP + 1** is enabled. If **NL at Max PP + 1** is not enabled, **FALSE** is displayed.

(3270 sessions only)	
FF - Any Position / Column 1 (3270 sessions only)	If this item is selected, **TRUE** is displayed if **FF - Any Position / Column 1** is enabled. If **FF - Any Position / Column 1** is not enabled, **FALSE** is displayed.
Ignore color while printing (3270 & VT sessions only)	Select this option to ignore the colors in PS and print in black & white.
Replace FF by LF (3270 and 5250 sessions only)	Select this option to replace a form feed by the number of lines entered in the edit box.
Automatic Orientation	If this item is selected, **TRUE** is displayed if automatic page orientation is enabled. If automatic page orientation is not enabled, **FALSE** is displayed.
Printer Font Codepage	The printer font code page used for printing on the workstation.
No CR between Fields	If this item is selected, **TRUE** is displayed if **No CR between Fields** is enabled. If **No CR between Fields** is not enabled, **FALSE** is displayed.
Bold as Normal	If this item is selected, **TRUE** is displayed if **Bold as Normal** is enabled. If **Bold as Normal** is not enabled, **FALSE** is displayed.
Use Raster Fonts	If this item is selected, **TRUE** is displayed if **Use Raster Fonts** is enabled. If **Use Raster Fonts** is not enabled, **FALSE** is displayed.

Show Wallpaper

You can specify a bitmap file as a background in the session window. You can use the default graphic or another monochrome, 16-color, 256-color, or 24-bit file.

Print Status Dialog → Show Dialog

You can have a printer status dialog displayed along with the session window. This option is available only for 5250 sessions.

Print Status Dialog → Contain in Session Window

You can display a printer status dialog that is tied to the session window. When the session window is moved or minimized, the printer status dialog is moved with it. This option is available only for 5250 sessions.

Printing

You can use Z and I Emulator for Windows to print from display or printer sessions:

- From display sessions, you can print all (**Print Screen**) or part (**Trim Print**) of the screen of your session window on a workstation printer.

  To print only part of the session window, drag the mouse to create a trimming rectangle around the part of the window you want to print and then select **Print Screen** from the **File** menu.
- With printer sessions, you can print files directly from a host system to a workstation printer. Refer to the online help for more information.

  Configure a printer session to designate a workstation printer as a system printer that will use either the printer definition tables (PDTs) provided with Z and I Emulator for Windows or the Windows® printer drivers. Refer to the online help for more information.

To print, the following methods apply:

- You can use Windows® printer drivers that you configure through the session **File → Printer Setup** menu.
- You can use printer definition tables (PDTs), which give greater control over the print data stream.
- For 5250 only: You can use Host Print Transform, where the host formats and builds the printer commands.

For more information about printing, refer to *Emulator User's Reference*.

Print Screen Collection functions

Using the **Collect Screen** function, you can add a capture of all or part of the screen to a collection of captures.

To add the current screen (or part of the screen) to the collection, click **File → Print Screen Collection → Collect Screen**.

To print and purge all the collected screens, click **File → Print Screen Collection → Print and Purge Collection**.

To print and keep all the collected screens, click **File → Print Screen Collection → Print and Keep Collection**.

To preview the collected screen and select collected screens to be printed or purge, click **File → Print Screen Collection → Prpcess Collection**.

All the collected screens can be deleted without printing by clicking **File → Print Screen Collection → Purge Collection**. An individual screen or part of the collection cannot be deleted.

The **File → Print Screen Collection → Print Collection on Exit** option ensures that the collected screens are printed before you close or disconnect the session. This option is enabled by default. To end the session without printing the collected screen, clear the Print Collection on Exit option. All the collected screens are then deleted when you close or disconnect the session.

Note: The Collect Screen feature works independently of the normal **Print Screen** function. You can still use Print Screen to print individual screens, while collecting multiple screens.

You can add the **Collect Screen** and **Print Collection** functions to the toolbar, a popup keypad, or a custom keyboard map. The settings in the Page Setup dialog are used (shared with the normal Print Screen function).

In PDT mode, there is an option available for printing more than one screen in a page. Refer to *Administrator's Guide and Reference* for more information.

Collecting Print Jobs (5250 Printer Session)

You can collect 5250 print jobs and print them as a single job or in a group. The collected print jobs are stored in a .SCS file.

You can set the following .WS profile keywords to specify the path and file name for the .SCS file.

```
[Printers]
SCSFile=<filename>.scs
SCSPath=<local path>
```

The functions associated with this feature are listed below. The functions can be mapped to the keyboard, popup keypad, mouse button, or toolbar button.

- **Collect Mode**

  When Collect Mode has been started, print jobs that have been sent are saved in the .SCS file. They are not printed immediately.
- **Print Collection**

  The print jobs that have been saved are sent to the printer as a single job.
- **Purge Collection**

  The collected print jobs are deleted.

Refer to the online help for details about mapping the functions.
The CombineJobs profile keyword enables you to collect the jobs for printing, while maintaining them as individual jobs (instead of one job in the .SCS file). Specify the .WS keyword as follows:

```
[Printers]
CombineJobs=N
```

If you set CombineJobs to **N**, the Print Collection function sends the separate, collected jobs to the printer. While in Collect Mode, if the keyword is set to **Y** or is not specified, the print jobs are combined as a single job in the .SCS file.

Using the Windows Printer Driver

To set up your printer to use a Windows® printer driver:

1. Click **File → Printer Setup** in the session window.

   The Printer Setup window lists the supported printers.
2. Select the printer driver to be used from the **Printer** list box. **DEFAULT** will cause the use of the Windows® default printer.

   Note:

a. The **DEFAULT** selection is shown when the .WS file specifies `printer=DEFAULT` in the [printers] stanza.
b. When this selection is made, no message appears before the job is printed.
c. When a printer has been selected for a session, the name of that printer is displayed in the status bar of the session window.

3. If desired, click on the check box to **Show this dialog before every print**.
4. Confirm that the **Use PDT file** check box is not selected and then select **OK**.

   Z and I Emulator for Windows will now use the printer driver you selected, and the Printer Setup window is closed.

Using Printer Definition Table (PDT) Files

Printer Definition Table files define the transfer of characters and control codes to a printer, and the printer output format. If a PDT file is used, the Windows® printer driver is not used, and Z and I Emulator for Windows generates print output based on printer control information defined in the PDT file.

Refer to *Administrator's Guide and Reference* for more information about PDT files.

To use PDT files:

1. Click **File → Printer Setup** in the session window.

   The Printer Setup window appears.
2. Select the port to be used from the **Printer** list box.

   Selected PDT files are available for the port selected here.
3. Select **Setup** and then specify the paper size of the selected printer driver.
4. Select the **Use PDT file** check box and then select **Select PDT**.

   The Select PDT file window appears.
5. To use an existing PDT file, select a PDT file to be used and then select **OK**.

Using Host Print Transform (5250 only)

When configuring a 5250 printer session, the HPT mode may be selected. To use Host Print Transform (HPT), do the following:

1. From the Session Parameters panel, click **Advanced**.
2. Select **HPT Yes**. You can then enter the following parameters:
   - Printer Module
   - Drawer 1, Drawer 2, and Envelope form names
   - Code Page 899, **Yes** or **No**
   - Customizing Object and Library (optional)

Image Print Transform

Z and I Emulator for Windows allows use of Image Print Transform in 5250 print sessions, when using Host Print Transform. Refer to the most recent IBM® iSeries™, eServer™ i5, or System i5™ printing reference for more information on this feature.

Page Setup

Z and I Emulator for Windows allows you to set Page Setup parameters, such as the maximum number of lines per page, the maximum number of columns, and fonts. You can also add a header or footer to a page.

For detailed Page Setup information and instructions, refer to *Emulator User's Reference*.

Scalable (Truetype) APL Font Support on Printers

Because special APL characters are not provided for printers, the APL fonts provided for displays are used when APL characters are printed. In some cases, APL characters are printed rather small. To print larger APL characters, you should install the Z and I Emulator for Windows AICAPL font, using the Windows **Control Panel → Font** dialog.

ZipPrint (3270 Only)

Use ZipPrint to print PROFS® notes, calendars, CMS files, XEDIT workspaces, and 3270 session screens.

Preparing to Use ZipPrint

Before you can use ZipPrint, DDE/EHLLAPI must be enabled for the sessions. To do this, click **Settings → API** and refer to the online help for detailed instructions. DDE/EHLLAPI is enabled by default.

Note: By default, PROFS-oriented functions of ZipPrint are U.S. English PROFS®. You can customize ZipPrint for other languages. Start ZipPrint before you start any display sessions.

Starting ZipPrint

Start ZipPrint before you start any display sessions. The ZipPrint menu is added to the menu bar of the specified session window; then you can use it from the menu bar the same as with the other functions.

Start ZipPrint by selecting the **ZipPrint** icon in the Z and I Emulator for Windows program folder. This starts ZipPrint for Session A only.

You can also start ZipPrint by placing it as the first command in a Z and I Emulator for Windows batch file.

For more information about ZipPrint, including information about using ZipPrint for additional emulator sessions, search for ZipPrint in Help.

Using ZipPrint

Keep the following consideration in mind when using ZipPrint.

ZipPrint uses the Z and I Emulator for Windows File Transfer function to print VM/CMS notes and files. On slower communication lines such as SDLC, Async (IIN), or when using a large packet or block size, you might experience a file transfer timeout. If this happens, you should increase the *timeout delay* as follows:

1. Click **Settings → Transfer** in the session window.
2. Increase the timeout value to 150 seconds or longer.

Editing

You can edit the contents of your session window using the Windows® clipboard and the **Edit** menu.

Note: When using copy/cut functions, Z and I Emulator for Windows takes the entire contents from the session window and places it on the clipboard. To copy or cut only marked sections from the session window, you need to update the Cut/Copy options. To update the Cut/Copy options, do the following:

1. Click **Settings → Edit**.
2. From the Edit Options window, select the **Cut/Copy** tab.
3. From the Cut/Copy page, select the **Only if a trim-rectangle is marked** check box.
4. Close the Edit Options window.

Undo

Cancels the most recent Edit operation, except for Copy Link, and restores the contents of the session window and the clipboard accordingly.

Cut

Copies the marked area into the clipboard and removes it from the display session window.

Copy

Copies (or duplicates) the marked area into the clipboard without removing it from the display session window.

Copy Append

Copies the marked area into the clipboard without removing it from the display session window. If there is already data in the clipboard, Copy Append adds the new data to it.

Copy As Image

Copies the marked area as a bitmap into the clipboard. If no area is marked, Copy As Image captures the entire presentation space as bitmap.

Note: Undo functionality is not supported for Copy As Image.

Paste

Overlays the current contents of the clipboard into the session window, starting at the current cursor position.

Paste Next

If not all data was pasted, Paste Next is enabled and the remaining clipboard data can be pasted.

Clear

Removes the marked area of the session window. The clipboard contents are not altered.

Copy Link

Supports the DDE Copy Link function. To start a link between Z and I Emulator for Windows and another application program, mark an area of the session window, select **Copy Link**, and then select **Paste Link** in the other application program.

Note: The command you should use for Paste Link or Paste Special depends on the application program you are using.

Find

Find text in Presentation Space of a display session. The text found is highlighted on screen. The search can be case sensitive or insensitive.

Send to Scratch Pad

Sends the selected contents of the Presentation Space into the Scratch Pad of the corresponding session.

Unmark

Removes the clipping (or marking) rectangle. The session window and the clipboard contents are not altered.

Select All

Marks the entire session window.

Edit Options

Paste Options

You can control how text is pasted before and after protected fields, and how tabulated text appears after it is pasted. The following Paste functions are available.

Field Wrap

Check this box if you want pasted data that falls onto a protected field to move to the next unprotected field. If you do not check this box, any data that falls onto an unprotected field is lost.

Line Wrap

Check this box to allow pasting of copied text across lines.

Don't Split Words

Check this box to avoid words being split across fields and lines. The text being pasted into fields is split on word boundaries, which breaks the text and starts the new word in the next field. If one word is being pasted into a field, but the field is not long enough to hold the word, then as much of the word as possible is put into the field, and the rest of the word is carried on to the next field.

Note: If the Field Wrap or Line Wrap option is not enabled, the word break option is not available.

Paste to marked area

Check this box to restrict pasting to a marked area, if it exists. If the marked area doesn't exist, pasting will take place at the current location.

Stop pasting when protected line encountered

Check this box to have the pasted text stop when it comes to a protected line on the emulator screen. If you do not check this box, the paste continues.

Tab Character Processing

Advance to next tab stop

You can choose to align tabulated text at specified tab stops. For example, if you choose Advance to next tab stop 4 column(s), your tabulated text is advanced to the column position that is the next multiple of 4.

Replace with n space(s)

You can choose to replace tab stops with a certain number of spaces. For example, if you choose replace with 3 spaces, each tab stop in your original text becomes 3 spaces.

The default setting is to replace each tab character with one space.

Paste data to fields

You can choose to have tabulated text placed in subsequent unprotected fields. With this option, when a tab character is encountered, the following text data will be pasted into the next unprotected field of the emulator session.

Note: This option is available only for 5250 sessions.

Cut/Copy Options

You can control the size of the copy area and how +/- signs behave with signed numeric fields (5250 only).

Only if a trim rectangle is marked

Select this box if you want to copy only the trim rectangle that has been marked on the session screen. The default is to copy the entire screen if no rectangle is marked.

Autocopy

This option enables you to automatically copy the selected text to the clipboard. When an existing selected area is moved to another screen area, the text inside the new selected area is automatically copied to the clipboard.

Force Leading +/-

On a signed numeric field, the Force Leading +/- option will force the +/- sign to be at the beginning of the field rather than at the end.

Note: This option is available only for 5250 sessions.

Trim Options

You can control the behavior of the trim rectangle.

Classic Box Style

This option allows you to draw a normal box like Trim Rectangle.

Trim Rectangle sizing handles

Check this box to add "handles" to the trim rectangle, allowing you to modify the size.

Trim Rectangle remains after edit function

This option keeps the trim rectangle active after completing the trim.

Expand Trim Rectangle during drag

This option causes the trim rectangle to step to character boundaries while it is being sized.

Use solid Trim Rectangle

This option changes the appearance of the trim rectangle. Instead of the standard outline boundary, the trim area appears as a solid box.

Windows Style

This option allows you to make uneven selection in the PS. When this option is selected, solid Trim rectangles without sizing handles are drawn by default.

Note: Undo functionality under the Edit menu is disabled for this functionality.

Editing by Linking to Windows Application Programs

Linking to Windows® application programs supporting **Paste Link** lets you paste session-window data to the windows of those application programs. You can run **Copy Link** when DDE/EHLLAPI is usable.

Confirming the DDE/EHLLAPI Settings

To check whether DDE/EHLLAPI is currently set to usable status, do the following:

1. Click **Settings → API**.
2. Make sure the **DDE/EHLLAPI** check box is selected.

   If the box is checked, DDE/EHLLAPI is set to usable status. Continue to step 4 on page 81.
3. If the box is not checked:
   a. Click the **DDE/EHLLAPI** check box and then click **OK**.
   b. Stop and then start your session to enable the new settings.
4. If DDE/EHLLAPI was already set to usable status, select **OK**.

Using Copy Link and Paste Link

1. Mark the session window area for which **Copy Link** should be issued.
2. Select **Copy Link** from the Edit menu.

   If the session window is already linked with an application program, **Copy Link** appears in gray and cannot be selected. In this case, force the application program to end the linkage, or stop the application program. Then you can select **Copy Link**.
3. Start the Windows® application program for the window to which an area should be copied.
4. Specify the location for which **Paste Link** should be run.
5. Issue **Paste Link** or **Paste Special** by using the menu for the application program.

   The contents of the marked area are pasted into the specified location in the window of the application program.

   Copy Link is now completed.

When the contents of the marked area in the session window are updated during linking, the contents of the area pasted to the window of the linked application program are also updated.

See the online help for more information about the Copy Link and Paste Link functions.

Copying Table Data to a Spreadsheet

You can use the **Cut**, **Copy**, **Copy Link**, or **Copy Append** choices in the Edit menu to copy data in the session window to the window of a Windows® spreadsheet application program.

To use **Copy**, select the **Paste** or **Paste Link** choice in the application program window into which data is to be copied.

Data in the marked area can be copied in the following three data formats, depending on the format supported by the spreadsheet for the window to which data is to be copied:

Sylk format

Data format for general-purpose spreadsheets, such as Multiplan

BIFF format

Data format for Microsoft® Excel

Wk3 format

Data format for Lotus® 1-2-3

Note: Whether application programs, such as Excel or Lotus® 1-2-3, also support these data formats in subsequent versions depends on individual application program specifications.

Individual items of data in tables of the session window are divided automatically such that they are suitable for spreadsheets, and they are copied into individual cells of tables in the application program.

Copying Marked Data without Dividing It into Cells

To paste data in the marked area per line, without dividing it into individual cells, add the following lines to the workstation profile:

```
[Edit]
Sylk=N       (If Sylk format data is not divided into cells)
Biff3=N      (If Biff3 format data is not divided into cells)
Wk3=N        (If Wk3 format data is not divided into cells)
```

Copying Lines Containing Only Operational Signs

If data in the marked area contains signs, such as +, -, =, or |, the signs are regarded as being ruled lines of the table. Once they are removed, only numeric data is copied.

Table 10. Table Data in Marked Area

	1	2	3	4
1990	60	-63	71	+58
1991	+69	69	90	80
1992	71	+80	80	-30

Table 11. Data Copied to Spreadsheet

	1	2	3	4
1990	60	-63	71	58
1991	69	69	90	80
1992	71	80	80	-30

To copy these signs without replacing them with null characters, add the following lines to the workstation profile:

```
[Edit]
MaskGridCharacter=N
```

Copying Data in Cells As Text Data

Data in the marked area is treated as numeric data by default. Therefore, currency symbols, such as $, and punctuation marks, such as commas, are removed before copying. To copy data containing such signs and marks as text data rather than numeric data, add the following lines to the workstation profile:

```
[Edit]
ConvertToNumeric=N
```

Data in the marked area containing signs and marks is then copied as text data. In addition, all numeric data that does not contain signs and marks is also copied as text data.

Transferring Files

Z and I Emulator for Windows enables the transfer of one or more files between a host system and workstation. You can define file transfer options in advance to help you transfer a variety of files quickly and easily.

Note:

PCT400 was withdrawn from marketing 3/98.

With Z and I Emulator for Windows, you can perform the following file transfer functions:

Send files to the host system

Send files using the **Send File to Host** from the **Actions** menu; or by clicking the **Send** button on the tool bar; or, when using 3270 sessions, the SEND command from the DOS prompt.

You can also send files by using an EHLLAPI or DDE application or a macro that invokes file transfer.

Receive files from the host system

Receive files using the **Receive File from Host** from the **Actions** menu; or by clicking the **Receive** button on the tool bar; or, when using 3270 sessions, the RECEIVE command from the DOS prompt.

You can also receive files by using an EHLLAPI or DDE application or a macro that invokes file transfer.

Data Transfer

For 5250 sessions, click **Transfer** from the **Appearance** menu; then select **Data Transfer** on the property page with the **General** tab. This causes the invoking of data transfer functionality when one of the above actions is taken. If **Data Transfer** is not selected, normal file transfer is invoked.

Create, test, replace, and delete templates

Create a template to have Z and I Emulator for Windows automatically generate a workstation or host file name and transfer type when you select a file to be sent or received.

Note: It is not possible to define a file transfer template with the long file name naming convention.

Define transfer types

Define up to 16 transfer types for each host system. Text, binary, and append (except for CICS®) are initially set as transfer types.

Select, create, and customize translation tables

Select translation tables to define which translation table is used during file transfer.

Import or export files (PC/3270 only)

Import/Export is an office system communication program and an application program run on the IBM® Customer Information Control System (CICS®). The import/export function makes it possible to import or export Final Form Text (FFT), Revisable Form Text (RFT), and PC documents.

When you export a file from the host, your workstation receives the file you exported and an interchange document profile (IDP) file. Before you can import a file to your workstation, you need to create an IDP file with transmission information.

Create interactive document profile (IDP) files (PC/3270 only)

An IDP file contains document header information, has the same name as the file to be transferred, and has the extension .IDP.

To create an IDP file, select **Transfer** from the **Appearance** menu.

Note: When you transfer a file on a Telnet5250 session, you cannot transfer a file that includes FFEF. The current version of the iSeries™, eServer™ i5, or System i5™ Telnet program misinterprets FFEF in the file as an end-of-record marker.

ASCII Host Data Transfer

When you are transferring files between two computers, specific protocols must be followed. Files can be transferred only when your PC uses the same protocol as the host. Z and I Emulator for Windows supports the XMODEM and YMODEM public domain protocols.

For XMODEM, Z and I Emulator for Windows uses the XMODEM and XMODEM1K protocols. XMODEM is a block-oriented, error checking protocol that is a single file, half duplex protocol. XMODEM1K is the same as XMODEM, except that it uses larger 1024 byte (1K) packets.

The YMODEM protocol is similar to the XMODEM1K protocol in that it transfers data in 1K packets, but it also allows multiple files to be sent in one transfer.

The YMODEMG protocol transfers multiple files like YMODEM, but performs no error detection or error correction. It can be much faster than YMODEM, but requires an error-free data connection.

With Z and I Emulator for Windows, you can perform the following data transfer functions to or from ASCII hosts:

Send any type of files to the host system

Send files using the Transfer menu using XMODEM, YMODEM, XMODEM1K, or YMODEMG.

Receive files from the host system

Receive files using the Transfer menu and XMODEM, YMODEM, XMODEM1K, or YMODEMG.

Create, test, replace, and delete templates

Create a template to have Z and I Emulator for Windows automatically generate a workstation or host file name and transfer type when you select a file to be sent or received.

Setting Up the Appearance of a Session Window

You can use the following functions to define the appearance of your session window. These options are in the **Settings → Appearance** menu.

Display Setup

Customize a variety of characteristics, such as the cursor, pointer, rule line, and trimming styles, graphics, sound, and color palette, in the display session.

Color Mapping

Set the colors used in session windows.

Font

Choose the font to use for display session windows, the style, and whether it will be an automatic sizing font or a fixed size font. If you use a fixed size font, you can also choose its size. The set of fonts from which you can choose depends on the type of display you are using.

Note: You cannot change the font size when the session window is maximized.

Window Setup

Change the appearance and title of the session window and change the session icon.

Sounds

Z and I Emulator for Windows enables customization of program sounds through the Windows Control Panel. You can configure specific program sounds using sound files included with the Z and I Emulator for Windows product.

A **Mute** function can be used to silence all program sounds. This option is available from the **Settings → Appearance → Display Setup → Sound** dialog.

Tool Bar Setup

The tool bar displays under the menu bar in your session window to allow quick access to the Z and I Emulator for Windows functions, commands, and defined macros.

Use the tool bar pop-up menu to quickly and easily create, edit, and delete tool bar items, as well as to save and load customized tool bars. When you customize the tool bar, you can change the order of items, add and delete items, change the function, title, or graphic associated with any item, change the fonts, colors, and other tool bar visual style elements. The settings are stored in a .BAR file.

To customize your tool bar, select **Tool Bar → Tool Bar Style** from the **Settings** menu, or display the Tool Bar pop-up menu by clicking on the right mouse button while pointing at any part of the tool bar.

For information about customizing the tool bar, refer to the online help.

If you want to hide the tool bar, see Showing or Hiding the Menu Bar, Status Bar, or Tool Bar on page 86.

Showing or Hiding the Menu Bar, Status Bar, or Tool Bar

You can show or hide the menu bar, status bar, or tool bar. If the menu bar is displayed, enable or disable status bar or tool bar from the session **View** menu. You can also do the following:

- Click the upper left corner of the session window to display the system menu.
  - **Hide Menu Bar** appears when the menu bar is shown.
  - **Show Menu Bar** appears when the menu bar is not shown.
  - **Hide Status Bar** appears when the status bar is shown.

    - **Show Status Bar** appears when the status bar is not shown.
    - **Hide Tool Bar** appears when the tool bar is shown.
    - **Show Tool Bar** appears when the tool bar is not shown.
    - **Show Expanded OIA** appears when the expanded OIA is not shown.
    - **Hide Expanded OIA** appears when the expanded OIA is shown.
    - **Hide Quick Connect Bar** appears when the Quick Connect bar is shown.
    - **Show Quick Connect Bar** appears when the Quick Connect bar is not shown.
- To hide the menu bar, status bar, tool bar, or expanded OIA, select **Hide Menu Bar**, **Hide Status Bar**, **Hide Tool Bar**, **Hide Expanded OIA**, or **Hide Quick Connect Bar**.
- To show either the menu bar, status bar, tool bar, or expanded OIA, select **Show Menu Bar**, **Show Status Bar**, **Show Tool Bar**, **Show Expanded OIA**, or **Show Quick Connect Bar**.

Window Setup

For some Windows® operating systems, if you clear the **Maximized Style → With Title Bar** option in the session **Settings → Appearance &rarrow; Windows Setup** dialog, then the **Minimize All Windows** option on the Microsoft Windows® taskbar may not have any effect. To minimize the window, press Alt-Space and click **Minimize**.

Customizing the Color Mapping using a configuration File

This feature enables users to apply the color mapping configuration to a session using a color mapping configuration file (CMP). You can import, modify and save the color mapping configuration in a CMP file, and can also customize the "Default" color mapping configuration for all the profiles using the default color mapping configuration file (‘DefaultColorConfig.CMP’).

The colors of the various parts of the host-session window can be customized; each component has its own default foreground and background colors, which are determined by the attributes (base or extended) sent to the screen by the host application to which the session is connected.

Use the Category tree control to change a category, or place your mouse pointer on the section of the Presentation Space screen you wish to change the color for and click the left mouse button. On the Color Mapping tab, this will set the correct Category/Element, as well as the current colors for that field in the color pulldowns and the Sample text.

The color modification pulldowns give 16 basic colors, each with a name and a color preview. Click the right-hand button to open up the standard Windows color palette, where you can further modify the colors.

How to Enable/Disable the creation of CMP

The **‘EnableCMP’** parameter in the Color section of *pcswin.ini*, can be set to ‘Y’ or ‘N’ respectively to enable or disable the creation of Color Mapping files.

For Example:

```
[Colors]

EnableCMP=Y
```

When enabled, the Color mapping dialog will have four additional options in the ‘File' menu:

1. **Open ->** Imports the color mapping changes from the selected color mapping file (.CMP) to the current session.
2. **Save ->** Saves the changes made by the User to the currently imported or the associated color mapping file.
3. **Save as->** Copies the color mapping configuration set in the color mapping dialog to a new file.
4. **Exit ->** Exits the color mapping dialog.

This feature enables users to apply the color mapping configuration to a session using a color mapping configuration file (CMP). You can import, modify and save the color mapping configuration in a CMP file, and can also customize the "Default" color mapping configuration for all the profiles using the default color mapping configuration file (‘DefaultColorConfig.CMP’).

The colors of the various parts of the host-session window can be customized; each component has its own default foreground and background colors, which are determined by the attributes (base or extended) sent to the screen by the host application to which the session is connected.

Use the Category tree control to change a category, or place your mouse pointer on the section of the Presentation Space screen you wish to change the color for and click the left mouse button. On the Color Mapping tab, this will set the correct Category/Element, as well as the current colors for that field in the color pulldowns and the Sample text.

The color modification pulldowns give 16 basic colors, each with a name and a color preview. Click the right-hand button to open up the standard Windows color palette, where you can further modify the colors.

Default Color Settings

If a user has enabled this function and needs to apply the "Default" color mapping configuration to all sessions, the *'DefaultColorConfig.CMP'* file should be saved to the ZIEWin Application Data folder (based on type of installation). To make a new default color mapping configuration file, rename any "*.CMP" file to *'DefaultColorConfig.CMP'*.

If a session does not have a Color Mapping file associated with it, the color mapping configuration from *'DefaultColorConfig.CMP'* would be used by default. If the session already has a CMP file connected with it, the color mapping configuration in the CMP file will override the *'DefaultColorConfig.CMP'* configuration.

For Example:

If a user has a Color mapping file associated with their session profile's file then the color mapping file would take priority over the *'DefaultColorConfig.CMP'* file file in terms of keyword precedence.

Session Profile file name: ABC.WS

Color mapping file: ABC.CMP (associated with the session profile) has content as below:

```
[Colors]

BaseColorNormalUnprotected=24D830 000000
```

DefaultColorConfig.CMP have below contents:

```
BaseColorIntensifiedUnprotected=F01818 000000
BaseColorNormalUnprotected= F01818 F01818
```

The Session profile (ABC.WS) will read the value of *BaseColorNormalUnprotected* from the 'ABC.CMP' file and *BaseColorIntensifiedUnprotected* from the '*DefaultColorConfig.CMP*' file.

Note:

1. If this feature is enabled, the colors in Session Profile file will be ignored.
2. *The "DefaultColorConfig.CMP"* has colors that users can set for all their session profile files. This could be different from the product default colors.
3. *The "DefaultColorConfig.CMP"* file needs to be created manually by the user. This does not get created automatically by the system.

Migration of Color mapping content

When a User enables this feature and launches a session, the color mapping configuration from the Session profile file (if it exists) is migrated once to the Color Mapping file.

Note: A single Color mapping file may be associated with several profile files. The color configuration of the last migrated profile will be stored in the common color mapping file.

When a User disables this feature and launches a session, the color settings from the Color Mapping file are migrated once to the Session profile file.

For Example:

Before Enabling this feature, the content in a Session profile has a Color section and a few colors customized in it:

Session Profile file name: ABC.WS

```
[Colors]

BaseColorNormalUnprotected=24D830 000000
```

After the user enable this feature, if there is no color mapping file associated with the session profile name then a new color mapping file will be created with the same name as the session profile file and will copy the color configuration to the newly created color mapping file.

File Name: ABC.CMP

```

BaseColorNormalUnprotected=24D830 000000
```

The Colors will not get deleted from the session profile name but it will be ignored.

Setting Up and Using the Assist Functions

The auxiliary functions described in this chapter let you operate the system more efficiently. **Keyboard/Macro/Script Function** can be customized from the **Actions** menu. The other functions can be customized from the **Edit** menu.

Keyboard, Macro, and Script Functions

A Keyboard/Macro/Script Function command lets you run scripts, macros or Z and I Emulator for Windows-supplied key functions without using the keyboard. Run the scripts, macros or key functions from the current cursor position in the session window.

Scripting Functions

You can write, execute, record and terminate VBScripts in the emulator environment. These scripts have access to the HACL automation API. The programming environment includes methods, class descriptions and properties. VBScript is a subset of the Visual Basic® programming language.

Macro Functions

A macro is a sequence of key or mouse actions and host commands that you can perform with a single action, such as a keystroke. Before you can use a macro function, you need to define it. For more information, see Macro/Script Setup and Use on page 93.

Key Functions

Z and I Emulator for Windows provides many key functions that can be assigned to the keys on the keyboard, the mouse buttons, or the buttons of the pop-up keypad. You can also use them to generate macros.

Hotspot Setup

A *hotspot* is an area of the session window on which you can double-click the left mouse button to perform a command or function. You do not need to use the keyboard. For example, you can double-click a function key number to perform the function.

Tip: Select **Show hotspots** to get three-dimensional (3D) hotspots; these only require a single click, and they stand out on your screen.

You can define the following actions for a hotspot:

- Click on a URL to connect to a World Wide Web site.
- Simulate function keys.
- Play a macro that has the same name as the character string you select on the session window.
- Enter the selected string at the cursor position.
- Simulate the Enter key at the cursor position.
- For VT, simulate two sets of function keys, PF1 through PF4 and F6 through F20.

Using Hotspots

Note: You must have a mouse to use hotspots

To use a hotspot:

1. Move the mouse pointer to the hotspot displayed in the session window.
2. Double-click the left button of the mouse, except for 3D hotspots, which only require a single click.

   Z and I Emulator for Windows determines whether you have specified a hotspot function that matches what appears at the position of the mouse pointer. If so, it processes the hotspot. When two or more hotspots are specified for a single character string, the *first retrieved hotspot* is processed.

   Hotspots are retrieved in the following order:

   a. Point-and-select (connect to Web site using URL)
   b. PFnn, FPnn, Fnn, nn
   c. Point-and-select (run the macro)
   d. Point-and-select (enter the selected string)
   e. Point-and-select (enter at the cursor position)

Keyboard Setup

You can use Keyboard Setup to modify the function defined for each key on the keyboard, except some reserved keys.

You can define the following functions for the keys.

- Performing a key function
- Playing a macro
- Entering characters

Note: By default, the Enter function is assigned to the Ctrl key. To change this assignment or, if the Enter key does not work properly, you need to customize your keyboard. For 3270 and 5250 sessions, you can use the keyboard map files provided with Z and I Emulator for Windows. Refer to *Emulator User's Reference* for more information about keyboard mapping and functions.

Keyboard File

When you specify a key, you can save the new keyboard layout in a file (.KMP). If you create two or more keyboard files, you can alternate between them as required.

To assign a function to a key on the keyboard:

1. Select **Keyboard** from the **Settings** menu or click the map icon on the tool bar.
2. When the Keyboard Setup window appears, select **Customize**.

   Note: Select **Spain** from the Language menu during keyboard setup if you want Catalan support.

3. Assign the key functions, referring to the online help for detailed instructions.
4. Save your changes and exit the Customize Keyboard window.
5. Select **OK** after completing the setup.

You can reset either the entire keyboard or specific keys to defaults:

- To reset the entire keyboard, set the current keyboard to **Default** in the Keyboard Setup window.
- To reset specific keys, select a key in the Customize Keyboard window and then select **Default** from the Current Actions for Selected Key box.

Note: There are seven keys that you cannot redefine and they are gray or dimmed in Keyboard Setup; these keys are: Alt, AltGr, Print Screen, Scroll Lock, CapsLock, NumLock, and Shift.

Customizing the VT Emulator Keyboard

If you are using a VT emulator session, you can represent ASCII control characters in the character strings that you define for your customized keyboard.

Use the # character to represent the CTRL key, following it by any character from the following list (only upper-case alphabetics are allowed):

```
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
```

Use ## to represent an actual #. For example, `123##45` represents `123#45`.
If you follow # with any other character than those shown, you will get an error message:

```
PCSKBD160 - Unrecognized key-action: "character string
```

For example, the error for the string "#a is

```
PCSKBD160 - Unrecognized key-action: "#a
```

The set of control characters is as follows:

Character pair	Control	Notes
#@	Control-@	NULL
#A	Control-A	
through		
#Z	Control-Z	
#[	Control-[	ESC
#\	Control-\	
#]	Control-]	
#^	Control-^	
#_	Control-_	

Because the *.KMP file can be edited, there is a run time check to ensure that the character string is correct. If `"###` is processed, then `#` will be displayed, followed by a beep to signify that the entire character string was not played. If `"123#a456` is processed, then `123` will be displayed, followed by a beep.

Macro/Script Setup and Use

A macro is a sequence of key or mouse actions and host commands that you can perform with a single action, such as a keystroke. You can edit an existing macro or create a new macro by selecting **Macro/Script** from the **Appearance** menu.

A script is a VBScript program that is a subset of the Visual Basic® programming language. For information on VBScript, see the online help pages.

Using a Macro or Script

You can use a macro or script in various ways. Table 12: Macro Use Examples and Settings on page 93 lists examples of how you can set up and then use macros and scripts.

Table 12. Macro Use Examples and Settings

If you want to...	Do this...
Automatically play a macro or script when your session starts.	Set an **Auto-Start Macro/Script**.
Play a macro or script while attached to a host application.	**Start Playing Macro/Script**.
Play a macro or script with the keyboard or macro function.	Use the **Keyboard/Macro/Script Function**.
Click on a hotspot to play a macro or script.	Set a **Hotspot**.
Assign a macro or script to a pop-up keypad button.	Customize the **Pop-Up Keypad** file.
Press the mouse button to play a macro.	Customize a **Mouse** file.
Press a key on the keyboard to play a macro or script.	Customize a **Keyboard** file.

Creating a Macro

You can create a macro manually or by recording some of your interactions with the host system, such as your logon procedure.

To create a macro manually:

1. **Macro/Script** from the **Appearance** menu.
2. When the Macro/Script Setup window appears, select **Customize**.
3. When the Customize Macro/Script window appears, select **File**, then **New**, then **Macro**, and edit the macro. You can type statements directly or select functions, characters, or even other macros, from the list of Key Actions in the **Select a Key-Action** subpanel.

   Refer to the online help for detailed information.
4. Click **File**, then **Save** to save the macro file.

Note:

1. Macros that are converted to XML are intended for use in ZIEWeb (Z and I Emulator for Web) and will not function in Z and I Emulator for Windows emulation sessions. Use the ZIEWeb Macro Manager to import a converted Z and I Emulator for Windows macro into ZIEWeb. These converted macros will not appear in the list of available Z and I Emulator for Windows macros.

Macro Statements

You can use the following statements when you create a macro:

Key function

Use Z and I Emulator for Windows-supplied key functions

Macro

Define a macro within another macro but you cannot create a macro in a permanent loop where the first macro is repeatedly called at the end of the macros.

Character

Use any characters in the **Character** list in the Customized Macro window.

Character string

Use character strings you can type from the keyboard. You must use a double quotation mark (“) at the beginning of the character string.

Wait condition

Specify wait conditions to stop the process until the specified time has elapsed or the status satisfying the specified condition occurs.

Tokens

You can use tokens, such as goto and run, to add logic.

Macro Loop Considerations

If you use a GOTO and a label in a macro to create a loop, you may experience unpredictable behavior if that loop executes a large number (more than 1000) of iterations.

Creating a Script

You can create a script manually or by recording some of your interactions with the host system, such as your logon procedure. Scripts, however, allow a higher level of programming control unavailable with macros.

To create a script:

1. **Macro/Script** from the **Appearance** menu.
2. When the Macro/Script Setup window appears, select **Customize**.
3. When the Customize Macro/Script window appears, select **File**, then **New**, then **Script**, and edit the script. The **Select a Key-Action** subpanel, in gray, is not available for creating scripts.

   Refer to the online help for detailed information.
4. Click **File**, then **Save** to save the script file.

Configuring a Macro or Script to Autostart

To configure a macro or script to autostart, do the following:

1. Click **Settings → Macro/Script**.

   The Macro/Script Setup window appears.
2. Set a macro or script to autostart when your workstation starts.
3. Refer to the online help for detailed instructions and then select **OK** after completing your setup.

Auto-start macro support

To specify which macro should be started automatically after start of a session, add the PCSWS option /M=<*mymacro*>, where <*mymacro*> is the name of the Z and I Emulator for Windows macro or script file. See the following example:

```
C:\ZIEWin\PCSWS.EXE C:\AppData\LAN1.WS /M=mymacro
```

If the specified macro or script does not exist, the following error message will be returned:

PCSKBD400-
The file: <macro name> is not a Z and I Emulator for Windows macro/script-file

Configuring a Java Applet to Autostart

To run a Java applet automatically, you can add the function **Run Applet** to a macro. You can add this function to an autostart macro, so that an applet can be run at session startup.

1. Click **Settings → Macro/Script**.

   The Macro/Script Setup window appears.
2. Select **Run Applet** from the Function drop-down list and type the name of the class.

Refer to the online help for syntax and detailed instructions.

Note: The applet class specified must exist in the same directory path as the .WS file in which the macro is included.

Recording Macros or Scripts

To start macro or script recording, use the following procedure:

1. Click **Start Recording Macro** from the **Actions** menu.
2. Type the macro or VBScript name.
3. Select the record format.

   If you are recording a macro format file for Express Logon, select the **Enable** check box and enter the Express Logon Feature (ELF) Application ID.
4. Configure the other options and click **OK**.

Note: When recording a macro, the processing of a nondisplayable field is controlled by the setting of the parameter HideNonDisplayDataOnRecord=Y in the [Keyboard] stanza of the .WS file. The hidden fields are ignored during the recording session.

To cancel macro or script recording, click **Actions → Cancel Recording Macro**. The recording operation is canceled, and the macro or script is not saved.

To pause macro or script recording, click **Actions → Pause Recording Macro**. The recording operation stops. To restart recording, click **Resume Recording Macro**.

To end macro or script recording, click **Actions → Stop Recording Macro**. Recording ends and the macro is saved in the specified file.

Playing Macros and Scripts

To play a macro or script, click **Actions → Start Playing Macro/Script**, select the macro or script, and click **OK**. The selected macro starts to play.

To stop playing a macro or script , click **Actions → Quit Playing Macro/Script**. The macro or script stops playing.

Note: Z and I Emulator for Windows macro files more than 32KB in size cannot be played in the Z and I Emulator for Windows emulator session. If you want to play a macro file that is greater than 32KB, you will have to break the macro into multiple files.

Note: When playing a macro, the processing of a non-displayable field is controlled by the setting of the parameter HideNonDisplayDataOnRecord=Y in the [Keyboard] stanza of the .WS file; setting this causes a pop-up window to display, requiring your input. When this appears, type in the requested information and press **Enter** to continue.

ThisMacroName support

The ThisMacroName property is used to enable a running script to obtain the name of the macro file that is running. This is useful when writing a script that will be using the name of the macro file.

See the following example of ThisMacroName:

```
[ZIEWin SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=Example of usage of property ThisMacroName
[ZIEWin SCRIPT SOURCE]
OPTION EXPLICIT
Dim sName, sHandle

REM App Main
Main

sub Main()
 'Initialize the session
 autECLConnMgr.autECLConnList.Refresh
 sName = autECLConnMgr.autECLConnList(1).Name
 sHandle = autECLConnMgr.autECLConnList(1).Handle

 'Connect to the current session
 autECLSession.SetConnectionByName(ThisSessionName)
 sName = autECLSession.Name
 sHandle = autECLSession.Handle

 MsgBox("The current session name :")
 MsgBox(ThisSessionName)
 MsgBox("The macro name is :")
 'Should pop up correct macro file name in the message box
 MsgBox(ThisMacroName)
end sub
```

Express Logon Feature

The Express Logon Feature (ELF) allows a Z and I Emulator for Windows TN3270E user to logon to a host application without sending the user ID and password. This function is designed to be implemented only in certain circumstances by administrators of Z and I Emulator for Windows. Refer to *Administrator's Guide and Reference* for detailed information on implementation.

Recording an Express Logon Macro

To record an ELF macro, do the following:

1. Choose **Actions** from the WorkStation-window menu-bar.

   The Actions pull-down menu appears.
2. Choose **Start Recording Macro** from the menu.

   The **Record Macro/Script As** dialog-box is displayed.
3. Select the format of file to be recorded: **VBScript or Plain text Macro**, if you are recording an Express Logon Feature macro.
4. Select the Record User Wait Time to be used. This affects only unconditional wait statements. There is no change for the interval in conditional wait statements (interval with condition). For these statements the interval is a time-out value, processing normally continues when the condition is satisfied. In most cases the script will fail if the time-out value expires before the condition is satisfied.
   - Actual- the default no change to existing function.
   - None - unconditional wait statements will not be written to the macro/script file. You have to edit the script and add any required unconditional wait statements where they are needed.
   - Fixed - unconditional wait statements will be written to the macro/script file with a fixed wait time of 25 ms. You may edit the script file and either increase the time interval or remove the wait statement if it not needed.
5. Type the name of the file in which you want to save the macro.
6. If you are recording an Express Logon Macro, select the **Enable** check box and enter the Express Logon Feature (ELF) Application ID.
7. Click **OK**.

   The Record operation starts. In the Actions menu, Start Recording Macro/Script is changed to Stop Recording Macro/Script; Pause Recording Macro/Script and Cancel Recording Macro/Script are added.
8. Record the keystrokes.
   - If, while you are recording, you choose **Cancel Recording Macro/Script**:

     The Record operation is canceled and nothing is saved. Cancel Recording Macro/Script and Pause Recording Macro/Script are removed from the pull-down list. Stop Recording Macro/Script is changed to Start Recording Macro/Script.
   - If, while you are recording, you choose **Pause Recording Macro/Script**:

    The Record operation is paused and Pause Recording Macro/Script is changed to Resume Recording Macro/Script.

  - If you choose **Resume Recording Macro/Script**:

    The Record operation is resumed and Resume Recording Macro/Script is changed to Pause Recording Macro/Script.

9. When you want to end the recording, choose **Stop Recording Macro/Script.**

   The Record operation stops and the macro is saved in the file you named. Stop Recording Macro/Script is changed to Start Recording Macro/Script. Cancel Recording Macro/Script and Pause Recording Macro/Script are removed from the list.

Verifying an Express Logon Feature Macro

You can visually inspect an existing Macro recording of a host application logon to verify that the UserID and Password have been replaced by Express Logon Feature (ELF) tags. The procedure is as follows:

1. From the Action Bar, open the Macro file containing the recorded keystrokes by selecting **Settings > Macro/Script**.
2. Select the Macro file to be inspected and then select **Customize**.
3. Verify that the UserID has been replaced with two tags: the ELF Application ID and the ELF UserID placeholder. The Application ID tag consists of the following three words, each separated by a blank character: "elf", "applid", and the identifier of the host application that will be logged onto. The UserID placeholder is **)USR.ID(**.

   For example

   "myUserID

   should have been replaced with

   ")USR.ID(
4. Verify that the Password has been replaced with the ELF Password placeholder tag **)PSS.WD(**.

   For example

   "myPassword

   should have been replaced with

   ")PSS.WD(
5. Verify below entry must be available

   For Example:

   - For Plain Text macro : **elf applid TSOIPO1**
   - For VBSCript macro : **autECLSession.SetELFApplID " TSOIPO1**.

Updating an Existing Macro for Express Logon

You can manually update an existing Macro recording of a host application logon to use the Express Logon Feature (ELF). The procedure is as follows:

1. From the Action Bar, open the Macro file containing the recorded keystrokes by selecting **Settings > Macro/ Script**.
2. Select the Macro file you just recorded and then select **Customize**.
3. Replace the UserID recorded in the Macro with two tags: the ELF Application ID and the ELF UserID placeholder. The Application ID tag consists of three words, each separated by a blank character: "elf", "applid", and the identifier of the host application that will be logged onto. The UserID placeholder is **)USR.ID(**.

   For example, replace

   "myUserID"

   with

   ")USR.ID("
4. Replace the Password recorded in the Macro with the ELF Password placeholder tag **)PSS.WD(**.

   For example, replace

   "myPassword"

   with

   ")PSS.WD("
5. Add below new entry to convert Normal macro into ELF macro:

   For Example:

   - For Plain Text macro : **elf applid TSOIPO1**
   - For VBSCript macro : **autECLSession.SetELFApplID " TSOIPO1"**.

Mouse Setup

The **Mouse Setup** command lets you allocate functions to the right and left buttons of the mouse; this enables you to perform the following actions without having to use the keyboard:

- Running the Z and I Emulator for Windows-supplied key functions
- Running the user-defined macros
- Placing a character at the current cursor location

Mouse File

The user can save the functions defined for the mouse buttons in a mouse file (*.MMP). The user can create two or more mouse files and switch between them as required.

To set up the mouseand assign functions to the mouse buttons:

1. Select **Mouse** from the **Settings** menu.

   The current settings appear in the Mouse Setup window.
2. Set up the required items, referring to the online help for detailed instructions.
   If the user want to create or edit a mouse file:
   a. Select **Customize**.
   b. When the Customize Mouse window appears, allocate functions to the right and left buttons of the mouse. These appear in **Current Action for Mouse Button**.
   c. Select the required functions from **List of Key Actions**.
   d. Save your changes and then select **OK**.
3. Select **OK**.

   The mouse setup is complete.

Mouse Wheel Functionality

Mouse wheel scroll up and scroll down is mapped to the “PF7” and “PF8” function key (aid key) respectively for 3270 host.

Mouse wheel scroll up and scroll down is mapped to the “Roll Down” and “Roll Up” function key (aid key) respectively for 5250 host.

Using the mouse wheel scroll option, users can scroll up and down on the Mainframe green screen, in case if the host data exceeds more than one screen.

Example: User can scroll on z/OS syslogs.

Pop-Up Keypad Setup

The pop-up keypad is a small window in which some buttons are arranged. To display the pop-up keypad, put the pointer anywhere in the session window and press the right mouse button.

You can allocate the following functions to these buttons:

- A standard key function provided by Z and I Emulator for Windows
- A user-specified macro
- A character entry

To perform these functions, you only need to click a button on the pop-up keypad with the left mouse button.

To view the description of a short-name key function or macro, click the right mouse button while pointing at one of the function or macro names.

Pop-Up Keypad File

You can specify the number of buttons displayed in the pop-up keypad, the functions allocated to those buttons, and colors of the buttons. You can save the specified pop-up keypad contents in a pop-up keypad file (also referred to as a *poppad* file).

A pop-up keypad file (*.PMP) contains the information for the number of buttons displayed in the pop-up keypad, the functions allocated to those buttons,and color information. You can define which pop-up keypad file is allocated to the pop-up keypad.

Using the Pop-Up Keypad

To use a pop-up keypad:

1. With the mouse pointer anywhere in the session window, click the right mouse button.
2. Select **Pad 1, Pad 2, Pad 3**, or **Pad 4**.
3. Click the required button in the pop-up keypad.

To set up the pop-up keypad:

1. Select **Popup Keypad** from the **Settings** menu.
2. When the Popup Keypad Setup window appears, set the required items, referring to the online help for detailed instructions.

   If you want to edit the pop-up keypad file, select **Customize** and then make your changes.
3. Select **OK**. The pop-up keypad you selected is ready for use.

Tab Setup (VT only)

Tab Setup allows you to define tab stops for your VT sessions.

Web Browser Setup

Web Browser Setup allows you to define a preferred web browser and to use it rather than the one that comes with your operating system.

Managing Emulator Sessions

Z and I Emulator for Windows provides the following functions, in addition to those provided by Windows®, for those who work with several open session windows simultaneously. These functions allow you to manage your session windows easily and quickly. The Window menu has the following selections:

Jump

Use **Jump** to switch between the currently opened session windows.

You cannot use Jump to switch to a session window that is currently hidden. Instead, select **Show Session** from the Window menu to display the session window on the screen. Then, select **Jump**.

Hide Session

Use **Hide Session** to stop displaying a visible session window.

You cannot hide all sessions. At least one session is always shown.

Show Session

Use **Show Session** to display a session window that was previously hidden with **Hide Session**.

Use the View menu to display a previously-saved arrangement of windows, or to save an arrangement of windows.

Z and I Emulator for Windows can save and restore the following information relating to the session window view:

- Position and size of each window
- Window status (standard, minimized, or maximized)
- Window font

You can save view information for up to eight windows.

Getting Help

For more information about managing session windows, refer to the online help:

1. Select **Procedures** from the Help menu.
2. When the help window appears, scroll down to **Managing Workstation Windows®**.

   Select choices from that list for detailed information.

Online Emulator Session

Emulator sessions started from Session Manager online are displayed with “Online” Tag at end of Session Titles. Dialogs where create/modify or delete opertations for the .ws (Workstation profile),.bch - (Multiple Sessions or Batch),.pmp (Popup-Keypad Configuration).kmp (Keyboard Configuration).bar (Toolbar Setup), .mmp (Mouse Setup) .xlt (Translation Table) are always online

Detect and Repair

The user use the **Help → Detect and Repair** function to check the Z and I Emulator for Windows product integrity. The Detect and Repair operation performs a check on the installed Z and I Emulator for Windows files to determine whether the installation has been damaged. A subsequent repair is performed, if necessary.

Note: The user must stop all active sessions before starting the Detect and Repair function.

Check **Restore my shortcuts** if the user want original shortcuts restored. If the user have modified shortcuts since the original Z and I Emulator for Windows installation, the user might want to keep your shortcuts intact—in that case, do not choose this option.

In order to use the **Detect and Repair** function, the user must be authorized in the System Policy. The user might be prompted to provide the original Z and I Emulator for Windows installation source.

Z and I Emulator for Windows **Detect and Repair** is invoked from the Session Manager or from an emulator session window. Windows Installer repair of the Z and I Emulator for Windows product is invoked from the Windows® Add/Remove Programs function. Note the following differences in the operations.

Z and I Emulator for Windows Detect and Repair performs the following operations:

- Reinstalls a file if it is missing or corrupt, or if it is an older version.
- Rewrites all registry settings for the application in the LOCAL_MACHINE section of the registry.
- Rewrites all registry settings for the application in the CURRENT_USER section of the registry.
- Reinstalls all shortcuts (optional).

Alternatively, Windows Installer performs the following operations:

- Reinstalls a file if it is missing, or if it is an older version. Windows Installer does not check files for corruption.
- Rewrites all registry settings for the application in the LOCAL_MACHINE section of the registry.
- Rewrites all registry settings for the application in the CURRENT_USER section of the registry.
- Reinstalls all shortcuts. This function is not optional in Windows Installer.

Managed ZIEWIN and Interoperablity

This section provides detailed information about Managed ZIEWin and Interoperabilty between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients.

HCL Z and I Emulator for Windows uses Session Manager Online dialog to provide easy access to workstation profiles and batch files on the ZIE Server. With Session Manager Online users can create or start a single or multiple sessions and or batch files. Users can create their own profile to the ZIE server and migrate existing files such as the workstation profiles (*.WS) and batch files (*.BCH) that were stored on the ZIE Server.

This "How To" document aims to supplement additional detailed information in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN) as referenced below.

Steps to Install Using Managed:

Refer to Planning to Install Z and I Emulator for Windows on page 42

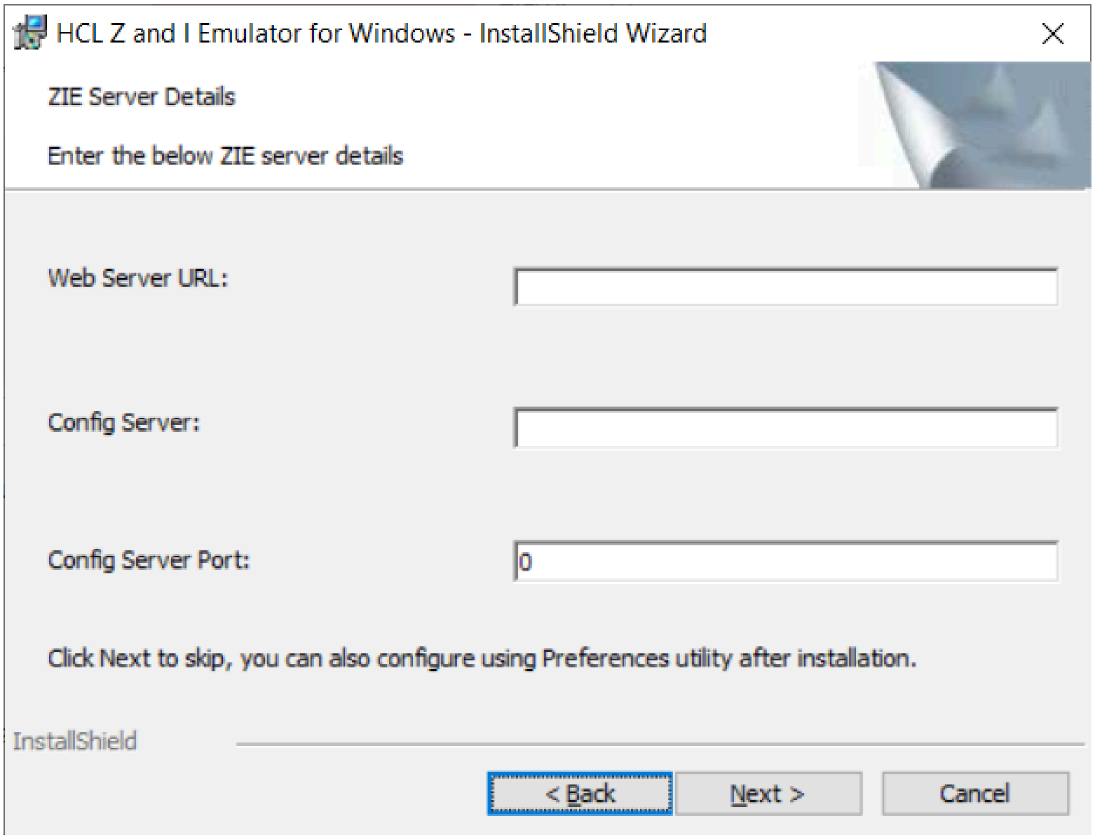
The steps provided in this document are applicable to all Windows 10 versions which are 64-bit OS level.

Prerequisites :

1. Download a copy of HCL Z and I Emulator for Windows 64-bit base package and HCL Z and I Emulator for Windows RP1.zip
2. A HCL Z and I Emulator for Web server is required for the Session Manager Online to work.
3. Create a folder (e.g. MPZiewin) in the ZIEWEB published directory.
4. Unzip and dump the HCL Z and I Emulator for Windows RP1 contents in *MPZiewin* folder.
5. Right click the *MPZiewin folder* > *Properties* > *Sharing* > *Advance Sharing* > *Put a check on Share this Folder*.
6. Click **OK** then **Close**.
7. Repeat steps 5-6 with ZIEWEB folder.
8. Obtain the IP address of the ZIEWEB server and use it at step 2 below.

Follow the steps below in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN):

1. There are two ways to input the ZIE Server configuration details. Choosing to go either way will have the same results.
   - During the ZIEWIN installation a new panel has been added.



HCL Z and I Emulator for Windows - InstallShield Wizard

ZIE Server Details

Enter the below ZIE server details

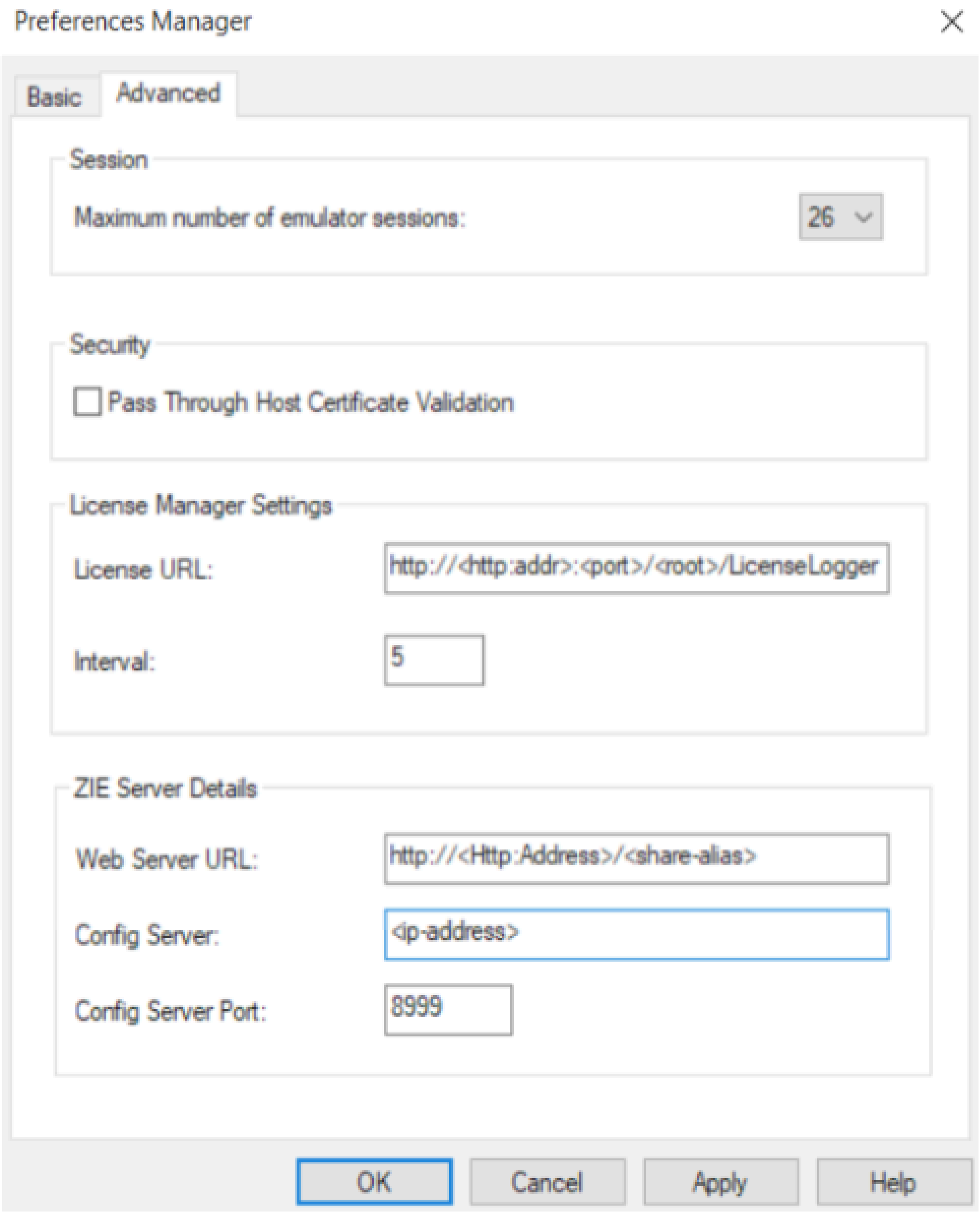
Web Server URL:

Config Server:

Config Server Port: 0

Click Next to skip, you can also configure using Preferences utility after installation.

- Preferences Manager - Click on Start > HCL Z and I Emulator for Windows > Preferences > Advanced



Maximum number of emulator sessions: 26

□ Pass Through Host Certificate Validation

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

2. Enter the configuration parameters based on the information below:
   - **Web Server URL** : The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file will be downloaded for installation. Installer or fix pack will be installed on the system by "Start or Configure Sessions - Online" program.>
   - **Config Server** : URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the HOD Embedded Server or on any configured Application Server.

     Example: http://< Application Server IP >/<Configured context root of the application>

For more details on WAR file deployment, refer to the technote <hyper-link>.

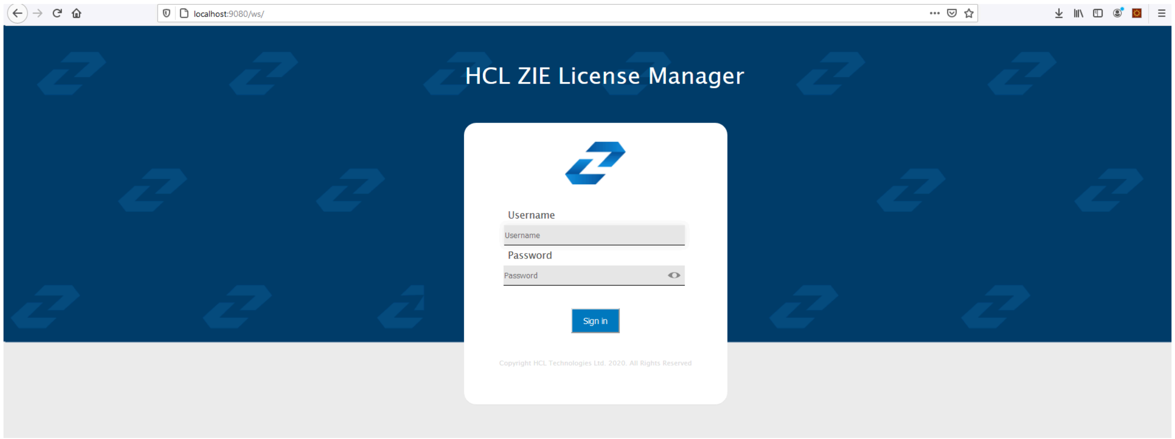
- **Config Server Port** : Port number of Application Server where interoperability module (.war file) is deployed.

  Example: 9080

3. Click **OK**.
4. *Open File Explorer* > *This PC* > *Map network drive.* Use the IP address of the Web Server along with the folder where the HCL Z and I Emulator for Windows RP1.msi is located. E.g. \\192.168.56.102\MPZiewin
5. Click **Save**.
6. Repeat steps 4-5 with ZIEWEB folder. E.g. \\192.168.56.102\ZIEWEB

Note: This completes configuring Managed HCL Z and I Emulator for Windows (ZIEWIN).

7. To validate that the configuration is correct, create a new Username or use an Existing User in the Session Manager Online. *Click on Start* > *HCL Z and I Emulator for Windows* > *Start or Configure sessions - Online*



Points to consider:

- When mapping the network drive ensure that client machine and ZIEWEB server is within the same network.
- The Session Manager Online checks for updates at startup. It is essential that HCL Z and I Emulator for Windows RP1 is in the ZIEWEB published directory.

Interoperability between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients

The interoperability feature allows the ZIEWin users to use the ZIEWin sessions from other HCL terminal emulator clients, such as ZIEWeb and ZIEWeb Client. ZIEWin users can use the “Session Manager Online” utility to store the new sessions and migrate the existing sessions to ZIE server, these sessions are then converted to ZIEWeb Session formats for the ZIEWeb and ZIEWeb Client usage.

Note: Interoperability feature is introduced in ZIEWin 2.1 version.

ZIEWin client communicates with the ZIE server over HTTP/HTTPS connectivity using JSON data format.

The interoperability feature is supported from ZIEWeb v2.1.0.0 & ZIEWeb Client v2.1.0.0 onwards and is applicable for 3270 Display, 5250 Display, 3270 Printer, 5250 Printer, and VT sessions.

Note: The session conversion happens only for ZIEWin to ZIEWeb sessions and not vice versa.

When the user stores the ZIEWin sessions using the “Session Manager Online” utility, they are converted to ZIEWeb sessions before saving them to the ZIE server . After storing to the ZIE server , users can log in from ZIEWin, ZIEWeb, or ZIEWeb Client to work with the stored ZIEWin sessions.

Using ZIEWin Sessions from ZIEWeb and ZIEWeb Client:

After the ZIEWin sessions are stored in the ZIE server , if any changes are made to the session definition from any of the clients, it is saved in the ZIE server . These session changes will be available to ZIEWin users after the next login.

Below is the list of supported parameters as part of the Interoperability feature.

Table 13. List of Supported Parameters for Interoperability

ZIEWIN Parameter	ZIEWEB Parameter
Primary Host Name or IP Address	Destination Address
Primary Port Number	Destination Port
Primary LU or Pool Name	LU or Pool Name
Screen Size	Screen Size
Host Code-Page	Host Code-Page
Auto-reconnect	Auto-reconnect
Backup 1 Host Name or IP Address	Backup 1 Destination Address
Backup 2 Host Name or IP Address	Backup 2 Destination Address
Backup 1 Port Number	Backup 1 Destination Port
Backup 2 Port Number	Backup 2 Destination Port
Backup 1 LU or Pool Name	Backup 1 LU or Pool Name
Backup 2 LU or Pool Name	Backup 2 LU or Pool Name
Enable Security	Protocol
Workstation ID	Workstation ID
Server Authentication	Server Authentication
Message Queue	Message Queue
Message Library	Queue Library
Send Personal Certificate to Server if it is Requested	Send a Certificate
Send Personal Certificate Trusted by Server	Certificate Source
Send Personal Certificate Based on Key Usage	Enable Key Usage
Machine Mode	Terminal Type (VT session)

Table 13. List of Supported Parameters for Interoperability (continued)

ZIEWIN Parameter	ZIEWEB Parameter
AutoWrap	AutoWrap (VT session)

Note: Only the listed parameters will be modified from ZIEWeb / ZIEWeb Client for a ZIEWin profile. If any other parameters are updated from ZIEWeb / ZIEWeb Client, there will not be any changes to the ZIEWin session. Users should modify the ZIEWin sessions either from ZIEWin or ZIEWeb / ZIEWeb Client at a time and should avoid simultaneous modifications from different clients.

Interoperability 2.1.0.0 Configuration Introduction:

ZIEWeb v3.0 (from v2.1.0.0 onwards) introduced interoperability between ZIEWin and ZIEWeb. This allowed ZIEWin sessions to be accessed through ZIEWeb and ZIEWeb Client after the session definitions were uploaded to the ZIE server .

Password provided during the user creation will be encrypted using AES 128-bit algorithm and will be sent to the server through the HTTP/HTTPS protocol as Json object. UID is added to the WS and BCH profile files for unique identification. Only Connection parameters are considered for the interoperability between ZIEWin and ZIEWeb Clients and vice-versa.

After the ZIEWin sessions are converted and stored in the ZIE server , any changes made to the common parameters from any of the clients will be saved on the ZIE server . These parameter changes will be available to ZIEWin users after the next login.

Steps to install:

1. Install the ZIEWeb v3.0.
2. Install the ZIEWin v3.0

WAR File Configuration:

The interoperability executable (**ZIEWeb_Interoperability.war**) is available under the lib directory of the product.

For Embedded Web Server:

If the Embedded Web Server is used, by default Interoperability application is running on context root "interop”. If the user needs to change the context root, add the following parameter to the configuration file **(*config.properties*)**, located in the ZIE server publish directory.

Example: InterOpContextPath=interop

The default ZIE server IP is 127.0.0.1 and the ZIE server port is 8999. If the user needs to connect to the ZIE server located on a different machine, then override the interoperability configuration by modifying the properties of ***“interop_overrides.xml”*** in the conf directory under the lib directory of the product.

Table 14. List of properties that can be used to configure the Interoperability

Property	Value	Description
ZIEWEB_SERVER_IP	127.0.0.1	ZIE server address

Table 14. List of properties that can be used to configure the Interoperability (continued)

Property	Value	Description
ZIEWEB_SERVER_PORT	8999	ZIEWEB Config Server port
Directory_Location	C:\\dir_location	Directory Location for logs

The user can utilize **ZIEWeb_Interoperability.war** file (available under the lib directory of the product) to deploy to different application servers such as WAS/Tomcat.

For WebSphere Application Server (WAS):

1. Log in to **WebSphere Application Server**.
2. Go to **Applications**.
3. Click WebSphere enterprise applications under **Application Type**.
4. Select **ZIEWeb_Interoperability.war file**.
5. Click on Initialize parameters for servlets link under **Web Module Properties** section.
6. Enter the required values.

Supported Application Servers: *Apache Tomcat and WAS*.

Limitations

1. Only connection parameters are considered for interoperability between ZIEWin and ZIEWeb and vice-versa.
2. Session creation from ZIEWeb / ZIEWeb Client will not be converted to a ZIEWin session.

Known Issues

1. For stored ZIEWin sessions, changes to any session parameters (not only the listed parameters) from ZIEWeb / ZIEWeb Client will be overridden or set to default when there is an update from **"Session Manager Online"** (ZIEWin Client).
2. If there are simultaneous profile updates from any of the two clients, the most recent update will be saved as the final copy in the ZIE server .
3. Modifications done in multiple sessions (add, delete sessions, or rename) from the ZIEWeb Clients do not reflect in the ZIEWin Client.
4. Saving/renaming profiles with special characters (Ex: \ / : * ? " < > |.) in ZIEWeb/ ZIEWeb Clients will result in unexpected behavior in the ZIEWin Client.

   How to setup Managed HCL Z and I Emulator for Windows (ZIEWIN)

   HCL Z and I Emulator for Windows uses Session Manager Online dialog to provide easy access to workstation profiles and batch files on the ZIE Server. With Session Manager Online users can create or start a single or multiple sessions and or batch files. Users can create their own profile to the ZIE server and migrate existing files such as the workstation profiles (*.WS) and batch files (*.BCH) that were stored on the ZIE Server.

   This "How To" document aims to supplement additional detailed information in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN) as referenced below.

   Steps to Install Using Managed: Refer to Planning to Install Z and I Emulator for Windows on page 42

The steps provided in this document are applicable to all Windows 10 versions which are 64-bit OS level.

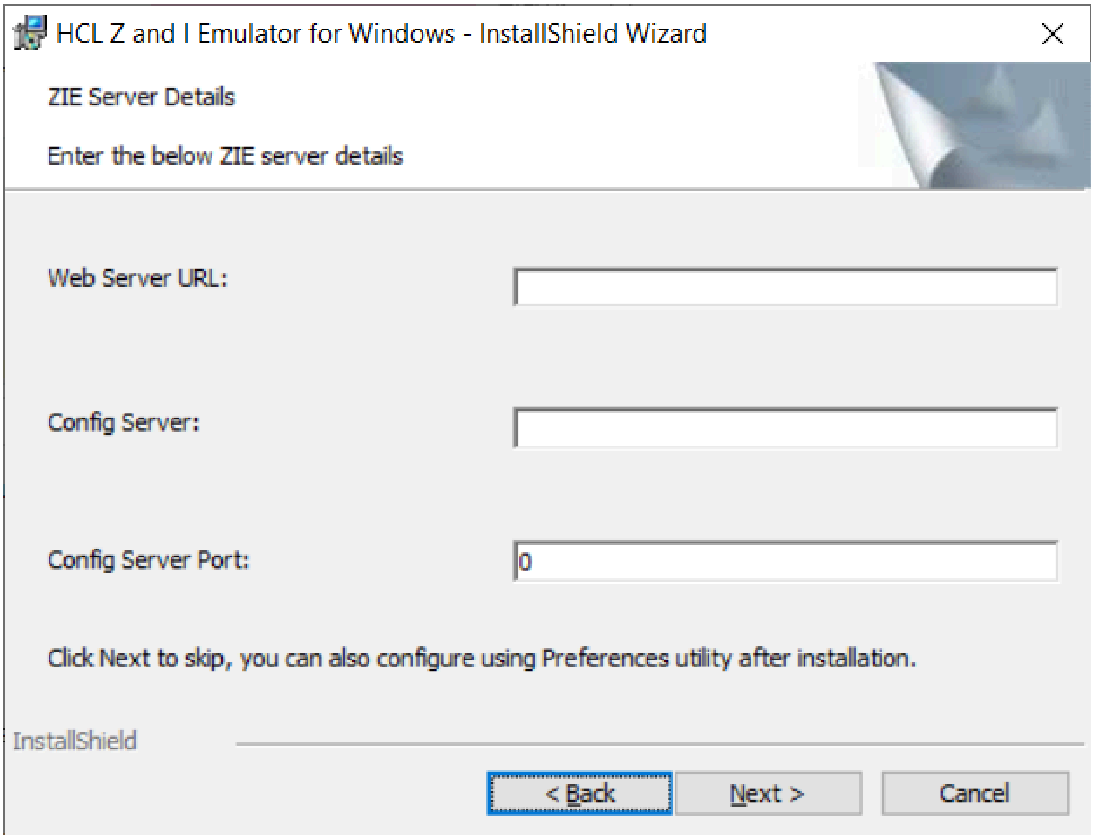
Prerequisites :

a. Download a copy of HCL Z and I Emulator for Windows 64-bit base package and HCL Z and I Emulator for Windows RP1.zip
a. A HCL Z and I Emulator for Web server is required for the Session Manager Online to work.
b. Create a folder (e.g. MPZiewin) in the ZIEWEB published directory.
c. Unzip and dump the HCL Z and I Emulator for Windows RP1 contents in *MPZiewin* folder.
d. Right click the *MPZiewin folder > Properties > Sharing > Advance Sharing > Put a check on Share this Folder*.
e. Click **OK** then **Close**.
f. Repeat steps 5-6 with ZIEWEB folder.
g. Obtain the IP address of the ZIEWEB server and use it at step 2 below.

Follow the steps below in setting up Managed HCL Z and I Emulator for Windows (ZIEWIN):

There are two ways to input the ZIE Server configuration details. Choosing to go either way will have the same results.

- During the ZIEWIN installation a new panel has been added.



HCL Z and I Emulator for Windows - InstallShield Wizard

ZIE Server Details

Enter the below ZIE server details

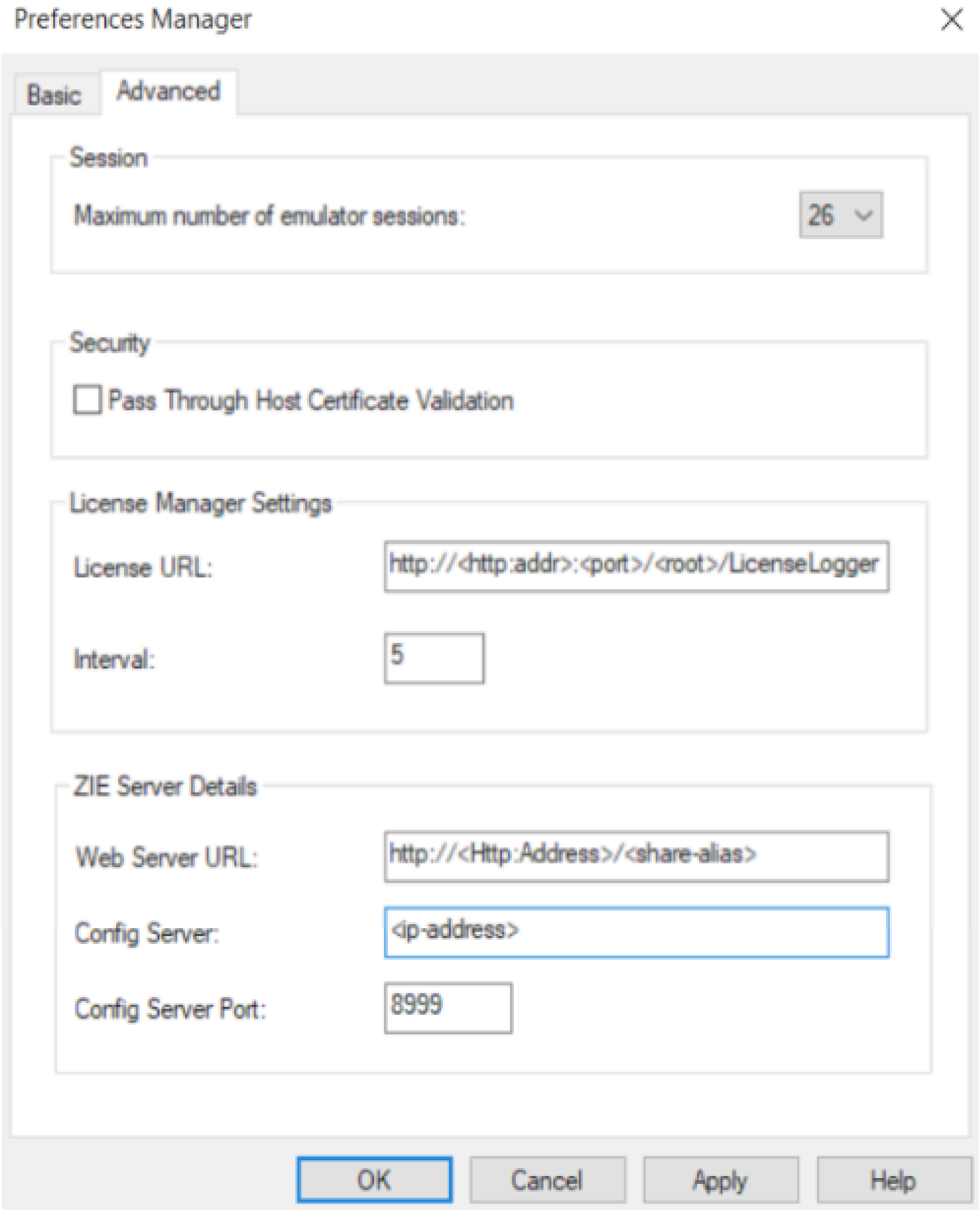
Web Server URL:

Config Server:

Config Server Port: 0

Click Next to skip, you can also configure using Preferences utility after installation.

- Preferences Manager - Click on Start > HCL Z and I Emulator for Windows > Preferences > Advanced



Preferences Manager

Session

Maximum number of emulator sessions: 26

Security

□ Pass Through Host Certificate Validation

License Manager Settings

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

ZIE Server Details

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

a. Enter the configuration parameters based on the information below:
   - **Web Server URL** : The URL of the Web Server from where HCL Z and I Emulator for Windows fix pack file will be downloaded for installation. Installer or fix pack will be installed on the system by "Start or Configure Sessions - Online" program.>
   - **Config Server** : URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the HOD Embedded Server or on any configured Application Server.

     Example: http://< Application Server IP >/<Configured context root of the application>

     For more details on WAR file deployment, refer to the technote <hyper-link>.

- **Config Server Port** : Port number of Application Server where interoperability module (.war file) is deployed.

  Example: 9080

b. Click **OK**.

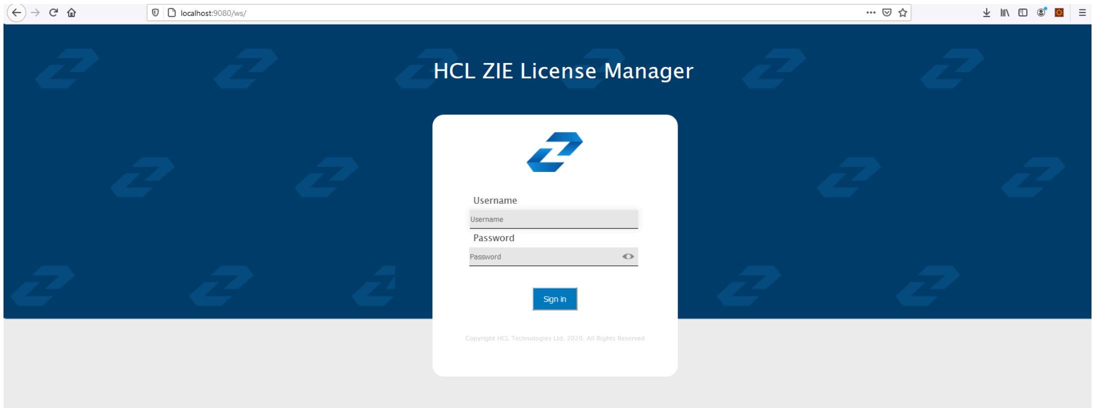
c. *Open File Explorer > This PC > Map network drive.* Use the IP address of the Web Server along with the folder where the HCL Z and I Emulator for Windows RP1.msi is located. E.g. \\192.168.56.102\MPZiewin

d. Click **Save**.

e. Repeat steps 4-5 with ZIEWEB folder. E.g. \\192.168.56.102\ZIEWEB

Note: This completes configuring Managed HCL Z and I Emulator for Windows (ZIEWIN).

f. To validate that the configuration is correct, create a new Username or use an Existing User in the Session Manager Online. *Click on Start > HCL Z and I Emulator for Windows > Start or Configure sessions - Online*



Points to consider:

- When mapping the network drive ensure that client machine and ZIEWEB server is within the same network.
- The Session Manager Online checks for updates at startup. It is essential that HCL Z and I Emulator for Windows RP1 is in the ZIEWEB published directory.

Utilities

Z and I Emulator for Windows provides the following utilities:

32-bit ODBC Administrator

Allows you to add, configure, or delete an ODBC data source.

Scratch Pad

The Scratch Pad is a lightweight text editor with the capability of normal edit operations, such as cut, copy and paste.

Note that Scratch Pad requires .NET Framework 3.5 or lower versions installed. The menu item is grayed out in Windows 8 or 8.1 as .NET Framework 3.5 is not available by default on these operating systems. Install .NET Framework 3.5 to resolve the problem.

Multiple Sessions

Provides the capability to run multiple host sessions using a single icon.

ZipPrint

Allows you to print PROFS® notes, calendars, CMS files, XEDIT workspaces, and 3270 session screens.

Convert Macro

Allows an existing Z and I Emulator for Windows Macro file to be converted to an XML or VBScript file.

Data Transfer

Transfers data from the iSeries™, eServer™ i5, or System i5™ to your workstation, or from your workstation to the iSeries™, eServer™ i5, or System i5™ (record-level data transfer).

iSeries Connection Configuration

Define connections to each iSeries™, eServer™ i5, or System i5™ host that will use the data transfer function.

Preferences

Configuration of certain advanced parameters.

PcsSound

The PcsSound utility, available under the product installation directory, allows you to:

- Assign labels to sound events.
- Associate sound files to sound events.
- Cleanup sound labels from the registry.
- Save the current sound scheme to a file.
- Restore a saved sound scheme.

Note: Saving and restoring a sound scheme will be useful while changing Windows themes.

32-Bit ODBC Administrator

ODBC is a programming interface that enables applications to access data in database management systems that use Structured Query Language (SQL) as a data access standard.

Use the following steps to set up the Z and I Emulator for Windows ODBC data source:

1. Select the **32-bit ODBC Administrator** icon from the Windows® Control Panel. The Data Sources window appears.
2. Select IBM® DB2® ODBC Driver data source in the Data Sources (Drivers) list. Then click **Finish**.
3. Click **Add Database**. The Z and I Emulator for Windows Add Database SmartGuide appears to prompt you through the set up.
4. Select the **Manually Configure a Connection to a DB2® Database** radio button.
5. Specify the information to set up the data source, by clicking **Next**.
6. When you are finished specifying the information, click **Done**.
7. You will be prompted to test the connection. To test the connection, click **OK**.

Note:

1. Z and I Emulator for Windows uses ODBC 32-bit drivers. Applications, such as Lotus® 1-2-3® included in the Lotus® SmartSuite® 96 package, require a 16-bit driver and will not work with Z and I Emulator for Windows. You should see your product vendor for a version that utilizes 32-bit ODBC drivers (Lotus 1-2-3® included in Lotus® SmartSuite® 97 for example).

Multiple Sessions

The Z and I Emulator for Windows Multiple Sessions Batch Program enables you to start several host sessions by clicking a single icon; the necessary commands are specified in a batch file (.BCH). You can include in a batch file other programs, which may communicate with a host session using the DDE or EHLLAPI interface.

ZipPrint

ZipPrint is a 3270 utility that enables you to print PROFS® notes, calendars, OV documents, CMS files, XEDIT workspaces, and 3270 session screens. By default, it uses the Windows® printer currently set up for the host session, but you can change it if you wish.

You do not have to install ZipPrint—you just start it. It adds itself to the menu bar of the sessions for which you define it, so you can use it in the same way as any other menu bar function. You must start ZipPrint before you start a session in which you want to use it.

ZipPrint needs the DDE/EHLLAPI, so you must make sure that this is enabled for the sessions for which you want to use ZipPrint. (It is enabled by default, but you should check that it has not been turned off.)

For more information about ZipPrint, see ZipPrint (3270 Only) on page 76.

File Transfer Considerations

ZipPrint uses the Z and I Emulator for Windows file transfer function to print VM/CMS notes and files. In order for this function to work correctly, you should use the VM/CMS host type for 3270 file transfer.

In the emulator session window, click **Settings → Transfer**. On the **File Transfer Settings → General** tab, select **VM/CMS** from the **Host Type** drop-down list.

On slow communications lines, if you are using a large packet or block size, you may experience a file transfer timeout. If you do, you should increase the timeout delay. To change the timeout delay, do the following:

1. From the session menu, click **Settings → Transfer**.
2. On the **General** tab, change the **File Transfer Timeout** to 150 seconds.

Convert Macro

The Convert Macro utility enables you to convert an existing Z and I Emulator for Windows Macro file to XML or a VBScript file.

Note: Macros that are converted to XML are intended for use in ZIEWeb (Z and I Emulator for Web) and will not function in Z and I Emulator for Windows emulation sessions. Use the ZIEWeb Macro Manager to import a converted Z and I Emulator for Windows macro into ZIEWeb. These converted macros will not appear in the list of available Z and I Emulator for Windows macros.

To use the conversion utility, click **HCL Z and I Emulator for Windows → Utilities → Convert Macro**.

To convert an existing macro to XML or a VBScript, do the following:

1. Select the name of an existing macro to be converted.

   Note: The macro must exist in the application data directory specified during installation.

2. Select **VBScript** or **XML** as the type of macro to which to convert.
3. Click **Convert**.
4. Enter a name for the new XML file or VBScript or accept the generated name. The extension will be added automatically.

   Note: When saving a converted XML macro you can choose where you would like to save it. You should not change the location of the converted VBScript macros.

5. Click **Save**.
6. Repeat the procedure to convert another macro, or click **Close** to end the application.

Data Transfer

Z and I Emulator for Windows Data Transfer enables you to transfer data between an iSeries™ system and your workstation. To use the Data Transfer function, select the **Data Transfer** icon.

Transferring data is quite different from transferring files, which is described in Transferring Files on page 83.

Requirements

Before you can transfer data with Z and I Emulator for Windows:

- IBM® PC Support/400 (5738-PC1) must be installed on your iSeries™, eServer™ i5, or System i5™, unless OS/400® Version 3 (or later) or i5/OS™ is installed.

There are two types of data transfer, depending on the direction of the transfer.

Data sending

Data is transferred from your workstation to the iSeries™, eServer™ i5, or System i5™.You can transfer data to any of the following destinations:

- Existing members in an existing iSeries™, eServer™ i5, or System i5™ physical file
- New members in an existing iSeries™, eServer™ i5, or System i5™ physical file
- New members in a new iSeries™, eServer™ i5, or System i5™ physical file

Note: You cannot transfer data from a workstation file to an iSeries™, eServer™ i5, or System i5™ logical file.

Data receiving

Data is transferred from the iSeries™, eServer™ i5, or System i5™ to your workstation.

While receiving data from the host, you can specify the data to be received and where the data is to be output.

Receivable data includes:

- An entire iSeries™, eServer™ i5, or System i5™ file
- Part of an iSeries™, eServer™ i5, or System i5™ file
- Data combined from several iSeries™, eServer™ i5, or System i5™ files
- Summary of record groups

Specify the following output destinations:

- Display
- Disk
- Printer

Also, you can specify the numeric value format.

For more information about data transfer, refer to *Emulator User's Reference*.

iSeries Connection Configuration Utility

The iSeries™ Connection Configuration Utility is used to define connections to each iSeries™, eServer™ i5, or System i5™ host that will use the data transfer function. The connection definitions are saved in an .NDC file in ASCII format.

You can use this utility for TCP/IP connections.

For more information on Data Transfer see Data Transfer on page 116.

To use the utility, click **Start → Programs → HCL Z and I Emulator for Windows → Utilities → iSeries Connection Configuration**; the resulting iSeries™, eServer™ i5, or System i5™ configuration screen has the following options:

Show IP Host Connections

Click this button to display and configure IP connections to the iSeries™, eServer™ i5, or System i5™ host.

Add

Click the type of connection and then click **Add**. Enter the **Host name** and **Alias** in the resulting dialog box.

Modify

Select a host name from the connection list, and then click **Modify** to edit the **Host name** and the **Alias** in the .NDC file.

Remove

Select a host name from the connection list, and click **Remove** to delete this connection definition from the .NDC file.

Note: When disabling a connection, if you want to preserve the connection definition but disable the connection, clear the checkbox next to the name in the connection list.

Global Parameters

Click this button to edit the **Extension list** and **Cache size**.

Extension List

The extension list parameter specifies the extension of a file on an iSeries™, eServer™ i5, or System i5™. You can specify more than one extension parameter in the extension list. The code pages of files with the specified extensions are translated from the EBCDIC code page to the ASCII code page when the file is transferred between the iSeries™, eServer™ i5, or System i5™ and the client. Up to three characters are allowed. There are two special cases:

- A dot alone (`.`) indicates that data for files having no extension should be converted.
- The character pair `.*` indicates that data for all files should be converted.

Cache Size

The cache size parameter specifies the number of kilobytes of iSeries™, eServer™ i5, or System i5™ data that is buffered in the read-ahead cache of the client. The default is 256 KB; the maximum is 4 MB. A value of zero requests that no cache be used. iSeries™, eServer™ i5, or System i5™ data can be retrieved in amounts that are first cached locally on the client. The client retrieves the data from the cache to populate the local device. This read-ahead caching reduces the number of times the client has to access the iSeries™, eServer™ i5, or System i5™ to retrieve the data.

Preferences

The Preferences utility provides a method for changing configuration and setup items.

Note:

1. Preferences set with the Preferences Utility pertain to you whenever you log on to the same user ID on the affected workstation; they apply to all of your sessions while you are logged on.
2. Preferences set using the session **Edit** menu apply to all sessions controlled by the workstation profile created or changed while using a session—when that session profile is used again, the preferences are applicable, regardless of the user ID at the time.
3. One of the capabilities of the Preferences Utility is to allow specification of a directory to be used for storing your profiles; this allows full control of your environment.

To access the Preferences Utility, click **Programs -> HCL Z and I Emulator for Windows -> Utilities -> Preferences** from the Windows **Start** menu. Select the **Basic** tab to change the preferences and select **Advanced** tab to change the maximum number of emulator sessions and pass through host certification validation.

Basic

Emulator Profile File Location

If the All User application data directory location was selected during installation of Z and I Emulator for Windows, you can specify the default location of workstation profiles.

Macro/Script Location

You can specify where emulator macro and script files are to be placed. This directory will be common to all sessions.

By default, macros are placed in the application data directory specified during the installation of Z and I Emulator for Windows, and the macro/script location field value is blank.

User Interface Language

You can view the language of the installed package by selecting **Help > About Z and I Emulator for Windows** from the session menu bar.

If your system was installed with multiple-language support, the **Basic** property page shows a section labeled **Select a default user interface language**. If you click on the radio button for **Z and I Emulator for Windows User Interface Language Preference**, it also shows a drop-down list box, with the language selected that is currently being used; you can select any other language from the list, and that language will be used for the user interface when you subsequently restart Z and I Emulator for Windows. Or you can click the radio button **User Default from Regional Settings** if you want the language to be that specified in the Windows® settings. You can also click the radio button **Post a Language Selection Dialog for each process**; this results in a pop-up dialog each time you start a new application from the Z and I Emulator for Windows group of programs.

Note: You might receive a warning message, if the selected language is incompatible with your system's current code page. You can ignore this, if you are planning to reboot your computer to select a new system locale with a compatible code page.

Advanced

Maximum number of emulator sessions

Specify the maximum number of Telnet emulator sessions. It can be either 26 or 52 sessions. Default is 26 sessions.

Pass Through Host Certificate Validation

Choose whether to enable or disable the default certificate validation process during SSL/TLS handshake. Default is enable the certification validation. Applicable only for Microsoft schannel provider.

Note: By default, schannel (MSCAPI) is responsible for validating the host certificate chain received during SSL/TLS handshake. Schannel runs several checks on the received certificate chain, one of which verifies that the signature affixed to the certificate is valid. The hash value computed on the certificate contents must match the value that results from decrypting the signature field using the public component of the issuer. To perform this operation, the user must own the public component of the issuer, either through some integrity-assured channel or by extracting it from another (validated) certificate. The default certificate validation process is exhaustive and runs several checks on the host certificate chain to successfully validate it. Enable this option, the user must effectively suppress the default validation done by schannel and the identity of the host would not be verified. As we are skipping the host certificate validation, the status bar is updated with the following message: “Skip the certificate validation since pass-through host certificate validation option is enabled.” Using this option is not recommended.

When the Host certificate is not added to the trusted root and “Pass-Through Host Certificate Validation” is enabled, a pop-up is displayed. Users can suppress this pop-up by adding the

"SuppressPassThroughPopup=Y" keyword under the "[Security]" section in the pcswin.ini file. By default, SuppressPassThroughPopup is disabled.

Configuration of License Manager settings

The License Manager settings can either be configured by providing the required server details in the 'InstallShield Wizard' at the time of ZIEWin installation itself, or can be added/updated in the 'License Manager Settings' section of the Advanced tab within the Preferences.

License Manager settings

- **License URL :**

  Specifies the HTTP URL of the License Manager Server to which the HCL Z and I Emulator for Windows session sends the license parameters. It is mandatory to configure this field to use the product. If this field is not configured the emulator session initiation will be aborted.

  For Example:

  http://<appserver-address>:<port-num>/<context-root>/LicenseLogger

  where,

  - *<appserver-address>* is the hostname or IP address of the server on which the License Manager is installed,
  - *<port-num>* is the port that is specified during the deployment of the application server and,
  - *<context-root>* is the location name that can be configured by the Administrator.

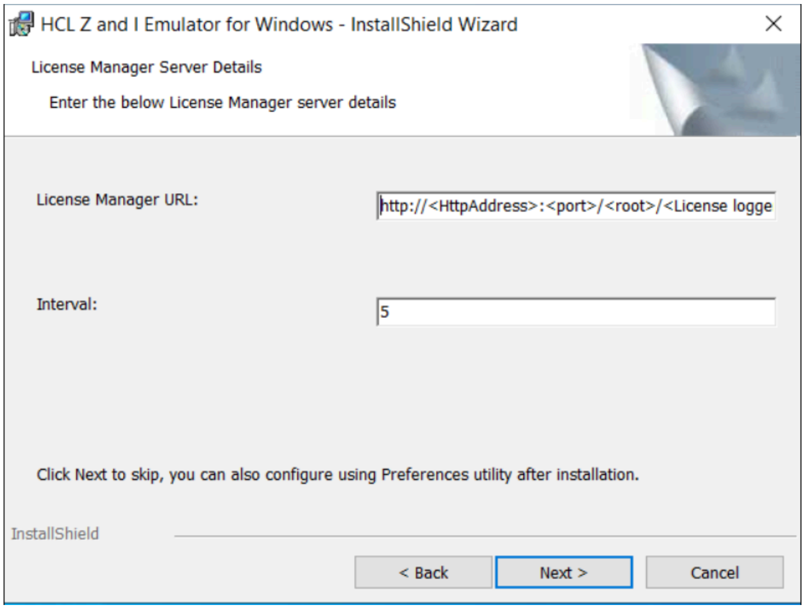
  For more information on the Installation of HCL ZIE License Manager, refer to HCL ZIE License Manager on page 6.
- **Interval :**

  Specifies the time period in minutes, after which the HCL Z and I Emulator for Windows session sends the license parameters. It is the request interval after which the server marks the client as timeout if the request is not sent. The minimum value is 5 minutes (which is the default), and the maximum value is 30 minutes.

  Note: The License Manager Settings set using the *'Preferences'* utility takes precedence over the values set during installation. If the installation is custom 'User Installation' where the application data location is *%appdata%* in the User directory, the values set in the 'Preferences' utility is applicable only for the current user.

1. **Configure the License Manager Settings during the GUI installation of ZIEWin**

   The *'License Manager URL'* and *'Interval'* fields can be set during installation, in the 'License Manager Server Details' panel. Users may skip the License Manager configuration at installation time and choose to configure it after installation using the 'Preferences' utility.



HCL Z and I Emulator for Windows - InstallShield Wizard

License Manager Server Details

Enter the below License Manager server details

License Manager URL: http://<HttpAddress>:<port>/<root>/<License logge

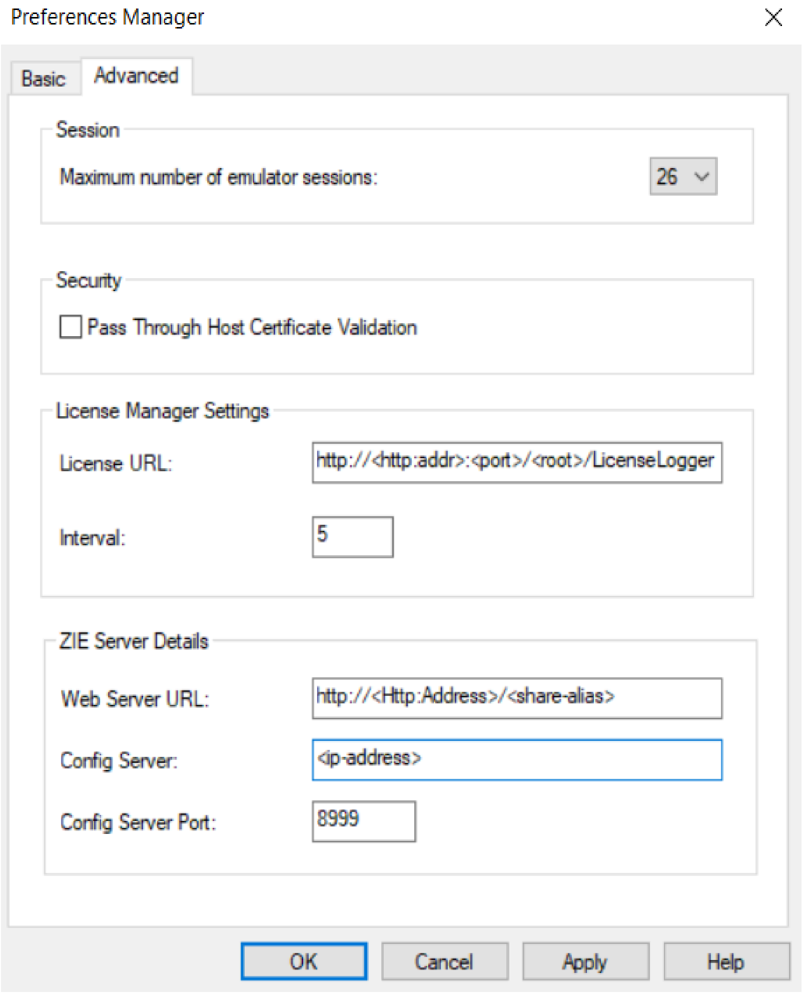
Interval: 5

Click Next to skip, you can also configure using Preferences utility after installation.

Figure : License Manager Settings section in the ‘Preferences Manager' panel ZIEWin.

2. **Configure the License Manager Settings using “Preferences” utility**

   The *'License URL'* and *'Interval'* fields can be set using the 'Preferences' utility, by entering values to the respective fields under the 'License Manager Settings' in the 'Advanced' tab.



Session

Maximum number of emulator sessions: 26

Security

☐ Pass Through Host Certificate Validation

License Manager Settings

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

ZIE Server Details

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

Figure : License Manager Settings section in the 'Preferences Manager' panel ZIEWin.

ZIE Server Details

Configuration of ZIE Server details

The ZIE Server can either be configured by providing the required server details in the 'InstallShield Wizard' at the time of ZIEWin installation itself, or can be added/updated in the 'ZIE Server Detail' section of the Advanced tab within the Preferences.

- **Web Server Details:** The URL of the Web Server from where Z and I Emulator for Windows installer or fix pack file will be downloaded for installation. Z and I Emulator for Windows prompts the user with a Pop-Up message for an Auto-Upgrade, if a newer version of ZIEWin is available on the Web Server. Upon receiving a confirmation from the user to upgrade, the latest version of installer or fix pack will be installed on the system by Z and I Emulator for Windows Session Manager ONLINE.
- **Config Server:** URL of the Application Server/Embedded Server, on which interoperability module (.war file) is deployed. It can be deployed on the Embedded Server or on any configured Application Server.

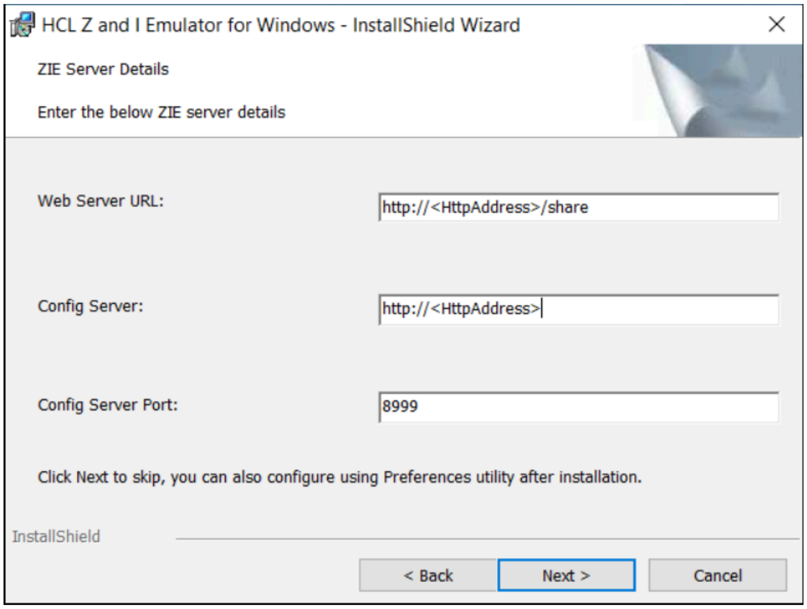
Example: http://< IP >/<Configured context root of the application>

For more information on ZIEWeb - ZIEWin interoperability and common parameters, refer to Interoperability between HCL Z and I Emulator for Windows and HCL Z and I Emulator for Web Clients on page 30

- **Config Server Port:** Port number of Application Server where interoperability module (.war file) is deployed. Example: 9080.

1. **Configure the ZIE Server Settings during the GUI installation of ZIEWIN**

   During the installation in the "ZIE Server Details" install panel user can configure the "Web Server URL", "Config Server" and "Config Server Port" fields. User may skip configuration during the installation and can configure using "Preferences" utility after installation.

   ZIE Server Details

   Enter the below ZIE server details

   Web Server URL: http://<HttpAddress>/share

   Config Server: http://<HttpAddress>

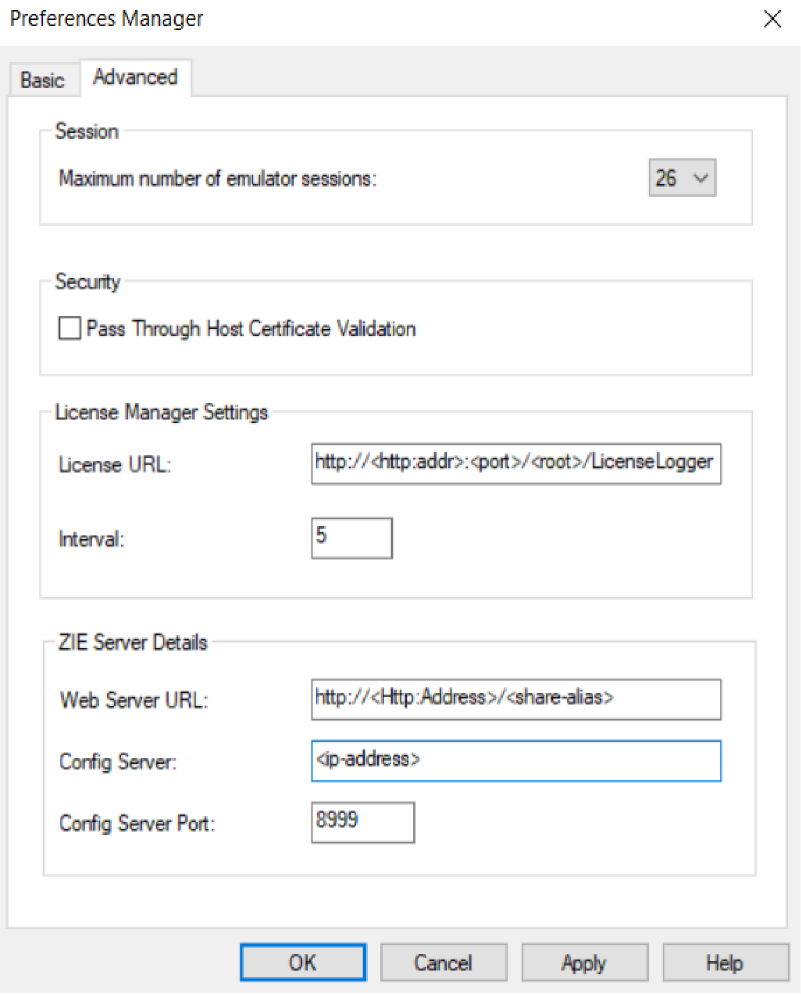
   Config Server Port: 8999

   Click Next to skip, you can also configure using Preferences utility after installation.

   Figure : ZIE Server Details section in the 'InstallShield Wizard' of ZIEWin.

2. **Configure the ZIE Server Settings using "Preferences" utility**

   In the "Preferences" utility, go to "Advanced" tab to find the section "ZIE Server Details", user can configure the fields "Web Server URL", "Config Server" and "Config Server Port" here. Please refer to the figure below.



Preferences Manager

Session

Maximum number of emulator sessions: 26

Security

□ Pass Through Host Certificate Validation

License Manager Settings

License URL: http://<http:addr>:<port>/<root>/LicenseLogger

Interval: 5

ZIE Server Details

Web Server URL: http://<Http:Address>/<share-alias>

Config Server: <ip-address>

Config Server Port: 8999

Figure : License Manager Server and ZIE Server Details section within the Advanced tab of Preferences utility in ZIEWin.

Standby/Hibernate

Choose whether to prompt for acceptance when the system attempts to go to standby or hibernate (power saving) mode. If you select **Standby/Hibernate without prompting**, Z and I Emulator for Windows allows the system to standby or hibernate without prompting you, even if sessions are connected. By default, this option is clear.

See Power Management on page 68 for more information.

Z and I Emulator for Windows FTP client

The Z and I Emulator for Windows FTP client implements the client functionality specified by File Transfer Protocol (FTP), which is the standard protocol for transferring files to and from remote machines running FTP servers. The FTP client enables file and directory upload and download, and directory navigation of remote and local file systems.

The Z and I Emulator for Windows FTP client supports the following servers:

- UNIX
- iSeries (AS/400)
- Windows
- z/OS MVS
- VMS

The following limitations apply:

- Secure connections (SSL/TLS) are not supported.
- The local file list does not support listing multiple local or LAN-attached drives.

  To view files on a different drive, type the drive letter of the drive you would like to view in the Directory field and click Enter. The new drive is displayed in the local file list.
- Code page conversion for files is not supported.
- Directory transfer is not supported on systems that do not have directory structures similar to Windows and UNIX. Such systems include the following:
  - VM
  - OS/390 or z/OS MVS services
  - OpenVMS
  - i5/OS and OS/400 Library File System

Command Line FTP

The command line FTP is used to achieve the FTP functionality over the command line. It is used to transfer files using FTP to and from a host with a UNIX file system, using pcsftpcmd.exe. It can be invoked from the command line using a set of parameters and switches. The functions supported by the executable are:

- Download a file
- Upload a file
- Delete a file (host side)
- Create a new directory (host side)

The necessary arguments for invoking the executable are:

hostname

This is the first argument and you need to specify the FTP server host name to which you want to connect.

username / password

These are the second and third arguments, which specify the user credentials to access the given host.

operation

In this argument you can specify the operation that needs to be performed. Possible operations are:

- **/d** - This switch is used for downloading a file from the host to the client system. This switch should be followed by the local directory where the file needs to be downloaded, a space and the complete path of the host file along with the name of the file that needs to be downloaded.
- **/u** - This switch is used for uploading a file from the client system to the host server. This switch should be followed by the complete path of the local directory along with the name of the file that needs to be uploaded, a space and the host directory where the file needs to be uploaded.
- **/FILE** - This switch is used to specify that a set of FTP commands are contained in a separate file and needs to be executed. This switch is followed by the complete path and the name of the file that contains the batch commands.
- **MKDR** - This switch is used to create a new Directory at the host. This is followed by the name of the directory to be created.
- **DELE** - This switch is used to delete a file from the host. This switch is followed by the complete path along with the name of the file to be deleted.

The transfer mode is set to AUTOMATIC, that is, the executable automatically checks to see if the extension of the file to be transferred is one of .log, .ini, .txt, .bat, .inf, in which case the mode is set to ASCII. For everything else it is set to BINARY mode.

Use one of the following switches to invoke help:

- -?
- /?
- -HELP

The issue here is that the password would be in plain text, which would be a security concern which can be overcome by requesting the customer to explicitly enter the password when the connection is being made.

The initial command line FTP being developed would entertain non-secure connections alone and would not support secure connections. The command line FTP currently under development would support only Windows/Unix style file systems. Other file systems would be supported in future releases.

If the user does not want to give the password in plain text along with the other parameters, a - can be entered in the password field, and then run the command which asks for the user to input the password dynamically.

The command line FTP is currently available only in English language.

Messages

Online messages are displayed during Z and I Emulator for Windows sessions, but a message does not always mean an error occurred. For example, a message might tell you that an operation is in progress or has been completed. A message can also prompt you to wait for the completion of an operation.

Press F1 to display help for the messages that appear.

Security-Related Messages

Z and I Emulator for Windows optionally utilizes Transport Layer Security (TLS) or Secure Sockets Layer (SSL) to establish sessions with servers; this might require input from you (for example, a password). Refer to *Administrator's Guide and Reference* for details.

Functions Restricted by System Policies

If your workstation is centrally administered, you may be shown a warning or pop-up error message whenever you attempt to use certain restricted functions. For example, if your ability to remap the keyboard is restricted, a message would be displayed when you select **Keyboard** from the **Settings** menu.

Contact your system administrator for further guidance. For further information about the systems policies provided with Z and I Emulator for Windows, refer to *Administrator's Guide and Reference*.

System Error Messages

If a page fault or similar system error message appears in a pop-up window, you can copy its contents into the Windows® clipboard. Use the following procedure:

1. Click **Details Command** on the pop-up window.
2. Mark the text that you want to copy.
3. Right click the marked text and then click **Copy**.
4. Start an editor, such as Notepad, and click **Edit → Paste**.
5. Save the file in case an HCL service representative needs this information to diagnose your problem.

OIA Messages

Z and I Emulator for Windows displays messages in the operator information area (OIA) or in a pop-up window. Messages from Z and I Emulator for Windows are displayed in the message window; messages from the host system are displayed in the OIA of the session window.

The bottom line of the session window is the OIA. The OIA indicator indicates the status of Z and I Emulator for Windows and information about the workstation, host system, and attachment method.

All of the OIA indicators, reminders, and messages are described in the online help.

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Installation Guide

Introduction

Z and I Emulator for Windows provides 3270, 5250, and VT emulation, connecting to z/OS™, z/VM™, eServer™ i5, iSeries™, System i5™, zSeries™, and ASCII systems.

Z and I Emulator for Windows uses Microsoft® Windows Installer technology for all installation procedures. This book details how to successfully install and customize installation of Z and I Emulator for Windows using the Windows Installer service. For more information on the Windows Installer service, see Microsoft Windows Installer on page 136. For detailed information on the Z and I Emulator for Windows product functionality, refer to *Quick Beginnings*.

The following sections discuss getting help when you are installing, configuring, or using Z and I Emulator for Windows.

Information Center

You can find documentation and links to other resources at the Z and I Emulator for Windows Information Center, at the following address:

https://help.hcltechsw.com/zie/ziewin/3.0/index.html

The Information Center contains reference material that is not found in this book, such as keyboard layouts and host code page tables.

The Z and I Emulator for Windows Information Center provides information in English, Chinese (Simplified), Chinese (Taiwan), Japanese and French.

Planning to Install Z and I Emulator for Windows

This chapter describes the companion products provided with Z and I Emulator for Windows and topics that should be considered before installing Z and I Emulator for Windows Version 3.0.

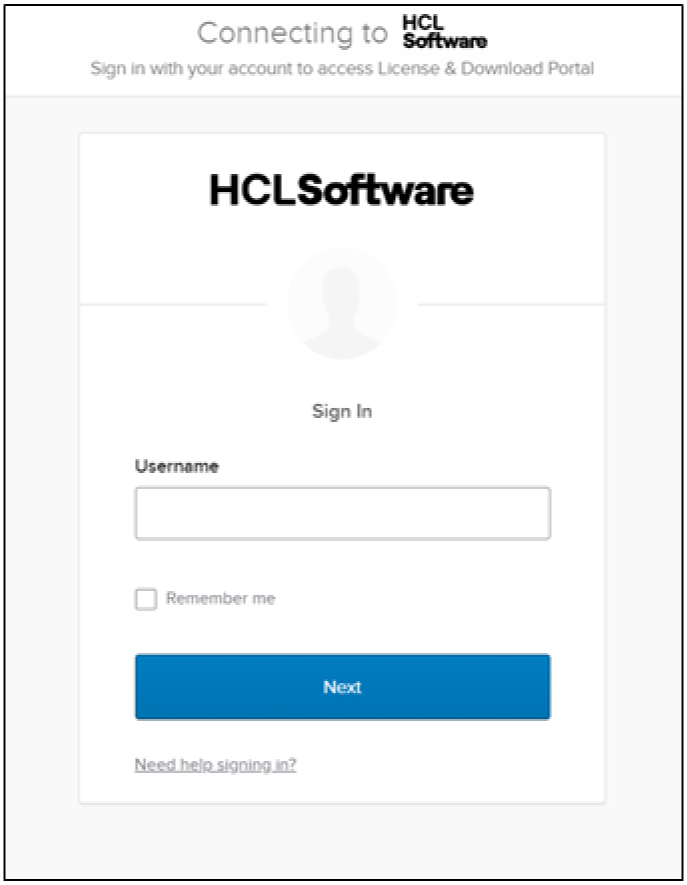
For instructions to install the HCL ZIE License Manager, refer to Installation of HCL ZIE License Manager on page 148, and refer the topic Configuration of License Manager settings to configure the ZIE License Manager for ZIEWin.

Downloading HCL Z and I Emulator for Windows

This section describes the procedure for downloading the HCL Z and I Emulator for Windows from the HCL Software License & Download Portal.

Execute the following steps to download the Z and I Emulator for Windows:

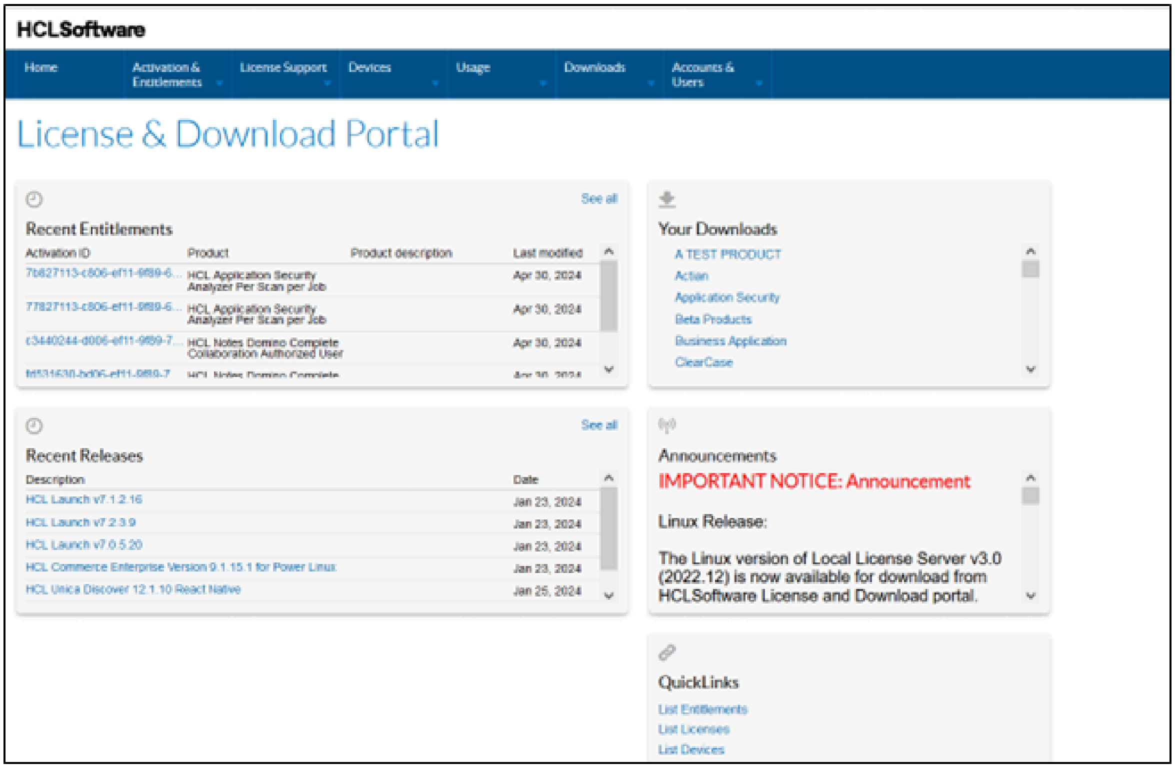
1. Browse the https://hclsoftware.flexnetoperations.com/flexnet/operationsportal/logon.do URL.
2. The **Login** page appears.
3. Enter the authorized login credentials and click **Login**.



Connecting to HCL Software

Sign in with your account to access License & Download Portal

4. After successful login, the homepage appears.



Recent Entitlements

Activation ID	Product	Product description	Last modified
7b827113-c806-ef11-9f89-6...	HCL Application Security Analyzer Per Scan per Job		Apr 30, 2024
77827113-c806-ef11-9f89-6...	HCL Application Security Analyzer Per Scan per Job		Apr 30, 2024
c3440244-d006-ef11-9f89-7...	HCL Notes Domino Complete Collaboration Authorized User		Apr 30, 2024
fd531630-bd06-ef11-9f89-7	HCL Notes Domino Complete		Apr 30, 2024

Recent Releases

Description	Date
HCL Launch v7.1.2.16	Jan 23, 2024
HCL Launch v7.2.3.9	Jan 23, 2024
HCL Launch v7.0.5.20	Jan 23, 2024
HCL Commerce Enterprise Version 9.1.15.1 for Power Linux	Jan 23, 2024
HCL Unica Discover 12.1.10 React Native	Jan 25, 2024

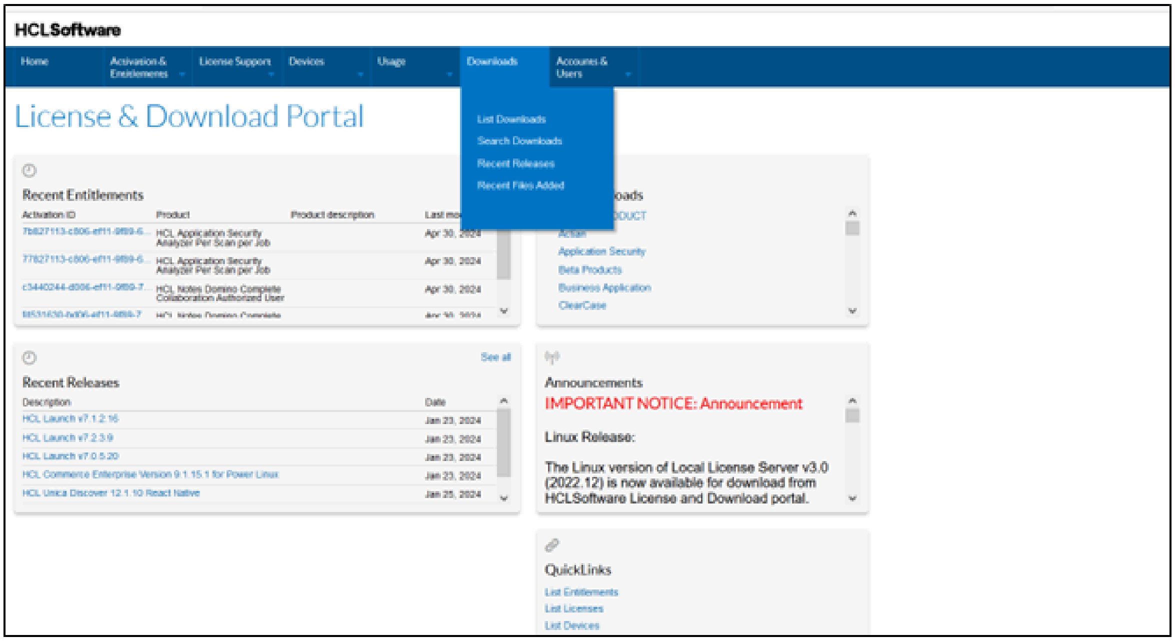
Announcements

IMPORTANT NOTICE: Announcement

Linux Release:

The Linux version of Local License Server v3.0 (2022.12) is now available for download from HCLSoftware License and Download portal.

5. In the **Downloads** tab, click on the **List Downloads** option.



Recent Entitlements

Activation ID	Product	Product description	Last mo
7b827113-c806-ef11-9f89-6...	HCL Application Security Analyzer Per Scan per Job		Apr 30, 2024
77827113-c806-ef11-9f89-6...	HCL Application Security Analyzer Per Scan per Job		Apr 30, 2024
c3440244-d006-ef11-9f89-7...	HCL Notes Domino Complete Collaboration Authorized User		Apr 30, 2024
fd531630-bd06-ef11-9f89-7	HCL Notes Domino Complete		Apr 30, 2024

Recent Releases

Description	Date
HCL Launch v7.1.2.16	Jan 23, 2024
HCL Launch v7.2.3.9	Jan 23, 2024
HCL Launch v7.0.5.20	Jan 23, 2024
HCL Commerce Enterprise Version 9.1.15.1 for Power Linux	Jan 23, 2024
HCL Unica Discover 12.1.10 React Native	Jan 25, 2024

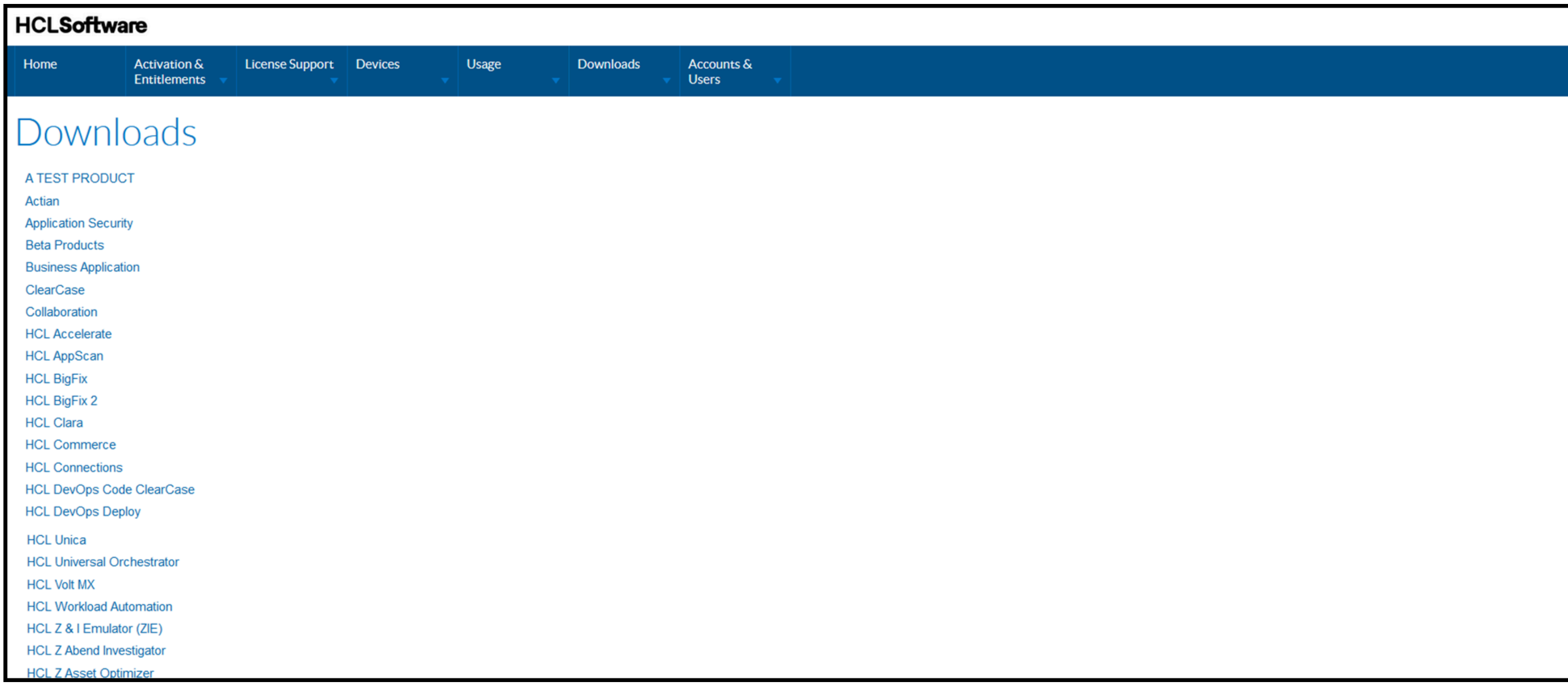
Announcements

IMPORTANT NOTICE: Announcement

Linux Release:

The Linux version of Local License Server v3.0 (2022.12) is now available for download from HCLSoftware License and Download portal.

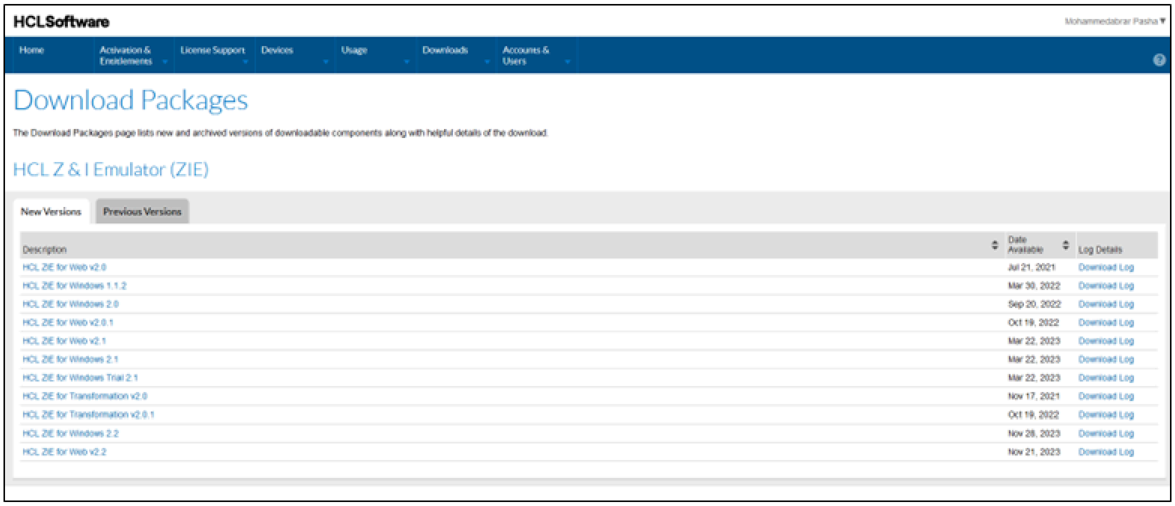
6. The **Downloads** page appears.
7. Select **HCL Z & I Emulator (ZIE)** from the list of products.



Downloads

A TEST PRODUCT
Actian
Application Security
Beta Products
Business Application
ClearCase
Collaboration
HCL Accelerate
HCL AppScan
HCL BigFix
HCL BigFix 2
HCL Clara
HCL Commerce
HCL Connections
HCL DevOps Code ClearCase
HCL DevOps Deploy
HCL Unica
HCL Universal Orchestrator
HCL Volt MX
HCL Workload Automation
HCL Z & I Emulator (ZIE)
HCL Z Abend Investigator
HCL Z Asset Optimizer

8. The **Download Packages** page appears.
9. Click on the required version/package under the **Description** column.



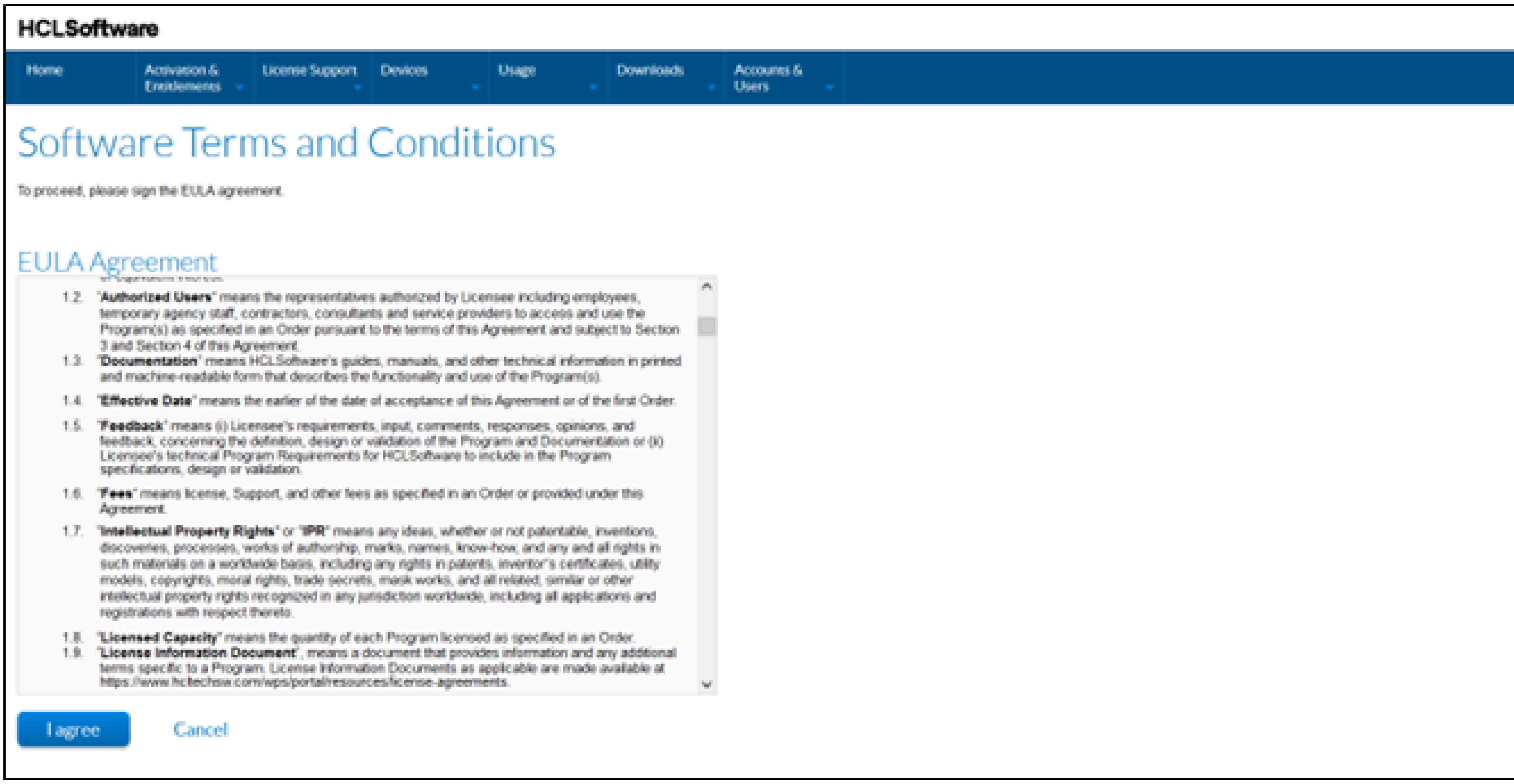
Download Packages

The Download Packages page lists new and archived versions of downloadable components along with helpful details of the download.

HCL Z & I Emulator (ZIE)

Description	Date Available	Log Details
HCL ZIE for Web v2.0	Jul 21, 2021	Download Log
HCL ZIE for Windows 1.1.2	Mar 30, 2022	Download Log
HCL ZIE for Windows 2.0	Sep 20, 2022	Download Log
HCL ZIE for Web v2.0.1	Oct 19, 2022	Download Log
HCL ZIE for Web v2.1	Mar 22, 2023	Download Log
HCL ZIE for Windows 2.1	Mar 22, 2023	Download Log
HCL ZIE for Windows Trial 2.1	Mar 22, 2023	Download Log
HCL ZIE for Transformation v2.0	Nov 17, 2021	Download Log
HCL ZIE for Transformation v2.0.1	Oct 19, 2022	Download Log
HCL ZIE for Windows 2.2	Nov 28, 2023	Download Log
HCL ZIE for Web v2.2	Nov 21, 2023	Download Log

10. The **Software Terms and Conditions** page appears.
11. Read the terms and conditions carefully and click the **I Agree** button.



Software Terms and Conditions

To proceed, please sign the EULA agreement.

EULA Agreement

1.2. "**Authorized Users**" means the representatives authorized by Licensee including employees, temporary agency staff, contractors, consultants and service providers to access and use the Program(s) as specified in an Order pursuant to the terms of this Agreement and subject to Section 3 and Section 4 of this Agreement.

1.3. "**Documentation**" means HCLSoftware's guides, manuals, and other technical information in printed and machine-readable form that describes the functionality and use of the Program(s).

1.4. "**Effective Date**" means the earlier of the date of acceptance of this Agreement or of the first Order.

1.5. "**Feedback**" means (i) Licensee's requirements, input, comments, responses, opinions, and feedback, concerning the definition, design or validation of the Program and Documentation or (ii) Licensee's technical Program Requirements for HCLSoftware to include in the Program specifications, design or validation.

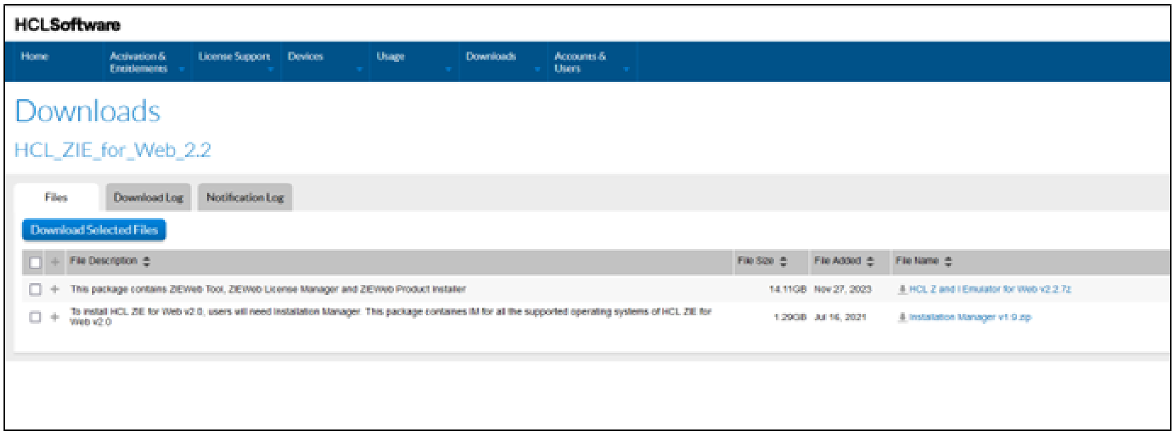
1.6. "**Fees**" means license, Support, and other fees as specified in an Order or provided under this Agreement.

1.7. "**Intellectual Property Rights**" or "**IPR**" means any ideas, whether or not patentable, inventions, discoveries, processes, works of authorship, marks, names, know-how, and any and all rights in such materials on a worldwide basis, including any rights in patents, inventor's certificates, utility models, copyrights, moral rights, trade secrets, mask works, and all related, similar or other intellectual property rights recognized in any jurisdiction worldwide, including all applications and registrations with respect thereto.

1.8. "**Licensed Capacity**" means the quantity of each Program licensed as specified in an Order.

1.9. "**License Information Document**", means a document that provides information and any additional terms specific to a Program. License Information Documents as applicable are made available at https://www.hcltechsw.com/wps/portal/resources/license-agreements.

12. The **Downloads** page appears.
13. Click on the required file name under the **File Name** column to download the product.



Downloads

HCL_ZIE_for_Web_2.2

□	File Description	File Size	File Added	File Name
□ +	This package contains ZIEWeb Tool, ZIEWeb License Manager and ZIEWeb Product Installer	14.11GB	Nov 27, 2023	HCL Z and I Emulator for Web v2.2.7z
□ +	To install HCL ZIE for Web v2.0, users will need Installation Manager. This package containes IM for all the supported operating systems of HCL ZIE for Web v2.0	1.29GB	Jul 16, 2021	Installation Manager v1.9.zip

Considerations Before Installing

Disk Space Requirements

Installation of Z and I Emulator for Windows to a drive other than the Windows® volume (the drive containing the Windows® folder) may still require as much as 180 MBs of available free space on the Windows® volume. This is due to the installation of files to the Windows® and system folders, as well as the caching of the Installer database by the Windows Installer service, and the use of temporary disk space by the Windows Installer service during the installation.

Migration Considerations

Z and I Emulator for Windows offers several migration options. To ensure that your session profiles, batch files, and other configuration information are migrated so you can use them with Version 3.0, see Custom Installation on page 140.

When migrating, you might be prompted to close all active Z and I Emulator for Windows sessions and actions.

Multi-Boot Environment Installation

If you want to install Z and I Emulator for Windows into a Z and I Emulator for Windows subdirectory that was originally installed under another operating system, you must remove the previous version first. Failure to do this may cause unpredictable results, including not being able to run Z and I Emulator for Windows from either operating system.

Coexistence support

iSeries Access

Co-existence is restricted for other Emulators that are similar to ZIEWin. HCL Z and I Emulator for Windows cannot be installed on a system where a similar product exists, and vice versa.

Coexistence support for iSeries Access PC5250 component is not provided. The iSeries Access PC5250 component must be removed prior to installing Z and I Emulator for Windows.

Following are some of the coexistence scenarios for which Users get appropriate notification messages :

- Installation of **HCL Z and I Emulator for Windows** over a similar product displays the following message :
  - "Installation of HCL ZIE for Windows cannot proceed further as a similar product is already installed on the system."
- Installation of other emulators similar to ZIEWin over **HCL Z and I Emulator for Windows** displays the following message :
  - "Setup has found a previous version of a similar product already installed on this system. You must uninstall the existing version before running setup again."

- Installation of **HCL Z and I Emulator for Windows** Refresh Pack Update Installer on a system, without having the Base/Refresh pack version installed, displays the following message :
  - "HCL Z and I Emulator for Windows RP** cannot continue as the Base version is not installed."

Windows x64 Platform Support

The x64-based versions of Microsoft Windows Server 2003, Microsoft Windows server 2008, Microsoft Windows 7, Microsoft Windows 8 and Microsoft Windows 10 are optimized to run native 64-bit programs, but do not support 32-bit drivers or 16-bit applications.

For these platforms, Z and I Emulator for Windows does not install the following features and libraries.

- 16-bit API support:
  - Standard EHLLAPI 16-bit interface
  - WinHLLAPI 16-bit interface
  - PCSAPI 16-bit interface

Installing Z and I Emulator for Windows

Z and I Emulator for Windows provides the below installation options:

- Administrative installation, including installing or running from source, where source medium is a network server. See Administrative Installation on page 150 for more information.
- Remote installation using Microsoft® Systems Management Server (SMS). See Remote Installation of Z and I Emulator for Windows on page 152 for more information.
- Installation and Uninstallation of HCL Z and I Emulator for Windows must be done manually using *setup.exe* or by using SCCM (Deployment tool). as the configuration and prerequisite checks will happen only with *setup.exe*.

Z and I Emulator for Windows also offers the ability to customize the installation procedure. For information on customizing with initialization file processing, including performing silent installations, see Installing Z and I Emulator for Windows Using an Initialization (response) File on page 149.

. Typical and custom setups are described in this chapter, as well as silent installation options. Additionally, this chapter provides an introduction to the Microsoft® Windows Installer service.

Microsoft® Windows Installer

Z and I Emulator for Windows utilizes the Windows Installer service. When the Z and I Emulator for Windows installation image is first run, it examines the target system and, if necessary, automatically installs the proper version of the Windows Installer service.

Setup.exe is the bootstrap loader that calls the Windows Installer service (msiexec.exe) and launches the installation dialogs. For a detailed description of the Microsoft® Windows Installer service, refer to the Windows Installer SDK available online at http://www.msdn.microsoft.com. For more information on setup.exe, see IPWI Command line parameters on page 153.

Note: The following should be taken into account when installing Z and I Emulator for Windows:

- In some cases, installation of Windows Installer triggers a reboot of the system. If you are required to reboot, upon subsequent startup you are taken immediately back to Windows Installer to continue installation of Z and I Emulator for Windows.
- After Windows Installer has been successfully installed, if the installation of Z and I Emulator for Windows fails or is cancelled by the user, Windows Installer rolls back all partially installed Z and I Emulator for Windows files and returns the system to its original state.
- You must be a member of the Administrator's group to perform these installations.
- Before you begin installation, make sure all other applications are stopped. If you are reinstalling Z and I Emulator for Windows or are upgrading Z and I Emulator for Windows, make sure that Z and I Emulator for Windows is not running before you start setup.

Upgrading Z and I Emulator for Windows

The HCL Z and I Emulator for Windows supports upgrade installation if an older version exists in the system. The upgrade works like a fresh installation; the existing version is uninstalled, and then upgrade installation is performed.

The application data created by the user in the previous installations gets retained after the upgrade.

The upgrade also supports different types of upgrade installation similar to a fresh installation.

- Administrative upgrade installation, including installing or running from source, where source medium is a network server. Refer to Administrative Installation for more information.
- Remote upgrade installation using Microsoft® Systems Management Server (SMS). Refer to Remote Installation of Z and I Emulator for Windows for more information.

Note: Execute the upgrade installation and uninstallation of the Z and I Emulator for Windows for Windows manually using *setup.exe* or SCCM (Deployment tool), as the configuration and prerequisite checks are performed only with *setup.exe*.

The HCL Z and I Emulator for Windows also offers the ability to customize the upgrade installation procedure. For information on customizing with initialization file processing, including performing silent installations, see Installing Z and I Emulator for Windows Using an Initialization (response) File.

Typical, custom setups and silent installation options available in the installation also apply to the upgrade installation. The upgrade installation supports auto-migration as well. For more information, refer to Migration Considerations.

Typical Installation

Typical installation selects all default features for installation. Features are defined as the specific functions of a program. See Feature Selection on page 141 for a list of default features. You can customize Z and I Emulator for Windows features by selecting the custom installation option (see Custom Installation on page 140).

To start a typical installation, click **Next** in the installation type panel. A panel appears, indicating that Z and I Emulator for Windows is computing the disk space requirements.

Note: Once the installation has passed this point, you cannot change the installation type. You would then need to cancel the installation on the Application Data panel, and begin a new installation.

To continue with the typical installation, use the following installation procedure.

1. The Application Data Location dialog opens. Select from the following application data location options:
    - User's application data folder ([UserProfile]\Application Data)
    - All users' common application data folder (All Users\Application Data)
    - Classic private directory

   Z and I Emulator for Windows uses multiple configuration files: user-class files can be stored individually by user profile, while system-class files are stored in a common location. Refer to *Quick Beginnings* for more information about user-class and system-class files and locations.

   If the [UserProfile]\Application Data location is selected, the following profile paths are used:

Operating System	**User-Class Directory (Current User)**[3]	**System-Class Directory**
Windows® Server 2003	C:\Documents and Settings\ %USERNAME%\Application Data\HCL\ZIE for Windows	C:\Documents and Settings\All Users\Application Data\HCL\ZIE for Windows
Windows Vista, Windows 7, Windows server 2008, Windows 10 x64	C:\Users\%USERNAME%\AppData\Roaming\HCL\ZIE for Windows	C:\ProgramData\HCL\ZIE for Windows

   If the All Users\Application Data location is selected, the following profile paths are used:

Operating System	**User-Class Directory (Current User)**[3]	**System-Class Directory**
Windows® Server 2003	C:\Documents and Settings\All Users\Application Data\HCL\ZIE for Windows	C:\Documents and Settings\All Users\Application Data\HCL\ZIE for Windows
Windows Vista, Windows 7, Windows server	C:\ProgramData\HCL\ZIE for Windows	C:\ProgramData\HCL\ZIE for Windows

Operating System	User-Class Directory (Current User)[3]	System-Class Directory
2008, Windows 10 x64		

If the classic Private directory location is selected, the following profile paths are used:

Operating System	User-Class Directory (Current User)[1, 2, 3]	System-Class Directory
Windows® Server 2003, Windows Vista, Windows 7 & Windows Server 2008, Windows 10 x64	C:\Program Files\HCL\ZIE for Windows\Private	C:\Program Files\HCL\ZIE for Windows\Private

[1] If the User Preference Manager (UPM) was set to a directory other than the default directory, Z and I Emulator for Windows will utilize that directory to store the user–class files. System–class files are always stored in the Private directory.

[2] For the classic Private directory locations, `C:\Program Files\HCL\ZIE for Windows` is the drive where Z and I Emulator for Windows is installed.

[3] The FTP Client configuration files are stored in the profile path mentioned the above, under the **FTP** folder.

Note: For installations on Windows x64 platforms, the directory path `Program Files` is replaced by `Program Files (x86)`.

After selecting your application data location, click **Next** to continue with the installation.

2. The Ready to Install the Program dialog opens. Click **Back** to change your previous settings, or click **Cancel** to terminate the installation process. Click **Install** to continue with installation.

The typical setup uses the C:\Program Files\HCL\ZIE for Windows directory for program installation.

Note: If there is not enough disk space on the C: drive, you are prompted to choose the custom installation setup type in order to choose an alternate installation destination.

After installation is complete, the Installation Complete dialog opens. Click **Finish** to exit the installation process.

After installation is complete, you are prompted to reboot the computer. You must reboot the computer before configuration changes take effect and you can use Z and I Emulator for Windows.

Note:

1. **Typical** installs the most common features for the applicable emulators.
2. **Typical** does not include API sample programs.

Custom Installation

Though the default feature selection for a custom setup is the same as for a typical setup, a custom configuration allows you to modify feature selection for your system. To continue with the custom installation setup:

1. Click the button to choose **Custom** setup type. Click **Next** to continue.
2. The English language product is automatically installed. Only the system default language locale and English are default. Select any additional languages that you want to install. Click **Next** to continue. A panel appears, indicating that Z and I Emulator for Windows is computing the disk space requirements.

   Note: Once the installation has passed this point, you cannot change the installation type. You would then need to cancel the installation on the Application Data panel, and begin a new installation.
3. The Custom setup dialog opens and asks you to select the program features that you want to install. Some features have subfeatures available. To view the subfeatures for a particular feature, click the plus sign (+) to the left of the feature name.

   Included in the Custom Setup window are **Feature Descriptions**. You can view the description of any feature by clicking on that feature and then reading the description section to the right of the feature selection tree. The feature description gives basic information about each feature, as well as the disk space required for installation. For more detailed information on disk space requirements for each feature, click the **Disk Space** button. For a description of available features, see Feature Selection on page 141. For a description of feature installation options, see Feature Installation Options on page 143.
4. From the Custom Setup dialog, you can change the directory where Z and I Emulator for Windows is installed. Click the **Change** button to choose another installation directory.
5. After making your feature selection choices and confirming the installation directory, click **Next** to continue with the installation.

   Note: If there is not enough space on the destination drive, you are prompted to choose another location.
6. The Application Data Location dialog opens. Select from the following application data location options:
   - User's application data folder ([UserProfile]\Application Data)
   - All users' common application data folder (All Users\Application Data)
   - Classic private directory

   See Typical Installation on page 137 for information on the profile paths used for each application data location. For more information about the location of application data, including workstation profiles, refer to *Quick Beginnings*. Click **Next** to continue.

7. The Automatic Migration Options dialog opens. By default, the **Automatic Migration of Profiles** box is checked. If you clear this check box, no migration occurs. The migration choices that are available to you are based on the information that you provided in the Application Data Location dialog. For information on the Application Data Location dialog, see .

   By default, the highest level of migration available for your application data location is selected. This is the recommended level of migration for your configuration. You can proceed with the installation using the default migration option, or you can choose another level of migration. For a description of the different levels of migration available for each Application Data Location, see .

   Click **Next** to continue with the installation.
8. The Ready to Install dialog opens. Click **Install** to complete the installation.

Feature Selection

The features and subfeatures available for Z and I Emulator for Windows are described in Table 15: Feature Selection Tree Contents on page 141. This table also identifies which features are installed by default.

Note: In the custom setup window, if the icon to the left of the feature name is white, that feature and all of its subfeatures will be installed. If the icon appears grey, that feature or one or more of its subfeatures will not be installed.

Table 15. Feature Selection Tree Contents

Feature	Description	Subfeatures Available	Default
3270 Emulation and Services	Your workstation can emulate a zSeries™ terminal (display, printer, or both). The emulator APIs (such as EHLAPPI, PCSAPI, DDE and) and utilities (such as Multiple Sessions, Menu Bar, , and Zip Print) are installed.	ZipPrint	Yes
3270 Emulation and Services	Your workstation can emulate an TN3270, TN3270E terminal (display, printer, or both). The emulator APIs and utilities (such as Multiple Sessions, Menu Bar, and Data Transfer) are installed.	ZipprintDefaultYes	
5250 Emulation and Services	Your workstation can emulate an iSeries™, eServer™ i5, or System i5™ terminal (display, printer, or both). The emulator APIs and utilities	Data Transfer • iSeries™ Connection Configuration	Yes

Table 15. Feature Selection Tree Contents (continued)

Feature	Description	Subfeatures Available	Default
	(such as Multiple Sessions, Menu Bar, and Data Transfer) are installed.		
VT Emulation	Your workstation can emulate an ASCII terminal. The emulator APIs (such as Multiple Sessions, and Menu Bar) are installed.	None	Yes
Fonts	Additional fonts are available, such as special 3270.	Fonts listed in dialog	Yes
Secure Sockets Layer	Allows encryption and authentication customization.	• MS CryptoAPI Security	No
Administrative and PD Aids	Diagnosis and update tools are included.	• Log Viewer • Information Bundler ◦ Internet Service	Yes
Utilities	Optional product utilities that can be installed.	• Convert Macro • Menu Bar Customization Utility • Multiple Sessions • User Preferences • FTP client	Yes
Emulator Programming APIs	APIs and sample programs.	.NET Interops • Register to GAC • Interops Sample Sample Programs for APIs • Host Access Class Library for C++ • Visual Basic • Miscellaneous APIs	No

Note:

1. The .NET Interops **Register to GAC** option is only available if the installation program detects that the .NET framework is present. However, the primary FTP client assembly will not be registered in the GAC.

   FTP Client requires .NET Framework v2.0 or higher to be installed on the system.

Feature Installation Options

Each feature and subfeature allows several installation options. To view the options available for each feature, click on the drop down icon to the left of the feature name. Select the desired installation type by clicking on it in the drop down menu. A description of each possible installation option follows:

- **This feature will be installed to run from CD** selects the feature to run from source where source medium is the installation image at the local workstation. This option is only available for top-level features and installs only the base files needed to run the feature.

  Note: If you are installing from a network, this option instead displays as, **This feature will be installed to run from network**. For more information on running from a network server, see Install to run from network on page 151.

- **This feature, and all subfeatures, will be installed to run from the CD** selects the main feature and all associated subfeatures to run from source where source medium is installation image at the local workstation. This installation option installs only the base files needed to run the features.

  Note: If you are installing from a network, this option instead displays as, **This feature, and all subfeatures, will be installed to run from the network**. For more information on running from a network server, see Install to run from network on page 151.

- **This feature will be installed when required** places a shortcut on the Z and I Emulator for Windows menu allowing the feature to be installed when the shortcut is selected. This installation option is also called *advertisement*.
- **This feature will not be available** deselects the feature for installation or advertisement.

Silent Installation

Z and I Emulator for Windows is installed silently by passing command-line parameters through setup.exe to the MSI (Windows Installer database) package. When running a silent installation, the user does not provide input via dialogs or see a progress bar during the installation process. Instead, installation occurs automatically using either a typical configuration or a custom configuration created during initialization file processing.

Note: When migrating through silent installation, all active Z and I Emulator for Windows sessions and actions will be closed without any prompting.

For details on initialization file processing, see Installing Z and I Emulator for Windows Using an Initialization (response) File on page 149. For information on performing a silent installation using setup.exe command-line parameters, see IPWI Command line parameters on page 153.

Auto-Upgrade for Standard Users

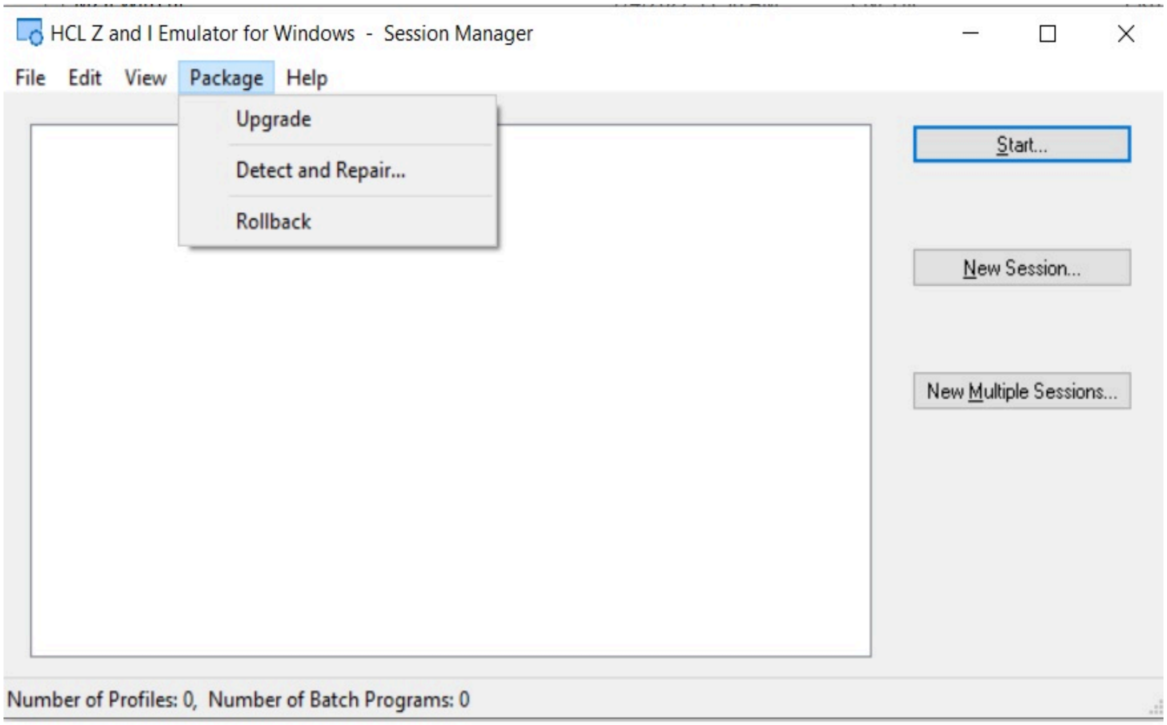
From 2.0 onwards, Auto-upgrade is enabled for Standard Users (Windows users who are not part of the **"Administrator"** group). *Upgrade* and *Rollback* is performed from the **"Package"** menu from the **Session Manager (Online/Offline)**.

Z and I Emulator for WindowsService" manages the Auto-Upgrade or Rollback of Z and I Emulator for Windows, and it can registered in the services during the **Base Pack installation**.

Steps to Rollback:

After the successful upgradation, to Rollback to the previous version, click **"Rollback"** menu item from the **"Package"** menu from the **Session Manager (Online/Offline)**.

Session Manager with disabled menu options



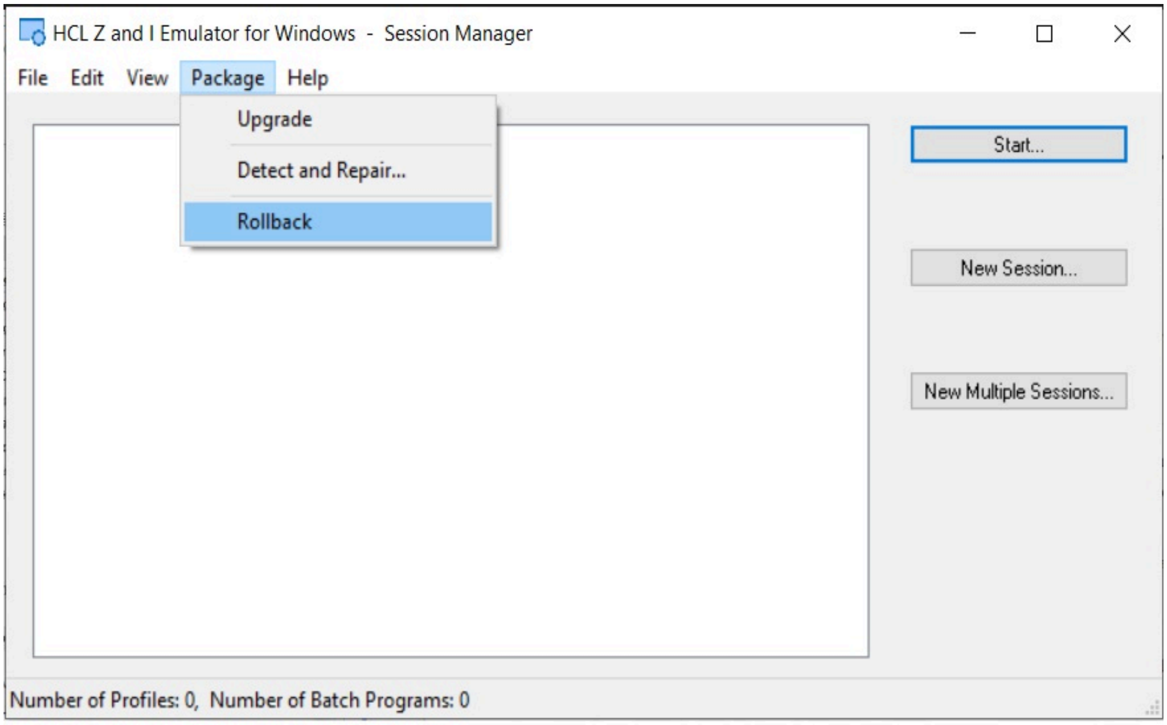
Number of Profiles: 0, Number of Batch Programs: 0

Steps to enable the Auto-Upgrade for a Standard User:

To enable the *“Upgrade”* and *“Rollback”* menu options for a Standard User, set the **“EnableUpdatePrivilege”** keyword value as *“YES”* in the **MZIEWIN.cnf** configuration file.

Note: By default, the **: "EnableUpdatePrivilege"** value is set to **NO**.

Session Manager with disabled menu options



Note: By default, *“Upgrade”* and *“Rollback”* menu options are enabled.

Pre-requisite

The base version Z and I Emulator for Windows version 3.0 full Installer must be installed in the system prior to the installation of higher version of RefreshPack update installer. The user must be Administrator to perform the Upgrade and Rollback for the base version is less than version 3.0

Note: From 2.0 onwards, the Auto-upgrade feature allows Windows users who are not part of the "Administrator" group (Standard User). A Standard User upgrade to the latest version with a base version of 3.0 or higher only.

Upgrade using ZIEWIN Refresh Pack Update Installer

Refresh Pack Update Installer is a light weight installer that contains only the fixes developed after the release of ZIEWIN version 3.0 base version. Each update installer is cumulative in nature; in other words, a new update installer also contain fixes from the previous update installers. This requires ZIEWIN version 3.0 base version or Refresh Pack installed.

When the ZIEWin Refresh Pack Update Installer is upgraded, the "Add/Remove Programs" shows both the latest Refresh Pack Update Installer version and ZIEWin base/refresh version.

As these Refresh Pack Update Installers are cumulative in nature, there will be only one Refresh Pack Update Installer installed in the system. If there is a lower version of Refresh Pack Update Installer already installed in the system, this update installer will uninstall the lower version before its installation.

If the Refresh Pack Update Installer is uninstalled manually, the ZIEWin version rolls back to its initial Base/Refresh pack version.

To uninstall the ZIEWin, it is recommended to always uninstall the Refresh Pack Update Installer first, followed by the base/Refresh pack version.

- Prerequisites on page 145
- Auto-Upgrade for Standard Users on page 144
- Steps to Install on page 147
- Steps to Install Manually on page 146
- Steps to Install Using the Start or Configure Sessions Online Utility on page 147
- Steps to Install Using Managed ZIEWIN on page 147

Steps to Install Manually

Find the following steps to install manually:

1. Ensure that Base or Refresh pack version is installed on the User's machine.
2. Extract the Refresh Pack Update installer zip to get the 'Z and I Emulator for Windows RP**.msi'.
3. Double click the 'Z and I Emulator for Windows RP**.msi' (The message, "Welcome to Installshield wizard for Z and I Emulator for Windows RP**." is displayed).
4. Click **Next**, to go to the next panel.
5. Click **Next**, to proceed with the installation in the Ready to Install panel.
6. Click **Install** and then click **Finish**.

Steps to Install Using the Start or Configure Sessions Online Utility

Find the following steps to install Using the Start or Configure Sessions Online Utility:

1. Ensure that the Base/Refresh pack is installed on the user machine.
2. Extract the Refresh Pack Update installer zip to get the Z and I Emulator for Windows RP**.msi" and *MZIEWin.cnf'* files.
3. The *cnf* is the upgrade configuration file that has the properties of refresh pack update installer.
4. Copy the Z and I Emulator for Windows RP**.msi' and *MZIEWin.cnf'* to the web server URL. This URL is provided by the user during the installation of the base/refresh pack or can be configured via the *'Preferences'* utility.
5. Invoke the *'Start or Configure Sessions - Online'* utility, if the installed version of ZIEWin is lower than the Refresh pack update installer version then the user will be prompted to upgrade.
6. Click **Yes** to proceed with the upgrade.

Steps to Install

Find the below steps to install:

1. Extract the **RefreshPack** zip file to get RefreshPack update installer Z and I Emulator for Windows RPx.msi" and configuration file **MZIEWIN.cnf** for version 3.0 which is higher than installed base version.
2. Double click Z and I Emulator for Windows RPx.msi. A message *"Welcome to Installshield wizard for Z and I Emulator for Windows RPx"* is displayed.
3. Click **Next**, it redirects to the next panel.
4. Click **Next** to install in Ready to Install the program panel.
5. Click **Install** and then click **Finish**.

Steps to Install Using Managed ZIEWIN

Find the following steps to Intsall:

1. After the base version is installed, unzip the **RefreshPack** zip file to copy the Z and I Emulator for Windows RPx.msi and **MZIEWIN.cnf** to the WebServerURL location provided.

   Note: Please refer to the section Web Server Details under the topic, HACP Server Details in the Quick Beginnings book.

2. From the ZIEWIN client, invoke the Start or Configure Sessions - Online. The user gets a notification about the latest available ZIEWIN version. To upgrade, refer to the Auto-Update of Z and I Emulator for Windows.
3. Click **Update**. The application is closed and the installer is downloaded to install the RefreshPack.

4. From 2.0 onwards, *Upgrade* and *Rollback*, is performed from **"Package"** menu from Session Manager (Online/Offline).

   Note: Base version of 3.0 or higher version.

Configuration Involved

Find the following configuration involved:

- The Z and I Emulator for Windows RPs are Cumulative.
- Supports Japanese the language.
- Adds all the files required for the Refresh Pack, which includes all APARs, Bugs, and PMRs.
- When a user installs Z and I Emulator for Windows 3.0, the Base Pack Z and I Emulator for Windows installation is auto-upgrade.
- The RefreshPack takes the backup of the base package files into a newly created folder BackupBase under the installation directory and restores them back to RP1.
- The RefreshPack overwrites a registry key under HKEY_LOCAL_MACHINE\SOFTWARE\HCL\Z and I Emulator for Windows\CurrentVersion\VersionNumber with RefreshPack version. The registry key is replaced back to the base version after the uninstallation.
- The new registry key is also added for the restricting coexistence under HKEY_LOCAL_MACHINE\SOFTWARE \HCL\Z and I Emulator for Windows\CurrentVersion\VersionNumber Value:- Fixpack_Version = 3.0.x and after uninstallation, the key is removed.
- Uninstallation of 3.0.x is also possible without affecting 3.0 base version
- A new ARP Entry Z and I Emulator for Windows 3.0.x is added to the control panel in addition to Z and I Emulator for Windows 3.0 and is removed after the uninstallation.

Installation of HCL ZIE License Manager

License manager is a tool that facilitates effective software management between end users and software vendors, thereby enabling organization to track and document the usage of the company's software products.

HCL ZIE License Manager is the license control tool used to track the license information for Mainframe Terminal +emulator products.

Steps to find the HCL ZIE License Manager installer files :

1. Download/copy the ***HCL_ZIE_for_Windows*.zip*** to the machine.
2. Extract the ***HCL_ZIE_for_Windows*.zip***.
3. To locate the 'License Manager.war' and 'License Manager.ear' files, extract the ***mmls*.zip*** folder and navigate to 'ZIE License Manager' sub-folder.

For more details on the Installation of HCL ZIE License Manager, see HCL ZIE License Manager on page 6.

To configure the ZIE License Manager for ZIEWin, see the topic Configuration of License Manager settings.

Installing Z and I Emulator for Windows Using an Initialization (response) File

Z and I Emulator for Windows provides an optional method of customization that allows property values and feature installation choices made during one installation to be automatically applied during subsequent installations. The initialization file (.ini) contains the properties and options for Windows Installer to use as initialization choices so that subsequent installations do not require users to provide installation input using dialogs. Then, future installations can be set to run silently using an initialization file.

Administrators create, save, and implement initialization files using command-line parameters. Z and I Emulator for Windows provides four command-line parameters:

- SAVEINI
- ONLYINI
- USEINI
- REMOVEINI

Each parameter, with a corresponding usage description, is described in the following sections.
Two sample initialization files are included on the Z and I Emulator for Windows installation image. These sample initialization files can be used during the installation if your workstation configuration matches the definitions in the sample. Z and I Emulator for Windows includes the following sample .ini files:

- **typical.ini** installs a typical setup
- **custom.ini** installs a custom setup to a user-defined path

The samples are defined for a first-time installation of Z and I Emulator for Windows.

Note: To ensure successful initialization file processing, use all syntax examples exactly as described.

Silent Installation Using Initialization File Processing

In order to ensure that property values and feature installation options designated in the initialization file are not overridden by users or to enhance the ease of installation, you can apply initialization files during silent installations.

To perform a silent installation using initialization file processing, type the following command:

```
E:\install\ZIEWin_pkgs\xxx\install\ZIEWin\setup.exe /s /v"/L*v
 \"%temp%\pcsinst.log\"
  USEINI=\"C:\Program Files\HCL\ZIE for Windows\ZIEWin.ini\" /qn"
```

where `xxx` is `mls`.

This process passes the silent installation command-line parameter (`/qn`) through setup.exe to the MSI package. For more information about setup.exe command-line parameters, see IPWI Command line parameters on page 153.
This parameter can also be added to commands that use system variables instead of path names.

Administrative Installation

An administrative installation copies a source image of Z and I Emulator for Windows installation files onto a network drive. The resulting location of this source image is called the installation point. After you complete an administrative installation, any user connected to the network can install Z and I Emulator for Windows to their own workstation by pointing to the installation point and running the setup. An administrative installation offers two installation choices to users:

- Installation directly to their system from the network server
- Installation to run from the network server

To begin an administrative installation, disable the AutoPlay function on your system or simply close the Z and I Emulator for Windows welcome window when it opens. With the installation image in the drive:

1. Open a command prompt and switch to the Z and I Emulator for Windows installation directory by typing

   ```
   E:
   ```

   where E: is the installation image drive.
2. At the command prompt, enter:

   -
     ```
     cd ZIEWin_pkgs\mls\install\ZIEWin
     ```

     for English.
3. From this directory, type

   ```
   setup.exe /a
   ```
4. The Windows Installer welcome dialog for Z and I Emulator for Windows opens. Click **Next** to continue with the installation.
5. The License Agreement dialog opens. Click the button to accept the terms of agreement. You can print the license agreement by clicking **Print**. If you decline the license agreement, the installation process terminates. Click **Next** to continue.

   Note: Windows® administrators have the option to accept the license agreement on behalf of all users. This allows users who install Z and I Emulator for Windows from the network server to skip the license agreement window during installation.
6. The Network Location dialog opens. You can type the desired network installation point in the command line or click **Change** to browse for a location.
7. Click **Install** to complete the installation process.

Note: To remove the source image of Z and I Emulator for Windows from your network server you must manually delete the source image directory from the network location.

Installing from Network Server

After the administrative installation is complete, any user connected to the network can install Z and I Emulator for Windows from the network server. To install from the network server:

1. Click **Run...** on the Windows® **Start** menu.
2. Type

   ```
   X:\MyLocation\setup.exe
   ```

   in the command line (where X: is your network server and MyLocation is the installation point designated in the administrative installation) or click **Browse** to browse for the location on the network.
3. The Windows Installer welcome dialog opens. Proceed with the installation as described in .

Installing to Run from Source, Where Source Medium Is a Network Server

After the administrative installation is complete, any user connected to the network can install Z and I Emulator for Windows to their workstation and designate any available features to run from source, where source medium is a network server (see Feature Selection on page 141 for a description of available features). In this scenario, feature shortcuts are placed on the Z and I Emulator for Windows menu. To install and run from the network server:

1. Click **Run...** on the Windows® **Start** menu.
2. Type

   ```
   X:\MyLocation\setup.exe
   ```

   in the command line (where X: is your network server and MyLocation is the installation point designated in the administrative installation) or click **Browse** to browse for the location on the network.
3. The Windows Installer welcome dialog opens. Proceed with the installation as described in , selecting **Custom** as your setup type.
4. In the Feature Selection dialog, click on the icon to the right of a desired feature to view its available installation options.
5. To select the feature to run from the network server, click on one of the following two options:
   - **This feature will be installed to run from network** to select a single feature to run from the network.
   - **This feature, and all subfeatures, will be installed to run from the network** to select the feature and all of its associated subfeatures to run from the network.
6. After making feature selection choices, proceed with the installation as described in Custom Installation on page 140.

Note: If you choose to run from source, all subfeatures are available, regardless of which subfeatures were selected or deselected using the feature tree.

Maintenance Installation of Z and I Emulator for Windows

After you have successfully installed Z and I Emulator for Windows on your system, users can perform maintenance installations to their Z and I Emulator for Windows program. The maintenance installation utility has three functions:

- **Modify** allows users to change their feature selection options. For details on changing feature tree selections, see Feature Selection on page 141.
- **Repair** analyzes the current configuration of Z and I Emulator for Windows and either repairs or reinstalls damaged features.
- **Remove** allows users to remove Z and I Emulator for Windows from their system.

Note:

1. When removing Z and I Emulator for Windows from your system, you are given the option to save the current program configuration for future installations of Z and I Emulator for Windows. If you choose to save the current settings, when you reinstall Z and I Emulator for Windows you are asked if you would like to use the previous settings to reinstall the product.
2. To remove a source image of Z and I Emulator for Windows created during an administrative installation, you must manually delete the source image directory from the network drive.

When the **Program Maintenance** dialog opens, select **Modify**, **Repair**, or **Remove** and click **Next**.

Note: To successfully run maintenance installation, the Z and I Emulator for Windows installation image must be available on either the installation image or on the network server. If you installed from a network server, the installation image must still be present at the original network location. If the installation image is not present, when you use the Modify or Remove utility to add features or to remove Z and I Emulator for Windows from your system, you may receive one of the following error messages:

- The feature you are trying to use is on a installation image or other removable disk that is not available.
- The feature you are trying to use is on a network resource that is not available.

To continue with maintenance installation you must either insert the installation image or browse the network to find the new location of the installation image.

Remote Installation of Z and I Emulator for Windows

Z and I Emulator for Windows supports remote installation using either Tivoli® Software Distribution or Microsoft® Systems Management Server (SMS) 2.0 Service Pack 2, or higher. Remote installation and uninstallation can be performed in a normal mode (attended) or silent mode (unattended).

Remote Installation Using SMS

A remote installation using SMS consists of the following steps:

1. Perform an administrative installation to copy Z and I Emulator for Windows installation files to the network (see Administrative Installation on page 150).
2. Create an SMS package containing the Z and I Emulator for Windows installation software.
3. Create an SMS job to distribute and install the software package.

Note: Z and I Emulator for Windows provides a sample SMS file, HCL Z and I Emulator for Windows.sms, for use in creating the SMS package. You can also create your own SMS file. An SMS file is the same as a Package Definition File (PDF) used in previous versions of Microsoft® SMS.

For detailed and up-to-date instructions on installing and deploying Z and I Emulator for Windows using SMS, refer to the SMS product documentation provided at http://www.microsoft.com/smsmgmt.

Remote Installation Using Active Directory Group Policy

Z and I Emulator for Windows can be distributed automatically to client computers or users via Microsoft Active Drirectory group policy.

For more information on how to distribute Z and I Emulator for Windows with Active Directory group policy, please refer to the Microsoft Windows knowledge base article at http://support.microsoft.com/kb/816102.

InstallShield Command-Line Parameters

InstallShield uses setup.exe as the bootstrap loader to call the Microsoft® Windows Installer service. Setup.exe can accept command-line parameters that allow you to perform administrative installations, run silent installations, and complete other administrative tasks. Using the /v parameter, other parameters can also be passed through setup.exe to the Windows Installer database (MSI package).

By default, setup.exe creates a verbose installation log with the file name pcsinst.log, and places it in the folder named by the environment variable `%temp%`. This behavior is overridden when command-line arguments are passed to the Windows® Installer using the /v parameter, as described in Parameter Descriptions on page 154.

Note: If `%temp%` points to a nonexistent folder and the /v flag is not used to override the default parameters passed to the Windows® Installer, then setup.exe will fail.

Setup.exe accepts the command-line parameters listed in Table 16: InstallShield Command-Line Parameters on page 154. Descriptions of each parameter are listed in Parameter Descriptions on page 154.

Table 16. InstallShield Command-Line Parameters

Parameter	Description
/v	Passes parameters to MSI package.
/s	Causes setup.exe to be silent.
/l	Specifies the setup language.
/a	Performs administrative installation.
/j	Installs in advertise mode.
/x	Performs setup uninstall.
/f	Launches setup in repair mode.
/w	Setup.exe waits for the installation to finish before exiting.
/qn	A Windows® Installer MSI parameter that causes everything but setup.exe to be silent. This sets the user interface level to zero.

Parameter Descriptions

Passing parameters to the MSI package

/v

The /v command-line parameter enables you to pass parameters supported by Windows Installer through setup.exe to the MSI package. For example, you can create and save a verbose log file to a location of your choice by passing the /L parameter through setup.exe to the MSI package. To create the log file, type:

```
E:\ZIEWin_pkgs\xxx\setup.exe /v"/L*v\"%temp%\pcsinst.log\"
```

where:

- `E:` is your installation image drive.
- `xxx` is `mls`.

For more information on supported command-line parameters and specific usage examples, refer to the Web site http://www.msdn.microsoft.com.

Note: The /v argument must be the last InstallShield parameter on the command line. Though supported Windows Installer parameters may be passed through to the MSI package, no InstallShield command-line parameters can follow the /v argument.

Running setup.exe silently

/s

To prevent setup.exe from displaying a progress bar, use the /s command-line parameter. To have setup run silently with no dialogs, pass the Windows Installer /qn command-line parameter through setup.exe using the /v parameter. Refer to Microsoft's documentation of command-line parameters for other /q user interface options. To run a silent installation, type:

```
E:\install\xxx\install\ZIEWin\setup.exe /s /v"/L*v
 \"%temp%\pcsinst.log\" /qn"
```

where:

- `E:` is your installation image drive.
- `xxx` is `mls`.

Note: You can pass an initialization file to the MSI package and run the installation silently using the /s /v /qn parameters in the following command:

```
E:\install\xxx\install\ZIEWin\setup.exe /s /v" /L*v
 \"%temp%\pcsinst.log\"
USEINI=\"C:\ZIE for Windows\ZIEWin.ini\" /qn"
```

where `xxx` is `mls`.

To install silently from source, where source medium is a network server, use the /s /v /qn parameters after pointing to the installation point in the command line.

To uninstall Z and I Emulator for Windows silently, use the /s parameter in conjunction with the /x parameter as shown in the following example:

```
X:\install\ZIEWin\setup.exe /s /x
```

where X: is the location of the Z and I Emulator for Windows installation directory.

Specifying the setup language

/l
The /l command-line parameter enables you to specify what language to use during setup by using the appropriate decimal language identifier. For a list of language identifiers, see . For example, to change the setup language to Czech, type:

```
E:\install\xxx\install\ZIEWin\setup.exe /l"1029"
```

Administrative installation

/a

Administrative installation installs a source image to the network server. This enables users with access to the network to install Z and I Emulator for Windows directly from the network server.

Advertise mode

/j

Advertisement enables users to install features of Z and I Emulator for Windows when they need them rather than during setup. Features that are available for installation are advertised with shortcuts on the user's system for later installation.

Uninstall mode

/x

Uninstall mode removes Z and I Emulator for Windows from your system.

Repair mode

/f

Launching setup.exe in the repair mode checks the key file of every installed feature and reinstalls any feature that is determined to be missing, corrupt, or an older version.

Abbreviations Used in This Book

API	Application Programming Interface
CPI-C	Common Programming Interface for Communications
EHLLAPI	Emulator High Level Language Application Programming Interface
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LAN	Local Area Network
LSP	LAN Support Program
MSI	Windows Installer Database
MSP	Windows Installer Patch
MST	Windows Installer Transform
TCP/IP	Transmission Control Protocol/Internet Protocol
WAN	Wide Area Network

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Emulator User's Reference

About This Book

HCL Z and I Emulator for Windows reference books are comprised of this *Emulator User's Reference* and an *Administrator's Guide and Reference*. This book is intended for users of Z and I Emulator for Windows.

Note:

1. *PC/3270* refers to the 3270 portion of the combined package.
2. *PC400* refers to the 5250 portion of the combined package.
3. *Workstation* refers to all supported personal computers.
4. *Windows®* refers to Windows® 7, Windows® 8/8.1, Windows® 10, Windows® Server 2008, and Windows® Server 2012. When information applies only to a specific operating system, this is indicated in the text.

Who Should Read This Book

This book is intended for the person who uses Z and I Emulator for Windows on a workstation to access hosts using 5250, 3270, or VT terminal emulation.

How to Use This Book

This book contains reference information that you might need to refer to when installing or operating Z and I Emulator for Windows.

Z and I Emulator for Windows is designed to use various communication adapters and to work with other workstation and host system software. Refer to the appropriate documentation for the products you use.

Command Syntax Symbols

Parentheses, brackets, ellipses, and slashes have the following meanings or uses:

( )

Parentheses enclose operands that govern the action of certain command options.

[ ]

Brackets indicate an optional command argument. If you do not use the optional item, the program selects a default.

...

Ellipsis after an argument indicates that you can repeat the preceding item any number of times.

/

For 3270, a slash must precede the Time Sharing Option Extensions (TSO/E) password. A slash must also precede parameters of DOS commands entered from the command line. For 5250, a slash must precede parameters of IBM® DOS commands entered from the command line.

\

A backslash is included as part of any directory name. An initial backslash indicates the first-level directory, and an additional backslash is inserted in the directory name to indicate another level.

All directives, operands, and other syntax can be typed in either uppercase or lowercase, unless otherwise indicated.

Where to Find More Information

The following sections discuss getting help when you are installing, configuring, or using Z and I Emulator for Windows.

Information Center

You can find documentation and links to other resources at the Z and I Emulator for Windows Information Center, at the following address:

https://help.hcltechsw.com/zie/ziewin/3.0/index.html

The Information Center contains reference material that is not found in this book, such as keyboard layouts and host code page tables.

The Z and I Emulator for Windows Information Center provides information in English.

Online Help

The help facility describes how to install, configure, and use Z and I Emulator for Windows. Online help is very extensive and includes information about every aspect of configuring and using Z and I Emulator for Windows. You can use Z and I Emulator for Windows online help just as you use the online help for Windows®.

Use help to obtain information about:

- Menu choices
- Operation procedures
- Operations in windows
- Meanings of the terms displayed in windows

- Causes of errors and the corresponding actions to take
- Mouse-based operations
- Operation without a mouse
- Detailed explanations of specific terms
- Further technical information about Z and I Emulator for Windows
- Detailed explanations of operator information area (OIA) messages

Z and I Emulator for Windows Library

The Z and I Emulator for Windows library includes the following publications:

- *Installation Guide*
- *Quick Beginnings*
- *Emulator User's Reference* (this document)
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Host Access Class Library*

In addition to the PDF documents, there are HTML documents provided with Z and I Emulator for Windows:

Quick Beginnings

The HTML form of *Quick Beginnings* contains the same information as the PDF version. The HTML files are installed automatically and can be accessed from the Help menus in the Session Manager and .WS session panels.

Related Publications

For information about local area networks (LANs), refer to the following publications:

- *IBM Local Area Network Technical Reference*
- *AS/400 Communications: Local Area Network (LAN) Guide Version 2*

Contacting HCL

This section lists ways you can reach HCL in case you encounter a problem or concern with Z and I Emulator for Windows. Depending on the nature of your problem or concern, we ask that you be prepared to provide the following information to allow us to serve you better.

- The environment in which the problem occurs:
    - Z and I Emulator for Windows configuration
        - Z and I Emulator for Windows version and manufacturing refresh level
        - The name of the workstation profile
    - Workstation configuration

        - The machine type and model, the system memory, the video adapter
        - The communication adapter you are using
        - Other adapters (especially communication adapters) installed
        - The printer type and model
        - Other devices installed, such as sound cards, modems, or fax machines
    - Software configuration
        - Windows® version and level
        - Communication and device-driver version and level
        - Other communication programs (such as Microsoft® Data Link Control) that are running and using resources
        - Printer driver version and level
    - Host configuration
        - The upstream host connection and configuration
    - FTP client configuration
        - The name of the FTP client configuration
        - The trace files
- Problem analysis information
    - Symptoms
    - Type of problem
    - OIA messages or error messages (if any)
    - Key factors related to the problem

If you have a technical problem, take the time to review and carry out the actions suggested here. Use your local support personnel before contacting HCL. You can also check the Hints and Tips at the Z and I Emulator for Windows support Web page for more information. Only persons with in-depth knowledge of the problem should contact HCL; therefore, support personnel should act as the interface with HCL.

Support Options

If you determine that you need to contact HCL, you can do any of the following:

- Access the Z and I Emulator for Windows Support page.

General Information

Z and I Emulator for Windows Highlights

Z and I Emulator for Windows brings the power of personal networking to your workstation by providing a variety of connectivity options supporting local area network (LAN) and wide area network (WAN) environments. Whether you need host terminal emulation, client/server applications, or connectivity, Z and I Emulator for Windows offers a robust set of communications, networking, and administrative features.

Z and I Emulator for Windows is a full-function emulator package with an easy-to-use graphical interface, which includes many useful features such as file transfer and dynamic configuration, and emulator APIs including the Host Access Class Library.

Z and I Emulator for Windows provides the following functions:

- **zSeries™ Connections**

  LAN

  Telnet3270

  VT-over-Telnet (TCP/IP)

  COM port

  Telnet 3270

  VT-over-Telnet (TCP/IP)
- **iSeries™ Connections**

  LAN

  Telnet5250 over TCP/IP

  VT over Telnet
- **ASCII Emulator Connections**

  LAN

  VT over Telnet

  COM port

  VT over Telnet
- **Log Viewer**
  - View Message Log, Trace Log, and Merged Log files
  - Summary and Detail views
  - Set default Message Log size and location
  - Filter and search Log files
  - Message Log entries Help
- **Trace Capability**
  - 3270/5250 emulator data
  - Connectivity data, such as LAN
  - User services data, such as node initialization
- **Sample Programs**
  - Located in \Z and I Emulator for Windows\samples subdirectory
- **Installation and Configuration**
  - Partial installation option
  - Program sharing on a network server

- Automatic detection of installed communication adapters
- Dynamic change of communication configurations
- Automatic Dial Facility)
- Silent Installation
- Verification of ASCII configuration

- **Host Session Function**
  - Up to 52 sessions
  - Variable screen size and automatic font scaling
  - Function settings (of the host code page, for example) for each session
- **Host Graphics Support**
  - Built-in vector graphics support for GDDM® and other graphics applications
- **File Transfer Function**
  - Easy operation through graphical user interface (GUI) windows
  - Batch transfer of multiple files
  - Concurrent file transfer through multiple sessions
  - Background file transfer
  - File transfer invocation by macro
  - VT File Transfer (XModem and YModem)
- **Edit (Cut and Paste) Function**

  You can use the clipboard to cut, copy, and paste a selected area. In addition, you can paste data in other applications, such as spreadsheet programs, that support the PasteLink function.
  - Support of spreadsheet data format (Sylk, BIFF3, Wk3 formats)
  - Copy Append
  - Paste Next
  - Paste to Trim Rectangle
  - Paste Stop at Protected Line
- **Graphical User Interface (GUI)**
  - Customizable 3D iconic tool bar
  - 3D-button hotspots
  - Pop-up keypad
  - Macro function, including record and play
  - VBScripts, including record and play
  - Keyboard-function setup and remapping
  - Mouse-button-function setup and remapping
  - Display setup (cursor type, graphics, sound, colors, for example)
  - Automatic font size adjustment or fixed font size
  - Window-appearance setup
  - Menu-bar customization
  - 3270 Light Pen emulation by using a mouse
  - Status bar with history
  - Page setup (Text and Graphics)

  - Revised Configuration Dialog
  - Online help
- **Print Function**
  - Printer session (for PC/3270: SCS, LU 3)
  - Graphics local print
  - Printing with the Windows printer drivers
  - Print function by printer definition table (PDT)
  - Multiple host-print functions in multiple sessions
  - PDF-to-PDT conversion tool
  - PC400 print function by OS/400® and i5/OS™ Host Print Transform (HPT)
  - PC400 printing supported by the iSeries™, eServer™ i5, and System i5™ Advanced Print Support Utility
- **Programming Interfaces**
  - 32-bit Emulator High-Level Language Application Programming Interface (EHLLAPI)
  - 32-bit Z and I Emulator for Windows API (PCSAPI)
  - 32-bit Automation Object API
- **PC400 Client Function**
  - Data transfer
  - Text Assist
  - Enhanced Programmable Terminal User Interface (ENPTUI)

Problem Analysis

This chapter describes the information that will help you analyze problems with Z and I Emulator for Windows, and ways to report a problem to HCL. For detailed information about contacting HCL, refer to *Quick Beginnings*.

For information about Z and I Emulator for Windows and support, refer to the following Web site(s):

- The Z and I Emulator for Windows support page provides links to code fixes, tips, newsgroups, support options, and services. To view this page or to submit a software defect report, go to the following Internet address:

  https://hclpnpsupport.hcltech.com/csm

Z and I Emulator for Windows provides several utilities to help you with problem analysis. They can be invoked by clicking their icons from the **Programs → HCL Z and I Emulator for Windows → Administrative and PD Aids** subfolder on the Windows® **Start** menu.

The following sections describe these utilities and how to use them.

Log Viewer

The Z and I Emulator for Windows log viewer utility enables you to view, merge, sort, search, and filter information contained in message and trace logs. Use the log viewer during problem analysis to work with message and trace log entries. The default name of the message log output file is PCSMSG.MLG; its file extension must be .MLG. The file extension for trace logs must be .TLG.

To view message or trace logs:

1. From the Administrative and PD Aids subfolder, click **Log Viewer**; or, from an active session, click **Actions → Launch → Log Viewer**.
2. From the list of logged messages, double-click a message to display the message text.

For more information about log viewer functions, refer to *Administrator's Guide and Reference*.

Trace Facility

The Z and I Emulator for Windows trace facility enables you to log trace information for certain Z and I Emulator for Windows functions.

To start a trace, perform the following steps:

1. From the **Administrative and PD Aids** folder, click **Trace Facility**; or, from an active session, click **Actions → Launch → Trace Facility**. The trace status on the title bar displays the current state:

   Active

   Trace data is being collected by the trace facility.

   Inactive

   No trace data is being collected.
2. From the main dialog box, click **Set Up** to set the desired trace system parameters.
3. Click **OK** to return to the main trace dialog box.
4. From the main trace dialog box, select the type of data you want to trace from the **Function Name**, **Component Name**, and **Trace Option** list boxes.

   Function Name

   A specific set of Z and I Emulator for Windows features, such as 3270/5250 Emulator or User Services.

   Component Name

   The name of a specific part of a function, such as API data (for the 5250 Emulator function) or Node Initialization (for the User Services function).

   Trace Options

   The options associated with a particular component, such as EHLLAPI (for the API component) or API trace (for the Node Initialization component).
5. Start tracing data by clicking **Start**, or apply changes to the trace options by clicking **Apply**.
6. Run the operation that you want to trace.
7. Optionally, stop the trace by clicking **Stop**.
8. Save the trace data to your hard disk by clicking **Save**.
9. Click **Format** to specify a formatted trace file name and to format the trace data. The Information Bundler utility should be used immediately after the trace is complete to ensure that the correct information is gathered.

> **Note:** If you have changed the default path for the formatted trace file, the Information Bundler will not find the trace information. Copy the trace files to the system-class application data directory.

10. Click **OK**.
11. Click **Clear** to clear the trace buffer where you saved a trace.
12. Use the log viewer to view the formatted trace log.

Enhanced trace buffers

Additional CSTrace buffers are provided, so that all trace records are captured during heavy trace loads.

Information Bundler

The Z and I Emulator for Windows Information Bundler utility gathers system files, trace and log files, and registry information into a .ZIP file. This file can be sent to support personnel, using the Internet Service utility. The Information Bundler should be executed immediately after the trace is complete to ensure that the correct information is gathered.

Start Information Bundler using one of the following methods:

- Click **Administrative and PD Aids → Information Bundler** from the Z and I Emulator for Windows program menu.
- In an active emulator session, click **Actions → Launch → Information Bundler** from the menu bar.

The X12345.ZIP file is created in the Z and I Emulator for Windows system-class application data directory. This file contains system and Z and I Emulator for Windows information. Refer to the installation documentation for the location of the system-class application data directory for each Windows® operating system.

NOTE : Information Bundler utility requires dotnet version 4.6.1 to work.

Considerations for Using Z and I Emulator for Windows Sessions

This chapter contains general hints and tips for using Z and I Emulator for Windows sessions. Supplementary information other than the items described in this book are included in the Readme HTML file in the Z and I Emulator for Windows directory.

Usage Notes for Sessions in OLE Documents

Changing Fonts

If you are using an In-Place embedded session, then changing the font face name, switching between automatic sizing and fixed size, or changing the size for a fixed size font can result in an incorrect display on the screen. To correct the display, adjust the size of the session object window slightly.

Initial Selection of Font

The initial font selection for a embedded or linked session is determined by its Session ID (a letter A - Z or a - z) just like a regular session. Therefore, the initial font might change if other sessions are already active. Further, to prevent In-Place embedded sessions from having adverse effects on subsequent sessions, font changes made during use of In-Place embedded sessions are not saved.

WordPad

Z and I Emulator for Windows session objects created in Microsoft® WordPad documents by the drag-and-drop method cannot be used after the document has been saved and closed. You should only create Z and I Emulator for Windows objects in WordPad by using the **Insert → Object** menu option.

Some versions of Microsoft® Word and Microsoft® WordPad incorrectly save the state of embedded objects that are displayed as icons. If you open a document that contains a Z and I Emulator for Windows session object that was created to display as an icon, and the object is activated, then it might activate in-place, instead of as a separate window.

WordPro

If you attempt to open a link to a Z and I Emulator for Windows session in a Lotus® WordPro document, WordPro might give unpredictable results. You should only use embedded Z and I Emulator for Windows objects in WordPro documents. You can use the **Display as Icon** option if a separate window is desired.

Updating Linked Files

Files that are linked into Word 97 or Excel 97 do not update automatically. You must manually save the linked file before your edits are reflected in the container window.

Inactivity Timeout for Communication Links

The Inactivity Timeout automatically disconnects a link after it has been idle for a specified length of time. Its purpose is to avoid excessive charges on dial-up links, such as switched-line connections . Inactivity Timeout is not recommended for other types of connections.

To enable Inactivity Timeout, add the following statement to the PU section of your workstation profile (.WS file):

```
[PU]
InactiveTimeout=xxx
```

The value `xxx`, in the range 1 to 999, is the number of minutes a link remains connected when there is no activity over it. The default value, InactiveTimeout=0, disables Inactivity Timeout.

Note: The Inactivity Timeout function monitors only attention keys (that is, the Enter, Clear, and PFx keys). It is recommended that you set a comparably longer value for `xxx` if, for example, you expect to key in large amounts of data on the screen before pressing the Enter key.

Environment Considerations

The following are environmental considerations for Z and I Emulator for Windows.

Virtual Memory

If you receive a message stating that the system is low on virtual memory, increase the size of the virtual memory paging file. If you receive this message while trying to open new host sessions or starting a Z and I Emulator for Windows function such as File Transfer, increase the amount of virtual memory.

Refer to the operating system documentation for instructions on how to increase the size of the paging file.

Emulator Session Icons

Emulator session icons that were not migrated during installation of Z and I Emulator for Windows Version 3.0 will not function correctly if they were not created in the application data directory that was specified during installation. The icons can be updated by using the **File → Import** option from the Session Manager. This option will not copy the prior icons to the application data directory specified during installation; the icons must be moved manually

Disabling CDRA Tables

This release uses the standard IBM® CDRA translation tables when converting between ASCII and EBCDIC. From some code page pairs, the standard tables differ from those that Z and I Emulator for Windows has used in the past. For code pages that were supported in prior releases, you can configure Z and I Emulator for Windows to use the old tables. A switch is available in PCSWIN.INI to disable the use of CDRA tables. This switch is located in the Translate section and is named UseOnlyZIEWin. This switch takes a binary value and is defaulted to FALSE (except for Japan, Korea, Taiwan, and PRC where it defaults to TRUE). For the code pages that are new to Version 3.0, you must use the standard tables. Setting the switch will apply to all sessions, as well as data transfer and command line file transfer.

Printing

You can use Z and I Emulator for Windows to print from display or printer sessions. This chapter provides details about printing and page setup options.

Setting Up the Printer

This section describes how to set up your printer with either a Windows® printer driver or a PDT file.

For an explanation of how to customize a PDT for PC/3270, see PDT Files (3270 and 5250) on page 178. For an explanation of how to customize a PDT for PC400, refer to *Administrator's Guide and Reference*.

For information about print processing for PC400, see Print Processing on page 244. For information about print processing for PC/3270, see Print Processing on page 205.

You can map a key sequence to bring up the Printer Setup dialog. There is no default key combination for this function. See Default Key Function Assignments on page 182 for more information about keyboard functions.

Defining a Printer for a Session

To choose or change the Windows® printer driver to be used, follow these steps:

1. Click **File → Printer Setup** in the session window.

   The Printer Setup window lists the available printers.
2. Select a printer from the list box. If desired, select the check box **Show this dialog before every print**.

   Note: The **Default** selection causes the current Windows® default printer to be used.
3. Confirm that **Use PDT** is not selected.

Page Setup Parameters

You can set Page Setup parameters, such as the maximum number of lines per page, the maximum number of columns, and fonts. These parameters are initially set to the defaults. Use this function to change specific control items.

Note: When a PDT file is used, this function cannot be used.

To set or change setup items:

1. Select **File → Page Setup** from the menu bar of the session window.

   The Page Setup window opens.
2. Select the tab that contains the parameters you want to change.

   Note: To switch from the current settings back to the defaults, select **Default**.
3. When all the items have been set, click **OK** or **Apply**.

Text Parameters

You can set the following general parameters for 3270, 5250, and VT display sessions.

Table 17. Page Setup Parameters — Text Tab

Parameter	Description
CPI	Specifies the number of characters to be printed per inch. If CPI was not set previously, a CPI value suitable for the font selected from the **Font** list box is assumed as the default.
LPI	Specifies the number of lines to be printed per inch. If LPI was not set previously, an LPI value suitable for the font selected from the Font list box is assumed as the default.
Maximum Number of Lines per Page	Specifies the maximum number of lines per page. A number in the range 1–255 can be specified. The default is *66*. When you set this to a value other than the default, Z and I Emulator for Windows uses it to scale the LPI and font to the best fit for the page.
Maximum Number of Characters per Line	Specifies the maximum number of characters per line. A number in the range 1–255 can be specified. The default is *132*. When you set this to a value other than the default, Z and I Emulator for Windows uses it to scale the CPI and font to the best fit for the page.
Font	Lists the fonts that can be used with the currently selected printer. The fonts in brackets [ ] are device fonts specific to the printer driver. The other fonts are graphics display interface (GDI) fonts printed in bit map formats.
Margins	This option maps the text on the screen to the printed page size. This option is disabled by default. It is available for the following sessions: • 3270 display and printer • VT display, including Printer Controller Mode
Top or Left Margin	Specify the decimal number (either inches or centimeters, depending upon your location), which represents the offset that will be reserved as the margin. **Note:** The value entered must represent the distance from the paper's edge. However, most Print Drivers enforce a nonprintable border (area). The default (original value shown) represents the Print Driver's enforced margin. Your input value cannot be less than that value; if it is, the Driver's value is substituted for your value. Some printers have an additional nonprintable border not represented in the Driver's value.
Use best fit scaling	This option maps the text on the screen to the printed page size. This option is disabled by default. It is available for the following sessions: • 3270 display and printer • VT display, including Printer Controller Mode

Text Options Parameters

You can set the following parameters by selecting the **Text Options** tab.

Table 18. Page Setup Parameters — Text Options Tab

Group	Options	Setting
Print Options These options are not available for the printer session.	**Suppress Null Lines**	Determines whether to delete lines containing only null or non-printable characters (null or non-printable field characters, and field attributes) or to print them as null lines. **Selected** Null lines are not printed. **Not selected** Null lines are printed. This option can be used when: • The **Print Screen** command is used • The combination of bits 2 and 3 of the WCC is not 00
	Print Nulls as spaces	Determines whether to print NULL control codes as null characters. **Selected** The codes are printed as blanks. **Not selected** The codes are treated as null characters. This option can be used when: • The **Print Screen** command is used
	Replace FF by LF	Select this option to replace a form feed by the number of lines entered in the edit box. The default is unchecked.

Table 18. Page Setup Parameters — Text Options Tab

(continued)

Group	Options	Setting
		Note: This option is only available on TN3270/TN3270E and TN5250.

Page Header and Footer Parameters

You create your own header and footer, and save up to five header and five footer configurations. Apply a saved header or footer by selecting it from the drop-down list.

Note:

1. A custom header or footer is associated with the specific session. A newly configured session will not have a header or footer.
2. If BestFit is enabled, the header and footer will be truncated at the Maximum Print Position (MPP), as determined by the BestFit parameters. You can allow multiple lines to prevent truncation (from the **Advanced** options).

To add items to a custom header or footer, do the following:

1. Select the desired alignment for the item (**Left**, **Center**, or **Right**).
2. Double-click on the item in the **Choices** box.

   The item is added to the alignment box.

You can manually reorder the items in an alignment box. Remove an item by manually deleting it from the box.

You can set the following parameters by selecting the **Header and Footer** tab.

Table 19. Page Setup Parameters — Header and Footer Tab

Group	Category	Parameter
Customize Header **Customize Footer**	**General**	The following information can be added to a header or footer: • **Date** • **New Line** • **Page Number** • **PC Name**

Table 19. Page Setup Parameters — Header and Footer Tab

(continued)

Group	Category	Parameter
		- **Time** - **PC User Name**
	Host Information	The following host details can be added to a header or footer: - **3270 Application Name** (3270 sessions only) - **Host Name** - **LU Name** - **Workstation ID** (5250 sessions only)
	Session Information	The following session details can be added to a header or footer: - **Short ID** - **Short Name**
Advanced Options	**Multiple Lines**	The following customization options are available: - **Allow multiple lines in header** - **Allow multiple lines in footer** If the header or footer does not fit on a single line, then it will be truncated at the Maximum Print Position. Select this option to allow multiple lines on the header or footer and prevent truncation.
	Page Number	The **Always start from** parameter specifies the starting value for the page number to be included in the header or footer. By default, the page number begins at 1.

Graphics Parameters (3270)

From a Z and I Emulator for Windows 3270 session, you can set additional parameters by selecting the **Graphics** tab.

Table 20. Page Setup Parameters — Graphics Tab (3270)

Parameter	Description
Scaling	By default, the screen size (display resolution) is mapped to the printed page size (printer resolution)—this is called **BestFit**. It is done automatically if you change either reso-

Table 20. Page Setup Parameters — Graphics Tab (3270) (continued)

Parameter	Description
	lution (including changing printers). The **/2**, **/3**, and **/4** values reduce the printed page size.
Black-on-White	Determines how the black pixels on the screen are printed. **Yes** Black pixels are printed as white pixels. Pixels other than black are printed as black pixels when you use a monochrome printer. When you use a color printer, they are printed in the same color as on the screen. **No** Black pixels are printed in black. Pixels other than black are printed as white pixels when you use a monochrome printer. When you use a color printer, they are printed in the same color as on the screen.

Orientation Parameters (5250)

When you use a PC400 printer session, you can set the following additional parameters by clicking the **Orientation** tab.

Table 21. Page Setup Parameters — Orientation Tab (PC400 Printer Session)

Group	Parameter	Description
Margins These margin settings are used only if **Use best fit scaling** is selected.	Top Margin	Bottom Margin is assumed to be equal to Top Margin.
	Left Margin	Right Margin is assumed to be equal to Left Margin.
Page Orientation Changes the default page orientation to specify how to print a document on the workstation printer. If the orientation is explicitly set by the iSeries™, eServer™ i5, or System i5™ page setup code, the explicit orientation is used.	Use automatic page orientation	If selected and the host does not explicitly set the orientation, the best orientation based on the host specified CPI, LPI, and page size will be used. If not selected and the host does not explicitly set the orientation the following drawer orientation will be used.
Drawer 1 orientation The default page orientation for the paper from drawer 1.	**Computer output reduction**	The document is printed in landscape. The font, pitch, and margins are set to appropriate values to fit on a page.
	Portrait	The document is printed in portrait.

Table 21. Page Setup Parameters — Orientation Tab (PC400 Printer Session)

(continued)

Group	Parameter	Description
	Landscape	The document is printed in landscape.
Drawer 2 orientation The default page orientation for the paper from drawer 2.	**Computer output reduction**	The document is printed in landscape. The font, pitch, and margins are set to appropriate values to fit on a page
	Portrait	The document is printed in portrait.
	Landscape	The document is printed in landscape.

When you use a PC400 printer session, you can set the following additional parameters by clicking the **Form Settings** tab. This option is available only when the printer and its driver support the change-source function.

Table 22. Page Setup Parameters — Form Settings Tab (PC400 Printer Session Only)

Parameter	Description
Form Settings	Specifies the form that should be selected when an application program specifies one of the following paper sources: • Drawer-one form • Drawer-two form • Envelope-hopper form Before using this function, you must configure the paper trays and forms in the printer-driver setup.

Table 23. Page Setup Parameters — Advanced Options Tab (PC400 Printer Session Only)

Option	Item to be set
Printer Font Code Page	Represents the code page being used for printing and displaying on the workstation.
No CR between fields	Represents not sending a CR when printing other fields on the same line.
Print bold as normal	Represents printing bold characters as not bold.
Display print status dialog	Represents showing a dialog window that will display showing printer status.
Use raster fonts	Represents allowing bitmap fonts for display and printing.

Display Sessions (3270 and 5250)

From display sessions, you can print all (**Print Screen**) or part (**Trim Print**) of the screen of your session window on a workstation printer. **Trim Print** is not available for PC400 sessions. For more information, refer to *Quick Beginnings* or the online help.

From a 3270 display session, you can also use the ZipPrint utility to print PROFS® notes, calendars, documents, CMS files, XEDIT workspaces, and host-session screens. See the online help for more information.

Print Screen Collection

The Print Screen Collection functions are available for 3270 and 5250 display sessions. You can capture all or part of the screen and add it to a collection of screen captures, and then print all the collected screen captures at the same time.

Collect Screens

Using the **File → Print Screen Collection → Collect Screen** feature, you can add a capture of all or part of the screen to a collection of captures.

To capture part of a screen, take the following steps:

1. Use the marking rectangle to mark an area of the screen.
2. Click **File → Print Screen Collection → Collect Screen** in the usual way. Z and I Emulator for Windows captures the entire screen image (including the white marking rectangle) and adds the screen image to the list of collected screen captures in the usual way, so that you can see the context of the area enclosed by the white marking rectangle.
3. Print the screen image. Z and I Emulator for Windows prints only the area that lies inside the white marking rectangle.

Note: The **Collect Screens** feature works independently of the normal **Print Screen** function. You can still use **Print Screen** to print an individual screen, while collecting multiple screens.

Print and Purge Collection

Using the **File → Print Screen Collection → Print and Purge Collection** feature, you can send the collected print screens to the printer. A status bar message indicates how many screens have been printed. The current Page Setup settings are applied to the printed screens. All the collected screens are purged.

Print and Keep Collection

Using the **File → Print Screen Collection → Print and Keep Collection** feature, you can send the collected print screens to the printer. A status bar message indicates how many screens have been printed. The current Page Setup settings are applied to the printed screens. All the collected screens are available for reprint.

Process Collection

Using the **File → Print Screen Collection → Process Collection** feature, you can preview the collected screens and select collected screens to be printed or purged.

On the **Process Print Screen Collection** window, you can perform the following tasks:

- Review the screens that have been collected.
- Select the check box next to one or more screens for processing.
- Print or delete selected screens.
- Scroll through the collected screens using the scroll bar.
- Click a collected screen to view a larger version of the screen.

Purge Collection

All the collected screens can be deleted without printing by clicking **File → Print Screen Collection → Purge Collection**. A confirmation message will be displayed. Click **Yes** to purge the collected screens.

Print Collection on Exit

The **File → Print Screen Collection → Print Collection on Exit** option ensures that the collected screens are printed before you close or disconnect the session. This option is enabled by default. To end the session without printing the collected screen, clear the **Print Collection on Exit** option. All the collected screens are then deleted when you close or disconnect the session.

When this option is disabled and you disconnect the session, a confirmation message will be displayed. Click **Yes** to purge the collected screens on exit.

You can add the **Collect Screens**, **Print and Purge Collection**, **Print and Keep Collection**, **Process Collection**, **Purge Collection**, and **Print Collection on Exit** functions to the toolbar, a popup keypad, a custom keyboard map, or a mouse customization, using the **Edit → Preferences** menu in the session window. The settings in the Page Setup dialog are used (shared with the normal Print Screen function).

In PDT mode, there is an option available for printing more than one screen on a page. Refer to *Administrator's Guide and Reference* for more information.

Replace FF with LF in GDI Print Mode

In Multiple Print Screen functionality, 3270/5250 host screens can be collected and then released to the physical printer. Each print screen that is collected is printed on a separate page, like individual print screens. However, in PDT mode using the BEL command, the Form Feeds (between two screens) can be converted to specified number of Line Feeds that is defined in the PDF/PDT file.

With this feature, the same functionality has been extended to GDI print mode, using the following workstation profile keyword:

```
[Printers]
ReplaceFFbyLF=<Byte value>
```

The possible byte values are as follows:

Byte Value	Action
00	No LF between screens
01 to 0xFE	LFs between screens
0xFF	FF after each screen

Byte Value	Action
No keyword	Default (FF after each screen)

Printer Sessions (3270 and 5250)

From printer sessions, you can direct printing from a zSeries™, iSeries™, eServer™ i5, or System i5™ to a workstation printer.

Note: When you use a host application which prints to your workstation's LPT1, you must first select the printer in the **Printer Setup** dialog of the **File** menu.

Configure a printer session to designate a workstation printer as a system printer that will use either a Windows® printer driver or a printer definition table (PDT) provided with Z and I Emulator for Windows.

- Use Windows® printer drivers for Z and I Emulator for Windows to print files based on printer setup parameters, such as scaling, duplex options, and page orientation, that you define in **Printer Setup**.
- Use PDT files for Z and I Emulator for Windows to print files based on page setup information, such as control codes and the printer output format, defined in the PDT. You can customize PDTs to define your own controls, by editing the corresponding printer definition file (PDF) and converting it to a PDT.

PDT Files (3270 and 5250)

PDTs (printer definition tables) are compiled from PDFs (printer definition files). PDFs contain printer commands that must be understood and supported by your printer.

The following are the basic printer languages:

PCL

Printer Control Language (Hewlett-Packard)

PPDS

ProPrinter Data Stream (IBM®)

ESC/P

Printer Control Language (Epson)

POSTSCRIPT

(No PDFs for this language)

Many printers support two or more of these languages. Most print drivers use a PJL (Printer Job Language) to switch between languages and to perform other job control functions, such as setting the number of copies.

You do not need a PDF for each different printer model; with the increasing number of models, PDFs are named for the printer language, not the printer model.

Older SBCS PDFs and PDTs are not shipped with Z and I Emulator for Windows, but are available at the product Web site. If you already have modified PDFs, any PDF and PDTs other than those in Table 24: Old Printer Definition Files on page 179 are retained during an install.

The End_Job statement in a PDF contains the printer commands that are sent to the printer at the end of each print job. If the End_Job contains a character defined as form feed (`FFF` in the PDT), a form feed (FF) is sent to the printer. It is not needed if the host application ends the job with the FF, as is commonly done. Some print drivers add the FF if needed, and most print drivers ignore extra FFs. So the FFF usually is protection against the host application not using a FF, and usually causes no problem. However, if you get an extra blank page, remove the FFF.

Table 24. Old Printer Definition Files

Printer Definition File (PDF) Name	Remarks
`ibm5577a`	(No FFF)
`ibm5577b`	(No FFF)
`lbp4`	

The supplied basic_ascii PDF does not contain any printer commands, which results in only ASCII text being sent to a printer or file. An accompanying PDT is also shipped. This PDF is for SBCS only.

PFT Migration

You can migrate a PC Support/400 Workstation Feature Printer Function Table (PFT) to a PDF for PC400. Refer to *Administrator's Guide and Reference* for more information.

Using PDT Files

To use a PDT file:

1. Click **File → Printer Setup** from the menu bar of the session window.

   The Printer Setup window opens.
2. Click the printer to be used from the list box.
3. Click **Setup**; specify the paper size.
4. Click **OK**.
5. Select the **Use PDT** check box, then click **Select PDT**.

   The Select PDT file window opens.
6. Do one of the following:
   - To use an existing PDT, select the PDT file to be used; then click **OK**.
   - To use a PDF that you have modified, you must first convert it to a PDT. To do so:
     a. Click **Convert PDF**.
     b. Select the PDF file to be converted from the list, then click **Convert**.

The window displays the result of the conversion. If there are any errors during the conversion, they are listed in the window.

c. When you select **Save List**, the window list is saved in *.LST file in the PDFPDT subdirectory.

To close without saving the list, click **Close**.

After the file is converted, control returns to the Select PDT file window and the converted PDT file appears in the list.

d. Select the PDT file; then click **OK**.

7. Click **OK** in the Printer Setup window.

Windows print driver for VT host printing

You can use the Windows print driver for VT host printing. This functionality adds to the existing PDT printing capability.

Collecting Print Jobs (5250 Printer Session)

You can collect 5250 print jobs and print them as a single job or in a group. The collected print jobs are stored in a .SCS file.

Note: This functionality is not supported in Host Print Transform mode.

You can set the following .WS profile keywords to specify the path and file name for the .SCS file.

```
[Printers]
SCSFile=<filename>.scs
SCSPath=<local path>
```

The functions associated with this feature are listed below. The functions can be mapped to the keyboard, popup keypad, mouse button, or toolbar button.

- **Collect Mode**

  When Collect Mode has been started, print jobs that have been sent are saved in the .SCS file. They are not printed immediately.
- **Print Collection**

  The print jobs that have been saved are sent to the printer as a single job.
- **Purge Collection**

  The collected print jobs are deleted.

Refer to the online help for details about mapping the functions.
The CombineJobs profile keyword enables you to collect the jobs for printing, while maintaining them as individual jobs (instead of one job in the .SCS file). Specify the .WS keyword as follows:

```
[Printers]
CombineJobs=N
```

If you set CombineJobs to **N**, the Print Collection function sends the separate, collected jobs to the printer. While in Collect Mode, if the keyword is set to **Y** or is not specified, the print jobs are combined as a single job in the .SCS file.

Printing to Disk

If you are using a PDT, you can save a host print-job or the contents of the session window (Print Screen) to a workstation file instead of printing it.

Two types of Print-to-Disk function are provided by Z and I Emulator for Windows:

Print-to-Disk Append

Appends multiple host print jobs or print screen jobs to a single workstation file.

Print-to-Disk Separate

Saves each host-print job or screen to a separate workstation file. You can specify the file name, but the extension is automatically assigned as a decimal number from 000 to 999. If you delete a file, its number will be re-used. When all 999 numbers have been used, the extension is automatically assigned a decimal number from 1000 to 9999.

Note:

1. Print-to-Disk is not available for the Print-Graphics function.
2. Print-to-Disk can be used only when you use a printer definition table (PDT) file.

To set up Print-to-Disk:

1. Click **File → Printer Setup** from the menu bar in the session window.

   The Printer Setup window lists the supported printers.
2. Select **Print to Disk Append** or **Print to Disk Separate** from the list box.
3. Click **Select PDT**.

   The Select PDT file window opens.
4. Select a PDT file from the list; then click **OK**.

   The Printer Setup window reopens.
5. Click **Setup**.

   The Select Print-to-Disk File window opens.
6. Specify a file name, drive, and path; then click **OK**.

   Note: If you specify the name of an existing file, subsequent print jobs are appended to the data in the original file in the case of Print to Disk Append.

Workstation Profile Parameter for Code Page

Occasionally a font does not support the desired code page. The wrong characters may be printed within the specific character set (Latin 2, for example). Z and I Emulator for Windows has a workstation profile parameter that allows the program to use a different code page that is supported by the desired font.

You can use the **PrinterFontCodePage** parameter if the following conditions are met:

- You can specify the printer font code page with which the desired font is encoded.
- Z and I Emulator for Windows provides the translation table for the host code page and the printer font code page.

However, because some Z and I Emulator for Windows releases might require manual adjustment of the workstation profile, try using different fonts before altering the .ws file. Fonts are listed in the Z and I Emulator for Windows Page Setup panel for all display sessions and 3270 host print sessions. For 5250 print sessions, the PCSPD.DAT file can be manually changed to control the fonts used. The Courier New font should support most languages and corresponding code pages.

To edit the .ws file, you must change the PrinterFontCodePage parameter to the value of the supported code page you wish to use. This option must be put in the [Printers] section, and is case-sensitive. See the following example for the proper parameter syntax. The parameter does not need to be placed immediately after the [Printers] section label.

```
[Printers]
PrinterFontCodePage=852
```

In this case, the desired font is encoded with code page 852. Z and I Emulator for Windows uses a different, existing translation table to translate data from EBCDIC to 852, versus using the standard Windows® code page.

This option is on the Page Setup panel for Z and I Emulator for Windows 5250 print sessions.

Key Functions and Keyboard Setup

This chapter contains information about keyboard setup and customizing mapped key functions.

Default Key Function Assignments

This section lists the functions assigned, by default, to each key on your keyboard.

For more information about each function, refer to the **Keyboard** choice on the **Help** menu.

You can change the default key assignments to the following default function tables, by selecting **Keyboard Setup** from the **Assist** menu.

When the Keyboard Setup window opens, select one of the following choices:

- 3270 for a 3270 keyboard layout
- 5250 for a 5250 keyboard layout
- 3270+5250 for a combined keyboard layout
- VT for a DEC VT220 keyboard layout

Z and I Emulator for Windows includes two .KMP keyboard map files that map the standard Win32 hotkeys for Cut, Copy, and Paste. You can use these keyboard map files or add the key values to an existing map file. See Win32 Cut, Copy, and Paste Hotkeys on page 192 for more information.

Setting the 3270 Keyboard Layout Default

To make the 3270 keyboard layout defaults available, do the following:

1. Click **Preferences → Keyboard** from the **Edit** menu. The Keyboard dialog box is displayed.
2. Select the **Default** radio button next to Current Keyboard.
3. Click **OK**.

Default Key Functions for a 3270 Layout

Table 25: Default Key Functions for a 3270 Layout on page 183 shows the default key functions for PC/3270. The key used is the same for all the supported keyboard types.

Table 25. Default Key Functions for a 3270 Layout

Function of Key	**Key**
APL	Ctrl+F8
Attention	Esc
Alternate Cursor	Alt+F11
Backspace	◄— (Backspace)
Back Tab	Shift+—►|
Back Tab Word	Alt+←
Break	Break
Change Format Toggle	Alt+F3
Change Screen	Ctrl+PageUp
Clear	Pause
Cursor Blink	Ctrl+F10
Cursor Down	↓ or 2(pad)
Cursor Left	← or 4(pad)
Cursor Right	→ or 6(pad)
Cursor Select	Ctrl+F9
Cursor Up	↑ or 8(pad)
Delete Character	Delete or .(pad)
Delete Word	Ctrl+Delete or Ctrl+.(pad)
Document Mode Toggle	Alt+F1

Table 25. Default Key Functions for a 3270 Layout

(continued)

Function of Key	**Key**
Dup	Shift+Insert²
Edit Copy	Ctrl+Insert
Edit Cut	Shift+Delete
Edit Paste	Shift+PageDown or Ctrl+Shift+Insert
Edit Undo	Alt+◄— (Backspace)
End Field	Pad End
Enter/Control	Shift+Ctrl
Erase EOF	End²
Erase Field	Shift+End²
Erase Input	Alt+End²
Fast Cursor Down	Alt+↓ or Alt+2(pad)
Fast Cursor Up	Alt+↑ or Alt+8(pad)
Field Mark	Shift+Home²
Graphic Cursor	Alt+F12
Highlighting Field Inherit	Alt+3(pad)
Highlighting Reverse	Alt+*(pad)
Highlighting Underscore	Alt+6(pad)
Home	Home or 7(pad)
Insert	Insert or 0(pad)
Jump Next	Alt+PageUp
Mark Down	Shift+↓
Mark Left	Shift+←
Mark Right	Shift+→
Mark Up	Shift+↑
Move Mark Down	Ctrl+↓ or Ctrl+2(pad)
Move Mark Left	Ctrl+← or Ctrl+4(pad)
Move Mark Right	Ctrl+→ or Ctrl+6(pad)
Move Mark Up	Ctrl+↑ or Ctrl+8(pad)
PA1	Alt+Insert²

Table 25. Default Key Functions for a 3270 Layout

(continued)

Function of Key	Key
PA2	Alt+Home[2]
PA3	Shift+PageUp[2]
Pause	Ctrl+F7
PF1 to PF12	F1 to F12
PF13 to PF24	Shift+F1 to F12
Play	Ctrl+F6
Print (Local Copy)	Not assigned
Quit (Device Cancel)	Alt+Left Ctrl
Record	Ctrl+F5
Reset/Control	Left Ctrl
Response Time Monitor	Ctrl+F11
Rule	Ctrl+Home
Sys Request	Shift+Esc
Tab Field	→| or Shift+→| (pad)
Tab Word	Alt+→
Test	Ctrl+PageDown
Word Wrap Toggle	Alt+F2

[2] Indicates the key on the main keyboard.

(pad)

Indicates a key on the numeric keypad.

Note: The Enhanced keyboard has some duplicated keys. The functions of the duplicated keys are the same except when you specify a single key. For example, Del means any Delete key, whereas Pad Del specifies only the Delete key on the numeric keypad.

Setting the 5250 Keyboard Layout Default

To make the 5250 keyboard layout defaults available, do the following:

1. Select **Preferences → Keyboard** from the **Edit** menu. The Keyboard dialog box is displayed.
2. Select the **Default** radio button next to Current Keyboard.
3. Click **OK**.

Default Key Functions for a 5250 Layout

Table 26: Default Key Functions for a 5250 Layout on page 186 shows the default key functions for iSeries™, eServer™ i5, or System i5™. The key used is the same for all the supported keyboard types.

Note:

1. If you use iSeries™ from the combined package, see Default Key Functions for the Combined Package on page 188.
2. The default key functions for a 5250 layout are not available by default. To make these functions available, perform the procedures in Setting the 5250 Keyboard Layout Default on page 185.

Table 26. Default Key Functions for a 5250 Layout

Function of Key	**Key**
Alternate Cursor	Ctrl+F11
Attention	Esc
Backspace	⟵ (Backspace)
Backtab	Shift+⟶|
Backtab Word	Alt+←
Begin Bold*	Ctrl+B
Begin of line*	Ctrl+4(pad)
Begin Underscore*	Ctrl+U
Bottom of Page*	Ctrl+2(pad)
Carrier Return	Ctrl+Enter or Ctrl+-(pad) or Ctrl++(pad)
Center Text*	Ctrl+C
Clear	Pause
Cursor Blink	Ctrl+F10
Cursor Down	↓ or 2(pad)
Cursor Left	← or 4(pad)
Cursor Right	→ or 6(pad)
Cursor Up	↑ or 8(pad)
Delete Character	Delete or .(pad)
Delete Word	Ctrl+Delete or Ctrl+.(pad)
Display Text Code	Alt+Insert
Dup	Shift+Insert
Edit Copy	Ctrl+Insert

Table 26. Default Key Functions for a 5250 Layout

(continued)

Function of Key	Key
Edit Cut	Shift+Delete
Edit Paste	Shift+PageDown or Ctrl+Shift+Insert
Edit Undo	Alt+◄— (Backspace)
End Bold/Underscore*	Ctrl+J
End of line*	Ctrl+6(pad)
End of page*	Ctrl+P
Enter/Control	Right Ctrl
Erase EOF	End or 1(pad)
Erase Input	Alt+End
Fast Cursor Down	Alt+↓ or Alt+2(pad)
Fast Cursor Up	Alt+↑ or Alt+8(pad)
Field Exit	Enter(pad) or ↵ (Enter)
Field Mark	Shift+Home
Field Minus (-)	-(pad)
Field Plus (+)	+(pad)
Half Index Down*	Ctrl+H
Half Index Up*	Ctrl+Y
Help	Alt+F1
Home	Home or 7(pad)
Host Print	Ctrl+Pause
Insert	Insert or 0(pad)
Insert Symbol*	Ctrl+A
Jump Next	Alt+PageUp
Mark Down	Shift+↓
Mark Left	Shift+←
Mark Right	Shift+→
Mark Up	Shift+↑
Move Mark Down	Ctrl+↓
Move Mark Left	Ctrl+←
Move Mark Right	Ctrl+→
Move Mark Up	Ctrl+↑

Table 26. Default Key Functions for a 5250 Layout

(continued)

Function of Key	**Key**
Next Column*	Ctrl+D
Next Stop*	Ctrl+N
Pause	Ctrl+F7
PF1 to PF12	F1 to F12
PF13 to PF24	Shift+F1 to F12
Play	Ctrl+F6
Quit	Alt+Left Ctrl
Record	Ctrl+F5
Required Backspace	Ctrl+← (Backspace)
Required Space*	Ctrl+Space
Required Tab*	Ctrl+→|
Reset/Control	Left Ctrl
Roll Down	9(pad) or PageUp
Roll Up	3(pad) or PageDown
Rule	Ctrl+Home
Stop Code*	Ctrl+S
System Request	Shift+Esc
Tab Field	→|
Tab Word	Alt+→
Test Request	Alt+Pause
Top of Page*	Ctrl+8(pad)
Word Underscore*	Ctrl+W

(*pad*)

Indicates a key on the numeric keypad.

*****

Indicates a Text Assist Key (SBCS only).

Default Key Functions for the Combined Package

Table 27: Default Key Functions for the Combined Package on page 189 shows the default key functions for the combined package. The key used is the same for all the supported keyboard types.

When you use the 3270+5250 keyboard layout, the key definitions for the 3270 and 5250 layouts are combined with those listed here.

Table 27. Default Key Functions for the Combined Package

Function of Key	Key
Change Screen	Not assigned
Character Advance	Shift+BackSpace
Help	Not assigned
Host Print	Not assigned
PA3	Not assigned
Roll Down	PageUp
Roll Up	PageDown
Printer Setup	Not assigned

Setting the VT Keyboard Layout Default

To make the VT keyboard layout defaults available, do the following:

1. Click **Preferences → Keyboard** from the **Edit** menu. The Keyboard dialog box is displayed.
2. Select the **Default** radio button next to Current Keyboard.
3. Click **OK**.

Default Key Functions for the VT Emulator Layout

Table 28: Default Key Functions for a VT Emulator Layout on page 189 shows the default key functions for VT220, VT100 and VT52. The key used is the same for all the supported keyboard types. The VT emulator keyboard gets selected as the default only when the VT Component is selected in the installation path.

Table 28. Default Key Functions for a VT Emulator Layout

Function of Key	Key
Backspace	← (Backspace)
Break	Ctrl+Pause
CAN	Ctrl+← (Backspace)
Cursor Down	↓ or 2(pad)
Cursor Left	← or 4(pad)
Cursor Right	→ or 6(pad)
Cursor Up	↑ or 8(pad)
Edit Copy	Ctrl+Insert
Edit Cut	Shift+Delete
Edit Paste	Shift+PageDown or Ctrl+Shift+Insert
Edit Undo	Alt+← (Backspace)
ESC	ESC

Table 28. Default Key Functions for a VT Emulator Layout

(continued)

Function of Key	Key
Jump Next	Alt+PageUp
New Line	↵ (Enter)
Mark Down	Shift+↓
Mark Left	Shift+←
Mark Right	Shift+→
Mark Up	Shift+↑
Move Mark Down	Ctrl+↓ or Ctrl+2(pad)
Move Mark Left	Ctrl+← or Ctrl+4(pad)
Move Mark Right	Ctrl+→ or Ctrl+6(pad)
Move Mark Up	Ctrl+↑ or Ctrl+8(pad)
PF6 to PF12	F6 to F12
PF13 to PF20	Shift+F1 to F8
Rule	Ctrl+Home
Tab Field	→| or Shift+→|
VT Enter	Shift+Enter(pad)
VT Find	End[2] or 1(pad)
VT Hold	Pause
VT Insert	Insert or 0(pad)
VT Next	Page Down[2] or 3(pad)
VT Numpad 0 to VT Numpad 9	Shift+0(pad) to Shift+9(pad)
VT Numpad Comma	Shift++(pad)
VT Numpad Minus	-(pad) or Shift+-(pad)
VT Numpad Period	Shift+.(pad)
VT PF1 to VT PF4	F1 to F4
VT Prev	Page Up[2] or 9(pad)
VT Remove	Delete or .(pad)
VT Select	Home[2] or 7(pad)
VT User F6 to VT User F12	Ctrl+F6 to F12

Table 28. Default Key Functions for a VT Emulator Layout

(continued)

Function of Key	Key
VT User F13 to VT User F20	Ctrl+Shift+F1 to F8

2

Indicates the key on the main keyboard.

(*pad*)

Indicates a key on the numeric keypad.

Note: The Enhanced keyboard has some duplicated keys. The functions of the duplicated keys are the same except when you specify a single key. For example, Del means any Delete key, whereas Pad Del specifies only the Delete key on the numeric keypad.

Keyboard Setup (3270 and 5250)

You can use Keyboard Setup to modify the function defined for each key on the keyboard, except some reserved keys.

You can define the following functions for the keys:

- Performing a key function
- Playing a macro
- Entering characters

Note: For 3270, the Enter function is assigned to the Ctrl key, by default. To change this assignment or, if you are using a non-IBM compatible keyboard and the Enter key does not work properly, you need to customize your keyboard. For 3270 and 5250 sessions, you can use the keyboard map files provided with Z and I Emulator for Windows (see Win32 Cut, Copy, and Paste Hotkeys on page 192).

Keyboard File

When you specify a key, you can save the new keyboard layout in a file (.KMP). If you create two or more keyboard files, you can alternate between them as required.

To assign a function to a key on the keyboard:

1. Click **Preferences → Keyboard** from the **Edit** menu or click the map icon on the tool bar.
2. When the Keyboard Setup window appears, select **Customize**.

    Note: Select **Spain** from the **Language** menu during keyboard setup if you want Catalan support.

3. Assign the key functions, referring to the online help for detailed instructions.

4. Save your changes and exit the Customize Keyboard window.
5. Click **OK** after completing the setup.

You can reset either the entire keyboard or specific keys to defaults:

- To reset the entire keyboard, set the current keyboard to **Default** in the Keyboard Setup window.
- To reset specific keys, select a key in the Customize Keyboard window and then select **Default** from the Current Actions for Selected Key box.

Note: You cannot redefine the following keys: Alt, AltGr, Print Screen, Scroll Lock, CapsLock, NumLock, and Shift.

Win32 Cut, Copy, and Paste Hotkeys

Z and I Emulator for Windows includes two .KMP keyboard map files that map the standard Win32 hotkeys for Cut, Copy, and Paste to Ctrl+X, Ctrl+C and Ctrl+V, respectively. You can use these keyboard map files or add the key values to an existing map file.

For 5250 sessions, the .KMP file provided is pcswinkb5.kmp. The remapping is given in Table 29: Win32 Keyboard Map Functions for a 5250 Layout on page 192.

Table 29. Win32 Keyboard Map Functions for a 5250 Layout

Function of Key	Key
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Enter	Enter
New Line	Right Ctrl

The keys PF7 and PF8 remains mapped to Roll Up and Roll Down, respectively.
For 3270 sessions, the .KMP file provided is pcswinkb3.kmp. The remapping is given in Table 30: Win32 Keyboard Map Functions for a 3270 Layout on page 192.

Table 30. Win32 Keyboard Map Functions for a 3270 Layout

Function of Key	Key
Edit Cut	Ctrl+X
Edit Copy	Ctrl+C
Edit Paste	Ctrl+V
Page Up	PF7
Page Down	PF8
Enter	Enter

Table 30. Win32 Keyboard Map Functions for a 3270 Layout (continued)

Function of Key	Key
New Line	Right Ctrl

Using Z and I Emulator for Windows 3270

Considerations for Using PC/3270 Sessions

This chapter contains hints and tips for using PC/3270 sessions. Supplementary information other than the items described in this book may be included in the Readme HTML file in the Z and I Emulator for Windows directory.

TN3270E Contention Resolution

Using the TN3270E function negotiation mechanism, Z and I Emulator for Windows negotiates with servers to enable the CONTENTION-RESOLUTION function described in the IETF *TN3270E Functional Extensions* Internet-Draft document. As with any other such negotiation, the server might accept or reject this function.

The CONTENTION-RESOLUTION function is supported only for display sessions.

Z and I Emulator for Windows negotiation for this function is enabled by default. It can be disabled by adding the following keyword to the .WS profile.

```
[Telnet3270]
TN3270EContentionResolution=N
```

Host-Session Window Operations

Cursor Color

By default, PC/3270 draws the underline cursor in white. If the background color of the current field is white, the color of the underline cursor automatically switches to black. PC/3270 draws the block and half-block cursors in the same color as the current field, reversing the background and forground colors. This behavior is identical to a 327x terminal.

If you want to change the default cursor color assignment, modify the pcswin.ini file and add the CursorColor value to the Session stanza, as follows:

```
[Session]
CursorColor=<red_value> <green_value> <blue_value>
```

Here, `<red_value>`, `<green_value>`, and `<blue_value>` are integers from 0 to 255 that specify the color intensity for each color primitive respectively. The color values range from `0 0 0` for black to `255 255 255` for white. PC/3270 then draws the underline cursor and the block cursors in this new color, mixing this new color with the existing screen colors using an XORed (exclusive or) operation. The value `0 0 0` (black) is not recommended because XORing 0 0 0 with any existing color results in the existing color, which makes the cursor invisible.
If you select a blinking cursor, it will always be drawn white, mixing the white with the existing screen colors using an XORed operation. You cannot change the cursor color when it is blinking.

Note: Cursor blinking is disabled by default in Windows Terminal Services sessions and under virtualization environments such as Citrix XenDesktop and Citrix XenApp. Refer to the appropriate vendor documentation to enable cursor blinking.

Releasing Insert Mode with Attention Keys

As on a non-programmable terminal, you can release insert mode when you press an Attention key. If you want this to happen, add this parameter to the [Keyboard] section of the workstation profile (`*.WS`):

```
[Keyboard]
ResetInsertByAttn=Y
```

Scroll Bar

If you choose **Font** from the Appearance menu in the host session window and choose **Fixed Size** from the Select Display Font window, the entire operator information area might not appear on the screen. If you specify **With Scroll Bar**, the OIA will not scroll. The session window size is restricted to be smaller than the screen size.

Scroll-Lock Key

When the Scroll Lock keyboard indicator is turned on, the cursor movement keys and the Page Up and Page Down keys are used to scroll windows only when you specify **With Scroll-Bar** in the Window Setup window. If you specify **Without Scroll-Bar**, you cannot use the Scroll Lock key, because the entire screen is displayed. For example, cursor-movement keys do nothing in Scroll Lock mode.

Customizing a Display Translation Table

PC/3270 displays the host EBCDIC character using the workstation (ANSI) graphic symbol so that the character defined by the zSeries™ EBCDIC host code page is displayed correctly using the same graphic symbol defined by ANSI. However, you might need your original translation, because your host or workstation application is not designed to use the standard translation.

You can use your original translation table if you refer to the following procedure as an example. Note that the data integrity caused by the user-defined table is your responsibility.

The following procedure is an example of how to remap left and right brackets.

1. Terminate all running 3270 sessions
2. Modify the PC/3270 workstation profile (*.WS).

   ```
   [Translation]
   IBMDefaultView=N
   DefaultView=C:\Z and I Emulator for Windows\PRIVATE\BRACKET.XLT
   ```

3. Create the display translation table file (.XLT). In this example, the following BRACKET.XLT file is created in the Z and I Emulator for Windows private subdirectory.

```
[Profile]
id=XLT
Description=User-defined Display Translation Table

[Option]
Replace=Y

[SB Xlate]
; EBCDIC=ANSI
; The next line displays EBCDIC X'AD' as
; an ANSI X'5B' (left bracket)
AD=5B
; The next line displays EBCDIC X'BD' as
; an ANSI X'5D' (right bracket)
BD=5D
```

4. Create your own keyboard layout (.KMP) if you need to enter your new left and right brackets graphic symbols:

```
[Keyboard]
KEY27=ansi dd
KEY28=ansi a8
```

   The information on the right should be lowercase characters. PC/3270 translates ANSI X'dd' into EBCDIC X'ad'. It is displayed as [by the table created in step 3.

5. Click on the PC/3270 icon corresponding to the modified workstation profile.

Support for Long File Names

Like Windows®, Z and I Emulator for Windows supports long file names. You can give any name (up to 255 characters) to a file; you are not limited to eight characters with a three-character extension. You can use spaces in the file name, but not the symbols /, \, :, *, ?, “, <, >, or |. In addition, the tilde (~) character should not be used in CMS or MVS host file names.

File Transfer Function

Host File Name and Reserved Words

You should not use the following words as a VM file name or file type, as a MVS data set name, or as a CICS® file name, because they are reserved for use as option commands:

- ASCII
- APPEND
- TIME
- CLEAR
- NOCLEAR

- SILENT
- QUIET
- PROGRESS
- BLANK
- CRLF
- BINARY
- NOCRLF

Changing the Packet Size When Import/Export Is Idle

When import/export is idle, select **Preferences → Transfer** from the **Edit** menu. When you change the packet size on the Setup window, end import/export, and then rerun it.

Wait Option for Multiple File Transfer

If multiple file transfers do not succeed, insert the following statement into the [Transfer] section of your .WS file:

```
[Transfer]
wait=1000
```

This parameter causes a 1000 msec (1 sec) delay between file transfers. If this does not help, you might need to increase the value again.

NOTRUNC and BLANK Options (SBCS Only)

If you want to add trailing blanks (spaces) to fill the logical record length for each record when downloading a text file, use the following options in the **Additional Options** edit field of the Transfer-Type Definition window.

NOTRUNC : for VM/CMS (PTF# UR35492)
NOTRUNC : for MVS/TSO (PTF# UR34797)
BLANK : for CICS®

Setting the VTAM® PSERVIC Statement

File transfer problems can occur if extended attribute support has not been set on in the VTAM® PSERVIC statement. For extended attribute support, set on the high order bit in PSERVIC byte1 (zero byte origin) as follows: x'xx80xxxxxx...'.

Entry assist feature in 3270 display session

The Entry Assist (DOC mode) features enable easier editing of text documents. The Entry Assist features are available only for 3270 display sessions.

Enabling DOC mode

Enabling DOC mode The Entry Assist features can be enabled by adding the keyword Docmode=Y to the .WS session profile. Set Docmode=N to disable Entry Assist.

When Entry Assist is enabled, the DOC indicator is displayed in positions 67, 68, and 69 of the Operator Information Area (OIA). Also, all the pasting functions on the Paste tab are ignored (see Pasting in DOC mode on page 199).

You can assign the mode toggle to a keyboard key or button in the Popup Keypad. The default key assignment for DOC mode is Alt+F1.

Word wrap

You can enable this function by adding the keyword DocmodeWordWrap=Y to the .WS session profile. When DOC mode is enabled, this function is enabled by default. When word wrap is enabled, the Word Wrap indicator is displayed in position 71 in the OIA.

When word wrap is enabled, a word that is being typed at the right margin is moved in its entirety to the first unprotected field in the next row, assuming that the unprotected field has enough left side blank space (spaces and nulls) to contain the word. The vacated area on the previous row is filled with spaces. See the following examples.

When word wrap is disabled:

```
Look in the diction
ary, please.
```

When word wrap is enabled:

```
Look in the
dictionary, please.
```

If the unprotected field does not have sufficient blank space at the left, then the word is not moved (same as when word wrap is disabled).

Start Column and End Column

You can set these values using the keywords DocmodeStartColumn=<value> and DocmodeEndColumn=<value> in the .WS profile. These settings control the left and right margins while the session is in DOC mode. For example, you can set the left margin to 10 and the right margin to 60. If insert mode is on, the following conditions apply:

- When you type a character at the right margin, the cursor returns to the left margin on the next row.
- Protected fields are skipped.

If insert mode is off, the following conditions apply:

- When you are typing data and a character is pushed beyond the right margin, the pushed character is wrapped to the beginning of the next row.
- Protected fields are skipped.

When insert mode is enabled, the ^ character is displayed at position 52 in the OIA.

New Line key

If you press the key (default is Enter key) for New Line, the cursor skips to the first unprotected character position in the next row (or in a subsequent row if necessary) that lies within the margins.

Tab stops

You can set tab stops values with the keyword DocmodeTabStops=<column1,column2,...> in the .WS profile. You can set the tab stops by specifying the number of the columns that you want for the tab stops, separated by commas (for example, 5,10,15,20,25).

When tab stops are set, pressing the Tab key causes the cursor to skip to one of the following, whichever comes first:

- The next tab stop in the same unprotected field on the same row. Note that tab stops cannot be defined outside the left or right margin.
- The first character position in the next unprotected field on the same row, if that character position is within the margins.
- The first character position in the next unprotected field in a subsequent row, if that character position is within the margins.

The following conditions apply when using tab stops:

- Characters that are skipped over as the result of a tab key are not set to blanks.
- Characters that lie within an unprotected field and that the cursor skips over as the result of a tab key are not set to blanks. However, nulls that the cursor skips over as the result of a tab key are set to blanks.

Nulls in an unprotected field straddling a margin

When the tab key causes the cursor to skip to the left margin of the next line, and an unprotected field straddles the left margin, nulls that lie outside the left margin in the unprotected field are converted to spaces. However, when the tab key causes the cursor to skip to the next line, and an unprotected field straddles the right margin of the preceding line, nulls that lie outside the right margin in the unprotected field are not converted to spaces.

Enable audible End of Line signal

You can enable this function by adding the keyword DocmodeEndofLineSignal=Y to the .WS profile. The function enables an audible signal when the cursor enters the column set for the End of Line signal column.

End of Line signal column

You can enable this function by adding the keyword DocmodeEndofLineSignalColumn=<value> to the .WS profile. You can set the column value at which you want the End of Line signal to be sounded. You could type in a value that would cause the audible signal to occur when the cursor approached the right margin. For example, if the right margin is 70, you might set the End of Line signal column to 65.

Pasting in DOC mode

When DOC mode is enabled, all pasting functions on the Paste tab are ignored and pasting is done according to the following rules.

If the cursor is within the margins:

- Start pasting characters at the cursor.
- Paste characters according to the same rules that are used when the user is typing characters.
- Follow the word wrapping rules if Word Wrap is enabled.
- Display the Too Much symbol in the OIA if the end of the last row is reached before all the data is pasted.

If the cursor is outside the right margin:

- Start pasting characters at the cursor.
- Paste characters into unprotected fields until the end of the row is reached.
- Start at the left margin of the next row and continue pasting according to the rules that apply when the cursor is within the margins.

If the cursor is outside the left margin:

- Start pasting characters at the cursor.
- Paste characters into unprotected fields until the left margin is reached.
- Continue pasting according to the rules that apply when the cursor is within the margins.

Graphic Functions

This section provides information, restrictions, and considerations for graphic functions.

Graphics Protocols

Z and I Emulator for Windows allows you to use host graphics applications, such as GDDM® and others. Two types of graphics are supported:

- Vector
- Programmed symbols

Two protocols are supported for vector graphics:

- Advanced
- Native

See Configuring Graphics in the online helps for a description of these protocols and to learn how to configure your sessions for graphics.

The following functions are supported:

- Multiple mixed alphanumeric and graphics host sessions
- Use of standard OS/2® printing and plotting facilities
- Creation of PIF (Picture Interchange Format) files
- Clipping graphics data into the clipboard

Vector Graphics

Vector graphics are computer graphics in which display images are generated from display commands and coordinate data. Z and I Emulator for Windows provides vector graphics support for the OS/2-Link (advanced) or the 3179G or GOCA (native) protocols. Choose the protocol that is appropriate for your host applications.

Advanced Protocol

Use the advanced protocol when you have GDDM® Version 2 Release 3 or later and are using any of the following operating systems:

- MVS
- VSE
- VM/SP
- VM/XA SP™

Note: The advanced protocol is not supported by the CICS® pseudo-conversational mode with versions of GDDM® earlier than Version 3, and not by IMS/VS at all. It is, however, supported by the CICS® pseudo-conversational mode with GDDM® Version 3 Release 1 or later.

The advanced protocol is equivalent to that used by OS/2-Link, so it supports the same subsystems. However, no download of code from the host system is required for Z and I Emulator for Windows because all the OS/2-Link graphics modules are integrated into the program.

Native Protocol

Choose the native protocol when you intend to use older GDDM® versions or non-GDDM host-graphics applications, such as those originally intended for use on 3270 nonprogrammable terminals as the 3179G, 3192G or 3472G. The native protocol also allows IMS/VS users to display GDDM® graphics.

Programmed Symbols

Raster graphics are displayed with programmed symbols, which are downloaded to your workstation. Z and I Emulator for Windows supports up to six sets (PSA through PSF) of triple-plane and multiple-color programmed symbols.

Use programmed symbols as the graphics type when you intend to use host graphics applications originally written for the 3279G terminal.

Graphics applications use one or both of these methods to display graphical screens. Z and I Emulator for Windows allows you to enable or disable support for vector graphics and programmed symbols. Choose the type of support that our host applications require.

Note: If you use the OS2-Link (advanced) protocol under the GDDM® program, do not choose programmed symbols. Also, do not choose programmed symbols when you use the OS2-Link protocol with other applications.

Enabling Programmed Symbol Sets

PC/3270 provides up to six sets of triple-plane programmed symbols, depending on the type of graphics support that you choose. By default:

- Two sets (PSA and PSB) of single-plane programmed symbols are usable if you choose both programmed symbols and vector graphics.
- Three sets (PSA, PSB, and PSE) of single-plane programmed symbols and three sets (PSC, PSD, and PSF) of triple-plane programmed symbols are usable if you choose programmed symbols, but not vector graphics.

You can change the number of programmed-symbol sets and triple or single planes available for each programmed-symbol set by editing the [3270] section of the workstation profile:

```
PSSPlanes=xxxxxx
```

Each *x* represents a number (0, 1, or 3) that indicates how many planes are to be available for each set; the first column indicates the number of planes for PSA, the second column for PSB, and so on. For example, to enable six triple-plane programmed symbol sets, enter the following:

```
PSSPlanes=333333
```

To enable two single-plane and two triple-plane sets, enter the following:

```
PSSPlanes=113300
```

How to Handle Errors Caused by Insufficient Memory

Graphic execution module PCSGRP.DLL uses a large amount of global memory for graphic drawing or printing. When the workstation has insufficient installed memory, results might not be correct. For example, an area might not be clearly shaded.

In this case, increase the amount of installed workstation memory by at least 1 MB. For host graphic printing, add 1 more megabyte.

Memory might have to be further extended depending on the host graphic application and printer driver used.

Drawing-Buffer Size

The drawing-buffer size varies depending on the contents set for Redraw of a graphic function.

To set **Redraw**, click **Preferences → Appearance → Display Setup** from the **Edit** menu in the session window. Select **Graphics** from **Category**.

Selecting **Host** from the optional items of **Redraw** requires no buffer.

If you select **Retained**, the graphic execution module stores all redrawing data into a buffer. Such a buffer is called a *retained buffer*. The buffer size varies depending on the complexity of the graphic data from an application program. For example, a simple table has a buffer size of 10 KB to 20 KB, while a complicated graphic image has a buffer size of 200 KB to 300 KB.

When you select **Bitmap** to set **Redraw**, the buffer size will be the same as the sum of the retained buffer size and compatible bit map size:

(Height) x (Width) x (Number of planes) x (Bits/Pixel) / 8 bytes

For example, when you select a 7x12 font for a VGA 16-Color Display Model 2 (24x80), the bitmap size is:

(7x80) x (12x24) x 1 x 4 / 8 = 80 KB

When you select a 12x20 font for an IBM® PS/55 High-Resolution 256-Color Display Model 2, the bitmap size is:

(12x80) x (20x24) x 1 x 8 / 8 = 460 KB

Using Bitmaps for Drawing

The graphic execution module uses a bit map compatible with the display unit to draw an area instruction in overpaint mode. An image instruction requires one plane bit map.

(Buffer for area)	=	(Area width) x (Area height) x (Number of planes) x (Bits/Pixel) / 8
(Image buffer)	=	(Image width) x (Image height) / 8

Print Buffer Size

The retained buffer must be used for printing. The retained buffer is the same size as that used for redrawing. This is also applied when you specified **Bit Map** for **Redraw** on the Display Setup window.

If graphic printing is called in Bitmap mode, the graphic printing module generates a bitmap compatible with the connected printer, draws an image on the bitmap, and transfers the bit image to the printer.

This operation is generally performed quickly. When memory is frequently swapped, the process slows down in proportion to the number of swap operations. If a large bit map is not allocated, the graphic printing module prints a graphic image normally using only the retained buffer.

Example:

Proprinter (240x144 DPI) character size:
Bitmap size = (240x8) x (144x11) x 1 x 1 / 8 = 380 KB

Example:

EPSON (ESC/P) (360x180 DPI color) character size:
Bitmap size = (360x8.5) x (180x11) x 3 x 1 / 8 = 2.3 MB

Note: With some printers, different printing results might be obtained in bitmap mode and non-bitmap mode. If the desired results are not obtained, change the current bitmap mode. For example, specify non-bitmap mode to print in bitmap mode.

Edit-Copy Buffer

An editing operation causes the graphic execution module to copy a bit map and DIBitmap to the clipboard. The bit map is compatible with the display; DIBitmap is a 4-bit/pixel bit map.

Printer Fonts

The printer driver can handle two font sets, the device font and GDI font. The device font is a hardware font built into the printer. The GDI fonts are System (without brackets) or other software fonts for Windows®.

When you select a font set for graphic printing from the Printer Control window, use the GDI fonts for the following reasons:

- In bitmap mode, a GDI font can be used for printing. However, the device font cannot be used, because an image cannot be drawn on a memory bit map when using the device font.
- In bitmap mode, the device font cannot be used for printing when OR and exclusive OR attributes are mixed.

Plotter

Because a plotter does not support a raster, the following restrictions are imposed on drawing. Use a plotter for figures and tables that have mainly lines.

- No shading is supported.
- Some shading patterns cannot be distinguished.
- Image order drawing requires much time, and the final printout is of poor quality.
- The OR and exclusive OR are not correctly reflected.

Hole in Screen Caused by Clearing a Graphic Character

When a character overlaps a graphic image, the graphic image is cleared at the position where the character is to be displayed. When you enter a null character or space having the transparent attribute at the position where a graphic image is displayed, the graphic image in that character cell is not cleared.

If you select **Host** or **Retained** to set **Redraw** on the Display Setup window, when characters in a graphic image are cleared, a hole appears in the graphic area. This is because these two modes do not have a bitmap image, and partial redrawing cannot be performed on the screen.

If you select **Bitmap** mode as **Redraw Graphics**, you can find no hole on the graphic region by the application that overrides any alphanumeric characters (as well as NULL and SPACE) on the graphic image.

To restore the screen, perform either of the following actions:

- Press the PA3 key to have the application program redraw the screen.
- Minimize and restore the graphic image retained in Retained mode, then redraw it or select Bitmap mode.

Note: When you change the setting of **Redraw** in the Display Setup window, the set contents are valid from the next drawing.

Miscellaneous Restrictions for Graphic Functions

If advanced protocol is selected, graphic functions cannot be used in the IMS/VS and CICS® pseudo-conversational mode with versions of GDDM® earlier than Version 3.

Considerations for Graphics Functions

Native-Graphics Datastream

If the host sends an Object Structured Field (Object Picture, Object Data, Object Control) with a zero value in the length field, Z and I Emulator for Windows rejects it and displays PROG754.

Printout to LPT1

When you use a host application that prints to your PC's LPT1, you must first select the printer in the Printer Setup dialog of the File menu.

Print Processing

Transferring Files

Z and I Emulator for Windows File Transfer enables you to transfer one or more files between a host system and workstation at the same time. Transfer types and translation tables can be defined in advance.

You can perform the following file transfer functions:

- Send files to the host system
- Receive files from the host system
- Use lists of files
- Create templates to define file names and transfer types
- Define transfer types
- Set transfer options
- Modify translation tables
- Import or export files (PC/3270 CICS only)
- Create interactive document profile (IDP) files (PC/3270 CICS only)
- Transfer files via the XMODEM or YMODEM protocols

Note:

PCT400 was withdrawn from marketing 3/98.

Host Requirements

For PC/3270 File Transfer in SBCS mode, you need one or more of the following host file-transfer programs (referred to as IND$FILE):

- IBM 3270-PC File Transfer Program, 5665-311 (MVS/TSO)
- IBM 3270-PC File Transfer Program, 5664-281 (VM/SP 2.1)
- IBM CICS/VS 3270-PC File Transfer Program, 5798-DQH (CICS/VS 1.5)

Sending Files to the Host System

To send a file from your workstation to the host system:

1. Sign on to the host system.
2. Click **Send File to Host** from the **Actions** menu of the session window. (You can also select the **Send** button on the tool bar.)

   The Send File to Host window opens.
3. To use a list file, click **Open List**. Select the list to be used for transfer. See Creating List Files on page 207 for details of how to create list files.

   If you do not want to use a list file, proceed to the next step.
4. Type the name of the **PC File** to be sent to the host system, or click **Browse** to select the file. If a template is provided for the file type you are transferring, the host file name and transfer type appear automatically.
5. Type the **Host File Name**. For MVS/TSO, you can click **Browse** to view the datasets and members on the host (3270 only). Select the files to send, then click **OK** to add the files to the transfer list.
6. Select the **Transfer Type**.
7. Click **Send**.

   The file is sent to the host system. The send status appears in the Send a File Status window.

Receiving Files from the Host System

To transfer a file from the host system to your workstation:

1. Sign on to the host system.
2. Click **Receive File from Host** from the **Actions** menu. (You can also select the **Receive** button from the tool bar.)

   The Receive File from Host window opens.
3. To use a list file, click **Open List**. Select the list to be used for transfer. See Creating List Files on page 207 for details of how to create list files.

   If you do not want to use a list file, proceed to the next step.
4. Type the name of the **Host File** to be received. You can also specify the host file name as follows:
   - **Using the Clipboard button**

     If you have copied one or more host file names to the clipboard, you can click the **Clipboard** button and paste the names into the transfer list. Select one or more of the pasted file names to be transferred and click **OK**.
   - **Using the Browse button**

     For MVS/TSO, you can click **Browse** to view the datasets and members (3270 only). Select one or more of the files to receive, then click **OK** to add the files to the transfer list.

   If a template is provided for the file type you are transferring, the PC file name and transfer type appear automatically.
5. Type the **PC File Name** or click **Browse** button to select a location for the file.
6. Select the **Transfer Type**.

7. Click **Receive**.

   The receive status appears in the **Receive a File Status** window.

Using List Files

If the same files are transmitted frequently, you can create a list of the files and save it.

A list file can be used for both Send and Receive. The default list file extension is .SRL.

Creating List Files

To create a list file:

1. Select **Receive File from Host** from the **Actions** menu or **Send File to Host** from the **Actions** menu of the session window; or click the **Send** or **Receive** buttons on the tool bar.

   The corresponding window opens.
2. Select a file to be transferred from the **Host-File Name** or **PC-File Name** list box by pointing to the name of a file to be selected. While holding down the Ctrl key, click the left mouse button.

   The file name, its corresponding workstation or host file name (according to the available templates), and the transfer type appear in the **Transfer List** part of the window.

   Note: You can also click the **Browse** button (for sending files) or the **Clipboard** button (for receiving files) to open the corresponding dialog box, which allows you to select files for transferring; when you click **OK**, the selected files are shown in the **Transfer List**.
3. Click the **Add to List** button to include a selected file in the **Transfer List**.
4. After all desired files have been selected, click **Save List**.

   The Save File-Transfer List File As window opens.
5. Enter or select a list name, and click **OK**.

Editing Lists

To edit the contents of a previously created list:

1. As explained in Sending Files to the Host System on page 205 and Receiving Files from the Host System on page 206, display the Send File to Host or Receive File from Host window.
2. Select **Open List**.

   The Open File-Transfer List File window opens.
3. Select the name corresponding to the list file to be edited, then click **OK**.
4. The contents of the selected list appear in the Send File to Host or Receive File from Host window.
5. Edit the contents of the list file.

Changing the contents of a list: Choose the file to be changed from the list, and overwrite the items to be changed in the text box; then click the **Update in List** button.

Removing a file from the list: Choose the file to be removed, and click **Remove from List**.

Adding a file to the list: Double-click the file to be added from the list of host or workstation files.

6. Select **Save List**.

   The Save File-Transfer List File As window opens.
7. Enter a name and then click **OK**.

Managing Templates

A *template* is a set of rules to be used by the workstation to automatically generate a workstation or host file name and transfer type when you specify a file to be sent or received.

You can have up to 32 templates. They are automatically numbered from 1 to 32.

When you specify a file to be transferred, the workstation scans the templates, starting from template 1. It uses the first matching template to generate a name for the transferred file and the transfer type.

To manage a template:

1. Click **Receive File from Host** from the **Actions** menu or **Send File to Host** from the **Actions** menu of the session window; or click the **Send** or **Receive** buttons on the tool bar.

   The Send File to Host or Receive File from Host window opens.
2. Select **Template**.

   The Template window opens. The contents of the window depend on the connected host system.

Adding Templates

The list box for the Template window lists the currently stored templates.

To add a template:

1. Select any template from the list box.

   The contents of the selected template appear under the list box.
2. Change the workstation or host file names or extensions by overwriting them; then select the transfer type. (For details of the transfer types, see Defining Transfer Types on page 209.)
3. Click **Add**.

   The window for determining where in the list to display the new template opens.

4. Select a template number and specify whether to display the new template before or after the template that has that number. Click **OK**.

   The new template is added to the list in the appropriate position.

Replacing and Deleting Templates

To change the contents of a currently stored template, or to delete a template:

1. Select the template to be changed or deleted.

   The contents of the selected template appear under the list box.
2. To change the contents, overwrite the appropriate part and then click **Replace**.

   To delete a template, click **Delete**.

   The selected template is changed or deleted, and the contents of the template list box are changed.

Testing Templates

To test the contents of an added or changed template:

1. Select the template to be tested from the list box.

   The number of the selected template appears in the Test Templates box in the lower part of the window.
2. Select or enter data for the following items:

   Test Mode

   Determine which mode is to be used for the test: the mode in which a file is transmitted from the workstation to the host system (send), or the mode in which a file is transmitted from the host system to the workstation (receive).

   Templates

   Determine which templates to test: only the template selected in step 1, or all registered templates.

   Source File

   Enter the name of the file to be used for the test.
3. Click **Test**.

   Target File indicates the name that has been generated by the template.

Note: Testing a template does not transfer a file.

Defining Transfer Types

Transfer types define the option information used for controlling file transfer. Up to 32 transfer types can be defined for each host system. Text, binary, and append (excluding CICS) are the defaults.

To add or change transfer types:

1. Click **Edit → Preferences → Transfer** from the session window.
2. Click the tab for your host type or modem protocol.

   The property page for the selected host or modem protocol opens. The items that appear depend on the selected host system.
3. Enter transfer-type names in the **Transfer Type** box, or select them from the drop-down list.
4. Select or enter the required items (see Items to Be Specified on page 210).

   To add or replace a transfer type, click **Save**. To delete a transfer type, click **Delete**.
5. A dialog box displays, asking for confirmation. Click **OK**.

Items to Be Specified

Choosing the appropriate property page enables you to set the items described in the following sections.

File Options

The file options that can be used depend on the type of the connected host system and the host code page selected when the session was configured. Table 31: Mode Values for File Transfer Options on page 210 lists the mode values for the file transfer options. Table 32: Transfer File Options on page 210 lists the transfer options.

Table 31. Mode Values for File Transfer Options

Mode	Host Code Page
SBCS	Others

Table 32: Transfer File Options on page 210 lists the options for PC/3270.

Table 32. Transfer File Options

File Option	Host System	Mode	Conversion Details
ASCII	VM/CMS MVS/TSO ICS	SBCS	Converts codes as follows when a file is sent: • Converts 1-byte workstation codes to EBCDIC codes • Converts RS (hex 1E) and US (hex 1F) to SO (hex 0E) and SI (hex 0F) Converts codes as follows when a file is received: • Converts EBCDIC codes to 1-byte workstation codes

Table 32. Transfer File Options (continued)

File Option	Host System	Mode	Conversion Details
CRLF	VM CMS MVS/TSO CICS	SBCS	Converts codes as follows when a file is sent: • Does not remove CRLF (hex 0D0A) from the end of each line. The code is treated as a delimiter for each record. • Removes EOF (hex 1A) from the end of the file. Converts codes as follows when a file is received: • Adds CRLF (hex 0D0A) to the end of each line. • Adds EOF (hex 1A) to the end of the file. Removes EOF from the existing file, and appends EOF to the end of the added file when APPEND is specified.
APPEND	VM/CMS MVS/TSO	SBCS	Appends the sent file to the existing host file. Appends the received file to the existing workstation file.
SRC	OS/400 i5/OS	SBCS	This option is good only for Send. When SRC is checked, the target file is stored as a member of the physical source file. If the file already exists on the Host system, this option is ignored.

Record Format

Valid only for VM/CMS and MVS/TSO when APPEND is not specified for file transmission. You can select any of the following:

- **Default**
- **Fixed** (fixed length)
- **Variable** (variable length)
- **Undefined** (undefined mode for MVS/TSO only)

If you select the **Default** value, the record format is selected automatically by the host system.

Specifying **Variable** for VM file transfer enables host disk space to be used efficiently.

Logical Record Length (LRECL)

Valid only for VM/CMS and MVS/TSO when APPEND is not specified for file transmission.

Enter the **logical record length** to be used (host record byte count) in the **LRECL** text box. If **Variable** and **Undefined Mode** are specified as the record format, the logical record length is the maximum record length within a file. The maximum value is 32767.

The record length of a file sent from a workstation to the host system might exceed the logical record length specified here. If so, the host file transfer program divides the file by the logical record length.

To send a file containing long records to the host system, specify a sufficiently long logical record length.

Because the record length of a workstation file exceeds the logical record length, a message does not appear normally if each record is divided. To display a message, add the following specification to the [Transfer] item of the workstation profile:

```
DisplayTruncateMessage = Y
```

TSO Allocation Parameter (MVS/TSO)

Valid only for MVS/TSO when **APPEND** is not specified for file transmission. The following items can be specified:

[Allocation Amounts]

Primary

Enter the number of tracks or cylinders allocated to this file transfer.

Secondary

If the primary allocation is not sufficient for the entire file transfer, enter additional storage capacity allocated to the file transfer.

[Allocation Units]

Tracks

Specify this parameter to allocate a host file by track. Ask your system manager whether to use tracks or cylinders as the unit.

Cylinders

Specify this parameter to allocate a host file in units of cylinders.

AVblocks

Specify this parameter to allocate a host file in units of blocks.

[Block size]

This item is used only to create a new data set. Enter the block size of a new host data set, in bytes, in the text box. If this item is omitted, the workstation assumes the value that appears in the **Logical Record Length** box. The maximum value is 32767. If **AVblocks** is selected, the block size is the block size of the new data set.

Additional Options

You can enter the required host command options in the **Additional Options** text box.

Setting General Transfer Options

To set advanced options:

1. Click **Edit → Preferences → Transfer** from the session window.

   The setup dialog is displayed.
2. Change the required settings on the property page labeled **General**.
3. Click **OK**.

The following sections contain information about the items which can be defined for file transfer options.

Host Type

You can specify from the drop-down list box the type of host (MVS/TSO, VM, or CICS) to which your workstation is connected.

Host Command

You can specify host command to be called when file transfer starts. If nothing is entered in this text box, IND$FILE or its equivalent for other countries is used for 3270 SBCS sessions.

Default PC Directory

You can specify the default directory that appears in the Send File to Host or Receive File From Host window. To select the directory, click the **Browse** button.

Default Partitioned Data Set (MVS/TSO Only)

You can specify the MVS partitioned data set to be used as the default.

Default VM Disk (VM Only)

You can specify the VM disk to be used as the default.

PC Code Page

When a file is transferred, EBCDIC codes are converted to 1-byte workstation codes, and vice versa. A valid value is automatically selected from among the following values for SBCS sessions: 437, 737, 806, 813, 819, 833, 850, 852, 854, 857, 858, 860, 861, 862, 863, 864, 865, 866, 869, 874, 912, 915, 916, 920, 921, 922, 1008, 1089, 1124, 1125, 1127, 1129, 1131, 1133, 1153, 1155, 1156, 1157, 1158, 1160, 1164, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, and

1258; —according to the host code page specified when the workstation is configured. For an explanation of how to select host code pages, see the online help for the host code page.

Packet Size

The amount of memory (in bytes) used by the workstation for transmission and reception. If a large value is entered, a file is transferred more quickly, but the memory overhead is larger. The default value is 12288. In the case of Telnet3270, you can specify a packet size larger than 8000 bytes by adding the following line to the Telnet3270 stanza in your workstation profile:

```
SendBufferSize=nnnn
```

File-Transfer Timeout

You can define the time the workstation waits for a response from the host system (in seconds). If the host system does not respond, the transfer is canceled, and an error message appears. A number in the range 20–65535 (or 0) can be specified. The default is 60 seconds for ASCII sessions; for all others, it is 30 seconds. Specify an appropriate value such that the error message does not appear too early. If you specify 0, a timeout is not set.

If a packet or block size is relatively large for low-speed lines, such as COM port lines, it is recommended that 150 seconds or greater be specified.

Extension for List-Files

You can change the default extension (.SRL) of file-transfer list files.

Clear Session Before Transfer

You can specify whether a Clear command is sent to the host system before a file is transferred. Choose any of these option buttons:

Default

A Clear command is sent before a file is transferred (VM/CMS or CICS only).

Yes

A Clear command is sent for MVS also.

No

A Clear command is not sent for any host system.

Show Status Window

You can choose the method of displaying the file-transfer-progress status.

In Session

When file transfer starts, the status window opens. The name of the file being transferred and the transfer progress appear.

In icon

When file transfer starts, the status icon appears on the screen. If the icon is restored, the status window opens.

Setting Up the Translation Table

You can create or edit the translation table to be used for sending or receiving files.

Changing the Translation Table

To change the translation table:

1. Click **Edit → Preferences → Transfer** from the session window.
2. Click the **Translation Table** tab on the resulting window.

   The Translation-Table Setup property page opens.
3. The table currently being used (default or the name of a user-defined table) is shown. Choose either **Default** or **User-Defined**.
4. If you choose **User-Defined**, enter a translation-table name in the **File Name** text box, or select a name by clicking **Browse**.
5. Click **OK**.

Customizing the Translation Table

You can create a user-specific translation table for transmission or reception, or you can edit an existing translation table.

To create or edit a translation table:

1. On the **Translation Tables** property page, click **Customize** in the Upload or Download window.

   The Customize Translation window opens.

   If you chose **Default** or if you chose **New** from the File menu, the default values appear in the table.

   Translation source codes

   PC code-points when an upload translation table is edited. Host code-points when a download translation table is edited.

Translation target codes

Host code-points when an upload translation table is edited. PC code-points when a download translation table is edited.

2. Double-click the code to be changed in the table, and change the value in the entry field that subsequently appears.
3. Click **Save** or **Save As** from the File menu.
4. If asked, enter a name in the Save Translation File As window and click **OK**.
5. Click **Exit** from the File menu of the Customize Translation window.

Import/Export (3270 CICS Only)

Import/Export is an office system communication program and an application program executed under the IBM Customer Information Control System (CICS).

Clicking **Import/Export** loads a module into workstation memory. You can then start Import or Export from a menu on the host screen.

When you export a document from the host, the workstation receives two files: one is the file itself, and the other is the interchange document profile (IDP) file, which contains document header information.

When you **Import** a file to a host system, it must be accompanied by an IDP file of the same name. If the necessary IDP file does not exist, you can create it as described in .

To transmit files using Import/Export:

1. Verify that the window of the host session is active and ready for file transfer.
2. Click **Import/Export** from the **Actions** menu of the session window.

   The minimized Import/Export Status window opens.
3. Click **Import** or **Export** from the host application menu.
4. Specify the host and workstation file names of the file to be transferred. Run Import or Export.

   When Import or Export starts, the Import/Export Status window is maximized.

   After the file is transferred, the window is closed.

File Transfer Commands for PC/3270

You can send data files to and receive them from IBM host systems that are running:

CICS/MVS

Customer Information Control System running under MVS

CICS/VSE

CICS running under Virtual Storage Extended

MVS/TSO

Multiple Virtual Storage/Time Sharing Option

VM/CMS

Virtual Machine/Conversational Monitor System

For more information on using these commands, click **Send File to Host** from the **Actions** menu and **File Transfer from Command prompt** in the help panel.

File Transfer Methods

You can transfer files in the following ways with PC/3270:

- By clicking **Receive File from Host** from the **Actions** menu or **Send File to Host** from the **Actions** menu of the workstation window
- By using the **SEND** and **RECEIVE** commands at the DOS command prompt
- By using an EHLLAPI application that invokes file transfer
- By using a macro that has send or receive commands as macro statements
- By clicking the **Send** or **Recv** icon on the tool bar

Requirements and Restrictions

Install the file transfer program, IND$FILE, on your host system. Ask your system administrator for additional file transfer procedures and precautions. An alternate host command name can be used by defining a DOS environment variable IND_FILE in AUTOEXEC.BAT or in a particular DOS box. For example:

```
SET IND_FILE = MYXFER
```

You should not use the following words as a VM file name or file type, as an MVS data set name, or as a CICS file name, because they are reserved for use as option commands.

ASCII, APPEND, TIME, CLEAR, NOCLEAR, SILENT, QUIET, PROGRESS,
BLANK, CRLF, BINARY, NOCRLF

If you want to send to or receive from a subdirectory other than \Z and I Emulator for Windows, you must specify the full path name.

Sending and Receiving Files from the DOS Command Prompt

The workstation is the point of reference for the SEND and RECEIVE commands: You send from the workstation to the host and receive from the host to the workstation.

To send or receive a file:

1. Make sure you are logged on to your host.
2. Make sure the **Ready** message of the host system is displayed, except if you are transferring files through the command option of the ISPF application.

   Note: In the latter case, you must specify the NOCLEAR option for the file transfer command.

   If your screen is blank, make sure that no applications are running and that your host session is not in a *holding* state.

   Note: If you receive any messages from host application programs while you are transferring files, the transfer might not succeed. To prevent messages from interfering, enter the appropriate host command to set messages off temporarily. When file transfer is finished, set messages on again.
3. Switch to your DOS window session or DOS full-screen session.
4. If you use a hard disk, make sure the SEND.EXE and RECEIVE.EXE files are in your current directory or in your path. If you want to send to or receive from a subdirectory other than \Z and I Emulator for Windows, you must specify the full path name.
5. Type the appropriate SEND or RECEIVE command at the DOS command prompt.

   Details on the SEND and RECEIVE commands and their options are explained in the following sections.

Using the VM/CMS SEND Command

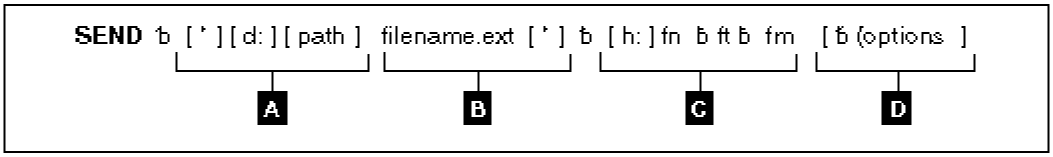
Use the following information when sending a file to VM/CMS:

Figure 1: VM/CMS SEND Command Syntax on page 218 shows the command and information that you must provide. Enter it as shown (including parentheses). You can use either uppercase or lowercase letters.

Note:

1. ƀƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 1. VM/CMS SEND Command Syntax


SEND ƀ ['] [d:] [path] filename.ext ['] ƀ [h:] fn ƀ ft ƀ fm [ƀ (options]
A
B
C
D

A

The workstation drive and path of the file to send.

B

The name of the workstation file to send.

C

Host session specifications for the file to be sent to the host.

`h:`

The short name of the session (which can be omitted if it is **a**)

`fn`

File name

`ft`

File type

`fm`

File mode

D

Optional changes made to the file during transfer. More than one option can be selected. Valid options are:

- APPEND
- ASCII
- CLEAR
- CRLF
- LRECL `n`
- NOCLEAR
- PROGRESS
- QUIET
- RECFM `x`
- TIME(`n`)

The parts of the VM/CMS SEND command are:

SEND

The command.

`d:`

The name of the diskette or hard disk drive on which the file is located.

`path`

The path to the subdirectory that the file is in.

`filename.ext`

The name of the file to be sent, including the extension.

`h:`

The short name of the host session to which you want to send the file. The default is `a:`.

`fn ft fm`

The name the file is to have on your VM/CMS disk. You must specify the file name (`fn`) and file type (`ft`). You can omit file mode (`fm`) if you want the file placed on your A-disk. You can create a new name or use a name that is already on your disk. If you use a new name, the file that you send is added to your disk. If you use the name of an existing file, the file that you send either replaces or is added to the old file. (Refer to the description of the APPEND option.)

(`options`

These options can be specified:

APPEND

Specifies that the file being sent is to be added to the end of an existing VM/CMS file. Omit this option if you want the file to replace an existing file. You cannot specify the LRECL `n` or RECFM `x` option if you use the APPEND option.

ASCII

Performs the following:

- Converts 1-byte workstation codes to EBCDIC codes.

CLEAR

Clears the workstation window at the beginning of the file transfer. **CLEAR** is the default.

CRLF

Specifies preserving of the carriage return and line feed codes. You need the ASCII and CRLF options for text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files, such as programs.

LRECL `n`

Specifies the file's record length. Include a record length only if you want the file to have a record length on your VM/CMS disk other than 80. Replace `n` with the record length you want. If you omit this option, the record length is set to 80 for fixed-length records or to a maximum of 80 for variable-length records.

NOCLEAR

Suppresses the sending of a Clear command at the beginning of the file transfer.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

RECFM `x`

Specifies the file record format. Use this parameter to specify variable-length or fixed-length records in the file. Replace `x` with V for variable or F for fixed. By default, the file has fixed-length records unless you specify the CRLF option; then the file has variable-length records unless you specify otherwise.

TIME(`n`**)**

Specifies the length of time `n`, in units of 30 seconds, that the program waits for a response from the host before it sends an error message. Replace `n` with an integer value in the range from 0 through 2184. If you specify 0, timeout will not be set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, of large block sizes, or for slow communication lines (such as and COM port), *5* (150 seconds) is recommended. There should be no blank spaces is between TIME and (`n`).

Command Syntax for Sending Files to VM/CMS

The following examples show the command syntax you can use to send files to a VM/CMS host. The parameters of the SEND command can be combined into a single set of parentheses.

- To send a workstation file from your default drive and add it as a new file on your VM/CMS A-disk:

  ```
  SEND pc.txt a:cmsfile script a (ASCII CRLF LRECL 72 RECFM V
  ```

  Note: If you use a command that exceeds one line, do not press Enter when you fill that line; continue typing your command.

  This command sends a workstation file named PC.TXT from your default drive to your host in your host session named **a**. You do not need to specify the workstation drive if the file you are sending is on the current drive. The command creates a new file, named CMSFILE SCRIPT, on your A-disk. The records in the file can vary in length up to 72 characters.
- To send a workstation file from your default drive to replace a file on your VM/CMS A-disk:

  ```
  SEND pc.txt a:cmsfile script a (ASCII CRLF
  ```

  This command sends a workstation file named PC.TXT from your default drive to your VM/CMS A-disk in your host session named **a**. You do not need to name the workstation drive if the file you are sending is on the default drive. The file replaces a SCRIPT file named CMSFILE. The new CMSFILE has the same record length and format as the old CMSFILE.

  If you do not have a file called CMSFILE SCRIPT on your A-disk, PC.TXT is added to your A-disk as a new file called CMSFILE SCRIPT. The records in the file are 80 characters long and have fixed length.
- To send a binary workstation file from a drive other than your default drive:

```
SEND a:pc.exe c:cmsfile exebin b (recfm v
```

This command sends a workstation file named PC.EXE from a diskette in drive A to your VM/CMS B-disk in your host session named **c**. It is a new file, or it replaces a file named CMSFILE.

When transferring a binary file, you must specify a variable record format (**recfm v**), otherwise, blank characters are added to the file.

- To send a file from your hard disk and add it to the end of a file on your VM/CMS A-disk:

```
SEND c:pc.txt cmsfile script a (ASCII CRLF APPEND
```

This command sends a workstation file named PC.TXT from your hard disk to your host session. You do not need to name the host session if you are sending to the **a** session. The file is added to the end of a script file named CMSFILE on your VM/CMS A-disk.

- To send a file from a subdirectory on your hard disk to your VM/CMS A-disk:

```
SEND c:\sd1\pc.txt cmsfile script a (ASCII CRLF
```

This command sends a file named PC.TXT from subdirectory SD1 on your hard disk to your host session. It replaces a SCRIPT file named CMSFILE on your VM/CMS A-disk.

Using the VM/CMS RECEIVE Command

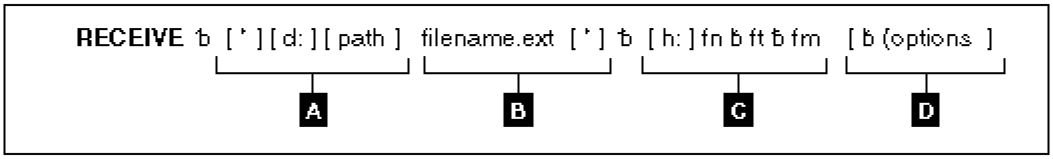
Use the following information when receiving a file from VM/CMS:

Figure 2: VM/CMS RECEIVE Command Syntax on page 222 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase letters.

Note:

1. ƀƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 2. VM/CMS RECEIVE Command Syntax


RECEIVE ƀ ['] [d:] [path] filename.ext ['] ƀ [h:] fn ƀ ft ƀ fm [ƀ (options]
A
B
C
D

A

The workstation drive and path of the file to be received.

B

The name of the workstation file to be received.

C

Host session specifications for the file to be received from the host.

`h:`

The short name of the session (which can be omitted if it is **a**)

`fn`

File name

`ft`

File type

`fm`

File mode

D

Optional changes made to the file during transfer. More than one option can be selected. Valid options are:

- APPEND
- ASCII
- BLANK
- CLEAR
- CRLF
- NOCLEAR
- PROGRESS
- QUIET
- TIME(`n`)

The parts of the VM/CMS RECEIVE command are:

RECEIVE

The command.

`d:`

The name of the diskette or hard disk drive on which the file is to be received.

`path`

The path indicating the directory to which the file is to be stored.

`filename.ext`

The name of the workstation file, including the extension. Use a new name or one that already exists. If you use a new name, the file that you receive is added to your diskette or hard disk. If you use the name of an existing file, the file that you receive either replaces or supplements the existing file. (Refer to the APPEND option.)

`h:`

The short name of the host session from which you want to get the file. The default is `a:`.

`fn ft fm`

The name of the file you want to receive from your VM/CMS disk. The file name `fn` is required.

(`options`

These options can be specified:

APPEND

Specifies that the file being received is to be added to the end of an existing file. Omit this part of the VM/CMS file that is received to replace an existing file.

ASCII

Performs the following:

- Converts EBCDIC codes to 1-byte workstation codes.

BLANK

This option is valid with the CRLF option. Use it to retain BLANK (x'40') at the end of each line.

CRLF

Specifies the carriage return and line feed codes. You need ASCII and CRLF for text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files, such as programs.

CLEAR

Clears the workstation window at the beginning of the file transfer.

NOCLEAR

Suppresses the sending of a Clear command at the beginning of the file transfer.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

TIME(`n`)

Specifies the length of time, in units of 30 seconds, that the program waits for a response from the host before it sends an error message. The value `n` is an integer value in the range from 0 through 2184. If you specify 0, timeout is not set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, of large block sizes, or for slow communication lines (such as and COM port), *5* (150 seconds) is recommended. There should be no blank spaces between TIME and *(n)*.

Command Syntax for Receiving Files from VM/CMS

The following examples show the command syntax you can use to receive files from a VM/CMS host. The parameters of the RECEIVE command can be combined into a single set of parentheses.

- To receive a file from your VM/CMS A-disk to your default drive for a workstationsession:

  ```
  RECEIVE pc.txt a:cmsfile script a (ASCII CRLF
  ```

  This command sends a SCRIPT file CMSFILE from your VM/CMS A-disk in a host session named A to your workstation session. It adds the file to your default drive (diskette or hard disk) with the name PC.TXT.
- To receive a file from your VM/CMS B-disk and replace a file on a drive other than your default:

  ```
  RECEIVE a:pc.txt a:cmsfile script b (ASCII CRLF
  ```

  This command sends a SCRIPT file named CMSFILE SCRIPT from your VM/CMS B-disk in a host session named A to a drive other than the default for your PC session. It replaces a file named PC.TXT on a diskette in drive A.
- To receive a file from your VM/CMS A-disk and add it to the end of a file on your hard disk:

  ```
  RECEIVE c:pc.txt a:cmsfile script a (ASCII CRLF APPEND
  ```

  This command sends a SCRIPT file named CMSFILE SCRIPT from your VM/CMS A-disk in a host session named A to your workstation session. It adds the contents of CMSFILE to the end of a file named PC.TXT on your hard disk.
- To receive a file from your VM/CMS A-disk and place it in a subdirectory on your default drive:

  ```
  RECEIVE \sd1\pc.txt a:cmsfile script a (ASCII CRLF
  ```

  This command sends a SCRIPT file named CMSFILE SCRIPT from your VM/CMS A-disk to your default drive. It creates or replaces a file named PC.TXT in a subdirectory named \SD1.

Using the MVS/TSO SEND Command

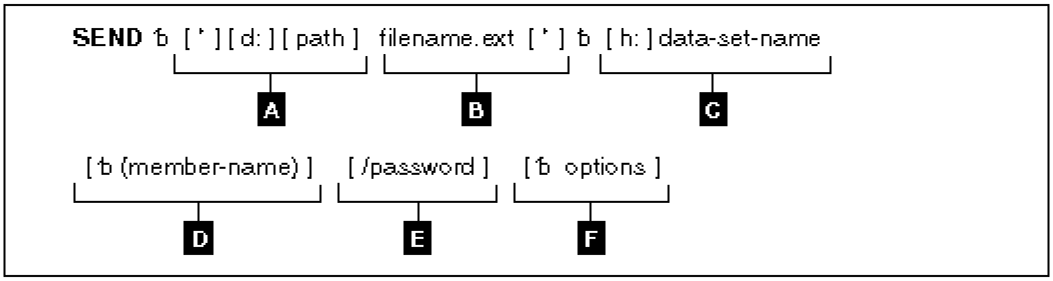
Use the following information when entering the SEND command to the MVS/TSO host:

Figure 3: MVS/TSO SEND Command Syntax on page 226 shows the command and information you must provide. Enter text as shown (including parentheses), except that you can use either uppercase or lowercase letters.

Note:

1. ƀ means to insert a space. There must ***not*** be a space between **h:** and **fn**.

Figure 3. MVS/TSO SEND Command Syntax


SEND ƀ ['] [d:] [path] filename.ext ['] ƀ [h:] data-set-name
A
B
C
[ƀ (member-name)] [/password] [ƀ options]
D
E
F

A

The workstation drive and path of the file to send.

B

The name of the workstation file to send.

C

The short name of the host session and the data set name of the file to send.

D

The member name if the file is in a partitioned data set.

E

The password of the data set if it has one.

F

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- APPEND
- ASCII
- BLKSIZE(n)
- CLEAR
- CRLF
- LRECL(n)
- NOCLEAR

  You must use the NOCLEAR option when you are transferring files while in ISPF command mode on the host.
- PROGRESS

- QUIET
- RECFM(`x`)
- SPACE(`n[,n1]`) `unit`
- TIME(`n`)

The parts of the MVS/TSO SEND command are:

SEND

The command.

`d:`

The name of the diskette or hard disk drive where the file is located.

`path`

The path indicating the directory where the file is located.

`filename.ext`

The name of the file to be sent. Include the extension if the file has one.

`h:`

The name of the MVS/TSO host session to which you want to send the file. You can omit this name if you have only one host. If you have more than one host, this is the short name of the MVS/TSO host session. The default short name is A.

`data-set-name`

The data set name that the file you send is to have on your MVS/TSO volume; this name is required. Enclose the data set name with the member name in single quotation marks if you are using a fully qualified data set name.

This option creates a new name or uses a data set name already on your TSO volume. If you use a new name, the file that you send is added to your MVS/TSO volume. If you use the name of an existing data set, the file you send either replaces or supplements the existing data set. Refer to the APPEND option.

(`member-name`**)**

The member name if the file is to be put into a partitioned data set. If you use member-name, you cannot use LRECL(`n`), BLKSIZE(`n`), RECFM(`x`), and SPACE(`n,[n1]`) `unit`.

Note: If someone else is using the partitioned data set, you cannot send a file to your MVS/TSO host.

/`password`

The password of the data set, if the data set has a password.

`options`

These options can be specified:

APPEND

Specifies that the file being sent is added to the end of an existing MVS/TSO data set. Omit this option if you want the file to replace an existing MVS/TSO data set. You cannot use LRECL(`n`), RECFM(`x`), SPACE(`n[,n1]`) `unit`, or BLKSIZE(`n`) options if you use the APPEND option.

Note: This option is not valid when sending data to a member of a partitioned data set.

ASCII

Performs the following:

- Converts 1-byte workstation codes to EBCDIC codes.

BLKSIZE(`n`)

Specifies the size of the blocks of data in a new data set on your MVS/TSO volume. This part is optional. To set the block size for a new data set, replace `n` with the new size. If you omit this option, the block size is determined in the following manner:

- If the record format is variable, the block size is 6233.
- If the record format is fixed, the block size is the largest multiple of the record length that is less than 6233:

```
BLKSIZE = LRECL * (6233/LRECL)
```

If you use the (member-name) or APPEND option, do not use this option.

CLEAR

Clears the workstation window at the beginning of the file transfer.

CRLF

Specifies the global use of carriage return and line feed codes. You need to specify ASCII and CRLF options for sending text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files.

LRECL(`n`)

Specifies the record length for a new data set on your MVS/TSO volume, where `n` is a whole number from 1 through 32760 representing the number of characters per record. If you want to set the record length for a new data set, replace `n` with the new length. If you omit this option, the record length is set to 80 for fixed-length records and to 255 for variable-length records. If you use the (member-name) or APPEND options, do not use this option.

NOCLEAR

Suppresses the sending of a Clear command at the beginning of the file transfer. This option is required for ISPF command mode.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

RECFM(`x`)

Specifies the record format for a new data set on your MVS/TSO volume, where `x` = V, F, or U. For variable-, fixed- or undefined-length records in the data set, replace the `x` with V, F, or U, respectively.

If you omit this option, the record format of the host data set is determined by the setting of the CRLF parameter: if you specify CRLF, the data set has variable-length records; if you do not specify CRLF, it has fixed-length records. If you use the (member-name) or APPEND options, do not use this option.

SPACE(`n[,n1]`) `unit`

Specifies an amount of space to be set aside for a new data set on your MVS/TSO volume. To set aside a certain number of blocks, tracks, or cylinders for the new data set:

- Provide `unit` as the type of space you want (AVBLOCK, TRACKS, or CYLINDERS).
- Give `n` as the amount of space that you want the data set to occupy (in the unit of measure you select).
- If the data set needs more space than you ask for with `n`, give `n,n1` where `n1` is the size of additional space to be used only when necessary.

These values are similar to the values on the ALLOCATE command of MVS/TSO.

If you omit this option, you get space for one block. The length of the block is set by the BLKSIZE(`n`) or LRECL(`n`) options. If you use the (`member-name`) or APPEND options, do not use this option.

TIME(`n`)

Specifies the length of time, in units of 30 seconds, that the program waits for a response from the host before it sends an error message. The value `n` is an integer value in the range from 0 through 2184. If you specify 0, timeout is not set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, of large block sizes, or for slow communication lines (such as and COM port), *5* (150 seconds) is recommended. There should be no blank spaces between TIME and *(n)*.

Command Syntax for Sending Files to MVS/TSO

The following examples show the command syntax you can use to send files from your workstation to an MVS/TSO host:

- To send a file from your default drive to replace a file on the MVS/TSO host:

  ```
  SEND pc.txt g:ds.script ASCII CRLF
  ```

  This command sends a workstation file named PC.TXT from your default drive to your MVS/TSO host in a host session named G. It creates or replaces a data set named DS.SCRIPT on your MVS/TSO volume.
- To send a file from a drive other than the default to your MVS/TSO host:

  ```
  SEND a:pc.txt g:ds.script ASCII CRLF
  ```

  This command sends a workstation file named PC.TXT from a diskette in drive A to your MVS/TSO host in a host session named G. It replaces a data set named DS.SCRIPT on your MVS/TSO volume.
- To send a file from your default drive to your MVS/TSO host and add it to the end of an MVS/TSO data set:

  ```
  SEND a:pc.txt g:ds.script ASCII CRLF APPEND
  ```

  This command sends a workstation file named PC.TXT from a diskette in drive A to your MVS/TSO host in a host session named G. It adds the file to the end of a data set named DS.SCRIPT on your MVS/TSO volume.
- To send a file to your MVS/TSO host and add it to the end of a data set that has a password:

  ```
  SEND a:pc.txt g:ds.script/odyssey8 ASCII CRLF APPEND
  ```

  This command sends a workstation file named PC.TXT from a diskette in drive A to your MVS/TSO host in a host session named G. It adds the file to the end of a data set named DS.SCRIPT on your MVS/TSO volume. This data set has a password of odyssey8.
- To send a file from a subdirectory on your hard disk to a partitioned data set on your MVS/TSO host:

  ```
  SEND c:\sd1\pc.txt g:ds.script (m1) ASCII CRLF
  ```

  This command sends a workstation file named PC.TXT from a subdirectory named \SD1 on your hard disk to your MVS/TSO host in a host session named G. It creates or replaces a member named M1 in a partitioned data set named DS.SCRIPT on your MVS/TSO volume.
- To send a file from your default drive and add it as a new data set on your MVS/TSO volume:

  ```
  SEND pc.txt g:ds.script/aeneid20 ASCII CRLF LRECL(132)
       BLKSIZE(132) RECFM(V) SPACE(20,10) TRACKS
  ```

  This command sends a workstation file named PC.TXT from your default drive to your MVS/TSO host. It adds the file as a new data set named DS.SCRIPT on your MVS/TSO volume. A password of aeneid20 is assigned. The records in the data set can vary in length up to 132 characters. Data blocks are the same length as the records. Twenty tracks are set aside for this data set. If more tracks are needed, they are added in groups of 10.

Using the MVS/TSO RECEIVE Command

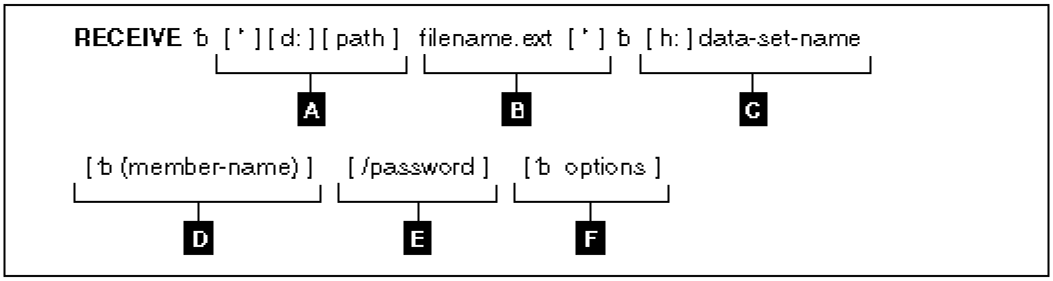
Use the following information when receiving a file from MVS/TSO:

Figure 4: MVS/TSO RECEIVE Command Syntax on page 231 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase letters.

Note:

1. ƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 4. MVS/TSO RECEIVE Command Syntax


RECEIVE ƀ ['] [d:] [path] filename.ext ['] ƀ [h:] data-set-name
A
B
C
[ƀ (member-name)] [/password] [ƀ options]
D
E
F

A

The workstation drive and path to the directory where the file is to be stored.

B

The name of the workstation file to receive.

C

The short name of the host session, and the data set name of the file you are receiving.

D

The member name if the file is put in a partitioned data set.

E

The password of the data set, if any.

F

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- APPEND
- ASCII

- BLANK
- CRLF
- PROGRESS
- QUIET
- TIME(`n`)

The parts of the MVS/TSO RECEIVE commands are:

RECEIVE

The command.

`d:`

The name of the diskette or hard disk drive where the file is to be located. Use A:, B:, C:, D: through Z:. This part is optional if the file is received on the current drive.

`path`

The subdirectory where you want the data set located. This part is optional.

`filename.ext`

The name the file is to have on your diskette or hard disk. Creates a new name or uses a name that is already on your diskette or hard disk.

If you use a new name, the data set that you receive is added to your diskette or hard disk. If you use the name of an existing file, the data set that you receive either replaces or supplements the existing file. (Refer to the APPEND option on page APPEND on page 233.)

`h:`

The short name of the MVS/TSO session where the data set is located. If you have only one host, this part is optional. Use this option if you have more than one host. The default short name is A.

`data-set-name`

The name of the data set or the partitioned data set that contains the member you want to send to your workstation session. You must use the qualified name. Enclose the data set name with the member name in single quotation marks if you are using a fully qualified data set name.

`(member-name)`

The member name of a partitioned data set to send to your workstation session. This part is optional. Use it only if the data set is a member of a partitioned data set.

`/password`

The password of the data set. Use it only if the data set has a password.

(`options`

These options can be specified:

APPEND

Adds the data set to the end of an existing file. Omit this part if you want the MVS/TSO data set to replace an existing workstation file.

ASCII

Performs the following:

- Converts EBCDIC codes to 1-byte workstation codes.

BLANK

This option is valid with the option CRLF; it retains BLANK (hex 40) at the end of each line.

CRLF

Specifies the use of carriage return and line feed codes. You need ASCII and CRLF for text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

TIME(`n`)

Specifies the length of time, in units of 30 seconds, the program waits for a response from the host before it sends an error message. Replace `n` with an integer value in the range from 0 through 2184. If you specify 0, timeout is not set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, of large block sizes, or for slow communication lines (such as and COM port), *5* (150 seconds) is recommended. There should be no blank spaces between TIME and (`n`).

Command Syntax for Receiving Files from MVS/TSO

The following examples show the command syntax you can use to receive files from your MVS/TSO host to your workstation:

- To receive a data set from an MVS/TSO host to the default drive for your workstation session:

  ```
  RECEIVE pc.txt g:ds.script ASCII CRLF
  ```

  This command sends a data set named DS.SCRIPT from your MVS/TSO volume in a host session named G to your OS/2 session. It creates or replaces the file on the default drive with the name PC.TXT.
- To receive a data set from an MVS/TSO host to a drive other than your default drive:

  ```
  RECEIVE A:pc.txt g:ds.script ASCII CRLF
  ```

This command sends a data set named DS.SCRIPT from your MVS/TSO volume in a host session named G. It replaces a file named PC.TXT on a diskette in drive A.

- To receive a data set from an MVS/TSO host and add it to a workstation file:

  ```
  RECEIVE a:pc.txt g:ds.script ASCII CRLF APPEND
  ```

  This command sends a data set named DS.SCRIPT from your MVS/TSO volume in a host session named G. It adds the data set to the end of a file named PC.TXT on the diskette in drive A.
- To receive a data set from an MVS/TSO host and place it in a subdirectory on your hard disk:

  ```
  RECEIVE c:\sd1\pc.txt ds.script ASCII CRLF
  ```

  This command sends a data set named DS.SCRIPT from your MVS/TSO volume in a host session named G. It creates or replaces a file named PC.TXT in a subdirectory named \SD1 on your hard disk.
- To receive a data set that has a password from an MVS/TSO host to your default drive:

  ```
  RECEIVE A:pc.txt g:ds.script/odyssey8 ASCII CRLF APPEND
  ```

  This command sends a data set named DS.SCRIPT from your MVS/TSO volume in a host session named G. The data set has the password odyssey8. The data set is added to the end of a file named PC.TXT on the diskette in drive A.
- To receive a member of a partitioned data set from an MVS/TSO host to your DOS session:

  ```
  RECEIVE c:\sd1\pc.txt g:ds.script (m1) ASCII CRLF
  ```

  This command sends a member named M1 from a partitioned data set named DS.SCRIPT in a host session named G. The member is placed on your hard disk in a subdirectory named \SD1. It replaces or creates a file named PC.TXT.
- To receive a member of a partitioned data set that has a password to your Windows session:

  ```
  RECEIVE a:pc.txt g:ds.script (m2)/ili1 ASCII CRLF APPEND
  ```

  This command sends a member named M2 from a partitioned data set named DS.SCRIPT in a host session named G. The data set has a password of ili1. The member is added to a file named PC.TXT on the diskette in drive A.

Using the CICS SEND Command

Please note the differences between the Z and I Emulator for Windows GUI and Command Line syntaxes. These two syntaxes are not interchangeable.

Using CICS SEND with the Z and I Emulator for Windows GUI

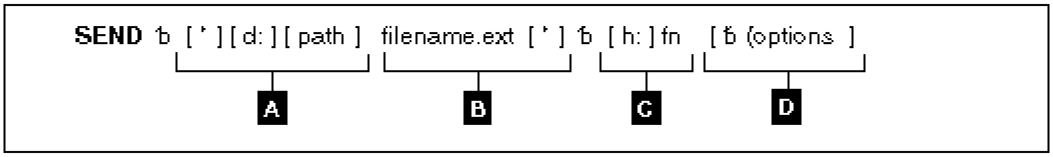
Use the following information when sending a file to CICS using the Z and I Emulator for Windows graphical user interface (GUI):

Figure 5: CICS SEND Z and I Emulator for Windows GUI Syntax on page 235 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase letters.

Note:

1. ƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 5. CICS SEND Z and I Emulator for Windows GUI Syntax



```
SEND ƀ [ ' ] [ d: ] [ path ]  filename.ext  [ ' ] ƀ  [ h: ] fn   [ ƀ (options  ]
```

A

The workstation drive and path of the file to send.

B

The name of the workstation file to send.

C

The short name of the host session, and the host file name of the file to send.

D

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- ASCII
- BINARY (for SBCS sessions)
- CLEAR
- CRLF
- NOCLEAR
- NOCRLF (for SBCS sessions)
- PROGRESS
- QUIET
- TIME(`n`)

Note: For SBCS sessions, the default options are ASCII and CRLF.

Using CICS SEND with the Z and I Emulator for Windows Command Line

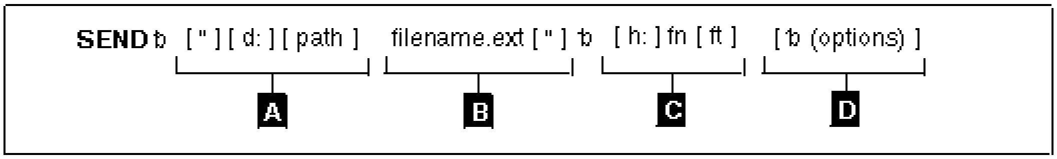
Use the following information when sending a file to CICS using the Z and I Emulator for Windows command line:

Figure 6: CICS SEND Command Line Syntax on page 236 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase letters.

Note:

1. ƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 6. CICS SEND Command Line Syntax


SEND ƀ ["] [d:] [path] filename.ext ["] ƀ [h:] fn [ft] [ƀ (options)]
A
B
C
D

A

The workstation drive and path of the file to send.

B

The name of the workstation file to send.

C

The short name of the host session (h:), the host file name (fn), and the file type (ft).

D

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- ASCII
- BINARY (for SBCS sessions)
- CLEAR
- CRLF
- NOCLEAR
- NOCRLF (for SBCS sessions)
- PROGRESS
- QUIET
- TIME(n)

Note: For SBCS sessions, the default options are ASCII and CRLF.

CICS SEND Command Description and Options

The parts of the CICS SEND command are:

SEND

The command.

`d:`

The name of the diskette or hard disk drive where the file is located.

`path`

The path to the subdirectory that the file is in.

`filename.ext`

The name of the file to be sent, including the extension.

`h:`

The short name of the host session where you want to send the file. If you have only one host, this part is optional. The default is session **A**.

`fn`

The name the file is to have on your CICS disk. You must specify the file name. You can create a new name or use a name that is already on the disk.

ft

The type of file in CICS. For use only with command line syntax, see Using CICS SEND with the Z and I Emulator for Windows Command Line on page 235.

(`options`

These options can be specified:

ASCII

Performs the following:

- Converts 1-byte workstation codes to EBCDIC codes.

The default is ASCII CRLF. You need these control terms for text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files.

Note:

1. CRLF and NOCRLF are mutually exclusive options.
2. BINARY and ASCII are mutually exclusive options.
3. The assumed defaults, if the optional parameters are omitted, are CRLF ASCII.

BINARY

Specifies that the data in the file is binary data. The data can be encrypted, compiled programs, or other data. It is not translated by the host file transfer program but copied unaltered into a temporary storage queue.

This option is valid for SBCS sessions only.

CLEAR

Clears the workstation window at the beginning of the file transfer.

CRLF

Specifies carriage return and line feed codes in the text file.

NOCLEAR

Suppresses the sending of a Clear command at the beginning of file transfer. This option is required for ISPF command mode.

NOCRLF

Specifies that the PC file does not consist of logical records delimited by carriage return and line feed characters. No concatenation or splitting of records is performed by the CICS file transfer program.

The file is written into a temporary storage using one item on the queue to represent each inbound data buffer. The items on the CICS temporary storage queue can be of different lengths, but none can be more than 32767 characters.

This option is valid for SBCS sessions only.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

TIME(`n`)

Specifies the length of time, in units of 30 seconds, the program waits for a response from the host before it sends an error message. Replace `n` with an integer value in the range from 0 through 2184. If you specify 0, timeout is not set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, large block sizes, or for slow communication lines (such as and COM port), 5 (150 seconds) is recommended. There should be no blank spaces between TIME and (`n`).

Command Syntax for Sending Files to CICS

The following examples show the command syntax you can use to send files from your workstation to your CICS host.

- To send a workstation file from your default drive and add it as a new file on your CICS host:

  ```
  SEND pc.txt a:cicsfile (ASCII CRLF)
  ```

  Note: Enter the complete CICS SEND command on one line.

  This command sends a workstation file named PC.TXT from your default drive on your workstation to your host session A. You do not need to provide the workstation drive name if the file you are sending is on the current drive. The command creates a new file named CICSFILE.
- To send a basic workstation file from a drive other than your default to replace a file on your CICS host:

  ```
  SEND a:myprog.exe a:basprog
  ```

  This command sends a workstation file named MYPROG.EXE from a diskette in drive A to your CICS host in your host session named A. It is written to a file named BASPROG, replacing any existing file by that name in host session A.

Using the CICS RECEIVE Command

Please note the differences between the Z and I Emulator for Windows GUI and Command Line syntaxes. These two syntaxes are not interchangeable.

Using CICS RECEIVE with the Z and I Emulator for Windows GUI

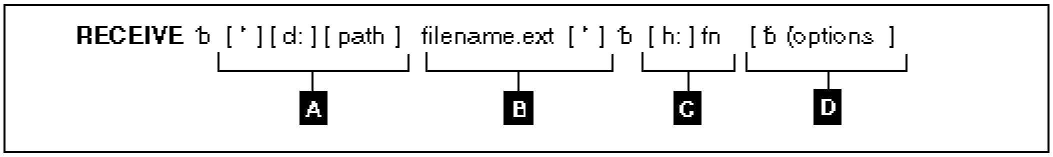
Use the following information when receiving files from CICS using the Z and I Emulator for Windows GUI:

Figure 7: CICS RECEIVE Z and I Emulator for Windows GUI Syntax on page 240 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase.

Note:

1. ƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 7. CICS RECEIVE Z and I Emulator for Windows GUI Syntax


RECEIVE ƀ ['] [d:] [path] filename.ext ['] ƀ [h:] fn [ƀ (options]
A
B
C
D

A

The workstation drive and path where the file is to be received.

B

The name of the workstation file.

C

The short name of the host session (h:) from which you are receiving the file, and the host file name (fn).

D

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- ASCII
- BINARY (for SBCS sessions)
- BLANK
- CLEAR
- CRLF
- NOCLEAR
- NOCRLF (for SBCS sessions)
- PROGRESS
- QUIET
- TIME(n)

Note: The default options for SBCS sessions are ASCII and CRLF.

Using CICS RECEIVE with the Z and I Emulator for Windows Command Line

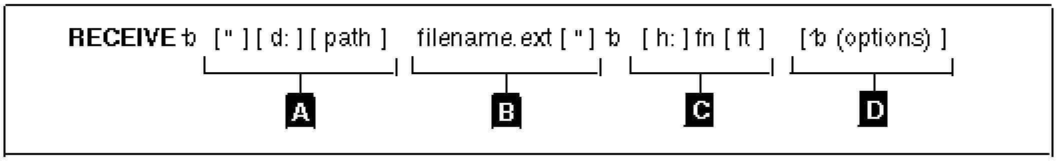
Use the following information when receiving files from CICS using the Z and I Emulator for Windows command line:

Figure 8: CICS RECEIVE Command Syntax on page 241 shows the command and information you must provide. Enter it as shown (including parentheses), except that you can use either uppercase or lowercase.

Note:

1. ƀ means to insert a space. There must *not* be a space between **h:** and **fn**.

Figure 8. CICS RECEIVE Command Syntax


RECEIVE ƀ ["] [d:] [path] filename.ext ["] ƀ [h:] fn [ft] [ƀ (options)]
A
B
C
D

A

The workstation drive and path where the file is to be received.

B

The name of the workstation file.

C

The short name of the host session (h:) from which you are receiving the file, the host file name (fn), and the file type (ft).

D

Optional changes made to the file during transfer. More than one option can be specified. Valid options are:

- ASCII
- BINARY (for SBCS sessions)
- BLANK
- CLEAR
- CRLF
- NOCLEAR
- NOCRLF (for SBCS sessions)
- PROGRESS
- QUIET
- TIME(n)

Note: The default options for SBCS sessions are ASCII and CRLF.

CICS RECEIVE Description and Options

The parts of the CICS RECEIVE command are:

RECEIVE

The command.

`d:`

The name of the diskette or hard disk drive where the file is to be received.

`path`

The path to the subdirectory where the file is to be located.

`filename.ext`

The name of the workstation file, including the extension. You can create a new name or use a name that is already on your workstation diskette or hard disk. If you use a new name, the file that you receive is added to your diskette or hard disk. If you use the name of an existing file, the file that you receive either replaces or supplements the existing file. Refer to the APPEND option.

`h:`

The short name of the CICS session where the data set is located. If you have only one host, this part is optional. The default session is A.

`fn`

The name of the file you want to receive from your CICS host.

ft

The type of file in CICS. For use only with command line syntax, see Using CICS RECEIVE with the Z and I Emulator for Windows Command Line on page 240.

(`options`

These options can be specified:

ASCII

Performs the following:

- Converts EBCDIC codes to 1-byte workstation codes.

The default is ASCII CRLF. You need ASCII and CRLF control terms for text or source files that you want to view or edit, such as SCRIPT files. You do not need them for binary files.

Note:

1. CRLF and NOCRLF are mutually exclusive options.
2. BINARY and ASCII are mutually exclusive options.
3. The assumed defaults, if the optional parameters are omitted, are CRLF ASCII.

BINARY

The data in the file is binary data. The data can be encrypted data, compiled programs, or other data. It is not translated by the host file transfer program but is copied without changes into the workstation file.

This option is valid for SBCS sessions only.

BLANK

This option is valid only when used with the CRLF option; it retains BLANK (hex 40) at the end of each line.

CLEAR

Clears the workstation window at the beginning of the file transfer.

CRLF

Specifies the use of the carriage return and line feed codes.

NOCLEAR

Suppresses the sending of a Clear command at the beginning of file transfer. This option is required for ISPF command mode.

NOCRLF

Specifies that the host computer file does not consist of logical records. The items in the temporary storage queue are sent in order and concatenated in your workstation into a single string of data.

This option is valid for SBCS sessions only.

PROGRESS

Shows a message indicating that the file transfer is in progress or has ended. Such messages do not show the current transferred bytes.

QUIET

Does not show any messages.

TIME(`n`)

Specifies the length of time, in units of 30 seconds, the program waits for a response from the host before it sends an error message. Replace `n` with an integer value in the range from 0 through 2184. If you specify 0, timeout is not set. The default is 1. To avoid a premature error message, specify an adequate value. In cases of large packet sizes, of large block sizes, or for slow communication lines (such as and COM port), 5 (150 seconds) is recommended. There should be no blank spaces between TIME and (n).

Command Syntax for Receiving Files from CICS

The following examples show the command syntax you can use to receive files from your CICS host to your workstation.

- To receive a file from your CICS host to your default drive for a workstation session:

  ```
  RECEIVE pc.txt A:cicsfile (ASCII CRLF)
  ```

  This command sends a file named CICSFILE from your CICS host in session A to your workstation session. It adds the file to your default drive (diskette or hard disk) with the name PC.TXT.
- To receive a basic file from your CICS host and replace a file on a drive other than your default:

  ```
  RECEIVE a:myprog.exe a:myprog
  ```

  This command sends a file named MYPROG from your CICS host in session A to a drive other than the default for your workstation session. It replaces a file named MYPROG.EXE on a diskette in drive A.

Configuring File-Transfer Code Translation

When you transfer a file between the host and the workstation using the ASCII option, the host-system file-transfer program performs translation from EBCDIC to ASCII or vice versa, according to the host and PC code pages specified during PC/3270 configuration. However, you might want to use different translation from that supplied. For the details, refer to Transferring Files on page 205.

Using Z and I Emulator for Windows 5250

Considerations for Using PC400 Sessions

This chapter contains hints and tips for using PC400 sessions. Supplementary information other than the items described in this book is included in the Readme HTML file in the Z and I Emulator for Windows directory.

Scroll Bar

When you click **Font** from the **Appearance** menu in the host session window and select **Fixed Size** from the Select Display Font window, the entire operator information area might not appear on the screen; the session-window size is restricted to be smaller than the screen size. If you specify **With Scroll Bar**, the OIA will not scroll.

Print Processing

Following are some additional considerations when printing with PC400.

Printing Bar Codes

This function requires OS/400® Version 4.2 or i5/OS™.

CPI/LPI of Device Fonts

If the printer driver cannot print with device fonts associated with the user-specified CPI/LPI, the print output can be generated with incorrect CPI/LPI values.

PCSERR999 Error Messages

Message `PCSERR999 - Z and I Emulator for Windows internal error:`*`module-name - xxxx`* might appear if there is insufficient memory. If any print jobs are queued in the print manager, delete those print jobs.

Disconnect in Testrequest to iSeries, eServer i5, or System i5 on Telnet 5250

Executing a Testrequest function when connected to an iSeries™, eServer™ i5, or System i5™ might cause the session to be disconnected. If you experience this problem, make sure that OS/400® APAR MA15053 has been applied on the iSeries™, eServer™ i5, or System i5™.

iSeries, eServer i5, or System i5 Host Print Problem

If you attempt to use the host print function (mapped to CTRL-Pause by default) while viewing a spooled print file, the ends of some of the lines might be wrapped incorrectly in the second generated spool file. This problem occurs with both 24X80 and 27X132 display modes. This problem has been fixed by a PTF on OS/400®. The APAR number is SA57195 and is available on PTF MF13596 for OS/400® V3R1.

Printable Area

Depending on the printer driver used, it might not be possible to use the entire surface of the paper for printing.

If the printing position is beyond the printable area, the page is automatically changed. When using a printer driver that allows you to set the margins, specify the minimum margins, thus maximizing the printable area.

PDT Mode

Printing using a PDT file is restricted as follows:

- Only the fonts specific to the printer being used are supported.
- Graphics are printed using the Windows® printer driver selected in **Printer Setting**, regardless of the PDT mode.
- Postscript printers are not supported. There are no PDF files for Postscript printers.

Setting the Code Page

The host code page, which is set in the Configuration panel, is used as the default. Use the Set Initial Condition (SIC) command to set the host code page.

You can change the code page by using Set GCGID Through GCID (SCG) command or Set CGCS Through Local ID (SCGL) command. The same code pages for the display session are available.

Data Transfer for PC400

This chapter explains file-description files and data conversions for the data transfer function. .

Data Transfer Function Overview

PC400 can transfer data between the host and workstation. The data transfer function can be invoked manually by clicking the **Data Transfer** icon. The Data Transfer application is automatically invoked from a 5250 session when you click **Send File to Host** from the **Actions** menu and **Receive File from Host** from the **Actions** menu. You can change this default to invoke normal file transfer functions; to do so, click **Preferences → Transfer** from the **Edit** menu, then click the **Data Transfer** radio button on the property page with the **General** tab.

Transferring data, described in this chapter, is quite different from transferring files, which is described in File Transfer for PC400 on page 340. The main differences are listed in the following table.

Table 33. Data Transfer Summary

Type of Transfer	Products required on an iSeries™, eServer™ i5, or System i5™	Access Method	Sending and receiving unit	Type of connection to an iSeries™, eServer™ i5, or System i5™
File Transfer	Z and I Emulator for Windows Tools (PCT/400 see Transferring Files on page 331)	• Transfer menu in the session window • EHLLAPI application that invokes File Transfer • DDE application that invokes File Transfer • Playing a macro that invokes File Transfer • Clicking the Send or Receive button on the tool bar	Entire file	Display session

Table 33. Data Transfer Summary

(continued)

Type of Transfer	**Products required on an iSeries™, eServer™ i5, or System i5™**	**Access Method**	**Sending and receiving unit**	**Type of connection to an iSeries™, eServer™ i5, or System i5™**
Data Transfer	PC Support/400 V2R2 or V2R3, OS/400® V3R1 or later, or i5/OS™[1]	Data Transfer icon or File Transfer selections from Actions menu	Field, record, or file in a database	• TCP/IP
[1] OS/400® and i5/OS™ provide the host transaction program for Data Transfer.				

Long Password Support

The Z and I Emulator for Windows Data Transfer utility supports 128-character case-sensitive passwords, when connecting to an iSeries™, eServer™ i5, or System i5™ host running i5/OS™ or OS/400®, V5R1 or later. This functionality is determined by the OS/400® or i5/OS™ system value QPWDLVL. Refer to the *iSeries Security Reference* (SC41-5302) for details.

Transferring Files from an iSeries, eServer i5, or System i5 System to a Workstation

When using a workstation, you can retrieve and use data from the following file types on an iSeries™, eServer™ i5, or System i5™:

- Physical database
- Logical database
- Distributed data management (DDM)

When retrieving files, you can do the following:

- Control which records (and which fields within a record) are retrieved
- Control the ordering of records and the ordering of fields within the record
- Select a subset of the records
- Group records into summary records
- Join two or more files
- Specify formats and separators of date and time fields
- Specify the decimal separator character

You can specify the following output destinations:

- Display
- Disk
- Printer

Transferring Files from a Workstation to an iSeries, eServer i5, or System i5

The PC→iSeries™ Transfer function enables the transfer of data from a workstation to an iSeries™, eServer™ i5, or System i5™ physical file. Data can be transferred to any of the following destinations:

- Existing members in an existing iSeries™, eServer™ i5, or System i5™ physical file
- New members in an existing iSeries™, eServer™ i5, or System i5™ physical file
- New members in a new iSeries™, eServer™ i5, or System i5™ physical file

Note: Data cannot be transferred from a workstation file to an iSeries™, eServer™ i5, or System i5™ logical file.

Transferring Data to Existing Members in an Existing File

Note the following considerations when transferring data from a workstation to an existing iSeries™, eServer™ i5, or System i5™ member.

- When data is transferred to an existing member, data in that member is replaced with that transferred from a workstation.
- When iSeries™, eServer™ i5, or System i5™ members already contain data, a message appears, indicating that the data in the existing members will be replaced with the data that is about to be transferred.
- Consider the effect of returning data that was previously transferred from the iSeries™, eServer™ i5, or System i5™ (such as when an iSeries™, eServer™ i5, or System i5™ master file is updated on a workstation).

  For example, you can transfer only the field subset of an iSeries™, eServer™ i5, or System i5™ file by issuing a transfer request from the iSeries™, eServer™ i5, or System i5™ to a workstation. In this case, when returning data from the workstation to the iSeries™, eServer™ i5, or System i5™, only the subset included in that iSeries™, eServer™ i5, or System i5™ file can be transferred. Other fields that had been defined in the iSeries™, eServer™ i5, or System i5™ file but not transferred, are filled with blanks if they are character fields or, if they are numeric fields, with zeros or the values specified at file creation.

  Therefore, the data must be transferred to another iSeries™, eServer™ i5, or System i5™ file and the transferred data must be embedded in the iSeries™, eServer™ i5, or System i5™ file by running the iSeries™, eServer™ i5, or System i5™ application program. Follow this procedure to control the update processing for an iSeries™, eServer™ i5, or System i5™ master file.

  To prevent users from transferring data to a certain iSeries™, eServer™ i5, or System i5™ file, check that the authority level for that file is defined correctly.

Transferring Data to New Members in an Existing File

You can transfer the data in a workstation file to new members in an existing iSeries™, eServer™ i5, or System i5™ file. The transfer function automatically creates these members in the specified file in the specified library. New members are created according to the file description in the existing file.

Be particularly careful when only the field subset of the iSeries™, eServer™ i5, or System i5™ file can be transferred from the iSeries™, eServer™ i5, or System i5™ to a workstation by the previous transfer request. When data is returned to the iSeries™, eServer™ i5, or System i5™, new members can receive only the subset defined in that iSeries™, eServer™ i5, or System i5™ file. Other character fields that are defined, but not transferred are filled with blanks. Numeric fields are filled with zeros or the valued specified at file creation. The date, time, and time-stamp fields use iSeries™, eServer™ i5, or System i5™ default values.

Transferring Data to New Members in a New File

By using a transfer request from a workstation to the iSeries™, eServer™ i5, or System i5™ system, you can transfer data to new members in a new iSeries™, eServer™ i5, or System i5™ file. This is one of the safest transfer methods, because data already stored in theiSeries™, eServer™ i5, or System i5™ file is not replaced with that transferred from the workstation.

There are two ways of transferring data to new members in a new iSeries™, eServer™ i5, or System i5™ file. The method used depends on the data to be transferred.

- For data that is broken up into fields, correct conversion is achieved by transferring it in units of fields. Specify use of the workstation file-description file at data transfer. In addition, specify *data* as the type of the eServer™ i5 or iSeries™ file.

  When an iSeries™, eServer™ i5, or System i5™ file and its members are created, the transfer function must access the description of the format of each field to be transferred in the iSeries™, eServer™ i5, or System i5™ file. You can get this description, called a field-reference file, from the iSeries™, eServer™ i5, or System i5™ file. To create an iSeries™, eServer™ i5, or System i5™ file and its members, specify the name of this iSeries™, eServer™ i5, or System i5™ field reference file, as well as the parameters for the other files and members. Note that only the fields to be transferred are defined in a new file.
- For data consisting only of text or source statement records, it is not necessary to break up the records into fields. In addition, the workstation file-description file is not required to transfer data. In other words, an iSeries™, eServer™ i5, or System i5™ physical source file is created.

Transferring Data to an iSeries, eServer i5, or System i5 Data File and Source File

You can transfer data to the following two types of iSeries™, eServer™ i5, or System i5™ physical files.

Physical data file

The members of a physical data file can contain numeric and character data of any iSeries™, eServer™ i5, or System i5™ data type. To transfer data to a physical data file, use the workstation file-description file to define how data is stored in a workstation data file. Besides this definition, the file description of the iSeries™, eServer™ i5, or System i5™ file is required to ensure correct conversion of the data.

When data is transferred to an existing iSeries™, eServer™ i5, or System i5™ file, the file description becomes part of the iSeries™, eServer™ i5, or System i5™ file. When data is transferred to a new iSeries™, eServer™ i5, or System i5™ file, the file description is included in the iSeries™, eServer™ i5, or System i5™ field-reference file.

Physical source file

Normally, a physical source file stores no data. It contains only text or source statements, as follows:

- The first part (field) of a source file always contains numbers indicating the order.
- The second part (field) of a source file always contains the date on which the file was created.
- The third part (field) of a source file contains the text of the file. This part can contain data fields of character type or zoned type only. Physical source files provide the optimum means of transferring text or source statements with a workstation.

Note the following considerations when transferring data to and from an iSeries™, eServer™ i5, or System i5™ physical source file:

- To transfer text from the iSeries™, eServer™ i5, or System i5™ to a workstation, specify the name of the source file and members in **FROM**. Specify an asterisk (*) in **SELECT**. This informs the iSeries™, eServer™ i5, or System i5™ that only text is transferred from the source file, with the order number and date fields excluded.
- The iSeries™, eServer™ i5, or System i5™ text must be stored in the workstation code text file. Normally, a workstation text editing program can be used to manipulate this workstation code text file.
- Specify that the file-description file is not to be stored for that workstation file. Because text is assumed to be a record consisting only of character data, it is not necessary to define fields.
- To return text from a workstation file to an iSeries™, eServer™ i5, or System i5™ file, specify the type of the workstation file containing the text. This is almost always workstation code text. Specification of the file-description file is not required.
- To create a new iSeries™, eServer™ i5, or System i5™ file and its members, specify a valid record length. This record length must be equal to the maximum record length of the workstation file, plus 12 bytes. This is because the transfer function automatically creates the order number and date fields when the file is transferred to the iSeries™, eServer™ i5, or System i5™ members. The order number and date fields together occupy 12 bytes.

Preparing for Data Transfer

The following topics describe the software products required to transfer data and the points you must understand before transferring data with PC400.

Required Software Products

To use Data Transfer, IBM® PC Support/400 (5738-PC1) must be installed on the iSeries™, eServer™ i5, or System i5™. IBM® PC Support/400 is not required with OS/400® Version 3 or later, or with i5/OS™.

Before using the data transfer function, run the router of PC400 or PC Support/400.

Transfer Function

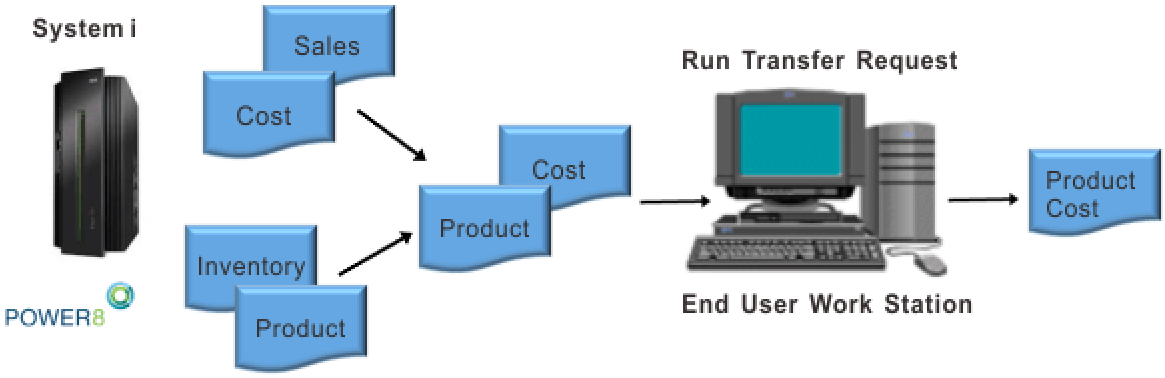
You can transfer only source programs, records, and the following information:

- Information organized for analysis
- Information used for decision making
- Information suited for computer processing

When using a spreadsheet, for example, you might want to use inventory data to create a cost analysis report. If there is no way to copy the data into the workstation, you must print the data from the iSeries™, eServer™ i5, or System i5™ and manually type it into a workstation file. With the transfer function, however, you can access the inventory database directly, select only the data needed for the report, process the data as required, then complete the report using that data.

You can also send data from the workstation to the host system for processing by iSeries™, eServer™ i5, or System i5™application. When a remote user is authorized to access the iSeries™, eServer™ i5, or System i5™ directly, he or she can access the created cost analysis report to compare with their results.

Figure 9: Data Transfer Example on page 251 outlines the joining of two files, transferring the information to the workstation, and creating a report.
Figure 9. Data Transfer Example



To transfer data by using PC400, you must create a transfer request. A transfer request provides the necessary information about the data you want to transfer.

Before creating a transfer request, you must have the answers to the following questions:

Where is the data located?
How much of the data do you want to transfer?
How should the data be sorted?
Where do you want data to be transferred?

When transferring data from the iSeries™, eServer™ i5, or System i5™ to a workstation, PC400 allows you to specify which data is to be transferred and whether the data is to be displayed or written to a workstation file.

In addition, a transfer request can be saved to a workstation file, allowing you to easily perform the same transfer at a later date. After a transfer request is saved, you can call the request to make changes or to run it again.

Data Transfer Program

PC400 data transfer is classified into two types, depending on the direction of the transfer:

- Transferring data from the workstation to the iSeries™, eServer™ i5, or System i5™ is called *data sending*.
- Transferring data from the iSeries™, eServer™ i5, or System i5™ to the workstation is called *data receiving*.

Data transfer can also be classified according to how the program is started, as follows:

- Data is transferred by interactively entering information such as *what data is transferred from which file to which file* on the screen. In this case, the interactive screen for sending is called the PC→iSeries™ Transfer window, and that for receiving is called the iSeries™→PC Transfer window.
- Data is transferred according to the information that has already been registered. The interactive screen is not necessary. This is called the *automatic transfer* of data.

In both cases, data transfer is performed by PCSFT5.EXE on the workstation and by the *PC Support/400* transfer program on the iSeries™, eServer™ i5, or System i5™.

The **Data Transfer** icon is registered in the PC400 folder by installing PC400. Double-clicking on this icon displays the iSeries™→PC Transfer window (for receiving). This icon includes:

```
   \Z and I Emulator for Windows\PCSFT5.EXE
```

The PC→iSeries™ Transfer window (for sending) opens when the registered contents are changed as follows:

```
   \Z and I Emulator for Windows\PCSFT5.EXE
```

The iSeries™→PC Transfer and PC→iSeries™ Transfer windows have a **Switch to SEND** button and **Switch to RECEIVE** button, respectively. By clicking either of these buttons, the window for sending can be switched to the window for receiving, and vice versa.

To perform *automatic transfer*, you must create transfer information, using the interactive screen window, and then save the information. You can then perform data transfer automatically by specifying the file name in which the data was saved.

For example, if you save transfer information to file TENSOU.TTO, contained in directory C:\Z and I Emulator for Windows\PRIVATE, run *automatic transfer* as follows:

```
“C:\Z and I Emulator for Windows\PCSFT5.EXE” “C:\Z and I Emulator for Windows\PRIVATE\TENSOU.TTO”
```

When you save the transfer information, register it as an icon in the PC400 folder. You can then transfer data automatically simply by double-clicking on this icon.

Data Concepts of the iSeries, eServer i5, or System i5 and Your Workstation

The basic components of data management are files, records, and fields. A *file* is an aggregate of records, referenced by a single name. Each record in a file contains one or more items of correlated information. Each item of information is called a *field*.

The iSeries™, eServer™ i5, or System i5™ and your workstation use different functions to store and group data, and to set the format.

Workstation Files

To transfer data from a workstation to the iSeries™, eServer™ i5, or System i5™, the transfer function uses a special-format workstation file, called a *file-description file*. Using this file, data is stored in a valid format and converted into a valid type.

A file-description file identifies the format of a workstation data file and contains a description of the fields in the data file. The file-description file also contains a name list of all the fields in the data file. This list reflects the order, as well as the names, in which each field appears within the data file. In addition, this list includes a description of the data type, length, and decimal position of each field. Using this information, the transfer function can recognize not only how data has been modified but also where a certain field exists in a file record.

When data is transferred from the iSeries™, eServer™ i5, or System i5™ to a workstation, you can use the transfer function to automatically create the file-description file. In this case, the information in the file-description file depends on the file description in the iSeries™, eServer™ i5, or System i5™ file.

You must create a file-description file with the same name as the workstation data file to transfer a workstation data file to the iSeries™, eServer™ i5, or System i5™.

Distributed Data Management (DDM) Files

Distributed data management (DDM) is one of the functions supported by iSeries™, eServer™ i5, and System i5™. This function is used to access database files that are stored on remote iSeries™, eServer™ i5, and System i5™ systems. To use the transfer function to access these database files, specify a DDM file name as the name of the iSeries™, eServer™ i5, or System i5™ file to be transferred. Refer to *DDM Guide* for details of how to use DDM files.

iSeries, eServer i5, or System i5 Files

The following list provides a simple explanation of the requirements for transferring data between the iSeries™, eServer™ i5, or System i5™ and a workstation.

Library

The iSeries™, eServer™ i5, or System i5™ library contains related objects that are used to generate significant groups. For example, the objects might be all the programs and files related to credit sales management. Using the library, you can group objects and find a desired file by name. The transfer function uses the library to locate an iSeries™, eServer™ i5, or System i5™ file.

File

iSeries™, eServer™ i5, or System i5™ files that you can manipulate consist of a file description and data stored in the file. PC400 processes an iSeries™, eServer™ i5, or System i5™ file, called a database file. The database file can be either a physical file or a logical file.

A *physical file* is a database file that contains data stored in records. It includes a description of the record format in addition to the data itself.

A *logical file* is a database file, that you can use to access data stored in one or more physical files. Logical files, like physical files, contain a file description. However, logical files do not contain any actual data. Instead, you can access fields in one or more physical files by using the record format included in the logical file description. When a logical file is transferred from the iSeries™, eServer™ i5, or System i5™ to a workstation, data is obtained from one or more physical files. You need only specify a logical file as the file to be transferred. The iSeries™, eServer™ i5, or System i5™ recognizes which physical file contains the actual data to be transferred.

Note: Data cannot be transferred from a workstation to logical files.

Member

Data records in a database file are grouped into several members. At least one member must be included in one file.

When data is transferred to and from the iSeries™, eServer™ i5, or System i5™, actual data transfer is done between file members. For example, a certain workstation file can be transferred to the iSeries™, eServer™ i5, or System i5™. In this case, the file members become new members of a new or existing iSeries™, eServer™ i5, or System i5™ file, or substitute for existing members in an existing iSeries™, eServer™ i5, or System i5™ file.

Record format

A record format describes the fields contained in a file record and the order in which these fields appear in the record. Record formats are stored in the file description. Both physical and logical database files can have one or more record formats.

Creating a Workstation-to-iSeries Transfer Request

To create a request for data transfer from a workstation to the iSeries™, eServer™ i5, or System i5™, do as follows.

1. Using the router session, establish attachment to the iSeries™, eServer™ i5, or System i5™ to which data is to be transferred.
2. Click the **Data Transfer** icon.
3. When the iSeries™→PC Transfer window displays, select **Switch to SEND**. The display is switched to the PC→iSeries™ Transfer window.

   To choose additional settings, select **Advanced**.
4. Specify each item. See Items to Be Specified on page 255 for details.

Items to Be Specified

The following section explains the items that you specify in the PC→iSeries™ Transfer window.

FROM

PC file name

This item is always required. It specifies the name of the workstation file containing the data to be transferred to the iSeries™, eServer™ i5, or System i5™. Specify this item using the following format. (Items inside brackets [ ] can be omitted.)

```
[d:][path-name]file-name[.ext]
```

A list of workstation files can be displayed by selecting **Browse**. You can limit the number of names listed. To limit the listing, specify a combination consisting of part of a file name and a global file name character (* or ?) in the input area of the workstation file list. For example:

- When you click **OK** with **/A:** specified, the displayed listing contains the names of all files in the current directory of the diskette inserted into drive A.
- When you click **OK** with **A:\SUPPLY\** specified, the displayed listing contains the names of all files under the **SUPPLY** path of the diskette inserted into drive A.
- When you click **OK** after specifying **B:*.XLS**, the displayed listing contains the names of all files having extension **XLS** in the current directory of the diskette inserted into drive B.

TO

System name

This item is always required. When the router program is active, this item specifies the default system name.

Library/File (Member)

This item is always required. It specifies the name of the iSeries™, eServer™ i5, or System i5™ physical file that will receive the data to be transferred from the workstation. You can specify either an existing file name or new file name.
Specify this item using the following format. (Items inside brackets [ ] can be omitted.)

```
[library-name/]file-name[(member-name[,record-format-name])]
```

library-name

This is the name of theiSeries™, eServer™ i5, or System i5™ library containing the iSeries™, eServer™ i5, or System i5™ file to which data is to be transferred. If no library is specified, *LIBL is used. To create a new file to receive transferred data, specify the library name.

When the input field is null and **Browse** is selected, the iSeries™, eServer™ i5, or System i5™ displays a list of all libraries defined in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list. You can modify this list by changing the job description. Run a change job description (CHGJOBD) command on the iSeries™, eServer™ i5, or System i5™.

file-name

This is the name of an iSeries™, eServer™ i5, or System i5™ physical database file. When data is transferred to an existing file, the data in that file is replaced with the transferred data. To create a new file to receive transferred data, specify a new file name of 1 to 10 characters.
To list the available files, do one of the following things:

- To list all files within all libraries defined in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list, specify ***USRLIBL** followed by a slash (/), then select **Browse**. If a slash (/) is not specified after the library name, the iSeries™, eServer™ i5, or System i5™ displays a list of library names rather than the file names.
- To list the names of the files in a certain library, specify the library name followed by a slash (/), then select **Browse**. You can also specify a part of a file name followed by an asterisk (*), then select **Browse**. The iSeries™, eServer™ i5, or System i5™ lists all the files whose names begin with the specified character string.

member-name

This is the name of a member in the specified iSeries™, eServer™ i5, or System i5™ file to which data is to be transferred. If this member name is not specified, data is transferred to the first member, *FIRST, in the iSeries™, eServer™ i5, or System i5™ file.

To transfer data to an existing file, specify the member name. The data within that file member is replaced with the transferred data.

To create a new member in an existing file or in a new file, specify a new member name of 1 to 10 characters.

By selecting **Browse** with a file name specified, the names of the members in that file are listed. When a left parenthesis, part of a member name, an asterisk (*), and a right parenthesis are specified, in this order, and then **Browse** is selected, the iSeries™, eServer™ i5, or System i5™ can list all member names beginning with the specified character string.

record-format-name

This is the name of the record format in the specified iSeries™, eServer™ i5, or System i5™ file. The record format name need not be specified except when a physical file contains more than one record format. Most physical files have only one record format. Before specifying a record format name, a member name or *FIRST must be specified as the member name.

When you transfer data to an existing file without specifying a record format name, it is assumed that the file has only one record format (*ONLY). Therefore, that record format is used.

When a new file is created with no record format name, QDFTFMT is used as the record format name.

Note: A library name, file name, member name, and record format name can be specified using up to 10 characters each. Each name must begin with one of the following characters: A to Z, ¥, #, or @. For characters subsequent to the first, the numbers 0 to 9, underscores, and periods can also be used.

Advanced Options

The following advanced options are available for PC→iSeries™ Transfer.

Use of File Description File

This item specifies whether a file-description file is used to transfer data to the iSeries™, eServer™ i5, or System i5™. The file-description file is required to transfer a workstation file, containing the data to be transferred (and converted), in fields. Such a workstation file can have either several fields or numeric data fields. To transfer a workstation file containing text (character data) only, the file-description file is not required. For details on creating a file-description file, see File-Description Files on page 296.

- Do not specify this item in the following case: a workstation file having only one field (for example, PC code character) is specified in **FROM**, while the iSeries™, eServer™ i5, or System i5™ file is a physical source file having the following record format.

```
  Field        Type          Length      "  " Decimal Places

Order number   Zoned            6                2
Date           Zoned            6                0
Data           Character     1 to 4096
               or Open
```

Note: When fields contain character data or zoned data only, the data portion can be broken down into several fields. The destination iSeries™, eServer™ i5, or System i5™ file contains the fields for order number and date. The workstation file, however, does not. This method is recommended when transferring text only between the iSeries™, eServer™ i5, or System i5™ and the workstation.

- Specify this item in all other cases. Two examples are:
  - Data is transferred from a workstation file having more than one field.
  - The iSeries™, eServer™ i5, or System i5™ file that receives the data is other than a physical source file having the record format described above.

File Description File Name

This item appears only when item **Use of File Description File** is specified.

This item is always required. It specifies the name of the workstation file-description file that describes the data to be transferred.

Upon transferring data from the iSeries™, eServer™ i5, or System i5™ to a workstation, a file-description file might have been created.

A file-description file must be created when the data has not yet been transferred from the iSeries™, eServer™ i5, or System i5™ to a workstation or when no file-description file exists.

PC File Type

This item appears only when **Use of File Description File** is not specified.

This item is always required. You must specify the type of the workstation file specified in the **FROM** field. The values provided by the iSeries™, eServer™ i5, or System i5™ are recognized as workstation code text. If the file type of a data file is *not converted*, the file can include nothing other than data that does not require conversion.

iSeries Object

This item is always required. It specifies whether the iSeries™, eServer™ i5, or System i5™ member to which data is transferred is a new member or an existing member. When data is transferred to a new member, this item also specifies whether the file to contain the new member is an existing file.

Create New Member

This item specifies that a new member, to which data is transferred, is created in an existing iSeries™, eServer™ i5, or System i5™ file.

Note:

1. To create a new member, you must have the following authorities:
   - *OBJOPR, *OBJMGT, and *ADD for the file that will include the new member
   - *READ and *ADD for a library that will contain the file

See *Security Descriptions* (SC41-8083) for details of object authorities.

2. To create a member to add to a file, the transfer function uses the iSeries™, eServer™ i5, or System i5™ default value for the add physical file member (ADDPFM) command.

When you specify this item, the following item must also be specified:

Member Text

This item is used to add an explanation of a new iSeries™, eServer™ i5, or System i5™ member. This explanation helps remind you of the contents of the member. This explanation appears, for example, when a list of all members in a file is requested (**Browse** is selected). If this item is left blank, no explanation is added to the new iSeries™, eServer™ i5, or System i5™ member.

To specify an apostrophe (') in the explanation, enter two apostrophes (' ').

Create New Member in New File

This item specifies that a new member, to which data is to be transferred, is created in a new iSeries™, eServer™ i5, or System i5™ file.

Note:

1. To create a new member in a new file, *READ and *ADD authorities are required for the library that will contain that file. Authority to use the create physical file (CRTPF) command of the iSeries™, eServer™ i5, or System i5™ is also required.
2. To create a new member in a new file, the transfer function uses the default value for the create physical file (CRTPF) command of the iSeries™, eServer™ i5, or System i5™. It does not, however, use the following values:

   (MAXMBRS[*NOMAX]). This indicates that the file can contain up to 32,767 members.

   (SIZE[*NOMAX]). This indicates that each member of the file can contain an unlimited number of records.

When this item is specified, also specify the following item:

Member Text

This item is optional. It is used to add an explanation of a new iSeries™, eServer™ i5, or System i5™ member. This explanation helps remind you of the contents of the member. This explanation appears, for example, when a list of all the members in a file is requested (**Browse** is selected). If this item is left blank, no explanation is added to the new iSeries™, eServer™ i5, or System i5™ member.

To specify an apostrophe (') in the explanation, enter two apostrophes (' ').

iSeries File Type

This item is always required. It specifies the type of iSeries™, eServer™ i5, or System i5™ file and the members to be created (same type for both).
Specify one of the following things:

- To create an iSeries™, eServer™ i5, or System i5™ physical source file and its members, specify **Source**. These members are created with two fields (order number and date) added to the beginning of the data transferred from the workstation file. A new iSeries™, eServer™ i5, or System i5™ source file and its members have the following record format:

  ```
  Field         Type       Length       Decimal Places

  Order number  Zoned         6            2
  Date          Zoned         6            0
  Data          Character   1 to 32755
                or Open
  ```

  Note that in an iSeries™, eServer™ i5, or System i5™ physical source file, each record can be up to 32 755 bytes in length. But, the maximum size of a source file created using the workstation-to-iSeries transfer function is 4,107 bytes. Also, this file must include the order and date fields. Therefore, the maximum amount of data that can be transferred is 4,096 bytes per record.

  The data portions of members inherit the workstation file characteristics. In other words, when a workstation file is a workstation code text file consisting of many records containing text, the created data fields will be the same.
- To create aniSeries™, eServer™ i5, or System i5™ physical data file and its members, specify **Data**. The file and members will contain only the data fields described in the file-description file.

The value of the **iSeries™ File Type** is assumed to be **Data** when a file-description file is used to transfer data. If a file-description file is not used for data transfer, the value of this item is assumed to be **Source**.

Field Reference File Name

This item appears only when **Use of File Description File** is specified for the creation of a new file.

When **Use of File Description File** is not specified, an iSeries™, eServer™ i5, or System i5™ physical source file is created. **iSeries™ File Type** and **Field Reference File Name** are not displayed. Instead, **Record Length** opens.

This item is always required. A new iSeries™, eServer™ i5, or System i5™ file is created using the field name in a file-description file and the field definitions in an iSeries™, eServer™ i5, or System i5™ field-reference file.

The format of a field-reference file name is as follows. (Items inside brackets [ ] can be omitted.)

```
[library-name/]file-name
```

library-name

This is the name of an iSeries™, eServer™ i5, or System i5™ library containing a field-reference file. If this library name is not specified, *LIBL is assumed. If you cannot find the desired library, selecting **Browse** displays a list of all libraries in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list. *USRLIBL of the library list can be changed by modifying the job description by executing a CHGJOBD command on the eServer™ i5 or iSeries™ processor.

file-name

This is the name of the iSeries™, eServer™ i5, or System i5™ physical database file containing the field definitions. Always specify this file name. When a library name is specified concurrently, use a slash (/) to delimit the library name and file name. If the desired file cannot be found, enter the library name and a slash, then select **Browse**. The system displays a list of files in that library. To list all the files in the libraries defined in *USRLIBL of theiSeries™, eServer™ i5, or System i5™ job library list, enter *USRLIBL/ then select **Browse**.

If you enter part of a file name followed by an asterisk (*) and then select **Browse**, the system displays a list of available file names, each beginning with the specified part of the name.

For example, enter ARLIB/AR* in the **Field Reference File Name** item, then select **Browse**. The system displays a list of all physical file names beginning with AR in library ARLIB.

Note: You must have *OBJOPR authority for the field-reference file to be specified. To list certain files, you must also have *OBJOPR authority for those files.

Record Length

This item is always required. It specifies the record length of an iSeries™, eServer™ i5, or System i5™ physical source file. When the data receiver is an iSeries™, eServer™ i5, or System i5™ physical source file, the specified value must include the length of the order number and date fields that are added to a workstation file at transfer (the total length of these two fields is 12 bytes).

Authority

This item is always required. It specifies the authority level of a new iSeries™, eServer™ i5, or System i5™ file.
Specify one of the following things:

- **Read/Write**. This enables other users to read from and write to the iSeries™, eServer™ i5, or System i5™ file and allows the file name to be displayed in lists. However, users cannot delete the file (*OBJOPR, *READ, *ADD, *OBJMGT, *UPD, and *DLT authorities). If other users might be transferring data from a workstation file to the iSeries™, eServer™ i5, or System i5™ file, specify **Read/Write** or **All**.
- **Read**. This enables other users to read from the iSeries™, eServer™ i5, or System i5™ file, and allows the file name to be displayed in lists. However, other users can neither write to the file nor delete it (*USE authority).
- **All**. This enables other users to read from and write to the iSeries™, eServer™ i5, or System i5™ file as well as delete it. The file name is displayed in lists (*ALL authority).
- **None**. This prevents other users (except for the system administrator) from writing to or deleting the iSeries™, eServer™ i5, or System i5™ file. The file name does not appear in lists (*EXCLUDE authority).

File Text

This item is optional. It is used to add an explanation of a new iSeries™, eServer™ i5, or System i5™ file. This explanation helps remind the user of the contents of the file. This explanation appears, for example, when a list of all files in a library is requested (**Browse** is selected). If this item is left blank, no explanation is added to the new iSeries™, eServer™ i5, or System i5™ file.

To specify an apostrophe (') in the explanation, enter two apostrophes (' ').

Replace Existing Member

This item transfers data to an existing iSeries™, eServer™ i5, or System i5™ member, specified in the **Library/File (Member)** item. The existing data in that iSeries™, eServer™ i5, or System i5™ member is replaced with the transferred data.

Saving, Opening, Changing, and Executing a Transfer Request

The following section explains how to save, open, change, and execute, as a file, information (transfer request) on data to be transferred.

Saving a Transfer Request

Save a transfer request when the request is likely to be executed repeatedly. This eliminates the need to create a transfer request every time data is to be transferred. To save a transfer request, do as follows:

1. Specify the information needed for transfer, using the PC→iSeries™ Transfer window.
2. After specifying the necessary information, click **Save** or **Save As** from the **File** menu.

   The Save Transfer Request File As window opens.
3. Specify each item, referring to the following explanation, then click **OK**.

File Name

Disk to which data is to be saved. Specify a file name or diskette file name. The default extension is TFR. Extension TFR identifies a file as a transfer request file.

Description

This item can be used to add an additional explanation of a transfer request, as required. The explanation can be up to 40 characters in length. This explanation is saved with the transfer request, and displayed in the list of transfer request names. It is, therefore, useful for identifying a transfer request.

4. The system asks whether the saved transfer request is to be registered in the PC400 folder.

   When you click **OK**, the transfer request is registered as an icon. Subsequently selecting this icon transfers data according to the contents of the registered data transfer request.

Opening and Changing a Saved Transfer Request

To open and change a saved transfer request, do as follows:

1. Display the PC→iSeries™ Transfer window.
2. Click **Open** from the **File** menu.
3. Specify the name of the file to be opened using the Open Transfer Request File window, then click **OK**.

   The PC→iSeries™ Transfer window reopens, and the transfer request information, saved to the specified file, appears for each item. This opens the saved transfer request.
4. Change the contents of the transfer request as necessary.
5. To save the changed contents, follow the procedure explained in Saving a Transfer Request on page 262.

Performing a Transfer Request

A transfer request can be performed in any of the following ways:

- By clicking the icon with which the transfer request has been registered
- By using the PC→iSeries™ Transfer window of the Data Transfer icon

Clicking the Icon with Which the Transfer Request Has Been Registered

This method can be used only when a transfer request has been saved as an icon by using the PC→iSeries™ Transfer window.

Clicking the corresponding icon starts data transfer.

Using the PC→iSeries Transfer Window

1. Before executing a transfer request, operations such as creating, opening, and changing a transfer request must be completed.

   Note: When data is transferred from a workstation to an existing member in an iSeries™, eServer™ i5, or System i5™ file, the transferred data replaces the existing data in that member.

2. Select **Send** from the PC→iSeries™ Transfer window.

   Data transfer starts.
3. After the transfer has been completed, click **Cancel** or **Exit** from the **File** menu.

Conversion Errors That Can Occur during Transfer

Upon executing a transfer request, a file-description file (when specified) is read from the disk or diskette to be processed. The iSeries™, eServer™ i5, or System i5™ and workstation exchange information, if the data is transferable.

The workstation transfers records, one at a time, from the file specified in **FROM**. Transferred records are converted and stored in the iSeries™, eServer™ i5, or System i5™ member specified in **TO**.

During this conversion process, conversion errors might occur. For example, the values in a workstation file might have to be rounded to fit the iSeries™, eServer™ i5, or System i5™ fields. Another example is the case where the record length of a workstation file differs from that expected by the iSeries™, eServer™ i5, or System i5™.

If such an error occurs, an error message is issued with the number of the workstation file record for which the error occurred and, sometimes, information about certain fields in that record.

If a severe error occurs, data transfer might stop. In such a case, stop the transfer request, correct the error, then rerun the transfer request.

When the error is not so severe, you can request that the system continue transferring data. By doing so, even if the same error occurs in another record, an error message does not appear and the transfer function automatically continues executing the transfer request.

Creating an iSeries-to-Workstation Transfer Request

To create a transfer request to receive data from the host, do as follows:

1. Click the **Data Transfer** icon.
2. When the PC→iSeries™ Transfer window opens, select **Switch to RECEIVE** to switch the display to the iSeries™→PC Transfer window.

   For the additional settings, click the **Advanced** button.
3. Which items are to be specified by the user vary with the data type, as follows:

- Entire iSeries™, eServer™ i5, or System i5™ file
- Part of an iSeries™, eServer™ i5, or System i5™ file
- Data combined from several iSeries™, eServer™ i5, or System i5™ files
- Summary of record groups

Before specifying each item, while referring to Items to Be Specified on page 267, note the following points regarding the data to be received.

Receiving an Entire iSeries, eServer i5, or System i5 File

This is the simplest way of transferring data from the iSeries™, eServer™ i5, or System i5™ to a workstation. All records in a file and all the data in each record are transferred.

The **FROM** items are as follows:

System name

This item specifies the name of the system.

Library/File (Member)

This item specifies the name of the iSeries™, eServer™ i5, or System i5™ file.

SELECT

Specifying an asterisk (*) for this item indicates that all fields are to be transferred, or lists all the fields in the iSeries™, eServer™ i5, or System i5™ file.

ORDER BY

This item is optional. It specifies how records are grouped. When this item is left blank, records are not grouped (data is transferred in the same order it appears in the iSeries™, eServer™ i5, or System i5™ file).

Receiving Part of an iSeries, eServer i5, or System i5 File

Only part of an iSeries™, eServer™ i5, or System i5™ file is transferred to the workstation.

The **FROM** items are as follows:

System name

This item specifies the name of the system.

Library/File (Member)

This item specifies the name of the iSeries™, eServer™ i5, or System i5™ file.

SELECT

This item specifies a field to be transferred.

WHERE

This item specifies the requirements that must be satisfied before records can be selected for transfer.

ORDER BY

This item is optional. It specifies how records are grouped. When this item is left blank, records are not grouped (data is transferred in the same order it appears in the iSeries™, eServer™ i5, or System i5™ file).

Receiving Data Combined from Several iSeries, eServer i5, or System i5 Files

The data to be transferred can be stored in two or more iSeries™, eServer™ i5, or System i5™ files. These files are assumed to be related. Based on this relationship, they can be linked or *joined*, as if all the data existed in a single file. The files can be transferred to the workstation after they have been joined. By using the iSeries™→PC Transfer function, this "join and transfer" function can be performed in a single step.

The **FROM** items are as follows:

System name

This item specifies the name of the system.

Library/File (Member)

This item specifies the names of all iSeries™, eServer™ i5, or System i5™ files from which data is to be transferred.

JOIN BY

This item specifies how to join or combine the data in each file.

SELECT

This item specifies a field to be transferred.

WHERE

This item specifies the requirements that must be satisfied before records can be selected for transfer.

ORDER BY

This item is optional. It specifies how records are grouped. When this item is left blank, records are not grouped (data is transferred in the same order as it appears in the iSeries™, eServer™ i5, or System i5™ file).

Receiving a Summary of Record Groups

A summary record is a single record that includes information on each set of records grouped from one or more iSeries™, eServer™ i5, or System i5™ files.

The **FROM** items are as follows:

System name

This item specifies the name of the system.

Library/File (Member)

This item specifies the names of all files from which data is to be transferred.

JOIN BY

This item is optional. It specifies the join conditions that must be satisfied before records can be joined.

GROUP BY

This item is optional. It must be specified only when the records of iSeries™, eServer™ i5, or System i5™ files are classified into several groups. To group all records into a single group, this item need not be specified.

SELECT

Specifying this item creates a summary record. The field names specified in **GROUP BY** can be specified.

WHERE

This item is optional. It specifies the requirements that each record to be grouped must satisfy. To group all records, this item need not be specified.

HAVING

This item is optional. It specifies the summary record to be transferred. To transfer all summary records, this item need not be specified.

ORDER BY

This item is optional. It specifies how summary records will be grouped. When this item is left blank, records are not grouped (data is transferred in the same order as it appears in the iSeries™, eServer™ i5, or System i5™ file).

Items to Be Specified

The following section explains the items to be specified using the iSeries™→PC Transfer window.

FROM

System name

This item specifies the name of the host system that contains the data to be received. When the router program is active, this item specifies the default system name.

Library/File (Member)

This item is always required. It specifies the name or names of one or more files used to store data to be transferred. Up to 32 file names can be specified. To specify several files, delimit them with commas and use JOIN BY, displayed after all **FROM** items have been specified. Only the file name must be specified. Do not specify a comma as a part of a file name. When the other optional items are not specified, they are assumed automatically. For example, the library name, member name, and format name can be assumed to be *LIBL, *FIRST, and *ONLY, respectively. When the cursor is on the input field of **FROM**, selecting **Browse** lists libraries, files, members, and formats.

Note: To transfer data from an iSeries™, eServer™ i5, or System i5™ physical file, you must have *USE authority for that file. To transfer data from an iSeries™, eServer™ i5, or System i5™ logical file, you must have *OBJOPR authority for that file and *READ authority for each subordinate file.

Specify file names as follows. (Items inside brackets [ ] can be omitted.) To specify several file names, delimit the names with commas.

```
[library-name/]file-name[(member-name[,record-format-name])],
[library-name/]file-name[(member-name[,record-format-name])],...
```

library-name

This is the name of the iSeries™, eServer™ i5, or System i5™ library that contains the iSeries™, eServer™ i5, or System i5™ file to be transferred. This iSeries™, eServer™ i5, or System i5™ file contains the data to be transferred from the iSeries™, eServer™ i5, or System i5™ to a workstation. If this library name is not specified, *LIBL is assumed. If you cannot find the desired library, selecting **Browse** displays a list of all libraries defined in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list. *USRLIBL of the library list can be changed by modifying the job description by executing the CHGJOBD command on the iSeries™, eServer™ i5, or System i5™.

file-name

This is the name of the iSeries™, eServer™ i5, or System i5™ physical file, logical file, or DDM file from which data is transferred. This file name must always be specified. To specify a file name and library name concurrently, delimit them with a slash (/). If you cannot find the desired file name, enter the library name followed by a slash, then select **Browse**. The system then displays a list of files contained in that library. To display a list of all the files in the libraries defined in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list, enter *USRLIBL/, then select **Browse**.

member-name

This is the name of the iSeries™, eServer™ i5, or System i5™ member containing the data to be transferred, or *FIRST. If this member is not specified, the system assumes *FIRST, and the first member of that file is used.

record-format-name

This is the name of the record format contained in the specified iSeries™, eServer™ i5, or System i5™ file, or *ONLY. Before specifying the record format name, specify the member name or *FIRST. If the record format name is not specified, the system assumes *ONLY, and the only record format for that file is used. To specify a record format name, delimit the record format name and member name with a comma.

When the specified iSeries™, eServer™ i5, or System i5™ file has several record formats, a record format name must be specified. If the file member name is not specified, a record format name cannot be specified.

Note:

1. A library name, file name, file member name, and record format name can be specified using up to 10 characters for each. Each name must begin with A to Z, ¥, #, or @. For characters subsequent to the first, 0 to 9, underscores, and periods can also be used.
2. When the **FROM** field remains blank or a comma is entered to specify the next file name, selecting **Browse** displays a list of libraries defined in *USRLIBL of the iSeries™, eServer™ i5, or System i5™ job library list.
3. Enter part of the file name, member name, or record format name, followed by an asterisk (*), then select **Browse**. The system displays a list of names beginning with the specified characters.

For example, you might want to transfer data from file member ITEMMBR1 (first member) of file ITEMMAST in library ITEMLIB. ITEMFMT is the only record format of this file. The specification will be as follows:

```
ITEMLIB/ITEMMAST(ITEMMBR1,ITEMFMT)
```

Alternatively, specify:

```
ITEMLIB/ITEMMAST
```

Receiving a Summary of Record Groups

The following information is necessary to receive summary records.

To transfer a summary record, do not leave this input area blank or specify an asterisk (*) (except when all the fields of the file specified at the prompt are specified in **GROUP BY**). The field names specified in **SELECT** (except for those specified in functions) must also have been specified in **GROUP BY**.

The functions and fields specified in **SELECT** return actual summary information for each group. Enter the field names and functions in **SELECT** in the order in which they are to be displayed.

Note: Null values are not included in the functions. When an entire value is null, the function output is set to null, except for **COUNT**. The **COUNT** output is 0.

The function format is as follows.

```
function (field-name)
```

This has the following meaning:

function

This is one of the following functions:

AVG

Transfers the average value of the specified fields for each record group. This function can be used only for numeric fields.

MIN

Transfers the minimum or lowest value of the specified fields for each record group.

MAX

Transfers the maximum or highest value of the specified fields for each record group.

SUM

Transfers the total value of the specified fields for each record group. This function can be used only for numeric fields.

COUNT

Transfers the total number of records that satisfy the **WHERE** condition for each record group. Specify **COUNT(*)**.

field-name

This is the field name defined with the record format specified in **FROM**.

Each function returns one value for each record group. In **SELECT**, several functions can be specified. To do so, delimit the functions by commas, as follows:

```
SUPPNO, AVG(PRICE), MIN(PRICE), MAX(PRICE)
```

This indicates that the average, minimum, and maximum values for PRICE are calculated for each supplier after SUPPNO has been selected. A summary record is transferred according to the function selection. Specify SUPPNO in **GROUP BY**, because SUPPNO has not been used for the functions.

Advanced Options

The following advanced options are available for iSeries™→PC Transfer.

JOIN BY

When several files have been specified in **FROM**, specify **JOIN BY**. When only one file has been specified in **FROM**, **JOIN BY** does not appear.

JOIN BY specifies how to link or join the records of the files specified in **FROM**. Each file specified in **FROM** must be joined with at least one other file that has been specified in **FROM**.

Use **JOIN BY** to specify one or more join conditions. The join conditions indicate the similarity of two files. Therefore, they indicate which records of one file are joined with those of another.

The join conditions are as follows:

```
field-name = field-name
```

Field name is the name of the field defined in the record format specified in **FROM**. The join conditions require two field names, one for each file to be joined.

Field names must be delimited by one of these:

=

Equal

<> or ><

Not equal

>

Greater than

>=

Greater than or equal to

<

Less than

<=

Less than or equal to

When specifying fields in **JOIN BY**, observe the following rules.

- Join a numeric field to another numeric field. The field lengths and types do not have to be identical.
- Join a character field to another character field. The lengths do not have to be identical.

The field name to be specified might have been defined in the files specified in **FROM**. When such a field name is used in the following items, prefix the field name with the file qualifier:

- JOIN BY
- GROUP BY
- SELECT
- WHERE
- HAVING
- ORDER BY

The file qualifier is the character T (uppercase or lowercase) followed by a one- or two-digit number. Use T1 for fields defined with the first record format, T2 for fields defined with the second record format, and so on. Delimit the file qualifier and field name with a period (`.`). See Receiving Records Using File Qualifiers on page 288 for details of the file qualifiers.

If the field name of the file specified in **FROM** cannot be found, select **Browse** when the cursor is on the **JOIN BY** input area. Then, a list of file qualifiers and field names of the files appears.

To join three or more files, or to join two files based on two or more common fields, two or more link conditions must be used. To specify several join conditions, join the conditions with AND. For example:

```
T1.EMPNO = T2.EMPNO AND T2.EMPNO = T3.EMPNO
```

In this case, records having the same value as EMPNO are joined between the first and second files specified in **FROM**. Then, such records are joined between the second and third files specified in **FROM**.

Up to 32 join conditions can be specified.

After **JOIN BY** is specified, each of **SELECT**, **WHERE**, and **ORDER BY** can be completed, by following the procedure described earlier in this chapter. To browse a field name that has been defined in several files, prefix the field name with a file qualifier.

GROUP BY

This item is required only to classify iSeries™, eServer™ i5, or System i5™ file records into several groups. When no value is specified in **GROUP BY**, all the records are treated as a single group.

If **GROUP BY** is not displayed, select **Group functions** at the bottom right of the screen. Then, **GROUP BY** appears. **GROUP BY** and **HAVING** are displayed concurrently. You can specify either, both, or neither.

When **GROUP BY** and **HAVING** are displayed but you do not want to specify either, select **Remove Group functions**. The two items disappear.

To classify several records into groups, specify one or more fields to act as the base for grouping. Records are grouped according to the field specified first, then by the field specified second, and so on. For example, suppose that the following groupings are specified:

```
SHIFT, DEPTNO
```

In this example, the records are first grouped by SHIFT. Records belonging to a single group will subsequently have the same value as SHIFT. Then, the records in each group are grouped by DEPTNO. When there is only one record having a certain SHIFT value, the group has only one record.

Delimit field names with commas. Blanks can be specified to improve readability. Up to 50 field names can be specified. These fields must have been defined in the record format defined in **FROM**.

If a field cannot be found, selecting **Browse** displays a list of all the fields contained in the record.

With **GROUP BY** specified, specify **SELECT** to transfer the summary record of each group.

SELECT

This item is always required. It specifies the field to be transferred or the function that indicates the type of summary information to be transferred.

The field to be specified must have been defined in the record format specified in **FROM**.

To transfer all the fields in the specified record, specify an asterisk (*) in this input field. (Specifying an asterisk causes all fields in the record to be transferred.)

Note: Up to 256 fields can be transferred. When more than 256 fields have been defined in a file, an asterisk cannot be used. In this case, specify the names by selecting the fields to be transferred.

To transfer fields by selecting from a record, enter the field names in the order in which the fields are arranged. One or more blanks can be placed between the field names to improve readability. However, the names must be delimited by commas, as follows:

```
ITEMNO,  QONHAND,  PRIC
```

You can also specify:

```
ITEMNO,QONHAND,PRICE
```

When records are transferred from an iSeries™, eServer™ i5, or System i5™ source file, specifying an asterisk (*) causes all fields in the file to be transferred, with the exception of the order number field and date field. (To transfer all the fields, including the order number field and date field, specify all the field names, including each data field name.)

A field can be specified repeatedly as required. However, bear in mind that no more than 256 fields can be selected. A list of field names can be displayed by selecting **Browse**.

WHERE

This item is optional. It specifies one or more conditions that records to be transferred must satisfy.

To transfer summary records, use this item to specify which records are to be grouped, then group the records. Using this item, you can specify one or more conditions that the record must satisfy to belong to a certain group. When **WHERE** is not specified, all records are grouped.

As the *conditions*, specify the test to be applied to the records in the specified file member. All the records in the specified file member are tested for the conditions specified here. Only those records that pass this test are transferred.

When **WHERE** is not specified, all records in the specified file member are transferred.

The condition format is as follows:

```
field-name  test  value
```

field-name

This must be a field substring or field name defined in the record format.

Fields or constants can be manipulated by specifying a supported function, with the results being used for comparison. The supported functions and usage are as follows:

SUBSTR

Returns the specified part of a character string. This function contains three parameters: the field name, starting position, and length of the returned substring. The following example returns the 20 characters starting from the 10th character of the FULLNAME field:

SUBSTR(FULLNAME 10 20)

VALUE

Returns the first non-null value in the parameter list. (If all parameters are null, null is returned.)

VALUE(DEPOSIT WITHDRAW BALANCE)

CURRENT

Returns DATE, TIME, TIMEZONE, or TIMESTAMP for the current system.

CURRENT(TIMEZONE)

DIGITS

Returns a character string representation of a numeric field.

DIGITS(EMPLOYEE#)

CHAR

Returns a character string representation of the date field, time field, or time-stamp field. The second parameter is used to specify the format of the Systems Application Architecture® (SAA®) of the string to be returned (supported values are USA, EUR, ISO, or JIS).

CHAR(DATEHIRE USA)

DATE

Returns the date of the time-stamp field.

DATE(TIMECRTD)

TIME

Returns the time of the time-stamp field.

TIME(TIMECRTD)

TIMESTAMP

Returns the time-stamp, combining the date field and time field.

TIMESTAMP(DATESEND TIMESEND)

YEAR

Returns the year of the date field or time-stamp field.

YEAR(DATEHIRE)

MONTH

Returns the month of the date field or time-stamp field.

MONTH(DATEHIRE)

DAY

Returns the date of the date field or time-stamp field.

DAY(DATEHIRE)

DAYS

Returns the day of the year, counted from January 1, of the date field or time-stamp field.

DAYS(DATEHIRE)

HOUR

Returns the time of the time field or time-stamp field.

HOUR(TIMESEND)

MINUTE

Returns the minute of the time field or time-stamp field.

MINUTE(TIMESEND)

SECOND

Returns the second of the time field or time-stamp field.

SECOND(TIMESEND)

MICROSECOND

Returns the microsecond of the time field or time-stamp field.

MICROSECOND(TIMECRTD)

test

This is the comparison type to be applied to fields or functions.
The following tests can be used. One or more blanks can be placed before and after these tests.

Note: Values are searched according to the exact characters specified by the user. In other words, when the user's specification consists only of uppercase characters, only uppercase character strings are returned. Similarly, when the specification consists only lowercase characters, only lowercase character strings are returned.

=

Equal

<> or ><

Not equal

>

Greater than

>=

Greater than or equal to

<

Less than

<=

Less than or equal to

LIKE

The field is similar to the specified value.

BETWEEN

The field is equal to one of two constants, or to a value between them.

IN

The field is the same as one of the values in the constant list.

IS

The field contains null values.

ISNOT

The field contains no null values.

Test usage is as follows:

Using the LIKE Test

The **LIKE** test checks the field specified with the field name for a character pattern specified as a value. The field to be specified must be a character field.

The values to be tested must be character-string constants. This string can contain any characters. A percent (%) character indicates a character string consisting of zero or more characters. A 1-byte underscore (_) character indicates any single 1-byte character. A 2-byte underscore (_) character indicates any single 2-byte character.

The following example explains how to use the **LIKE** test:

```
NAME LIKE '%ANNE%'
```

The previous example searches for names containing character string ANNE, such as ANNE, ANNETTE, and SUZANNE.

The following example searches for names beginning with character string ANNE, such as ANNE and ANNETTE.

```
NAME LIKE 'ANNE%'
```

The following example searches for names ending with character string ANNE, such as ANNE and SUZANNE.

```
NAME LIKE '%ANNE'
```

The following example searches for all names whose second character is A.

```
NAME LIKE '_A%'
```

The following example searches for all last names beginning with character J.

```
LSTNAM LIKE 'J%'
```

This has the same effect as the following example:

```
SUBSTR (LSTNAM,1,1) = 'J'
```

When the pattern does not include a percent character (%), the length of the character string must be identical to that of the field.

Using the BETWEEN Test

The **BETWEEN** test checks the fields specified in the field name for character strings or numeric values that are equal to or between the specified constants. The values to be tested must be two character-string constants or two numeric constants. The types of these constants must be identical to that of the field name specified by the user. Delimit the two constants with AND.

The following example searches for those records for which the price is between 50.35 and 75.3, inclusive:

```
PRICE BETWEEN 50.35 AND 75.3
```

The following example searches for those records for which the name begins with C:

```
NAME BETWEEN 'C' AND 'CZZZZZZZZZ'
```

The following example searches for those records for which the balance is between 0 and 5␠000.

```
BALDUE BETWEEN 0 AND 5000
```

This has the same meaning as the following expression.

```
BALDUE >= 0 AND BALDUE <= 5000
```

Note: Specify the values to be tested in the form of **BETWEEN** (minimum) AND (maximum). For instance, `BETWEEN 1 AND 10` is a valid specification. However, `BETWEEN 10 AND 1` returns no records.

Using the IN Test

The **IN** test checks the fields specified in the field name for the character strings or numeric values in the list specified as the value. The value to be tested must be a list of character-string constants or numeric constants. In addition, the types of these constants must be identical to that of the specified field. Delimit the constants with blanks and enclose them in parentheses. Up to 100 constants can be specified. The following example shows how to use the **IN** test:

```
NAME IN ('SMITH' 'JONES' 'ANDERSON')
```

This example searches for those records for which the name is SMITH, JONES, or ANDERSON.

The following example searches for the values in the **STATE** field for which the value is other than NY, MN, or TX:

```
NOT STATE IN ('NY' 'MN' 'TX')
```

Note: Values are searched according to the exact characters specified by the user. In other words, when the user's specification consists of only uppercase characters, only uppercase character strings are returned. Similarly, when the specification consists of only lowercase characters, only lowercase character strings are returned.

Using the IS Test

The **IS** test checks the fields specified in the field name for null values.
The following example searches for those records for which the commission field contains null values:

```
COMMISSIONS IS NULL
```

Using the ISNOT Test

The **ISNOT** test checks the fields specified in the field name for non-null values.
The following example searches for those records for which the commission field does not contain null values:

```
COMMISSIONS ISNOT NULL
```

In the test, logical AND and logical OR can be combined. When both AND and OR are specified, AND comparison is performed first. Up to 50 conditions can be specified. For example:

```
MONTH=2 AND LOC='MIAMI' OR LOC='CHICAGO'
```

In this example, each record to be selected must satisfy the following condition:

```
MONTH=2 AND LOC='MIAMI'
```

or must satisfy the following condition:

```
LOC='CHICAGO'
```

This command can be modified by using parentheses. For example:

```
MONTH=2 AND (LOC='MIAMI' OR LOC='CHICAGO')
```

In this example, each record to be selected must satisfy the following condition:

```
MONTH=2
```

and it must satisfy the following condition:

```
LOC='MIAMI' OR LOC='CHICAGO'
```

NOT can also be used. The following example selects items where data is transferred not only from those records in which the DEPT field is not equal to 470, but also from those records for which the DEPT field is equal to 470 and, additionally, STATE is equal to NY.

```
NOT (DEPT = 470) OR (DEPT = 470 AND
  STATE = 'NY')
```

Comparison can start from a certain line and end at the next line. However, a field name cannot start from a certain line and end at the next line. Field names must not exceed one line.

When a value to be tested is a character string enclosed in quotation marks, the value can start from a certain line and continue to the next line.

HAVING

This item is optional. It specifies which summary record is transferred.

Pay particular attention to the difference between **HAVING** and **WHERE**. **WHERE** operates on each record within a certain group. **HAVING**, on the other hand, operates only on summary records (records that contain summary information for each group).

With this item, you can specify one or more conditions that a summary record must satisfy prior to being transferred.

As the conditions, specify the tests that should be applied to the summary records. The specified test conditions are applied to all summary records, only those summary records that pass the tests are transferred. To transfer all summary records, leave the **HAVING** item blank.

The format of the conditions is as follows:

```
function (field-name)  test  value
```

This indicates:

function

This is a function supported for **SELECT**. See the description of **SELECT** in this section for details of these functions.

field-name

This is the field defined by the record format specified in **FROM**. A field name is acceptable even when it has not been specified in **SELECT**.

Test

This is the comparison type for functions. The types are listed below.

=

Equal to

<> or ><

Not equal

>

Greater than

>=

Greater than or equal to

<

Less than

<=

Less than or equal to

value

This is a function operating on certain fields or a constant. See WHERE on page 273 for details of constants, expressions, and tests.

Note: A comma is treated as a decimal point. Therefore, do not separate numbers with commas.

Test conditions can be combined by using logical AND or logical OR. When both AND and OR are specified, AND comparison is performed first. Up to 50 tests can be specified. By using parentheses, the operation order can be modified, or a description can be added to an operation. For example, you can specify:

```
COUNT(*) >=2 AND MAX(PRICE) > 100
```

In this case, the following conditions are applied concurrently: groups to be transferred must contain more than one record, and the summary records in such groups are transferred only when the maximum price is greater than 100.

If the desired field cannot be found, selecting Browse displays a list of the names of all fields in the record.

The type, length, digit, and number of decimal places of the value returned for each function are:

```
            Type     Length    Digit       Decimal Places
SUM         Packed     16       31    (Same as tested field)
AVG         Packed     16       31     31 (Total of the digit
                                           and decimal places
                                           of the field)
COUNT       Binary      4       10          0
MAX              (Same as tested field)
MIN              (Same as tested field)
```

ORDER BY

This item is optional. It specifies the order in which the requested records are grouped. When **ORDER BY** is not specified, record transfer is not done according to a certain order.

Records are grouped according to the field specified first. Those records having the same value in each field specified first are grouped by the field specified second, and so on. Records containing null values are grouped after all records without null values have been grouped.

For example, you can specify:

```
DEPT,NAME,PHONE
```

In this case, records are first grouped according to DEPT. Then, the records having the same value for DEPT are grouped by NAME. The records with the same DEPT and NAME values are finally grouped by PHONE.

When a field name is specified in **ORDER BY**, it must also have been specified in **SELECT**, or **SELECT*** must have been specified.

Fields can be grouped in ascending or descending order. To do this, specify one blank after a field name then enter ASC or DESC. The default value is ASC. For example, specify:

```
DEPT DESC, NAME ASC
```

This indicates that the DEPT fields are to be grouped in descending order, after which the NAME fields are to be grouped in ascending (alphabetic) order.

Absolute values (ABS) can be specified for numeric fields. To do this, add a blank after a field name then enter ABS. For those fields having negative values, the negative signs are ignored and the absolute values are used.

The total length of the fields to be specified must not exceed 120 digits.

Return Record at Missing Field Value

When joining records from several files, joining might fail because a record is missing. This item specifies whether records with missing fields are transferred.

When you specify that records with missing fields are to be transferred, the alternative values for the missing fields are transferred. These values are normally blanks for character fields and zeros for numeric fields.

When you do not specify transfer of records with missing fields, those records are not transferred.

Specify this item to transfer data records that have alternative values for missing fields.

Do not specify this item if data records that have alternative values for missing fields are not to be specified. In this case, only those data records created from those records that exist in all files specified in **FROM** are transferred.

TO

Output device

This item specifies where received data is to be sent.

Display

The received data is displayed on the screen.

Disk

The received data is written to a workstation diskette or hard disk file.

Printer

The received data is printed on the printer.

When **Disk** is selected as the output device, also specify the following items.

PC file

This item specifies the name of the workstation disk file or diskette file to which the data is to be written.

Replace old file

This item is always required. It specifies whether the records in the file specified by **PC File** are to be replaced with the transferred records.

The default value is **Replace old file**.

Workstation file type

This item is always required. It specifies the type of the workstation disk file or diskette file to which the transferred records are written.

The system default is PC code test.

Save transfer description

This item is always required. It specifies whether the workstation file description is written to a workstation file. This file description describes the transferred data and it is required to subsequently return data to the iSeries™, eServer™ i5, or System i5™.

The system default is **Save**.

Description file name

This item is always required. It appears only when **Save Transfer Description** is selected. The **File Description File Name** specifies the name of the workstation disk file or diskette file to which the file description is written.

This item automatically sets the desired file name. This file name is the same as that specified by the user for **TO**, but to which extension .FDF has been added. Extension .FDF indicates that this file is a file-description file.

The use of extension .FDF is recommended when using a unique file name. To specify a file name in this item, use the same format as that in **TO**. (Items inside brackets [ ] can be omitted.)

```
[d:][path-name]file-name[.ext]
```

After **Save File Description File** is specified or a name is specified for **File Description File Name**, the iSeries™→PC Transfer Request window reopens after the **Return** key is pressed. Using this screen, a transfer request can be changed, saved, or executed.

Saving, Opening, Changing, and Executing a Transfer Request

The following section explains how to save, open, change, and execute, as a file, the information (transfer request) on the data to be transferred.

Saving a Transfer Request

You should save a transfer request, especially when the request will be executed repeatedly. This eliminates the need to create a transfer request every time a request is executed. To save a transfer request, do as follows:

1. Specify the information needed for transfer, using the iSeries™→PC Transfer window. See Creating an iSeries-to-Workstation Transfer Request on page 264 for an explanation of how to specify the required data.
2. After specifying the necessary data, click **Save** or **Save As** from the File menu of the menu bar.

   The Save Transfer Request File As window opens.
3. Specify each item, referring to the following explanation, then click **OK**.

   File Name

   Disk to which data is to be saved. Specify a file name or diskette file name. The default extension is TTO. Extension TTO identifies a file as a transfer request file.

   Description

   This item can be used to add a short explanation of a transfer request, as required. The explanation can be up to 40 characters in length. This explanation is saved with the transfer request, and displayed in the list of transfer request names. It is useful, therefore, for identifying a transfer request.
4. The system asks whether the saved transfer request is to be registered in the PC400 folder.

   When you click **OK**, the transfer request is registered as an icon. Subsequently selecting this icon transfers data according to the registered data transfer request.

Opening and Changing a Saved Transfer Request

To open and change a saved transfer request:

1. Display the iSeries™→PC Transfer window.
2. Select **Open** from the **File** menu.

   The Open Transfer Request File window opens.
3. Specify the name of the file to be opened using the Open Transfer Request File window. Then click **OK**.

   The iSeries™→PC Transfer window reopens, with the information specified for each item for the transfer request displayed. This completes opening of the saved transfer request.
4. Change the contents, as necessary.
5. To save the changed contents, follow the procedure given in Saving a Transfer Request on page 282.

Executing a Transfer Request

You can execute a file transfer request in one of the following two ways:

- By selecting the icon with which the transfer request has been registered
- By using the iSeries™→PC Transfer window of the Data Transfer icon

Selecting the Icon with Which the Transfer Request Has Been Registered

This method can be used only when a transfer request has been saved as an icon by using the iSeries™→PC Transfer window.

Data transfer starts as soon as you select the icon with which a transfer request has been registered.

Using the iSeries→PC Transfer Window

1. Before attempting to execute a transfer request, all operations such as creating, opening, and changing a transfer request must have been completed.

   Note: When data is transferred from a workstation to an existing member in an iSeries™, eServer™ i5, or System i5™ file, the transferred data replaces the existing data in the member.

2. Select **Receive** from the iSeries™→PC Transfer window.

   Data transfer starts.
3. After the transfer has been completed, click **Cancel**, or click **Exit** from the **File** menu.

Status during Transfer

Display can be specified as the **output device**, when the current transfer request is created or changed. This sends the transferred record to the screen. On the screen, each record is displayed on one line.

Each field in a transferred record is converted from the iSeries™, eServer™ i5, or System i5™ data type to workstation code.

Note: The workstation receives the iSeries™, eServer™ i5, or System i5™ records in order and then writes them to a temporary file of the default directory in the default drive (usually, the directory in which PC400 is installed). The maximum number of records that can be transferred is 4096 records, limited by the amount of records that can be stored in free space of the default drive.

When **Disk** is selected as the **output device**, the following actions are performed:

1. The workstation file description is written to a workstation disk file or diskette file according to the **Save File Description File** specification. (If **Save File Description File** has not been specified, this procedure is not performed.)
2. The transferred records are written to a workstation disk file or diskette file.

Limited Usage of File Names and Field Names

For a transfer request from a workstation to the iSeries™, eServer™ i5, or System i5™, none of the following reserved words can be specified as a file name or field name:

CRTFILE	MBRTEXT
CRTMBR	PUBAUT
FILETEXT	RCDLEN
FILETYPE	REFFILE
INTO	

For a transfer request from the iSeries™, eServer™ i5, or System i5™ to a workstation, none of the following reserved words can be used as a file name or field name:

ABS	IS
AND	ISNOT
ASC	LIKE
AVG	MAX
BETWEEN	MIN
BY	NOT
COLUMNS	OPTIONS
COUNT	OR
DESC	ORDER
EXTRACT	PARTOUT
FROM	REPLACE
GROUP	SELECT
HAVING	SUBSTR
IN	SUM
INNER	TABLES
	WHERE

To use one of these reserved words as a file name or field name, use the reserved word in uppercase, enclosed in quotation marks:

```
TO MYLIB/"INTO"
```

Examples of Transfer Requests for Receiving

This section provides examples of transfer requests for receiving. The contents of this section provide supplementary information to help you better understand transfer requests for receiving.

This section describes how to transfer data from the iSeries™, eServer™ i5, or System i5™, based on the inventory control file INVENTORY and supplier file SUPPLIERS.

The INVENTORY file contains information about the various parts in stock. Each part has a three-digit identification number, PARTNUM. The INVENTORY file contains the names of parts (DESCRIPTION) and the quantity on hand (QONHAND) for each part.

```
       File: INVENTORY
Field name: PARTNUM  DESCRIPTION  QONHAND
             -------  -----------  -------
```

```
  Record 1:     209          CAM       50
         2:     221         BOLT      650
         3:     222         BOLT     1250
         4:     231          NUT      700
         5:     232          NUT     1100
         6:     207         GEAR       75
         7:     241       WASHER     6000
         8:     285        WHEEL      350
         9:     295         BELT       85
```

The SUPPLIERS file contains information about the suppliers of each part. Each supplier is identified by a two-digit number, SUPPNO. The SUPPLIERS file contains the number of parts delivered (PARTNO), their prices (PRICE), times of delivery (DELIVTIME), and ordered quantities (QONORDER). The parts listed in the SUPPLIERS file are the same as those listed in the INVENTORY file.

```
      File: SUPPLIERS
Field name: SUPPNO   PARTNO   PRICE   DELIVTIME   QONORDER
            ------   ------   -----   ---------   --------
  Record 1:     51      221     .30          10         50
         2:     51      231     .10          10          0
         3:     53      222     .25          15          0
         4:     53      232     .10          15        200
         5:     53      241     .08          15          0
         6:     54      209   18.00          21          0
         7:     54      221     .10          30        150
         8:     54      231     .04          30        200
         9:     54      241     .02          30        200
        10:     57      285   21.00          14          0
        11:     57      295    8.50          21         24
        12:     61      221     .20          21          0
        13:     61      222     .20          21        200
        14:     61      241     .05          21          0
        15:     64      207   29.00          14         20
        16:     64      209   19.50           7          7
```

Receiving Part of an iSeries, eServer i5, or System i5 File

Specify the following items:

Library/File (Member)	INVENTORY
SELECT	PARTNUM, QONHAND
WHERE	QONHAND < 100
ORDER BY	PARTNUM

In this case, only part of the INVENTORY file is to be transferred. Specifically, only the part number (PARTNUM) and quantity on hand (QONHAND) fields of the records for which the number of parts in stock is less than 100 (QONHAND < 100) are transferred. Records are transferred in ascending order of parts numbers (PARTNUM).
The following data is transferred:

```
    Field:   PARTNUM  QONHAND
             -------  -------
 Record 1:      207       75
        2:      209       50
        3:      295       85
```

Receiving Records Joined from Several iSeries, eServer i5, or System i5 Files

Two iSeries™, eServer™ i5, or System i5™ files, INVENTORY and SUPPLIERS, are assumed. Note that both files contain records including part number fields. The INVENTORY file contains inventory information about individual parts. The SUPPLIERS file contains information about purchasing and ordering.

You might want to transfer information on part numbers, part names, and the prices of the parts to be ordered from supplier 51. The desired fields are PARTNO (SUPPLIERS file), DESCRIPTION (INVENTORY file), and PRICE (SUPPLIERS file).

By comparing the data in the INVENTORY file and the SUPPLIERS file, the user can determine that supplier 51 provides part numbers 221 and 231, called BOLT and NUT, respectively, and that their prices are 30 cents and 10 cents, respectively. The following table summarizes this information:

```
    Field:   PARTNO   DESCRIPTION   PRICE
             ------   -----------   -----
 Record 1:    221     BOLT            .30
        2:    231     NUT             .10
```

The same results are available by joining the data in these two files by using the iSeries™→PC Transfer function. To do this, specify both files (INVENTORY and SUPPLIERS) in the **FROM** item. For **SELECT**, specify which fields are to be transferred (PARTNO, DESCRIPTION, and PRICE). For **WHERE**, specify which records are to be transferred (records for which SUPPNO = 51).

Respecify the relationship between the two files in **JOIN BY**. From these results, the user can determine, by checking the SUPPLIERS file, that part number 221 is delivered from supplier 51 at a cost of 30 cents. In addition, to determine the part name, the user must check the INVENTORY file for part number 221 and its product name. In other words, the user observes that data is joined from the records in both the SUPPLIERS file and the INVENTORY file and that those records have the same part number. Therefore, to link the two records in these files, the records must have the same part number.

In short, to obtain this information, specify:

Library/File (Member)	SUPPLIERS, INVENTORY
JOIN BY	PARTNO = PARTNUM
SELECT	PARTNO, DESCRIPTION, PRICE
WHERE	SUPPNO = 51
ORDER BY	PARTNO

Receiving Records Using File Qualifiers

To join records from several iSeries™, eServer™ i5, or System i5™ files, fields of the same type must be joined.

For example, the part number fields in the INVENTORY and SUPPLIERS files can have the same name PARTNO. To specify the desired PARTNO fields, you must specify which file contains those fields. To do so, file qualifiers are used.

A file qualifier is the character T (uppercase or lowercase) followed by a one- or two-digit number. Use a comma to delimit the file qualifier and field name. In the previous example, prefix T1. and T2. to the PARTNO field names. **T1.** indicates the first file of **FROM**, while T2. indicates the second.

To obtain the same information as in the previous example, specify:

Library/File (Member)	SUPPLIERS, INVENTORY
JOIN BY	T1.PARTNO = T2.PARTNO
SELECT	T1.PARTNO, DESCRIPTION, PRICE
WHERE	SUPPNO = 51
ORDER BY	T1.PARTNO

T1.PARTNO indicates the PARTNO fields in the SUPPLIERS file, while T2.PARTNO indicates the PARTNO fields in the INVENTORY file.

Qualifiers are not needed for the names of the DESCRIPTION, PRICE, and SUPPNO fields, because they exist in one file only. However, the user can specify the following qualifiers for clarity:

```
T2.DESCRIPTION, T1.PRICE, T1.SUPPNO
```

The following examples of joining several iSeries™, eServer™ i5, or System i5™ files describe more sophisticated techniques. You should now be familiar with the basics of how to join two files. For a more detailed explanation, refer to the following sections.

Receiving with Field Missing Records Joined

The joining of records from several files could fail because one or more records is missing. For example, the record containing part number 221 might not be found in the INVENTORY file. This means that the records that can be joined to the 1st, 7th, and 12th records in the SUPPLIERS file do not exist in the INVENTORY file. In this case, the PARTNO field and PRICE field for part number 221 can be determined, but the DESCRIPTION field cannot be determined. So, the DESCRIPTION field is missing.

To transfer field missing records, use **Return Record at Missing Field Value**.

When **Return Record at Missing Field Value** has been specified, the default iSeries™, eServer™ i5, or System i5™ values are transferred instead of the missing field values. The default values for character fields are blanks, while those for numeric fields are zeros. For example, if the INVENTORY file does not contain the part number 221 record, the result of the previous example will be as follows:

```
     Field: PARTNO   DESCRIPTION   PRICE
            ------   -----------   -----
```

```
 Record 1:    221                 .30
        2:    231   NUT           .10
```

If **Return Record at Missing Field Value** has not been specified, the field missing records are not transferred. For example, if the INVENTORY file does not contain the part number 221 record, the result of the previous example will be as follows:

```
    Field: PARTNO   DESCRIPTION   PRICE
           ------   -----------   -----
 Record 1:    231   NUT             .10
```

Receiving with Records in a Same File Joined

Records in the same file can be joined. In other words, a file can be repeatedly specified in **FROM**. For instance, data in certain records can be compared using this function.

For example, the SUPPLIERS file shows that several suppliers provide the same part. The user might want to know which supplier sets a price that is double, or greater than double, that of another. To transfer the necessary information to a workstation, specify:

Library/File (Member)	SUPPLIERS, SUPPLIERS
JOIN BY	T1.PARTNO = T2.PARTNO
SELECT	T1.PARTNO, T1.SUPPNO, T1.PRICE, T2.SUPPNO, T2.PRICE
WHERE	T1.PRICE > 2 * T2.PRICE
ORDER BY	T1.PARTNO

The same file has been specified in **FROM** twice. **JOIN BY** specifies that records having the same part number are joined. This creates a joined record containing information about two suppliers of a single part. The user can spot those records for which the price is double, or greater than double, that of another supplier.

Records in the **SUPPLIERS** file are compared, one by one, with all the records (including itself) in the **SUPPLIERS** file. When the same part number is found, the two corresponding records are linked. This processing is performed for each record in the **SUPPLIERS** file.

For each record, the first supplier's price is compared with the second supplier's price. When the first supplier's price is double, or greater than double, that of the second, only the record containing the first supplier price is kept.

The final result is as follows:

```
    Field:  T1.PARTNO   T1.SUPPNO   T1.PRICE   T2.SUPPNO   T2.PRICE
            ---------   ---------   --------   ---------   --------
 Record 1:        221          51        .30          54        .10
        2:        231          51        .10          54        .04
        3:        241          53        .08          54        .02
        4:        241          61        .05          54        .02
```

Specifying Records To Be Included in a Group

You might want to limit which records will be included in a group. To do so, use **WHERE**. The following example transfers the average and lowest prices of each part for those records for which the delivery time (DELIVTIME) is less than 30 days.

Library/File (Member)	SUPPLIERS
GROUP BY	PARTNO
SELECT	PARTNO, AVG(PRICE), MIN(PRICE)
WHERE	DELIVTIME < 30

The result is as follows:

```
   Field: PARTNO   AVG(PRICE)   MIN(PRICE)
         ------   ----------   ----------
Record 1:   221          .25          .20
       2:   231          .10          .10
       3:   222          .23          .20
       4:   232          .10          .10
       5:   241          .07          .05
       6:   209        18.75        18.00
       7:   285        21.00        21.00
       8:   295         8.50         8.50
       9:   207        29.00        29.00
```

Note that the conditions specified in **WHERE** are checked first, then the records that satisfy those conditions are included in the group.

Specifying Summary Records To Be Transferred

In some cases, you might want to transfer only summary records that satisfy certain conditions. The use of **HAVING** enables the selection of which summary records are to be transferred. **WHERE** is applied to certain records in a group, while **HAVING** is applied only to summary records.

The following example transfers the highest and lowest prices for each part. However, the summary records to be transferred are only those for which the highest price exceeds 10.00.

Library/File (Member)	SUPPLIERS
GROUP BY	PARTNO
SELECT	PARTNO, MAX(PRICE), MIN(PRICE)
HAVING	MAX(PRICE) > 10.00

The following table shows the result of removing unnecessary summary records by using **HAVING**

```
   Field: PARTNO   MAX(PRICE)   MIN(PRICE)
          ------   ----------   ----------
```

```
Record 1:    209        19.50        18.00
       2:    285        21.00        21.00
       3:    207        29.00        29.00
```

One summary record for an entire file can be transferred. To do this, specify only the summary function in **SELECT** and nothing in **GROUP BY**. As a result, an entire file can be recognized as one group, while one summary record can be transferred for the group.

You can concurrently use the concept of summarizing groups and that of joining records from several files. To obtain the desired results, do as follows:

1. Specify a file in **FROM**, and specify the join conditions to join the records in **JOIN BY**.
2. Specify the conditions in **WHERE** to remove unnecessary records.
3. Specify the fields used for grouping the remaining records in **GROUP BY**.
4. Specify the function in **SELECT**, then create summary records.
5. Specify the conditions in **HAVING** to remove unnecessary records.
6. Specify the items for grouping the final summary records in **ORDER BY**.

Functions Available from the Pull-Down Menu

The following section provides a simple explanation of the menu bar of the iSeries™→PC Transfer window and PC→iSeries™ Transfer window.

File

Transfer request files can be processed.

Create

Creates a transfer request file

Open

Displays the contents of an existing transfer request file

Save, Save As

Save the current settings to the transfer request file being used or to a new transfer request file, respectively

Exit

Terminates the operation started by selecting the **Data Transfer** icon

Setup (Only for iSeries→PC Transfer)

User Options

Time, date, and numeric value format for receiving can be specified.

Ignore Decimal Data Error

Specifies whether decimal data errors found in packed or zoned decimal fields upon executing requests are to be ignored. Selecting **Yes** to ignore decimal data errors and using existing indices can considerably reduce the time needed to execute a request. If this item is not specified, the transfer request creates indices again and modifies any detected decimal data errors. This requires extra processing time.

Time Format

Specifies a desired time format for fields of iSeries™, eServer™ i5, or System i5™ field type having a selected time. If no time format is specified, the default value in the workstation's national information file is used when the transfer request starts, and that in an existing transfer request is assumed when the request is called again.

Supported time formats are as follows:

HMS

Hours, minutes, seconds (hh:mm:ss)

ISO

International Standard Organization (hh.mm.ss)

USA

USA Standard (hh:mm AM or PM)

EUR

IBM® European Standard (hh.mm.ss)

JIS

Japanese Industrial Standard (hh:mm:ss)

DDS

iSeries™, eServer™ i5, or System i5™ DDS (Format given by iSeries™, eServer™ i5, or System i5™ file attribute)

DFT

iSeries™, eServer™ i5, or System i5™ default format (Host job default is used)

Time separator

Specifies enabled delimiters. The fields of the iSeries™, eServer™ i5, or System i5™ field type for the selected time must be in a format that supports delimiters.

When no delimiters are specified, the default value in the workstation's national information file is used when the transfer request starts, and that in an existing transfer request is assumed when the request is called again.

Supported time delimiters are as follows:

Colon

(:)

Period

(.)

Comma

(,)

Blank

( )

Null

(NULL) No Separator

Default value

(DFT) iSeries™, eServer™ i5, or System i5™ Default Separator

Date Format

Specifies the date format for fields of iSeries™, eServer™ i5, or System i5™ field type for the selected date.

If this date format is not specified, the default value in the workstation's national information file is used.

Supported values are as follows:

MDY

Month, day, year (mm/dd/yy)

DMY

Day, month ,year (dd/mm/yy)

YMD

Year, month, day (yy/mm/dd)

JUL

Julian (yy/ddd)

ISO

International Standard Organization (yyyy-mm-dd)

USA

USA Standards (mm/dd/yyyy)

EUR

IBM® European Standard (dd.mm.yyyy)

JIS

Japanese Industrial Standard (yyyy-mm-dd)

DDS

iSeries™, eServer™ i5, or System i5™ DDS (Format given by iSeries™, eServer™ i5, or System i5™ file attribute)

DFT

iSeries™, eServer™ i5, or System i5™ default format (Host job default is used)

Date separator

Specifies delimiters. The fields of the iSeries™, eServer™ i5, or System i5™ field type for the selected date must be in a format that supports delimiters.

When no date delimiters are specified, the default value in the workstation's national information file is used when the transfer request starts, and that in an existing transfer request is used when the request is called again.

Supported date delimiters are as follows:

Slash

(/)

Dash

(-)

Period

(.)

Comma

(,)

Blank

( )

Null

(Null) Delimiters are not used.

DFT

(DFT) iSeries™, eServer™ i5, or System i5™ default separator

Decimal separator

Specifies the decimal point character in an iSeries™, eServer™ i5, or System i5™ field whose type is packed decimal or zoned decimal.

When decimal points are not specified, the default value in the workstation's national information file is used when the transfer request starts, and that in an existing transfer request is used when the request is called again.

Supported decimal point delimiters are as follows:

Period

(.)

Comma

(,)

DFT

(DFT) - Default decimal separator

Sort Sequence

Specifies which sort sequence should be used for this transfer request.

iSeries job default

Sort by the table identified on the iSeries™, eServer™ i5, or System i5™ as the job sort table.

Hexadecimal

Sort by the internal hexadecimal representation.

User specified table

Sort by the table identified by the user in a subsequent prompt.

Shared Weight Table

Sort by the shared weight table associated with the language named in a subsequent prompt.

Unique Weight Table

Sort by the shared unique table associated with the language named in a subsequent prompt.

Changing the sort sequence affects the order in which records appear ***only*** if the **ORDER BY** clause is being used. The sort sequence affects all character comparisons that depend on the order of the alphabet. Such comparisons can occur in the **WHERE** clause, the **GROUP BY** clause, the **HAVING** clause, the **JOIN BY** clause, the **IN** predicate, the **LIKE** predicate, the **BETWEEN** predicate, the **MAX** function, and the **MIN** function. Comparison operations are =, <>, >, >=, and >=.

Sort Sequence Table Name

Type the name of the sort sequence table that you want to use for this transfer request. The format of the table name should be *library/table*. *LIBL and *CURLIB are allowed for the library name.

Translation Table

Translation tables for ASCII-to-EBCDIC translation or for EBCDIC-to-ASCII translation can be specified, created, and customized.

Current Table

Specifies whether the default translation or the user-defined translation table is to be used.

Host Code Page

Specifies the host code page to be used for translation.

Workstation Code Page

Specifies the workstation code page to be used for translation.

File Name

Specifies the file name of the user-defined table to be used for translation.

- To list all files in your workstation, click **Browse**.
- To customize the translation table, click **Customize**.

Signon Options

When configuring a Data Transfer session, the user specify the following signon parameters:

Use Kerberos principal, no prompting

This function enables Kerberos authentication, using the ticket generated by the Windows user credentials. This option is disabled by default.

Prompt as needed

The host prompt the user for signon information. For each host, the signon dialog is presented only once during the transfer session.

The user enable or disable the “Kerberos Signon Options” using the “Kerberos auto-signon” check box provided in the “Custom” installation UI. By default, “Kerberos auto-signon” is checked and “Kerberos Signon Options” is enabled.

The user disable the “Kerberos Signon Options” by unchecking the “Kerberos auto-signon” check box in the “Custom” installation UI.

For the “Typical” installation, “Kerberos Signon Options” is enabled by default and there are no UI options to enable or disable the “Kerberos Signon Options.”

Silent Installation

The user enable or disable the “Kerberos Signon Options” by setting the value of “KERBEROSSIGNON” property in the *custom.ini* file.

By default, “Kerberos Signon Options” is enabled and “KERBEROSSIGNON” property is set to “1” in the custom.ini file. The user disable the “Kerberos Signon Options” by setting the value of “KERBEROSSIGNON” property to “0” in the custom.ini. The user must pass *custom.ini* as an input during the reinstallation of Personal Communications.

Note: The user is not able to enable/disable the "Kerberos Signon Options" for the Refresh Pack Installer.

File-Description Files

A file-description file is a workstation file that contains all field descriptions of the data in the corresponding workstation data file. Each field descriptor contains the field name, data type, and field length. There is one field descriptor for each field in the workstation file.

A file-description file defines the following:

- The file type of the workstation file to be transferred. For an explanation of each file type, see Creating a File-Description File on page 297.
- The field names and order of these fields in each data record.
- The data type of each field in the workstation file.
- The size and number of decimal places of each field.

The workstation files require field definitions when the files are transferred. The field definitions describe the file as it exists on the workstation. These definitions contain data that is similar to the field definitions (DDS) required by iSeries™, eServer™ i5, or System i5™ files. The data must be defined for both the iSeries™, eServer™ i5, or System i5™ and the workstation files, because the field names from each file are needed to send the data to the iSeries™, eServer™ i5, or System i5™ and the data in each file might be in different formats.

A file-description file is created on request during the transfer process of data from an iSeries™, eServer™ i5, or System i5™ file to a workstation file. Therefore, you usually do not need to worry about the contents or the format of the file-description file. However, if you transfer data that has not been previously transferred to the system, you must create a file-description file.

Creating a File-Description File

You can create a file-description file using a workstation text editor. The file-description file must be an ASCII text file. Therefore, each record must end with a carriage return (CR) character (hex 0D) followed by a line feed (LF) character (hex 0A). All tab characters (hex 09) are treated as ASCII spaces. The last byte of the file must contain an end-of-file (EOF) character (hex 1A). Workstation editors that create ASCII text files usually use these special character designators, so normally you do not need to be concerned about them.

File-Description File Format

The format of the file-description file is as follows:

```
PCFDF [comment]
PCFT file-type-indicator [comment]
PCFO time-format,time-separator, date-format, date-separator, decimal-separator [comment]
PCFL field-name-1 data-type-1 length-1[/decimal-position-1][comment]
    .
    .
    .
PCFL field-name-n data-type-n length-n[/decimal-position-n][comment]
[* comment]
```

Items within brackets are optional. Use either uppercase or lowercase characters anywhere in the file.

PCFDF Entries

PCFDF is a keyword that identifies this file as a workstation file-description file. It must appear in the first line of the file, starting in column 1. A comment is the only other entry allowed on the first line. If you type a comment, it must be separated from the PCFDF keyword by a space.

PCFT Entries

PCFT is a keyword that identifies this record as containing the file type indicator. It is followed by an indicator identifying the type of file in which the data is stored. It must appear only once, and must start in column 1, after the PCFDF record and before any PCFL records. An optional comment can follow this file-type indicator if separated from the indicator by at least one space.

Following is an example of a PCFT entry:

```
PCFT 4 BASIC RANDOM FILE
```

Table 34: File-Type Indicators on page 298 shows the valid file-type indicators.

Table 34. File-Type Indicators

Indicator	File Type
1	ASCII text
2	DOS random
3	BASIC sequential
4	BASIC random
5	Data interchange format (DIF**)
6	No-conversion file
7	Reserved
8	DOS random type 2
9	BIFF format

PCFO Entry

The PCFO entry is optional. PCFO is a keyword that identifies this record as containing information about the date and time formats, time stamp, and separator characters for applicable formats. It must appear only once and must start in column 1, after the PCFT record and before any PCFL records. If there is no PCFO entry, the information or characters assigned as defaults for the host system are used.

Table 35: Time Formats on page 299 shows the valid time formats.

Table 35. Time Formats

Indicator	Format Name	Time Format
1	HMS	hh:mm:ss
2	ISO - International Standards Organization	hh.mm.ss
3	USA - USA standard	hh:mm AM or PM
4	EUR - European	hh.mm.ss
5	JIS - Japanese Industrial Standard Christian Era	hh:mm:ss
6	DDS	Format given by iSeries™, eServer™ i5, or System i5™ file attribute
7	DFT	Host job default is used
*	Unspecified	Host job default is used

Table 36: Time Separators on page 299 shows the valid time separators.

Table 36. Time Separators

Indicator	Separator
1	Colon (:)
2	Period (.)
3	Comma (,)
4	Blank ( )
5	Null (N)
6	Default (D) (host job default)
*	Unspecified (host job default)

Table 37: Date Formats on page 299 shows the valid date formats.

Table 37. Date Formats

Indicator	Format Name	Date Format
1	MDY	mm/dd/yy
2	DMY	dd/mm/yy
3	YMD	yy/mm/dd
4	Julian	yy/ddd
5	ISO	yyyy-mm-dd
6	USA	mm/dd/yyyy
7	EUR	dd.mm.yyyy
8	JIS	yyyy-mm-dd
9	DDS	Format given by iSeries™, eServer™ i5, or System i5™ file attribute
10	DFT	Host job default is used
*	Unspecified	Host job default is used

Table 38: Date Separators on page 300 shows the valid date separators.

Table 38. Date Separators

Indicator	Separator
1	Slash (/)
2	Dash (-)
3	Period (.)
4	Comma (,)
5	Blank ( )
6	Null (N)
7	Default (D) (host job default)
*	Unspecified (host job default used)

Table 39: Decimal Separators on page 300 shows the valid decimal separators.

Table 39. Decimal Separators

Indicator	Separator
1	Period (.)
2	Comma (,)
*	Unspecified (workstation default used)

Following is an example of a PCFO entry:

```
PCFO 1,1,1,1,1 OPTIONS SETTINGS
```

PCFL Entries

PCFL identifies a definition for a field. Enter a PCFL entry in the file-description file for each field in the data file. The PCFL records must be in the same order as the fields they define in the data file.

Define as many as 256 PCFL records in the file-description file and start PCFL records in column 1. If you enter more than 256 PCFL records, you receive an error message. You cannot continue a record on one line, and only the first 80 characters of a record are used.

Following is an example of a PCFL entry:

```
PCFL CUSTNAME 1 20  CUSTOMER NAME
```

Each PCFL entry contains the following things:

- The keyword, PCFL, starting in column 1 and followed by a space. This identifies the record as a field description.
- The field name, followed by a space. This must match the name that exists in the field definitions on the iSeries™, eServer™ i5, or System i5™ and can be from 1 to 10 characters.

- The indicator for the data type. Table 40: Data Type Indicators on page 301 shows the indicators that represent the data type of the data in the field. Follow the specified indicator with a space.
- The size of the field (in bytes) as it is stored in the workstation file. The length specification can be from 1 to 4 characters.

Table 40. Data Type Indicators

Indicator	Data Type
1	ASCII[1]
2	ASCII numeric
3	Hexadecimal
4	Binary
5	Zoned
6	Packed
7	BASIC integer
8	BASIC single-precision floating point
9	BASIC double-precision floating point
10	EBCDIC
11	EBCDIC zoned
12	EBCDIC packed

[1] Includes date, time, and time stamp except for files that are not converted.

The data type indicator you enter must be valid for the file type entered earlier. Any other data types are not valid and are diagnosed as errors during a data transfer to the iSeries™, eServer™ i5, or System i5™.

Table 41: Valid SBCS Data Types for File Types on page 301 shows the valid single-byte character set (SBCS) data types for each file.

Table 41. Valid SBCS Data Types for File Types

File Type	Valid Data Type
ASCII text	ASCII ASCII numeric
DOS random	ASCII Binary Hexadecimal ASCII Packed Zoned
BASIC sequential	ASCII ASCII numeric

Table 41. Valid SBCS Data Types for File Types

(continued)

File Type	Valid Data Type
BASIC random	ASCII BASIC double-precision floating point BASIC integer BASIC single-precision floating point Hexadecimal
DIF	ASCII ASCII numeric
No-conversion	Binary EBCDIC EBCDIC packed EBCDIC zoned Hexadecimal
DOS random type 2	ASCII Binary Hexadecimal Packed Zoned
BIFF format	ASCII ASCII numeric

Note: ASCII (SBCS) includes date, time, and time stamp types if converted. EBCDIC includes date, time, and time stamp if not converted.

For numeric fields in BASIC sequential and DIF files, a size specification must be present. However, because the data in these fields is of variable length, the data transfer function assumes a maximum length of 65 characters. This length more than covers the largest possible exponential ASCII numeric value. The size specifications for character fields must be the maximum size of any data item in that field.

Table 42: Allowable Data Length Limits for Personal Computer SBCS Data Types on page 302 shows the allowed data length limits for each workstation data type. These are the maximum lengths you can specify for size in the PCFL entry.

Table 42. Allowable Data Length Limits for Personal Computer SBCS Data Types

Personal Computer Data Type	Data Length Limit (in Bytes)
ASCII	4093
ASCII numeric	33 (65 for DIF and BASIC sequential)
BASIC double-precision	8 (only allowed length)
BASIC integer	2 (only allowed length)
BASIC single-precision	4 (only allowed length)
Binary	4
EBCDIC	4093
Hexadecimal	2048
Packed decimal (ASCII and EBCDIC)	16
Zoned decimal (ASCII and EBCDIC)	31
Time HMS [1] USA ISO, EUR, and JIS [1] DDS, DFT	• 8 • 8 • 8 • 8 or 10 [2]

Table 42. Allowable Data Length Limits for Personal Computer SBCS Data Types

(continued)

Personal Computer Data Type	Data Length Limit (in Bytes)
Date MDY, DMY, YMD Julian ISO, EUR, JIS, USA (see note 1) DDS, DFT	• 8 • 6 (only allowed length) • 10 • 6, 8, or 10 [2]
Time stamp	• 26

Notes:

1

These abbreviations appear in the time and date parameter sections.

HMS

Hours Minutes Seconds

EUR

IBM® European Standard

JIS

Japanese Industrial Standard Christian Era

ISO

International Standards Organization

2

The length is determined by the format defined in the host file for DDS, or from the iSeries™, eServer™ i5, or System i5™ job default (DFT keyword).

Table 43: Allowable Data Length Limits for iSeries, eServer i5, or System i5 Data Types on page 303 shows the allowed data length limits for each iSeries™, eServer™ i5, or System i5™ data type.

Table 43. Allowable Data Length Limits for iSeries™, eServer™ i5, or System i5™ Data Types

iSeries, eServer i5, or System i5 Data Type	Data Length Limit in Bytes [1]
Binary	2 or 4 (only allowed lengths)
EBCDIC	4096
Hexadecimal	2048
Packed decimal (EBCDIC)	16
Zoned decimal (EBCDIC)	31

Table 43. Allowable Data Length Limits for iSeries™, eServer™ i5, or System i5™ Data Types

(continued)

iSeries, eServer i5, or System i5 Data Type	Data Length Limit in Bytes [1]
Time	
HMS	8
USA	8
ISO, EUR, and JIS	8
DDS, DFT	8 or 10 [2]
Date	
MDY, DMY, YMD	8
Julian	6 (only allowed length)
ISO, EUR, JIS, USA	10
DDS, DFT	6, 8, or 10 [2]
Time stamp	26

Notes:

1

The data length limits for the workstation and the system data fields are different in some cases. For these cases, the transfer function attempts to fit the workstation data into the system field. If the data does not fit into the field, a message is displayed. Refer to Data Conversions on page 305 for more details.

2

The length is determined by the format defined in the host file for DDS, or from the iSeries™, eServer™ i5, or System i5™ job default (DFT keyword).

If there is a decimal position associated with the data in that field, place a forward slash (/) and then the number of decimal positions after the length specification. There are no spaces between the length, slash, and decimal position specifications.

The decimal position specification refers to the number of positions from the right-hand byte of the resulting decimal number. Do not specify a decimal position for floating-point numbers unless the data type is one of the following types:

- ASCII numeric
- Binary
- Packed
- Zoned

Note: The number of decimal positions in a field ranges from 0 to 9 or the maximum number of decimal digits in this number, whichever is smaller. The data transfer function might round the number to fit it into the field. Refer to Data Conversions on page 305 for more details.

Comment Entries

Enter comment lines anywhere in the file-description file, observing the following restrictions:

- The last element of the field-descriptor entry specification is a comment. This is an optional entry for your information only, and must be separated from the size entry by a space. PCFL entries created by the data transfer function (RTOPC) do not contain a comment field.
- Precede the comment with an asterisk (*) as the first nonspace character in the line.
- Do not exceed 80 characters in length.
- Do not make the comment the first record in the file-description file.

Following is an example of a comment:

```
* This is a comment
```

File-Description File Example

Following is an example of a file-description file for an inventory file:

```
PCFDF
PCFT 3 BASIC SEQUENTIAL FILE
* ITEM INVENTORY FILE
PCFO 1,1,1,2,1 OPTIONS SETTINGS
PCFL ITEMNO 2 8  ITEM NUMBER
PCFL ITEMDESC 1 20  DESCRIPTION OF ITEM
PCFL COLOR 1 8  COLOR
PCFL WEIGHT 2 7/2  ITEM WEIGHT
PCFL PRICE 2 7/2  PRICE PER ITEM
PCFL INSTOCK 2 6  ITEMS IN STOCK
```

Data Conversions

The data transfer function needs data conversions for transferring data from the system to the workstation, and vice versa. For both types of transfers, the necessary conversion depends on the record size, the type of data being transferred, the type of workstation file being used, the system data type, and, in some cases, the data length.

Record Size

Each transferred record contains data indicating whether each field contains a null value. There is a restriction on the maximum data record that can be sent or received from the iSeries™, eServer™ i5, or System i5™ because of this data.

The following formula determines the maximum record length that can be transferred:

- 4096 - (number of fields in the record + 2) = (maximum record length)

Data Types

The data transfer function supports the following system data types:

- Date
- Time
- Time stamp
- Binary data
- Character data
- Hexadecimal data
- Packed decimal data
- Zoned decimal data

The data transfer function supports the following workstation data types:

- BASIC numeric data, including:
  - Double-precision data
  - Integer data
  - Single-precision data
- Binary data
- Character data, including:
  - ASCII
  - EBCDIC
- Hexadecimal data
- Packed decimal data
- Zoned decimal data
- ASCII numeric data

Date, Time, and Time-Stamp Data Types

Date, time, and time-stamp values can be used in certain arithmetic and character operations and are compatible with certain character constants, but they are neither characters nor numbers.

A date is a three-part value (year, month, and day) designating a point in time on the calendar. The range of the year is 0001 to 9999. The range of the year for a non-SAA format is 1940 to 9999. The range of the month is 1 to 12. The range of the day is 1 to *x*, where *x* depends on the month.

A time is a three-part value (hour, minute, and second) designating a time of day under a 24-hour clock. The range of the hour is 0 to 24 and the range of the other values is 0 to 59.

A time stamp is a seven-part value (year, month, day, hour, minute, second, and microsecond) that designates a date and time including the specified microseconds. The maximum length of the time stamp is a character string of 26.

Dates, times, and time stamps can be assigned to result fields. A valid character-string representation of a date can be compared with a date field, or a valid character-string representation of a time can be compared with a time field.

BASIC Numeric Data

Double-Precision Data

Double-precision data is defined only for the workstation. The iSeries™, eServer™ i5, or System i5™ does not support this data type. BASIC applications use double-precision data. This data type is a positive or negative number from 2.938735877055719 x 10^{-39} to 1.701411834604692 x 10^{38}. Double-precision numbers are stored in 8 bytes, with 7 bytes representing the mantissa and 1 byte representing the exponent.

Integer Data

Integer data is defined only for the workstation. BASIC applications use integer data. Integer data is stored in 2 bytes and represents a whole number from -32768 to 32767.

Single-Precision Data

Single-precision data is defined only for the workstation. The iSeries™, eServer™ i5, or System i5™ does not support this data type. BASIC applications use single-precision data. This data type is a positive or negative number from 2.938736 x 10^{-39} to 1.701412 x 10^{38}. Single-precision numbers are stored in 4 bytes, with 3 bytes representing the mantissa and 1 byte representing the exponent and sign.

Binary Data

This data represents signed or unsigned numbers in twos complement form. Binary numbers of 1, 2, 3, or 4 bytes in length are allowed on the workstation, but the iSeries™, eServer™ i5, or System i5™ allows only numbers 2 or 4 bytes in length. The bit on the left side of the high-order bit determines the sign of the number (0 for positive, 1 for negative). The system stores the data with the high-order byte on the left side of the field, whereas the workstation stores the data with the high-order byte in the right-hand position of the field.

The decimal position, if specified by the file description, represents the number of decimal digits to the right of the decimal point. The file description specifies the presence of a decimal position.

For example, the binary number 3BF5 is equivalent to the decimal number 15349, and the binary number FFB4 is equivalent to the decimal number -76.

Character Data for SBCS

You can think of this data as a string of bits that represents particular characters and symbols.

The tables used to translate characters from ASCII to EBCDIC and from EBCDIC to ASCII contain the following kinds of values:

- Values where the workstation ASCII characters and iSeries™, eServer™ i5, or System i5™ EBCDIC characters match exactly
- Values where a substitute character is chosen for a character that cannot be translated

The data transfer function uses tables to translate data from ASCII to EBCDIC and EBCDIC to ASCII. You can change these default tables using the translation table utility (TRTABLE).

Note: ASCII (SBCS) data includes date, time, and time stamp types if converted. EBCDIC data includes date, time, and time stamp if not converted.

Hexadecimal Data

You can think of this data as a string of bits representing base 16 numbers. For example, you can represent hex 3D with the following string of bits:

0011 1101

Packed Decimal Data

For both the iSeries™, eServer™ i5, or System i5™ and the workstation, each half-byte represents a value from 0 through 9. The hexadecimal value in the half-byte on the right side of the right-hand byte specifies the sign.

For the iSeries™, eServer™ i5, or System i5™, a value of hex B or hex D in this half-byte represents a negative number.

For DOS random files, only the last half-byte (the half-byte that contains the sign) is changed. For the sign half-byte, the workstation uses hex 3 to indicate a positive number or hex B to indicate a negative number.

For example, X'0865431F' appears as X'08654313'.

For DOS random type-2 files, the last half-byte (the half-byte that contains the sign) is not changed. The sign convention used on the workstation and on the host system is the same.

For example, X'0865431C' appears as X'0865431C'.

The decimal position, if specified, represents the number of decimal digits to the right of the decimal point. The presence of a decimal position is specified in the file description.

Zoned Decimal Data

This data is represented in a form in which each byte corresponds to one decimal digit. Each of these bytes is stored in character form. For example, the digit 7 is stored on the iSeries™, eServer™ i5, or System i5™ as F7, which is the EBCDIC representation, and is stored on the workstation as 37, which is the ASCII representation.

The size of each digit is determined by its half-byte on the right side. Valid values for the half-bytes are decimal 0 through 9.

The sign in both the iSeries™, eServer™ i5, or System i5™ and workstation zoned decimal fields is specified by the hexadecimal value in the left half-byte of the right byte of the field. For the iSeries™, eServer™ i5, or System i5™, a hex B or hex D in this half-byte represents a negative number (for example, X'F6D2' represents -62).

For DOS random files, zoned decimal fields from the system change from EBCDIC to ASCII, as do character fields, except that the sign half-byte in the workstation field is changed to a hex 3 to indicate a positive number or a hex B to indicate a negative number.

For DOS random type-2 files, zoned decimal fields from the system change from EBCDIC to ASCII, as do character fields, except that the sign half-byte in the workstation field is changed to a hex 3 to indicate a positive number or a hex 7 to indicate a negative number.

The decimal position, if specified, represents the number of decimal digits to the right of the decimal point and is specified by the file description.

ASCII Numeric Data

The data transfer function defines ASCII numeric data to represent any numeric value stored in ASCII format. This is not a valid iSeries™, eServer™ i5, or System i5™ system data type. The number -123.45 in ASCII format is:

2D 31 32 33 2E 34 35

The decimal point and sign are stored explicitly for ASCII numeric data. The character on the left displays the sign (space or plus (+) for positive, minus (-) for negative). Leading zeros to the left of the decimal point change to spaces. The decimal point, if any, is added in the correct position.

BASIC sequential and DIF file types also support another form of ASCII numeric data called exponential numbers.

An exponential number is a decimal number followed by the letter E or D and a signed integer of two or three digits. E represents a single-precision number and D represents a double-precision number. The exponent portion (E or D and the integer) represents *"times 10 to the power of the integer specified"*.

For example, the number -1.0E+03 (representing -1.0 x 10^{3} in ASCII numeric format) is:

2D 31 2E 30 45 2B 30 33

For example, the number 9.5D-15 (representing 9.5 x 10^{-15} in ASCII numeric format) is:

39 2E 35 44 2D 31 35

Personal Computer File Types

The following workstation file types are supported:

- ASCII text files
- BASIC random files
- BASIC sequential files
- DIF files
- BIFF files
- DOS random files
- DOS random type-2 files
- No-conversion files

ASCII Text Files

ASCII text files are normally used with programs that work with text (such as editors and print routines). The characteristics of an ASCII text file are as follows:

- Records consist of ASCII characters.
- A carriage return character (hex 0D) and a line feed character (hex 0A) delimit each record from the next.
- Workstation records in an ASCII file can be variable in length due to truncation of trailing blanks at the end of an iSeries™, eServer™ i5, or System i5™ record.

Transferring Data to ASCII Text Files

When you create an ASCII text file, the data coming from the iSeries™, eServer™ i5, or System i5™ changes as follows:

- Hexadecimal fields change to equivalent ASCII characters for each half-byte. For example, X'D3' expands to ASCII 4433 and is written to the file. When displayed by an editor or printed, the string appears as D3.
- EBCDIC character fields change byte by byte and are mapped into ASCII characters as defined by the translation tables.
- Date, time, and time-stamp data is mapped into ASCII characters as defined by the translation tables.
- Variable-length and null fields are converted to fixed lengths, and trailing blanks (for character, hexadecimal, date, time, and time-stamp data) or zeros (for binary, zoned, and packed,) are added to the maximum length of the field.

> **Note:** Some nondisplayable EBCDIC characters are translated into ASCII control characters on the workstation. If EBCDIC character fields contain nondisplayable data, you might get unexpected results and your ASCII text file might appear to be corrupted.

For example, X'05' in an EBCDIC field is translated to an ASCII X'09', which is an ASCII control character for horizontal tab. Most workstation text editors process this tab character so that the data in your workstation text file appears to be shifted to the right when viewed.

One possible solution to this problem is to define these fields on the host system as hexadecimal fields instead of character fields.

- Binary fields change to ASCII numeric. For example, X'FFD3' with no decimal position expands to ASCII 20202020202020202D3435. When displayed by an editor or printed, the string appears as -45.

Note: The length of the ASCII field depends on the length of the binary field.

A binary field on the iSeries™, eServer™ i5, or System i5™ is either 2 or 4 bytes long. The resulting ASCII field length is from 6 to 11 bytes, including the sign. Another byte is added for a decimal point.

Table 44: Binary-to-ASCII Field Length Mapping on page 311 shows the mapping between binary field lengths and their ASCII lengths.

Table 44. Binary-to-ASCII Field Length Mapping

Binary Length	ASCII Length	Value Range
2	6	-32768 to 32767
4	11	-2147483648 to 2147483647

- Zoned decimal fields are changed to ASCII numeric. For example, EBCDIC F0F0F9F5F2D6 with a field length that indicates two digits to the right of the decimal point expands to ASCII 20202D39352E3236. When displayed by an editor or printed, the string appears as -95.26. The resulting workstation field length is equal to the length of the system field plus 1 for the sign and 1 for the decimal point, if specified.
- Packed decimal fields change to ASCII numeric. For example, X'871D' (no decimal point) changes to ASCII 2D383731. When displayed by an editor or printed, the string appears as -871.

  Since two decimal digits are packed into 1 byte, the length of the resulting workstation field is equal to two times the length of the iSeries™, eServer™ i5, or System i5™ field, plus 1 for the decimal point (if specified). This length always includes the sign. A minus sign (-) indicates negative, and a space indicates positive.

Transferring Data from ASCII Text Files

When you transfer data from ASCII text files to system files, the data changes as follows:

- ASCII character data changes to EBCDIC character, date, time, or time-stamp data (based on the iSeries™, eServer™ i5, or System i5™ field type) on a byte-to-byte basis, or to hexadecimal data by changing 2 ASCII bytes into 1 hexadecimal byte.
- ASCII numeric data changes to iSeries™, eServer™ i5, or System i5™ binary, zoned decimal, or packed decimal data, depending on the specified data type.

The field lengths on the iSeries™, eServer™ i5, or System i5™ and the workstation are different because of the explicit way minus signs and decimal points are stored in ASCII numeric fields. Each field changes individually, to ensure that the resulting field length matches the specifications for that field. The data transfer function tries to fit the workstation data into the system field.

- For null-capable iSeries™, eServer™ i5, or System i5™ fields, null values (except date, time, and time stamp) cannot be reliably detected and are not uploaded. For variable-length iSeries™, eServer™ i5, or System i5™ fields, trailing blanks are removed and the field is converted to the variable-length format.

Errors When Transferring Data from ASCII Text Files

When you transfer data from a workstation ASCII text file to an iSeries™, eServer™ i5, or System i5™ file, the following errors can occur:

- A data field in the ASCII text file is too long for a field in the iSeries™-, eServer™ i5-, or System i5™-defined file. In this case, the data is truncated. This occurs when the description file defines the character data as longer than the field length specified for the system file.

  If the data transfers to an EBCDIC field, this error occurs only if the extra bytes are not spaces.

  If the data transfers to a hexadecimal field, this error occurs only if the extra bytes are not zeros. These extra bytes are truncated so the data fits into the specified field.
- The value of numeric data is too large for the system field. The maximum value is used. This error occurs when:
  - Numeric data in the field does not fit into the specified number of bytes for the field.
  - The decimal value of a numeric field contains more digits than were specified for the field.

  The value of the field is set to the maximum value possible for the number of bytes and digits specified by the iSeries™, eServer™ i5, or System i5™.
- Data in this field has too many decimal positions. The number is rounded. This error occurs when the number of decimal positions in the field is greater than the number of decimal positions specified on the iSeries™, eServer™ i5, or System i5™. These extra bytes are significant because the data rounds up if the first extraneous digit is 5 or greater, and rounds down if it is less than 5.
- Data in this field is incorrect or does not match the data type. This error occurs when:
  - Nonnumeric data is found in a field that the file descriptions defined as numeric. The transfer request ends to prevent transferring incorrect data to the file.
  - ASCII numeric data is found that does not match the format the file description specified. An incorrectly positioned decimal point within the field could cause this error.
  - A value other than X'30' through X'39', minus, plus, or decimal point is found. A duplicated decimal point or minus is found. The transfer request ends to prevent transferring incorrect data to the file.
- Data for this field is missing. The default values are used. This error occurs when a data field is defined, but the data is not in the file. This means that the end of the record is reached before all of the defined data is found.

The field or fields for which data has been defined but not found then fill with default values and transfer to the file. The default values are EBCDIC spaces for character fields, or zeros for numeric and hexadecimal fields.

To supply your own default values, use the default (DFT) keyword in the data description specifications (DDS) for the file.

- Extra data is found at the end of this record. The extra data is not transferred. Data found at the end of this record and not defined by the system data definitions or workstation file-description file is not transferred to the system file, because no definitions exist to define the data and how it should change.

When you transfer data from an ASCII text file to an iSeries™, eServer™ i5, or System i5™ file without using a file-description file, any extra data found past the record length specified for the file is not transferred.

BASIC Random Files

BASIC random files are the most general-purpose BASIC file type. They contain fixed-length records with:

- No delimiters between fields or records
- No end-of-file marks

Transferring Data to BASIC Random Files

When you create a BASIC random file, system data changes as follows:

- Hexadecimal fields do not change.
- Change from a system binary field depends on the field length:
    - Fields of 2 bytes, with no decimal positions to the right of the decimal point, change to 2-byte BASIC integer values. The only change is that the order of the bytes reverses.
    - Fields of 2 bytes, with decimal positions to the right of the decimal point, change to BASIC single-precision numbers.
    - Fields of 4 bytes change to BASIC double-precision numbers.
- EBCDIC character, date, time, and time-stamp fields change byte by byte and are mapped into ASCII characters as defined by the translation tables.
- Variable-length and null fields are converted to fixed lengths, and trailing blanks (for character, hexadecimal, date, time, and time-stamp data) or zeros (for binary, zoned, and packed data) are added to the maximum length of the field.
- Zoned decimal fields change into one of the following BASIC variables depending on the field length and the number of decimal positions:
    - Zoned decimal fields of 4 bytes or less with no positions to the right of the decimal point change to a BASIC integer of an equivalent value.

      A zoned decimal field of 4 bytes or less, but with a decimal point, falls into the following category.

    - Zoned decimal fields up to 7 bytes (including those that did not fall into the previous category) change to a BASIC single-precision number of an equivalent value.
    - Zoned decimal fields greater than 7 bytes change to a BASIC double-precision number of an equivalent value.
- Packed decimal fields change into one of the following BASIC variables depending on the length of the field:
    - Packed decimal fields of 2 bytes or less with no positions to the right of the decimal point change to a BASIC integer of an equivalent value.

      A packed decimal field of 2 bytes or less, but with a decimal point, falls into the following category (up to 4 bytes).
    - Packed decimal fields of up to 4 bytes (including those that did not fall into the previous category) change to a BASIC single-precision number of an equivalent value.
    - Packed decimal fields greater than 4 bytes change to a BASIC double-precision number of an equivalent value.

Note: Changes between binary, packed decimal, and zoned decimal numbers with decimal points are not equivalent to their BASIC number counterparts, because BASIC uses a binary number format that does not always change into exact decimal fractions.

Transferring Data from BASIC Random Files

When you transfer data from BASIC random files to system files, the data changes as follows:

- Hexadecimal fields transfer to the system file as unchanged hexadecimal data. The field lengths as stored on the workstation should be the same as the field lengths as stored on the system.
- ASCII character, date, time, and time-stamp data changes to EBCDIC character data byte by byte.
- For null-capable iSeries™, eServer™ i5, or System i5™ fields, null values (except date, time, and time stamp) cannot be reliably detected and are not uploaded. For variable-length iSeries™, eServer™ i5, or System i5™ fields, trailing blanks are removed and the field is converted to the variable-length format.
- Numeric fields from BASIC random files (BASIC integers, single-precision floating-point numbers, and double-precision floating-point numbers) change to system binary data, zoned decimal data in EBCDIC format, or packed decimal data in EBCDIC format.

Note: Because the change of floating-point numbers into decimal fractions is not always exact, each number automatically changes into the most precise number possible with respect to the system field length. If you want more precision, specify a larger system field size.

Errors When Transferring Data from BASIC Random Files

When you transfer data from a workstation BASIC random file to a system file, the following errors can occur:

- Data in this field is too short for the system field. The data is padded. This error occurs when the file contains character or hexadecimal data shorter than the field length specified on the system. This error can occur if the workstation field is defined as shorter than the system, or if the data in the last record of the file is too short. Character fields are padded on the right with EBCDIC spaces, and hexadecimal fields are padded with zeros.
- Data in this field is too long for the system field. The data is truncated. This error occurs when the workstation file-description file defines character or hexadecimal data as longer than the field length specified on the system.

  For character data, this error occurs only if the extra bytes are not spaces. For hexadecimal data, this error occurs only if the extra bytes are not zeros. These extra bytes are then truncated so that the data fits into the specified iSeries™, eServer™ i5, or System i5™ field.
- The value of numeric data is too large for the system field. The maximum number is used. This error occurs when:
  - Numeric data in the workstation field does not fit into the specified number of bytes for the system field.
  - The decimal value of a numeric field contains more digits than are specified for the system field.
- Data in this field has too many decimal positions. The number is rounded down to zero. In BASIC random processing, this error occurs if the value of the number is too small to fit into the specified field.

  For example, the number 0.00001 does not fit into a system zoned field specified as being 2 bytes in length and 2 decimal positions to the right of the decimal point. In this example, the resulting value is zero.
- Data for this field is missing. The default values are used. This error occurs when a data field is defined, but the data is not in the file. This means that the end of the file is reached before all of the defined data is found. For BASIC random files, this error occurs only on the last record in the file, since there are no explicit record delimiters.

  When this error occurs, the field or fields for which data is defined, but not found, are filled with default values and are transferred to the iSeries™, eServer™ i5, or System i5™ file. These default values are EBCDIC spaces for character fields and zeros for numeric fields.

  To supply your own default values, use the Default (DFT) keyword in the DDS for the file.

When you transfer data from a BASIC random file to an iSeries™, eServer™ i5, or System i5™ file, any data shorter than the record length defined for the system file is padded with EBCDIC spaces.

Because there are no record delimiters in BASIC random files, this error can occur only on the last record of the file. This probably indicates that the record length of the system file does not match the record length of the workstation file.

BASIC Sequential Files

BASIC uses BASIC sequential files for sequential processing (for example, INPUT and WRITE statements). The fields written are considered either character or numeric. Characteristics of BASIC sequential files are as follows:

- Both numeric and character fields are written as displayable characters. However, character strings are distinguished from numeric strings by the ASCII double quotation marks (X'22') that surround them.

  Therefore, character data in BASIC sequential files cannot contain ASCII double quotation marks, because they are interpreted as the end of the character string.
- Fields are delimited by ASCII commas (X'2C'). Therefore, commas are not allowed as date, time, or decimal separators.
- Each record is delimited from the next by a carriage return character (X'0D') and a line feed character (X'0A'). The end-of-file character is X'1A'.
- Records and fields are variable in length.

Transferring Data to BASIC Sequential Files

The following list describes how iSeries™, eServer™ i5, or System i5™ data created by a BASIC-sequential-file-defined data definition changes:

- Hexadecimal fields change to equivalent ASCII characters for each half-byte. Double quotation marks surround them.

  For example, X'F3' expands to ASCII 22443322 and is written to the file.
- EBCDIC character, date, time, and time-stamp fields change byte by byte and are mapped into ASCII characters as defined by the translation tables. ASCII double quotation marks are added before and after the character string.
- Null fields are represented by the absence of the field (comma comma, or by a single comma if the null field is the last field of the record).
- For null fields, successive commas in the file will result in a null value being sent to the iSeries™, eServer™ i5, or System i5™ if the field is null-capable.
- In variable-length fields, if the iSeries™, eServer™ i5, or System i5™ field is variable length, the field is converted to the iSeries™, eServer™ i5, or System i5™ variable-length format.
- Binary fields change to ASCII numeric. Leading zeros to the left of the decimal point and trailing zeros to the right of the decimal point are removed.

  For example, X'FFD3' appears as ASCII 2D3435. When displayed on an ASCII device, the string appears as -45.
- Zoned decimal fields change to ASCII numeric. Leading zeros to the left of the decimal point and trailing zeros to the right of the decimal point are removed.

  For example, EBCDIC F0F0F9F5F2D6 with a field length that indicates two digits to the right of the decimal point expands to ASCII 2D39352E3236. The string appears as -95.26 when an editor displays it or it prints.
- Packed decimal fields change to ASCII numeric. Leading zeros to the left of the decimal point and trailing zeros to the right of the decimal point are removed.

  For example, X'871F' (no decimal point) changes to ASCII 383731. The string appears as 871 when an editor displays it or it prints.

Transferring Data from BASIC Sequential Files

When you transfer data from BASIC sequential files to iSeries™, eServer™ i5, or System i5™ files, the data changes as follows:

- ASCII character, date, time, and time-stamp data changes to EBCDIC character data on a byte by byte basis and to hexadecimal by changing 2 ASCII bytes into 1 hexadecimal byte.
- ASCII numeric data translates to system binary, zoned decimal, or packed decimal data, depending on the specified data type. The lengths of the system data and the workstation data might be different because the minus signs and decimal points are stored in ASCII numeric fields, and leading and trailing spaces are stripped away.

  BASIC might create exponential numbers in these files. The data transfer function also changes these numbers.

  Each translated field is individually verified to ensure that the resulting field length matches the specifications for that field. The data transfer function tries to fit the workstation data into the system field.

Errors When Transferring Data from BASIC Sequential Files

When you transfer data from a BASIC sequential file to a iSeries™-, eServer™ i5-, or System i5™-defined file, the following errors can occur:

- Data in this field is too long for the iSeries™, eServer™ i5, or System i5™ field. The data is truncated. The file-description file defines character data as longer than the field length specified for the file.

  If the data transfers to an EBCDIC field, this error occurs only if the extra bytes are not spaces. If the data transfers to a hexadecimal field, this error occurs only if the extra bytes are not zeros. These extra bytes are truncated so that the data fits into the specified iSeries™, eServer™ i5, or System i5™ field.
- The value of numeric data is too large for the system field. The maximum value is used. This error occurs when:
  - Numeric data in the workstation field does not fit into the specified number of bytes for the system field.
  - The decimal value of a numeric field contains more digits than were specified for the system field.

  The value of the field is set to the maximum value possible for the number of bytes and digits specified by the iSeries™, eServer™ i5, or System i5™.
- Data in this field has too many decimal positions. The number is rounded. This error occurs when the number of decimal positions in the workstation field is greater than the number of decimal positions specified on the system. The extra bytes are significant, because the data is rounded up if the first extraneous digit is 5 or greater, and is rounded down if it is less than 5.
- Data in this field is incorrect or does not match the workstation data type. This error occurs when a field defined as numeric by the file description contains nonnumeric data. This could also result if a character or hexadecimal field contains a numeric field, or if a numeric (zoned, packed, or binary) field contains a character field.

When this error occurs, the transfer request ends to prevent transferring incorrect data to the system file.

- Data for this field is missing. The default values are used. This error occurs when a data field is defined, but the data is not in the file. This means that the end of the record is reached before all of the defined data is found.

  When this error occurs, the field or fields for which data has been defined, but not found, are filled with default values and transferred to the iSeries™, eServer™ i5, or System i5™ file. These default values are EBCDIC spaces for character fields, or zeros for numeric fields.

  To supply your own default values, use the default (DFT) keyword in the DDS for the file.
- Data in this field exceeds the workstation field size. The data is lost. This error occurs when extra data, not defined by the file-description file, is found at the end of a character field. The extra bytes are truncated and are not transferred to the system file.
- Extra data found at the end of the record. The extra data is not transferred. This error occurs when extra data is found at the end of the record, and has not been defined by the system data definitions or workstation file-description file. This extra data is not transferred to the system, because no definitions exist to define the data and describe how it should change.

Data Interchange Format Files

Data Interchange Format (DIF) files represent data in rows and columns. DIF files contain character and numeric data (positive and negative decimal numbers).

DIF is used for data interchange between spreadsheet programs and other application programs.

The data transfer function supports only the following two data types within DIF files:

- **Character data:** The data in a character cell (think of a *cell* as one field in one record) must be enclosed in double quotation marks if there is an embedded space in the string. However, if the string begins with a quotation mark, it must also end with a quotation mark.
- **Numeric data:** The numeric data supported by the data transfer function consists of a decimal number that can contain a minus sign or a decimal point or both. The data transfer function also supports exponential numeric data.

Transferring Data to DIF Files

When creating a DIF file, system data changes as follows:

- Hexadecimal fields change to equivalent ASCII characters for each half-byte. Double quotation marks surround them.
- EBCDIC character, date, time, and time-stamp data changes byte by byte and is mapped into ASCII characters as defined by the translation tables. ASCII double quotation marks are added before and after the character string.

- Binary fields change to ASCII numeric. Leading zeros to the left of the decimal point, and trailing zeros to the right of the decimal point, are removed.
- Zoned decimal fields change to ASCII numeric. Leading zeros to the left of the decimal point, and trailing zeros to the right of the decimal point, are removed.

  For example, EBCDIC F0F0F9F5F2D6 with a field length that indicates two digits to the right of the decimal point expands to ASCII 2D39352E3236. When displayed or printed, the string appears as -95.26.
- Packed decimal fields change to ASCII numeric. Leading zeros to the left of the decimal point, and trailing zeros to the right of the decimal point, are removed.

  For example, X'871D' (no decimal point) changes to ASCII 2D383731. When displayed or printed, the string appears as -871.
- If untranslatable data is found, the entire field becomes an error cell. An error cell results when untranslatable data is found when a DIF file is created or when a not valid calculation is done using the DIF file with a spreadsheet program.

Transferring Data from DIF Files

If an error cell is found when data is transferred from a DIF file to the iSeries™, eServer™ i5, or System i5™, one of the following things can occur, depending on the type of data in the file:

- If the system field is a character (EBCDIC) field, it is filled with untranslatable characters (hexadecimal zeros) and is transferred to the system. A message appears, telling you how many bytes of untranslatable data have transferred.
- If the system field is a hexadecimal, zoned, packed, or binary field, you receive an error message telling you that the data in this cell is incorrect, and that the data was not transferred to the system.

When you transfer data from a system file to a DIF file, the field names are placed in the first record and you can consider them column headings. When you transfer DIF files back to the system, the first row must either be these field names (exactly as they are defined on the system) or data. If the first row does not consist of field names, the file is processed as if it contains only data.

No DIF header information is used when sending the file to the iSeries™, eServer™ i5, or System i5™. To correctly transfer a DIF file to the system, ensure that the file is in the correct format (row and column). It is essential that the field names, if present, make up the first row of data. The subsequent records make up the remaining rows of data. Therefore, when you transfer the data to the iSeries™, eServer™ i5, or System i5™, the file must be saved in the same format as originally created by the data transfer function.

When you transfer data from DIF files to iSeries™, eServer™ i5, or System i5™ files, the data changes as follows:

- ASCII character, date, time, and time-stamp data is changed to EBCDIC character data or to hexadecimal data. ASCII-to-EBCDIC conversion is done byte by byte. ASCII-to-hexadecimal conversion is done by changing two ASCII bytes to one hexadecimal byte.
- ASCII numeric data changes to system binary, zoned decimal, or packed decimal data, depending on the data type the system specifies.

The lengths of the fields on the system and the workstation can be different, because of the explicit way minus signs and decimal points are stored in ASCII numeric fields. This means that each field changes individually, to ensure that the resulting field length matches the system specifications for that field. The data transfer function tries to fit the workstation data into the system field.

- In null fields, a NULL DIF character field results in a null value being sent to the iSeries™, eServer™ i5, or System i5™ field if the field is null-capable.
- If the iSeries™, eServer™ i5, or System i5™ field is variable-length, the field is converted to the iSeries™, eServer™ i5, or System i5™ variable-length format.

Errors When Transferring Data from DIF Files

When you transfer data from a workstation DIF file to a system file with data definitions, the following errors can occur:

- Data in this workstation file is not valid, or the version of this workstation file is not supported. The DIF file does not follow the standard DIF format. Processing ends, and no more records are transferred.
- Data in this field is too long for the iSeries™, eServer™ i5, or System i5™ field. The data is truncated. The workstation file-description file defines character or numeric data as longer than the field length specified for the system file.

  For character data, this error occurs only if the extra bytes are not spaces. For hexadecimal data, this error occurs only if the extra bytes are not zeros. The extra bytes are truncated so that the data fits into the specified iSeries™, eServer™ i5, or System i5™ field.
- The value of numeric data is too large for the system field. The maximum value is used. This error occurs when:
  - Numeric data in the workstation field does not fit into the specified number of bytes for the iSeries™, eServer™ i5, or System i5™ field.
  - The decimal value of a numeric field contains more digits than are specified for the system field.

  The value of the field is set to the maximum value possible for the number of bytes and digits the system specifies.
- Data in this field has too many decimal positions. The number is rounded. The number of decimal positions in the workstation field is greater than the number of decimal positions specified on the system. The data is rounded up if the first extraneous digit is 5 or greater, and is rounded down if it is less than 5.
- Data in this field is incorrect or does not match the workstation data type. One of the following things has occurred:
  - A numeric field contains nonnumeric data.
  - A character or hexadecimal field contains a numeric field or a numeric (zoned, packed, or binary) field contains a character field.
  - An iSeries™, eServer™ i5, or System i5™ hexadecimal or numeric (zoned, packed, or binary) field contains a DIF error cell.

  When this error occurs, the transfer request ends to prevent the transfer of incorrect data to the system file.

- Data for this field is missing. This occurs when a data field is defined, but the data is not in the file. This means that the end of the record is reached before all of the defined data is found. If the host field is null-capable then a null is inserted; otherwise, the default values are used.

  When this error occurs, the field or fields for which data is defined, but not found, are filled with default values and are transferred to the system file. These default values are EBCDIC spaces for character fields, or zeros for numeric fields.

  To supply your own default values, use the Default (DFT) keyword in the DDS for the file.
- Data in this field exceeds the field size. The data is lost. This error occurs when extra data, not defined by the file-description file, is found at the end of a character field. The extra bytes are truncated and are not transferred to the system file.
- Extra data is found at the end of this record. The extra data is not transferred. This error occurs when there is extra data at the end of the record, and the iSeries™, eServer™ i5, or System i5™ data definitions or file-description file have not defined it. This extra data is not transferred to the system, because no definitions exist to define the data and how it should change.

BIFF Files

The BIFF file format is used by Microsoft® Excel. In a BIFF file, data is expressed in lines and columns. A BIFF file contains character and numeric data (both positive and negative decimal values).

BIFF format versions 4 and 8 are supported for 5250 Data Transfer. Both BIFF4 and BIFF8 support 256 columns, which is the maximum for a Microsoft Excel worksheet. Documentation on both formats is freely available from the Microsoft Web site.

BIFF4 handles data for Microsoft Excel V2, V3, and V4. The format supports a maximum of 16 384 rows.

BIFF8 is a superset of BIFF4 and stores data as an OLE compound document. BIFF8 handles data for Microsoft Excel V5, V7 (Excel 95), V8 (Excel 97), and V9 (Excel 2000). The format supports a maximum of 65 536 rows.

The transfer facility supports only the following two data types for a BIFF file:

- Character data
- Numeric data

Transferring Data to BIFF Files

When a BIFF file is created, the system data is converted to equivalent Excel cell data.

If untranslatable data is found, the entire field is treated as an error cell.

Transferring Data from BIFF Files

If an error cell is found during data transfer from a BIFF file to the iSeries™, eServer™ i5, or System i5™, either of the following things can occur depending on the data type of the file:

- If the system field is a character (EBCDIC) field, the error cell containing untranslatable characters (hexadecimal zeros) is transferred to the system. A message indicating how many bytes of untranslatable data were transferred is displayed.
- If the system field is a hexadecimal, zoned decimal, packed decimal, or binary field, an error message indicating that the data in this cell is not valid and thus has not been transferred to the system is displayed.

When you transfer data from a system file to a BIFF file, the first record contains field names, which can be treated as column headers.

To return a BIFF file to the system, the first line must contain these field names (as defined in the system) or data. If the first line does not contain field names, the file is regarded as containing data only.

When a file is sent to the iSeries™, eServer™ i5, or System i5™, cell information (such as the character size and font information) is ignored. This means that cell information is lost, even if the contents of a BIFF file that have been sent to the iSeries™, eServer™ i5, or System i5™ are retransmitted to a workstation.

When you transfer data from a BIFF file to an iSeries™, eServer™ i5, or System i5™ file, the data is converted as follows:

- ASCII character cell data is converted to EBCDIC character data or hexadecimal data; 1-byte ASCII data is converted to 1-byte EBCDIC data.
- ASCII numeric cell data is converted to a binary number, or a zoned or packed decimal number, depending on the data type specified in the system.

When you transfer data from a BIFF file to the iSeries™, eServer™ i5, or System i5™, the following specific processing is performed:

- When you transfer data to a BIFF file, the first record contains the names of the fields to be transferred, which can be treated as column headers. To return a BIFF file to the iSeries™, eServer™ i5, or System i5™, the first line must contain the same field names (as defined in the iSeries™, eServer™ i5, or System i5™) or data. If the first line or the first set does not contain a character field that exactly matches the iSeries™, eServer™ i5, or System i5™ field, the file is treated as being a file with no column headers, and only data is processed.
- When you transfer a BIFF file to the iSeries™, eServer™ i5, or System i5™, header information is not used.
- To ensure correct transfer of a BIFF file to the iSeries™, eServer™ i5, or System i5™, the file format must be valid (lines and columns). Data for each set or line must correspond to one record in the iSeries™, eServer™ i5, or System i5™ file.

Errors When Transferring Data from BIFF Files

When you transfer data from a BIFF file on a workstation to the system file with the data definition, the following errors can occur:

- Data in this workstation file is not valid, or the version of this workstation file is not supported. The BIFF file does not conform to the standard BIFF format. Processing terminates, and no more records are transferred.
- Data in this field is too long for the corresponding iSeries™, eServer™ i5, or System i5™ field. The data is truncated. A file-description file defines character or numeric data that is longer than the field specified in the system file.
    - For conversion from ASCII to EBCDIC, this error occurs if a file-description file defines ASCII data that is longer than the field specified on the iSeries™, eServer™ i5, or System i5™.

      During conversion from ASCII to hexadecimal, this error will occur if a file-description file defines ASCII data that is twice as long as the field specified on the iSeries™, eServer™ i5, or System i5™. This is because 2-byte ASCII data is converted to one hexadecimal character.
    - A truncation error only occurs if excess bytes are other than blanks (X'20') during conversion from ASCII to EBCDIC, or other than zeros (X'30') during conversion from ASCII to hexadecimal. Truncating these excess bytes enables data to fit into the specified iSeries™, eServer™ i5, or System i5™ fields.
- Numeric data is too long to fit into the corresponding iSeries™, eServer™ i5, or System i5™ field. The maximum value is assumed. This error occurs under either of the following conditions:
    - Numeric data in a workstation field is too long to fit into the number of bytes specified for the iSeries™, eServer™ i5, or System i5™ field.
    - The number of decimal digits in a numeric field exceeds the number of digits specified for the iSeries™, eServer™ i5, or System i5™ field.

  The field value is set to the maximum value that can be specified for the number of bytes, and that for the number of digits, specified for the iSeries™, eServer™ i5, or System i5™.
- Data in this field contains too many decimal places. The data is rounded off. The number of decimal places in a workstation field is greater than the number of decimal places specified for the system. If the first excess digit is 5 or more, the data is rounded up. Otherwise, it is rounded down.
- Data in this field is not correct, or its type does not match the type of workstation data. One of the following things has occurred:
    - A numeric field contains other than numeric data.
    - A character field or a hexadecimal field contains a number, or a numeric (zoned or packed decimal, or binary) field contains characters.
    - A hexadecimal field or a numeric (zoned or packed decimal, or binary) field for the iSeries™, eServer™ i5, or System i5™ contains a BIFF error cell.

  If this error occurs, the transfer request terminates to avoid transferring incorrect data to the system file.
- Data for this field is missing. This error occurs if the data field is defined, but the file does not contain any data. This means that the end of the record is reached before all defined data has been found.

  If this error occurs (that is, if data is defined for one or more fields, but it is not found there) the fields containing the default value are transferred to the system file. The default value is EBCDIC spaces for a character field and zeros for a numeric field.

  To specify a user-specific default value, use the default value (DFT) keyword in DDS for the file.
- Data in this field exceeds the size of a workstation field. Data is lost. This error occurs if excess data, not defined in the workstation file-description file, is found at the end of the field. For character data, excess bytes

are truncated, and not transferred to the system file. For numeric data, the entire field is converted to zeros and transferred to the system file.

- Excess data is found at the end of this record. The excess data is not transferred. This error occurs if such excess data is not defined in the iSeries™, eServer™ i5, or System i5™ data definition or in the workstation file-description file. This excess data is not transferred to the system, because the data and the conversion method are not defined.

DOS Random Files

DOS random files are fixed-length files used by the DOS random read and write routines. The characteristics of DOS random files are as follows:

- There are no end-of-record or end-of-file markers.
- Records are delimited by their constant length, relative positions in the file, and the total length of the file.

Note: DOS random and DOS random type-2 files are identical, except for the way in which the signs are represented for packed decimal and zoned decimal numbers.

Transferring Data to DOS Random Files

When creating DOS random file data definitions, system data changes as follows:

- Binary fields on the iSeries™, eServer™ i5, or System i5™ and the workstation are represented as two-complement numbers, so it is unnecessary to change individual bytes. The workstation uses the convention of storing numeric values with the least significant byte in the left-hand byte position. The data transfer function then reverses the order of the bytes in the binary fields.

  For example, X'CEF3', coming from the system as a 2-byte binary number (representing the value -12557), appears as X'F3CE'.
- EBCDIC character, date, time, and time-stamp data changes byte by byte and is mapped into ASCII characters as defined by the translation tables.
- Variable-length and null fields are converted to fixed length, and trailing blanks (for character, hex, date, time, and time stamp) or zeros (for binary, zoned, and packed) are added to the maximum length of the field.
- Hexadecimal fields do not change.
- Packed decimal fields do not change except for the last half-byte, which contains the sign. The workstation uses X'3' to indicate a positive number and X'B' to indicate a negative number in the sign half-byte.

  For example, X'0865431F' appears as X'08654313'.
- Zoned decimal fields from the system change from EBCDIC to ASCII, as do character fields, except that the sign half-byte in the workstation changed field is X'3' to indicate a positive number and X'B' to indicate a negative number.

  For example, EBCDIC X'F0F1F2F5F2D6' appears as ASCII X'3031323532B6'.

Transferring Data from DOS Random Files

When you transfer data from DOS random files to iSeries™, eServer™ i5, or System i5™ files, the data changes as follows:

- ASCII character, date, time, and time-stamp data changes to EBCDIC character data on a byte by byte basis.
- Binary fields in the workstation file are stored in an order reversed from what the system file expects. These bytes reverse and transfer to the system file.
- Hexadecimal fields do not change. The field length on the system should be the same as the field length on the workstation.
- For packed decimal fields, only the last half-byte (the byte that contains the sign) is changed. The host system uses X'F' to indicate a positive number and X'D' to indicate a negative number for the sign half-byte.

  For example, X'08654313' appears as X'0865431F'.
- Zoned decimal fields on the workstation change from ASCII to EBCDIC , as do character fields. The last half-byte (the half-byte that contains the sign) in the workstation field is changed to X'F' to indicate a positive number and X'D' to indicate a negative number.

  For example, ASCII X'3031323532B6' appears as EBCDIC X'F0F1F2F5F2D6'.
- For null-capable iSeries™, eServer™ i5, or System i5™ fields, null values (except date, time, and time stamp) cannot be reliably detected and are not uploaded. For variable-length iSeries™, eServer™ i5, or System i5™ fields, trailing blanks are removed, and the field is converted to the variable-length format.

Errors When Transferring Data from DOS Random Files

When you transfer data from a DOS random file to an iSeries™, eServer™ i5, or System i5™ file, the following errors can occur:

- Data in this field is too short for the system field. The data is padded. This error occurs when the workstation file contains character or hexadecimal data shorter than the specified field length. It also occurs if the length of the workstation field is defined as less than the system field, or if the data in the last record of the file is too short. Character fields are padded on the right with EBCDIC spaces. Hexadecimal fields are padded on the right with zeros.
- Data in this field is too long for the system field. The data is truncated. This error occurs when the workstation file-description file defines character or hexadecimal data as longer than the field length specified for the system file.

  For character data, this error occurs only if the extra bytes are not spaces. For hexadecimal data, this error occurs only if the extra bytes are not zeros. These extra bytes are truncated so that the data fits into the specified field.
- The value of numeric data is too large for the system field. The maximum value is used. This error occurs when:

    - Numeric data in the workstation field does not fit into the specified number of bytes for the iSeries™, eServer™ i5, or System i5™ field.
    - The decimal value of a numeric field contains more digits than were specified for the iSeries™, eServer™ i5, or System i5™ field.

  The value of the field is set to the maximum value possible for the number of bytes and digits specified by the system.
- Data in this field has too many decimal positions. The number is rounded. This occurs when the number of decimal positions in the workstation field is greater than the number of decimal positions specified on the system. The extra bytes are significant, because the data rounds up if the first extraneous digit is 5 or greater, and rounds down if it is less than 5.
- Data in this field is incorrect or does not match the workstation data type. This error occurs when nonnumeric data appears in a field defined as numeric by the file descriptions. When this occurs, the transfer request ends to prevent transferring incorrect data to the system file.
- Data for this field is missing. The default values are used. This error occurs when a data field is defined, but the data is not in the file. This means that the end of the file is reached before all the defined data is found.

  When this error occurs, the field or fields for which data has been defined, but not found, fill with default values and transfer to the system file. Default values are EBCDIC spaces for character fields, or zeros for numeric fields.

  To supply your own default values, use the default (DFT) keyword in the DDS for the file.

When you transfer data from a DOS random file to a system file without data definitions, any data shorter than the record length defined for the system file is padded with EBCDIC spaces.

Because DOS random files have no record delimiters, this error occurs only on the last record and probably indicates that the record length of the system file does not match that of the workstation file.

DOS Random Type-2 Files

DOS random type-2 files are fixed-length files used by the DOS random read and write routines. The characteristics of DOS random type-2 files are as follows:

- There are no end-of-record or end-of-file markers.
- Records are delimited by their constant length, relative positions in the file, and the total length of the file.

Note: This workstation file type is identical to the DOS random file type, except that the internal sign representation for packed decimal and zoned decimal data types follow Systems Application Architecture® (SAA®) standards. Some workstation applications, such as applications written in IBM® COBOL/2™

programming language, need to have the signs for packed decimal and zoned decimal data types represented this way. Use the DOS random type-2 file type for those workstation applications.

Transferring Data to DOS Random Type-2 Files

When you create DOS random type-2 file data definitions, system data changes as follows:

- Binary fields on the iSeries™, eServer™ i5, or System i5™ and the workstation are represented as two complement numbers, so it is unnecessary to change individual bytes. The workstation uses the convention of storing numeric values with the least significant byte in the left-hand byte position. The data transfer function then reverses the order of the bytes in binary fields.

  For example, X'CEF3', coming from the system as a 2-byte binary number (representing the value -12557), appears as X'F3CE'.
- EBCDIC character, date, time, and time-stamp fields change byte by byte and are mapped into ASCII characters as defined by the translation tables.
- Variable-length and null fields are converted to fixed length, and trailing blanks (for character, hex, date, time, and time stamp) or zeros (for binary, zoned, and packed) are added to the maximum length of the field.
- Hexadecimal fields do not change.
- Packed decimal fields do not change. The sign convention used on the workstation and on the host system is the same.

  For example, X'0865431C' appears as X'0865431C'.
- Zoned decimal fields from the system change from EBCDIC to ASCII, as do character fields. However, the sign half-byte is changed to a 3 to indicate a positive number or a 7 to indicate a negative number when the data is sent to the workstation.

  For example, EBCDIC X'F0F1F2F5F2D6' appears as ASCII X'303132353276'.

Transferring Data from DOS Random Type-2 Files

When you transfer data from DOS random type-2 files to iSeries™, eServer™ i5, or System i5™ files, the data changes as follows:

- ASCII character data, date, time, and time stamp data change to EBCDIC character data on a byte by byte basis.
- Binary fields in the workstation file are stored in an order reversed from what the system file expects. These bytes reverse and transfer to the system file.
- Hexadecimal fields do not change. The field length on the system should be the same as the field length on the workstation.
- For packed decimal fields, the last half-byte (the half-byte that contains the sign) is not changed unless the sign half-byte is less than X'A' (represented by values 0 through 9). If the sign half-byte is less than X'A', it is changed to X'F' on the host system.

  For example, X'865431D' appears as X'0865431D', but X'08654318' appears as X'0865431F'.

- Zoned decimal fields on the workstation change from ASCII to EBCDIC, as do character fields. However, the sign half-byte is changed to an F to indicate a positive number or a D to indicate a negative number when the data is sent to the host system.

  For example, ASCII X'303132353276' appears as EBCDIC X'F0F1F2F5F2D6'.
- For null-capable iSeries™, eServer™ i5, or System i5™ fields, null values (except date, time, and time stamp) cannot be reliably detected and are not uploaded. For variable-length iSeries™, eServer™ i5, or System i5™ fields, trailing blanks are removed and the field is converted to the variable-length format.

Errors When Transferring Data from DOS Random Type-2 Files

When you transfer data from a DOS random type-2 file to an iSeries™, eServer™ i5, or System i5™ file, the following errors can occur:

- Data in this field is too short for the system field. The data is padded. This error occurs when the workstation file contains character or hexadecimal data shorter than the specified field length. It also occurs if the length of the workstation field is defined as less than the system field, or if the data in the last record of the file is too short. Character fields are padded on the right with EBCDIC spaces. Hexadecimal fields are padded on the right with zeros.
- Data in this field is too long for the system field. The data is truncated. This error occurs when the workstation file-description file defines character or hexadecimal data as longer than the field length specified for the system file.

  For character data, this error occurs only if the extra bytes are not spaces. For hexadecimal data, this error occurs only if the extra bytes are not zeros. These extra bytes are truncated so that the data fits into the specified field.
- The value of numeric data is too large for the system field. The maximum value is used. This error occurs when:
  - Numeric data in the workstation field does not fit into the specified number of bytes for the iSeries™, eServer™ i5, or System i5™ field.
  - The decimal value of a numeric field contains more digits than were specified for the iSeries™, eServer™ i5, or System i5™ field.

  The value of the field is set to the maximum value possible for the number of bytes and digits specified by the system.
- Data in this field has too many decimal positions. The number is rounded. This occurs when the number of decimal positions in the workstation field is greater than the number of decimal positions specified on the system. The extra bytes are significant, since the data rounds up if the first extraneous digit is 5 or greater, and rounds down if it is less than 5.
- Data in this field is incorrect or does not match the workstation data type. This error occurs when nonnumeric data appears in a field defined as numeric by the file descriptions. When this occurs, the transfer request ends to prevent transferring incorrect data to the system file.
- Data for this field is missing. The default values are used. This error occurs when a data field is defined, but the data is not in the file. This means that the end of the file is reached before all the defined data is found.

When this error occurs, the field or fields for which data has been defined, but not found, fill with default values and transfer to the system file. Default values are EBCDIC spaces for character fields, or zeros for numeric fields.

To supply your own default values, use the default (DFT) keyword in the DDS for the file.

When you transfer data from a DOS random type-2 file to a system file without data definitions, any data shorter than the record length defined for the system file is padded with EBCDIC spaces.

Because DOS random type-2 files have no record delimiters, this error occurs only on the last record and probably indicates that the record length of the system file does not match that of the workstation file.

No-Conversion Files

No-conversion files, defined by the data transfer function, consist of data that has not changed. For example, when data transfers from the system to a workstation no-conversion file, the data transfers exactly as it is stored on the iSeries™, eServer™ i5, or System i5™. Date, time, and time-stamp data transfers to EBCDIC character data on the workstation.

Transferring Data to No-Conversion Files

When you transfer data from the iSeries™, eServer™ i5, or System i5™ to a no-conversion file, the data transfers exactly as it is stored on the system.

Variable-length iSeries™, eServer™ i5, or System i5™ fields are converted to fixed-length fields, and trailing EBCDIC blanks are added to the maximum length of the field.

Date, time, and time-stamp data is converted to EBCDIC character data.

Variable-length and null fields are converted to fixed length, and trailing EBCDIC blanks (for character, hex, date, time, and time stamp) or EBCDIC zeros (for binary, zoned, and packed) are added to the maximum length of the field.

Transferring Data from No-Conversion Files

The data types that exist in a no-conversion file are EBCDIC system data types only. When a no-conversion file transfers to the system, the data transfer function performs no data change or translation. Date, time, and time-stamp data transfers to EBCDIC character data on the workstation.

However, the data transfer function verifies that all numeric data is in the correct EBCDIC format. If any numeric data is found that is not in the correct EBCDIC format, that data and any remaining data does not transfer.

Errors When Transferring Data from No-Conversion Files

When you transfer data from a workstation no-conversion file to a system file, the following errors can occur:

- Data sizes are not equal. When you transfer no-conversion files, the length and decimal position specifications for the system and the workstation must match exactly. If not, no records transfer.
- Data in this field is too short for system field. The data is padded. This error occurs when the workstation file contains character or hexadecimal data shorter than the field length specified for the system file. This could occur if the data in the last record of the file is too short. Character fields are padded on the right with EBCDIC spaces. Hexadecimal fields are padded with zeros.
- Data in this field is incorrect or does not match the workstation data type. The transfer request ends to prevent transferring incorrect data to the system file. This error occurs when a field defined by the file descriptions as numeric contains nonnumeric data.

  Note: The data is verified assuming that the data is in EBCDIC format. If you want to transfer data in another format, do not use data definitions or file descriptions, and specify the record lengths defined on the system and the workstation in the same way.

- Data for this field is missing. The default values are used. This error occurs when a data field has been defined, but the data is not in the file. This error can occur only in the last record of the file, since no-conversion files have no explicit record delimiters.

  When this error occurs, the field or fields for which data has been defined but not found fill with default values and transfer to the system file. These default values are EBCDIC spaces for character fields, or zeros for numeric fields.

  To supply your own default values, use the default (DFT) keyword in the DDS for the file.

iSeries, eServer i5, or System i5 System-to-PC Performance Considerations

Transferring data from the iSeries™, eServer™ i5, or System i5™ to the workstation depends on the following performance considerations:

- The system workload.
- How many records have to be looked at to complete the transfer.
- If more than two files are joined. You need extra iSeries™, eServer™ i5, or System i5™ resources to join records from more than one file.
- If **GROUP BY** fields are specified.
- If complicated **WHERE** or **HAVING** comparisons are specified.

These factors and others influence the time needed to determine which data should be transferred. For example, the time needed to receive the first record of a transfer in which all the records are chosen is less than the time needed to start transferring a smaller group of records based on complicated **WHERE** or **HAVING** values. However, transferring all the records in a large file is sometimes impractical or unnecessary.

The iSeries-to-workstation data transfer function uses many functions within the iSeries™, eServer™ i5, or System i5™ to determine the fastest method of selectively retrieving records. When it selects a smaller group of records to transfer, the iSeries-to-workstation data transfer function uses the existing access paths whenever possible to improve performance.

For the iSeries-to-workstation data transfer function to consider using an existing access path (logical file), the access path must meet the following conditions:

- It must be defined to the data that transfers.
- It must have either *DELAY or *IMMED maintenance.

When you meet these conditions, you must then match the transfer request to the access path. The following considerations might be helpful when you define your transfer request:

- The time it takes to select records based on **WHERE** clause values is less when the following things are true of the **WHERE** field:
  - It is compared with a constant.
  - It is the first key field in an existing access path defined to the data to be transferred.
- A transfer request containing a **GROUP BY** or **ORDER BY** clause or both can work better if the key fields in the access path are in the same order as specified on the **GROUP BY** or **ORDER BY** clauses.
- A transfer request containing a **JOIN BY** clause can work better when:
  - An access path exists over the file that you are joining to.
  - The field you are joining to is a primary key field in the access path.
  - You are not returning records with missing fields.

Transferring Files

Z and I Emulator for Windows File Transfer enables you to transfer one or more files between a host system and workstation at the same time. Transfer types and translation tables can be defined in advance.

You can perform the following file transfer functions:

- Send files to the host system
- Receive files from the host system
- Use lists of files
- Create templates to define file names and transfer types
- Define transfer types
- Set transfer options
- Modify translation tables
- Transfer files via the XMODEM or YMODEM protocols

Note:

PCT400 was withdrawn from marketing 3/98.

Host Requirements

For PC400 File Transfer in SBCS mode, you need one of the following host file-transfer programs (referred to as APVAFILE):

- Z and I Emulator for Windows Tools/400 8mm Tape — 46H8350
- Z and I Emulator for Windows Tools/400 1/2 inch Tape — 85G9973
- Z and I Emulator for Windows Tools/400 1/4 inch Tape — 85G9969

Sending Files to the Host System

To send a file from your workstation to the host system:

1. Sign on to the host system.
2. Click **Send File to Host** from the **Actions** menu of the session window. (You can also select the **Send** button on the tool bar.)

   The Send File to Host window opens.
3. Specify the name of the workstation file to be sent to the host system by entering the name in the **PC File** text box, or click the **Browse** button to open a dialog box for selecting the file.
4. Enter the name under which the file will be stored on the host; then enter or select the **Transfer Type**. If a template is provided for the file type you are transferring, the host file name and the transfer type appear automatically.

   Using List Files: Select **Open List**; then select the list to be used for transfer. See Creating List Files on page 333 for details of how to create list files.
5. Click **Send**.

   The file is sent to the host system. The send status appears in the Send a File Status window.

Receiving Files from the Host System

To transfer a file from the host system to your workstation:

1. Sign on to the host system.
2. Click **Receive File from Host** from the **Actions** menu. (You can also select the **Receive** button from the tool bar.)

   The Receive File from Host window opens.
3. Specify the name of the host file to be received. Enter the name in the **Host File** text box, or specify it as follows:

> **Using the Clipboard button:** If you have copied one or more host file names to the clipboard, you can paste the names into the transfer list; click the **Clipboard** button to open a dialog box for this. Select one or more of the pasted file names to be transferred. Then click **OK**.

4. Enter or modify the suggested name under which the file will be stored on the workstation, and enter or select the **Transfer Type**; or click the **Browse** button to open a dialog box for selecting a location for the file.

   > **Using List Files:** Select **Open List**, and select the list to be used for transfer. (See Creating List Files on page 333 for an explanation of how to create list files.)
   >
   > If a template is provided for the file type you are transferring, the workstation file name and the generated transfer type appear automatically.

5. Click **Receive**.

   The receive status appears in the **Receive a File Status** window.

Using List Files

If the same files are transmitted frequently, you can create a list of the files and save it.

A list file can be used for both Send and Receive. The default list file extension is .SRL.

Creating List Files

To create a list file:

1. Click **Receive File from Host** from the **Actions** menu or **Send File to Host** from the **Actions** menu of the session window; or click the **Send** or **Receive** buttons on the tool bar.

   The corresponding window opens.
2. Select a file to be transferred from the **Host-File Name** or **PC-File Name** list box by pointing to the name of a file to be selected. While holding down the Ctrl key, click the left mouse button.

   The file name, its corresponding workstation or host file name (according to the available templates), and the transfer type appear in the **Transfer List** part of the window.

   > **Note:** You can also click the **Browse** button (for sending files) or the **Clipboard** button (for receiving files) to open the corresponding dialog box, which allows you to select files for transferring; when you click **OK**, the selected files are shown in the **Transfer List**.

3. Click the **Add to List** button to include a selected file in the **Transfer List**.
4. After all desired files have been selected, click **Save List**.

   The Save File-Transfer List File As window opens.
5. Enter or select a list name, and click **OK**.

Editing Lists

To edit the contents of a previously created list:

1. As explained in Sending Files to the Host System on page 332 and Receiving Files from the Host System on page 332, display the Send File to Host or Receive File from Host window.
2. Select **Open List**.

   The Open File-Transfer List File window opens.
3. Select the name corresponding to the list file to be edited, then click **OK**.
4. The contents of the selected list appear in the Send File to Host or Receive File from Host window.
5. Edit the contents of the list file.

   Changing the contents of a list: Choose the file to be changed from the list, and overwrite the items to be changed in the text box; then click the **Update in List** button.

   Removing a file from the list: Choose the file to be removed, and click **Remove from List**.

   Adding a file to the list: Double-click the file to be added from the list of host or workstation files.
6. Select **Save List**.

   The Save File-Transfer List File As window opens.
7. Enter a name and then click **OK**.

Managing Templates

A *template* is a set of rules to be used by the workstation to automatically generate a workstation or host file name and transfer type when you specify a file to be sent or received.

You can have up to 32 templates. They are automatically numbered from 1 to 32.

When you specify a file to be transferred, the workstation scans the templates, starting from template 1. It uses the first matching template to generate a name for the transferred file and the transfer type.

To manage a template:

1. Click **Receive File from Host** from the **Actions** menu or **Send File to Host** from the **Actions** menu of the session window; or click the **Send** or **Receive** buttons on the tool bar.

   The Send File to Host or Receive File from Host window opens.
2. Select **Template**.

   The Template window opens. The contents of the window depend on the connected host system.

Adding Templates

The list box for the Template window lists the currently stored templates.

To add a template:

1. Select any template from the list box.

   The contents of the selected template appear under the list box.
2. Change the workstation or host file names or extensions by overwriting them; then select the transfer type. (For details of the transfer types, see Defining Transfer Types on page 336.)
3. Click **Add**.

   The window for determining where in the list to display the new template opens.
4. Select a template number and specify whether to display the new template before or after the template that has that number. Click **OK**.

   The new template is added to the list in the appropriate position.

Replacing and Deleting Templates

To change the contents of a currently stored template, or to delete a template:

1. Select the template to be changed or deleted.

   The contents of the selected template appear under the list box.
2. To change the contents, overwrite the appropriate part and then click **Replace**.

   To delete a template, click **Delete**.

   The selected template is changed or deleted, and the contents of the template list box are changed.

Testing Templates

To test the contents of an added or changed template:

1. Select the template to be tested from the list box.

   The number of the selected template appears in the Test Templates box in the lower part of the window.
2. Select or enter data for the following items:

   Test Mode

   Determine which mode is to be used for the test: the mode in which a file is transmitted from the workstation to the host system (send), or the mode in which a file is transmitted from the host system to the workstation (receive).

   Templates

   Determine which templates to test: only the template selected in step 1, or all registered templates.

Source File

Enter the name of the file to be used for the test.

3. Click **Test**.

   Target File indicates the name that has been generated by the template.

Note: Testing a template does not transfer a file.

Defining Transfer Types

Transfer types define the option information used for controlling file transfer. Up to 32 transfer types can be defined for each host system. Text, binary, and append (excluding CICS®) are the defaults.

To add or change transfer types:

1. Click **Preferences → Transfer** from the **Edit** menu of the session window.
2. Click the tab for your host type or modem protocol.

   The property page for the selected host or modem protocol opens. The items that appear depend on the selected host system.
3. Enter transfer-type names in the **Transfer Type** box, or select them from the drop-down list.
4. Select or enter the required items (see Items to Be Specified on page 336).

   To add or replace a transfer type, click **Save**. To delete a transfer type, click **Delete**.
5. A dialog box displays, asking for confirmation. Click **OK**.

Items to Be Specified

Choosing the appropriate property page enables you to set the items described in the following sections.

File Options

The file options that can be used depend on the type of the connected host system and the host code page selected when the session was configured. Table 45: Mode Values for File Transfer Options on page 336 lists the mode values for the file transfer options. File Transfer for PC400 on page 340 lists transfer options.

Table 45. Mode Values for File Transfer Options

Mode	Host Code Page
SBCS	Others

Logical Record Length (LRECL)

Enter the **logical record length** to be used (host record byte count) in the **LRECL** text box. If **Variable** and **Undefined Mode** are specified as the record format, the logical record length is the maximum record length within a file. The maximum value is 32767.

The record length of a file sent from a workstation to the host system might exceed the logical record length specified here. If so, the host file transfer program divides the file by the logical record length.

To send a file containing long records to the host system, specify a sufficiently long logical record length.

Because the record length of a workstation file exceeds the logical record length, a message does not appear normally if each record is divided. To display a message, add the following specification to the `[Transfer]` item of the workstation profile:

```
DisplayTruncateMessage = Y
```

Additional Options

The required host command options can be entered in the **Additional Options** text box.

Setting General Transfer Options

To set advanced options:

1. Select **Preferences → Transfer** from the **Edit** menu of the session window.

   The setup dialog is displayed.
2. Change the required settings on the property page labeled **General**.
3. Click **OK**.

The following sections contain information about the items which can be defined for file transfer options.

Data Transfer

You can choose whether the Data Transfer function (see Data Transfer for PC400 on page 246) is to be used instead of the normal file transfer function.

Host Command

You can specify host command to be called when file transfer starts. If nothing is entered in this text box, APVAFILE is used for 5250 sessions.

Default PC Directory

You can specify the default directory that appears in the Send File to Host or Receive File From Host window. To select the directory, click the **Browse** button.

Default Library

You can specify the iSeries™, eServer™ i5, or System i5™ library to be used as the default.

PC Code Page

When a file is transferred, EBCDIC codes are converted to 1-byte workstation codes, and vice versa. A valid value is automatically selected from among the following values for SBCS sessions: 437, 737, 806, 813, 819, 833, 850, 852, 854, 857, 858, 860, 861, 862, 863, 864, 865, 866, 869, 874, 912, 915, 916, 920, 921, 922, 1008, 1089, 1124, 1125, 1127, 1129, 1131, 1133, 1153, 1155, 1156, 1157, 1158, 1160, 1164, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, and 1258; —according to the host code page specified when the workstation is configured. For an explanation of how to select host code pages, see the online help for the host code page.

File-Transfer Timeout

You can define the time the workstation waits for a response from the host system (in seconds). If the host system does not respond, the transfer is canceled, and an error message appears. A number in the range 20–65535 (or 0) can be specified. The default timeout is 30 seconds. Specify an appropriate value such that the error message does not appear too early. If you specify 0, a timeout is not set.

If a packet or block size is relatively large for low-speed lines, such as or COM port lines, it is recommended that 150 seconds or greater be specified.

Extension for List-Files

You can change the default extension (.SRL) of file-transfer list files.

Show Status Window

You can choose the method of displaying the file-transfer-progress status.

In Session

When file transfer starts, the status window opens. The name of the file being transferred and the transfer progress appear.

In icon

When file transfer starts, the status icon appears on the screen. If the icon is restored, the status window opens.

Setting Up the Translation Table

You can create or edit the translation table to be used for sending or receiving files.

Changing the Translation Table

To change the translation table:

1. Select **Preferences → Transfer** from the **Edit** menu of the session window.
2. Click the **Translation Table** tab on the resulting window.

   The Translation-Table Setup property page opens.
3. The table currently being used (default or the name of a user-defined table) is shown. Choose either **Default** or **User-Defined**.
4. If you choose **User-Defined**, enter a translation-table name in the **File Name** text box, or select a name by clicking **Browse**.
5. Click **OK**.

Customizing the Translation Table

You can create a user-specific translation table for transmission or reception, or you can edit an existing translation table.

To create or edit a translation table:

1. On the **Translation Tables** property page, click **Customize** in the Upload or Download window.

   The Customize Translation window opens.

   If you chose **Default** or if you chose **New** from the File menu, the default values appear in the table.

   Translation source codes

   PC code-points when an upload translation table is edited. Host code-points when a download translation table is edited.

   Translation target codes

   Host code-points when an upload translation table is edited. PC code-points when a download translation table is edited.
2. Double-click the code to be changed in the table, and change the value in the entry field that subsequently appears.
3. Click **Save** or **Save As** from the File menu.

4. If asked, enter a name in the Save Translation File As window and click **OK**.
5. Click **Exit** from the File menu of the Customize Translation window.

File Transfer for PC400

File transfer is designed so that you can use it in the following cases:

- To store a workstation file on the iSeries™, eServer™ i5, or System i5™ for a backup
- To edit a source file of an iSeries™, eServer™ i5, or System i5™ program with a workstation editor, and send the file edited on the workstation to the iSeries™, eServer™ i5, or System i5™.
- To distribute workstation documents and programs to the iSeries™, eServer™ i5, or System i5™ users

PC File Transfer with the CRLF Option

If the CRLF option is specified, the transfer program checks for new-line characters. If the record length is reached before a new-line character is found, the record is divided at this point; one sentence of a workstation file will become two or more records. Particularly, specify a sufficiently long record length when retransmitting a workstation file containing 2-byte characters.

By default, the message `records segmented.` is not displayed. To display the message, do the following:

1. Look for the profile for the session you will use. Normally, this will be in the application data directory under the name *filename*.WS (*filename* is a user-specified file name).
2. Use an editor to insert the following sentence into the [Transfer] section. If there is no [Transfer] section, first enter [Transfer]. Be careful to enter it correctly.

```
[Transfer]
DisplayTruncateMessage=Y
```

The next time the session is started, this specification becomes active.

Transfer to a Physical Source File

An iSeries™, eServer™ i5, or System i5™ physical source file contains 12 bytes of information for each record as internal information: 6 bytes are for a record number, the other 6 bytes are for a date. When you transfer a file from a workstation using file transfer, the date field contains 000000. If the APPEND option is not specified, the record number is incremented by 1, up to a maximum of 9999. Otherwise, it is incremented from the nearest integer, greater than the number of the last record in the original file (for example, 24 for 23.1). If the number of records exceeds 9999, the next and all subsequent record numbers are 9999.

Use the source specifications input utility (SIU) to renumber records when saving the file after editing.

Transfer to a Physical File

A file, such as a PC program, that does not require the processing of the contents of an iSeries™, eServer™ i5, or System i5™ file or the reading of data, should be transferred to a physical file with the BINARY transfer type. Because data is not converted, if the data is subsequently retransmitted from the iSeries™, eServer™ i5, or System i5™ to a workstation, the original workstation file can be re-created exactly. If the data is converted, however, data might not be restored to its original form, depending on the contents of the conversion table.

For the maximum number of members (MAXMBRS), a physical file attribute, the default value is 1. When a physical file is created during file transfer, MAXMBRS is 1.

When a file is transferred from a workstation to a physical file, the default file name **xxxBIN** is assumed (**xxx** is a workstation file extension.) If you transfer more than one file, an error occurs when the second and subsequent files are transferred: `The TRANS58 file or member cannot be created. File transfer terminates`. A file should be created with the expected file attribute before it is transferred from a workstation to the iSeries™, eServer™ i5, or System i5™.

Using the DSPMBRLST Command

For file transfer from the iSeries™, eServer™ i5, or System i5™ to a workstation, the Paste function can be used. If the name of the Library/File(Member) to be transferred is copied with the Copy function of the Edit menu, it can be displayed as the host file candidate to be transferred on the transfer request screen by clicking **Paste**. This is particularly convenient when transferring more than one file at a time.

Use the DSPMBRLST command to list iSeries™, eServer™ i5, or System i5™ files or members. The command format is as follows:

```
DSPMBRLST LIB(lib-name) FILE(file-name)
```

LIB parameter

The LIB parameter contains the target library name. The default value is *USRLIBL. Extensive specification, such as *ALL, ***** for generic name, is possible, but is time-consuming. iSeries™, eServer™ i5, or System i5™ files or members are listed more efficiently if a specific name is specified.

FILE parameter

The FILE parameter contains the target file name. There is no default value. The parameter must be specified. *ALL and ***** for generic name can be specified.

Executing this command lists Library/File(Member) on the screen. If they cannot be listed on one screen, **MORE...** is displayed in the lower right corner of the screen. Use the next page or the preceding page key to scroll the screen. Create a list for Paste with the Copy or the CopyAppend function of the Edit menu, as required.

Restrictions for Transferred File Size

A file that is more than 1 040 000 bytes cannot be transferred correctly.

Using Z and I Emulator for Windows VT

VT Emulation

For connection to ASCII hosts, Z and I Emulator for Windows provides the **VT Emulator** for VT340, VT100, and VT52 terminals. ASCII hosts commonly use these terminal control sequences as standards for session presentation, and many ASCII-host application programs assume a VT-compatible terminal. VT emulation allows your personal computer or workstation to operate as if it were a VT terminal. Software that is designed to operate a VT340, VT100, or VT52 terminal should work correctly with the Z and I Emulator for Windows VT emulator.

Although the keyboard layout on VT terminals is similar to that of the personal computer, there are some exceptions. See Default Key Functions for the VT Emulator Layout on page 189 for the default mapping of keys for VT emulation.

For file transfer to and from ASCII hosts, using the XMODEM and YMODEM protocols, see Using XMODEM and YMODEM on page 355.

VT connections to non-ASCII hosts, such as the IBM® zSeries™ are also possible if you have the appropriate communication devices.

Configuring a VT Session

Use the **Customize Communication → ASCII Host** panel to select values for the parameters that define your ASCII host session. There are two types of parameters: Session and Link.

Customizing the VT over Telnet Attachment

1. Click **Communication** from the WorkStation-window menu bar.
2. Click **Configure** from the **Communication** menu.

   The Customize Communication window opens.
3. Select the ASCII host and then select the **LAN** or **COM Port** interface.

   The available attachments appear.
4. Select **VT over Telnet** attachment.
5. Click **Session Parameters**.

   The Session Parameters — ASCII Host window opens.
6. Set the Session Parameters (see Session Parameters on page 343).
7. Click **Link Parameters**.

   The TelnetASCII window opens.
8. Enter the host name or IP address.
9. Optionally enter the port number, change the terminal ID, or select the **Auto-reconnect** check box.
10. Click **OK** until the Customize Communication window closes.

    Customization is complete.

Session Parameters

These parameters correspond to setup choices on a VT340 terminal.

Online/Local

In the **Online** state, the emulator receives data from the host computer, and can send data to it. In the **Local** state, data you enter on the keyboard appears on the screen, but is not sent to the host; data from the host is held, and not presented on the screen until you change the state to **Online**.

Operating Mode

Select **Char** if the host does not echo the characters you type on your keyboard. The VT emulator displays them as it sends them to the host.

Select **Echo** if the host echos your keyboard characters for display. The VT emulator displays them only as they return from the host.

If you see doubled characters, you should select **Echo** instead of **Char**. **Echo** is the default.

Machine Mode

There are four machine modes. These are:

VT340 mode, with 7-bit controls

This is the default. This mode is recommended for most applications.

VT340 mode, with 8-bit controls

The emulator is set for an 8-bit environment with 8-bit controls.

VT100 mode

This mode is intended for situations requiring strict compatibility with the VT100 terminal. In general, the VT340 7-bit mode is appropriate for applications that expect a VT100.

VT52 mode

This mode is only for applications designed for the VT52 terminal.

Screen Size

You can choose the number of rows and columns that the session screen displays. The choices are

- Rows: 24, 36, 48, 72, and 144
- Columns: 80 and 132

The defaults are 24 rows and 80 columns.

Type of Host Code-Page

The choices are for the host code page are National, PC, and Multinational. Multinational which selects the 8–bit DEC Supplemental Graphic Character Set is the default. If you select National, then you must select a country from the Host Code Page pull-down list. The PC option selects the PC Code Page 437.

ISO Latin 9 (ISO 8859-15) character set support for ASCII (VT) sessions

Support of the ISO Latin 9 (ISO 8859-15) character set is available for ASCII (VT) sessions.

Optional Parameters

These parameters correspond to setup choices on a VT340 terminal.

Reverse Screen Image

Check this box to reverse the foreground and background colors.

User Feature Lock

Check this box to lock the following functions so that the host cannot change them.

- Auto Repeat
- Keyboard Lock
- Reversed Screen Image
- Tab Stops

Auto Wrap

Check this box if you want the VT emulator to start a new line whenever the current row of characters reaches the end of line.

Auto-Answer Back Message

Check this box if you want the VT emulator to send a message automatically to the host, once a connection has been established.

Move Cursor on Mouse Click

Select this option if you want the cursor to move when you click the left mouse button in the session window presentation space.

Answer Back Message

Enter the message, which is a maximum of 31 characters, to send to the host when communication is established.

Conceal

If you check this box, your answerback message is not displayed in the configuration window. After you conceal your message, the **Conceal** box has no effect, and the message remains concealed until it is changed.

User Defined Key Lock

Check this box to lock user-defined keys. For example, you can select **User Defined Key Lock** and define the values of the F6 to F20 keys. These keys are then locked with those values and cannot be redefined by the host.

Transparent Mode

Check this box to cause the VT emulator to display control characters rather than interpreting them.

VT ID

The attributes of the selected model are sent to the host computer. Choose one of the following: **VT100 ID**, **VT101 ID**, **VT102 ID**, **VT220 ID**, **VT240 ID**, **VT320 ID**, or **VT340 ID**.

History Logging

When this option is enabled, text is logged into the VT history window as it scrolls off the screen from the top margin row. The top and bottom margins are set when the host application defines the scrolling region.

History Logging Buffer Size

Use this list to select one of the available sizes for the history log buffer. The choices are 16KB, 32KB, 64KB, 128KB, and 512KB. The default is 64KB.

History Logging – Enhanced

Data erased due to the Erase in Display command is scrolled into the history window. See Enhanced History Logging on page 355 for more information.

Advanced ASCII Host

The **Advanced** button takes you to the Advanced ASCII Host dialog. The Advanced Options dialog contains all of the configuration options needed for the Local editing feature of VT340 Emulation. The following list defines these configuration options. Default settings are indicated in bold.

Graphics Cursor

Determines whether the graphics input cursor is shown when in graphics mode. Possible values are **Enabled** or Disabled.

Sixel Scrolling

When this option is selected, a sixel graphics image scrolls to the next row when the last column is reached. Possible values are **Enabled** or Disabled.

MacroGraph Reports

Controls the ability of the host to retrieve stored macro graph procedures. Possible values are Enabled or **Disabled**.

Edit Mode

Selects whether local editing is available and the current mode of operation. Possible values are **Unavailable**, Interactive, or Edit.

Erasure Mode

Determines which characters can be erased in edit mode. Possible values are **Unprotected** or All.

Edit Key

Determines how the VT340 emulation switches between interactive and edit mode. Possible values are **Immediate** or Deferred.

Transmit

Determines how the VT340 emulation sends a block of data to the host system in edit mode. Possible values are **Immediate** or Deferred.

Application Keys

Determines how the unshifted function keys F6 through F20 work in edit mode. Possible values are **Disabled**, Immediate, Prefix, or Suffix.

Guarded Area

Determines whether protected characters can be sent to the host system. Possible values are **All** or Selected.

Selected Area

Determines whether the VT340 emulation can send all characters or only selected characters to the host system. Possible values are **All** or Selected.

Multiple Area

Determines whether VT340 emulation can send all selected areas on the page, or only the area selected with the cursor. Possible values are **Multiple** or Single.

VT131 Transfer

When **Line Transmit Mode** is disabled, this feature selects an ANSI-style or VT131-style data transmission. Possible values are **ANSI** or VT131. The size of the block depends on the **Transfer Termination Mode** value.

EOL Characters

Allows you to select characters used to indicate the end of a line (EOL) in a data block. By default the VT340 emulation sends a carriage return (CR). Up to six hexadecimal characters can be specified.

EOB Characters

Allows you to select characters used to indicate the end of a data block (EOB). This feature has no default. Up to six hexadecimal characters can be specified.

Page Coupling

Determines whether to automatically display a new page when the cursor moves to a new page in page memory. Possible values are **Enabled** or Disabled.

Line Transmit Mode

Allows you to send characters one line at a time to the host system. Possible values are **Disabled** or Enabled.

Transfer Termination Mode

When **Line Transmit Mode** is disabled, this feature determines whether the VT340 emulation sends a partial page or the scrolling region. Possible values are **Enabled** or Disabled.

Space Compression Mode

Determines how the VT340 emulation sends unused character fields and spaces in a data block. Possible values are **Disabled** or Enabled.

Link Parameters

The **Configure Links** button take you to a panel for configuring the details of the connection to the ASCII host computer. The panel you see depends upon the attachment type that you chose for your ASCII host. There are two types:

- VT over Telnet

Configuring Links for VT over Telnet

The VT over Telnet attachment is an application that uses TCP/IP (Transmission Control Protocol/Internet Protocol) and that enables remote logon to an ASCII host. TCP/IP provides connectivity functions for both local area networks (LAN) and wide area networks (WAN) and includes the ability to route information between LANs and WANs. The major TCP/IP networks—the Internet—use a standardized addressing procedure to ensure that IP addresses are unique and that communication between enterprises is possible.

The VT over Telnet attachment for Z and I Emulator for Windows requires a TCP/IP stack that supports the Windows® Sockets Version 1.1 interface. WSOCK32.DLL must be in the Windows® system directory or the current path to provide the interface for the stack program and to support the Windows® Sockets V1.1 interface.

For the VT over Telnet attachment, you must define the following attachment parameters.

- Host Name or IP Address (mandatory)
- Port Number (optional)
- Terminal ID (optional)
- Auto-reconnect (optional)

Host Name or IP Address

Specify either the alphabetic name of the target host or its numeric IP address.

Host Name

The name of the target host is a string

Host IP Address

The IP address of the target host is in dotted-decimal notation—for example: 0.0.0.0

Port Number

Specify the decimal number of the target host's Telnet port. The default, 23, is the standard Telnet port.

Terminal ID

The VT emulator and the Telnet server use the terminal ID for negotiating an appropriate connection. Ask your Telnet administrator for your host's correct terminal ID. When the default box is selected, the default values are selected from the Machine Mode, as shown in the following table:

Machine Mode	Default Terminal ID
VT340	DEC-VT220
VT100	DEC-VT100
VT52	DEC-VT52
ANSI	ansi

Auto-reconnect

If the session is disconnected from the host, and if this box is selected, you will be re-connected automatically.

The default is not selected.

Using A VT Session

Your Z and I Emulator for Windows VT session works as if you were using a VT340, VT100, or VT52 terminal. For mainframe VT, iSeries™, eServer™ i5, or System i5™ connections, the protocol converters have defined VT keyboard sequences, such as F1 or PA1.

The following tables are provided:

- Characters generated by VT Compose Key
- Characters displayed in transparent mode
- OIA line display messages

Refer to *Administrator's Guide and Reference* for default mapping of the VT340 keyboard to the PC keyboard, as used by the Z and I Emulator for Windows VT emulator.

Compose Key

The VT emulator supports the VT340 compose key for generating special characters on the display. Before using the compose key, define a key combination that represents it.

Using the compose key involves three separate actions:

1. Press and release the compose key.
2. Press and release the first character (see Table 46: Character Generation (Special Characters) on page 349).
3. Press and release the second character.

The first and second characters may be typed in either order, except when the table specifies that they must be entered as shown,
Table 46: Character Generation (Special Characters) on page 349 shows the appearance and name of each special character, the character pair that generates the character, and an indication whether the order of entering the characters is significant.

Table 46. Character Generation (Special Characters)

Generated Character		Compose Key, Plus This Pair		
Appearance	**Description**	**First**	**Second**	**Order**
Á	A acute	A	'	either
á	a acute	a	'	either
Â	A circumflex	A	^	either
â	a circumflex	a	^	either
À	A grave	A	`	either
à	a grave	a	`	either
Å	A ring	A	*	either
		A	°	either
å	a ring	a	*	either
		a	°	either
Ã	A tilde	A	~	either
ã	a tilde	a	~	either
Ä	A umlaut	A	"	either
ä	a umlaut	a	"	either
Æ	AE ligature	A	E	as shown
æ	ae ligature	a	e	as shown
´	apostrophe	'	space	either
@	at sign	a	a	either
		A	A	either
\	backslash	/	/	either
\	backslash	/	<	either
Ç	C cedilla	C	,	either
ç	c cedilla	c	,	either
¢	cent sign	c	/	either
		C	/	either
		c	|	either

Table 46. Character Generation (Special Characters)

(continued)

Generated Character		Compose Key, Plus This Pair		
Appearance	**Description**	**First**	**Second**	**Order**
		C	|	either
^	circumflex accent	^	space	either
}	close brace	)	-	either
]	close bracket	)	)	either
»	close French quote	>	>	either
@	commercial at	a	a	either
		A	A	either
©	copyright mark	c	o	either
		C	O	either
		c	0	either
		C	0	either
°	degree sign	0	^	either
		°	space	either
		#	space	either
É	E acute	E	'	either
é	e acute	e	'	either
Ê	E circumflex	E	^	either
ê	e circumflex	e	^	either
È	E grave	E	`	either
è	e grave	e	`	either
Ë	E umlaut	E	"	either
ë	e umlaut	e	"	either
a	feminine ordinal indicator	a	_	either
		A	_	either
½	fraction one-half	1	2	as shown
¼	fraction one-quarter	1	4	as shown
ß	German ess-tset	s	s	either
µ µ	Greek mu	/	u	as shown
		/	U	as shown
»	guillemets, closing	>	>	either
«	guillemets, opening	<	<	either
Í	I acute	I	'	either
í	i acute	i	'	either
Î	I circumflex	I	^	either

Table 46. Character Generation (Special Characters)

(continued)

Generated Character		Compose Key, Plus This Pair		
Appearance	**Description**	**First**	**Second**	**Order**
î	i circumflex	i	^	either
Ì	I grave	I	`	either
ì	i grave	i	`	either
Ï	I umlaut	I	"	either
ï	i umlaut	i	"	either
¡	inverted exclamation	!	!	either
¿ ¿	inverted question mark	?	?	either
º	masculine ordinal indicator	o	_	either
		O	_	either
µ µ	micro sign	/	u	as shown
		/	U	as shown
• ▪	middle dot	.	^	either
Ñ	N tilde	N	~	either
ñ	n tilde	n	~	either
#	number sign	+	+	either
Ó	O acute	O	'	either
ó	o acute	o	'	either
Ô	O circumflex	O	^	either
ô	o circumflex	o	^	either
Ò	O grave	O	`	either
ò	o grave	o	`	either
Ø	O slash	O	/	either
ø	o slash	o	/	either
Õ	O tilde	O	~	either
õ	o tilde	o	~	either
Ö	O umlaut	O	"	either
ö	o umlaut	o	"	either
Œ Œ	OE ligature	O	E	as shown

Table 46. Character Generation (Special Characters)

(continued)

Generated Character		Compose Key, Plus This Pair		
Appearance	**Description**	**First**	**Second**	**Order**
œ Œ	oe ligature	o	e	as shown
{	open brace	(	-	either
[	open bracket	(	(	either
«	open French quote	<	<	either
¶	paragraph sign	p	!	either
±	plus-or-minus sign	+	-	either
£	pound sterling sign	l	-	either
		L	-	either
		l	=	either
		L	=	either
"	quotation mark	"	space	either
§	section sign	s	o	either
		S	O	either
		s	!	either
		S	!	either
		s	0	either
		S	0	either
'	single quote	'	space	either
ß	ss German	s	s	either
^{1}	superscript 1	1	^	either
^{2}	superscript 2	2	^	either
^{3}	superscript 3	3	^	either
~	tilde	~	space	either
Ú	U acute	U	'	either
ú	u acute	u	'	either
Û	U circumflex	U	^	either
û	u circumflex	u	^	either
Ù	U grave	U	`	either
ù	u grave	u	`	either
Ü	U umlaut	U	"	either
ü	u umlaut	u	"	either
|	vertical line	/	^	either
Ÿ	Y umlaut	Y	"	either

Table 46. Character Generation (Special Characters)

(continued)

Generated Character		Compose Key, Plus This Pair		
Appearance	**Description**	**First**	**Second**	**Order**
Ÿ				
ÿ	y umlaut	y	"	either
¥	yen sign	y	-	either
		Y	-	either
		y	=	either
		Y	=	either

Transparent Mode

Table 47: Character Generation (Transparent Mode) on page 353 shows the symbol displayed for each character and control code when the VT emulator is in transparent mode. The characters at AA and BA are the feminine and masculine ordinals, respectively. The characters at 1E, 1F, 80, and 9E are underlined, although they may not appear underlined on the output.

Table 47. Character Generation (Transparent Mode)

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	@	P		0	@	P		p		p	Ÿ Ÿ	°	À	Ð	à	ð
x1	A	Q	!	1	A	Q	a	q	a	q	¡	±	Á	Ñ	á	ñ
x2	B	R	"	2	B	R	b	r	b	r	¢	²	Â	Ò	â	ò
x3	C	S	#	3	C	S	c	s	c	s	£	³	Ã	Ó	ã	ó
x4	D	T	$	4	D	T	d	t	d	t	¤	´	Ä	Ô	ä	ô
x5	E	U	%	5	E	U	e	u	e	u	¥	µ µ	Å	Õ	å	õ
x6	F	V	&	6	F	V	f	v	f	v	¦ ¦	¶	Æ	Ö	æ	ö
x7	G	W	'	7	G	W	g	w	g	w	§	• ▪	Ç	×	ç	÷

Table 47. Character Generation (Transparent Mode) (continued)

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x8	H	X	(	8	H	X	h	x	h	x	¨	¸	È	Ø	è	ø
x9	I	Y	)	9	I	Y	i	y	i	y	©	¹	É	Ù	é	ù
xA	J	Z	*	:	J	Z	j	z	j	z	a	o	Ê	Ú	ê	ú
xB	K	[	+	;	K	[	k	{	k	{	«	»	Ë	Û	ë	û
xC	L	\	,	<	L	\	l	|	l	|	¬	¼	Ì	Ü	ì	ü
xD	M	]	-	=	M	]	n	}	m	}	-	½	Í	Ý	í	ý
xE	N	^	.	>	N	^	m		n			¾	Î	Þ	î	þ
xF	O	_	/	?	O	_	n	Œ Œ	o	œ œ	¯ —	¿ ¿	Ï	ß	ï	ÿ

OIA Line Display Messages

During VT emulation, messages unique to VT can appear in certain columns of the OIA line. These columns display only VT messages, and do not display any of the messages that would appear there in 3270 or 5250 mode. Table 48: OIA Line Display Messages (VT only) on page 354 shows the meaning of each VT-specific message. Columns that are not mentioned in the table show messages common to all Z and I Emulator for Windows modes.

Table 48. OIA Line Display Messages (VT only)

Columns	Message	Meaning
1 through 7	VT340 7	Machine mode is VT340, seven-bit control.
	VT340 8	Machine mode is VT340, eight-bit control.
	VT100	Machine mode is VT100.
	VT52	Machine mode is VT52.
	VTANSI	Machine mode is VTANSI.
9 through 12	LOCK	Keyboard is locked.
30 through 39	OVERSTRIKE	New characters replace the character at the cursor position in Local Edit mode.
	INSERT	New characters move characters in page memory to the right in Local Edit mode.
61 through 64	HOLD	Screen is in hold mode.
66 through 69	EDIT	Local Edit mode is enabled.
71 through 72	Pn (n=1 through 6)	Current page number.

History Logging

You can use the Windows® scroll bar control to view history data from the current VT session. When configuring the session, select the **History Logging** option and specify the size of the log (see Configuring a VT Session on page 342).

When history logging is enabled, text is logged into the VT history window as it scrolls off the screen from the top margin row. The top and bottom margins are set when the host application defines the scrolling region, using the DECSTBM command sequence (`(ESC [ Pn ; Pn r)`).

Enhanced History Logging

When the host application sends the Erase in Display (ED) command sequence to erase a portion of the emulator screen, the contents can be logged into the VT history window before being erased. To enable this functionality, select the **History Logging – Enhanced** option, when configuring the VT session parameters.

The format of the host application ED command is `ESC [ Ps J`, where `Ps` is one of the following values:

0

Erases the screen contents from the cursor position to the end of the screen. This is the default setting.

1

Erases the screen contents from the beginning of the screen up to and including the cursor position.

2

Erases the entire screen contents.

When the ED command setting is 0 or 1, and **History Logging – Enhanced** is enabled, then the portion of the screen that is about to be erased will be logged into the history window before being erased. When the parameter value is 2, the entire screen contents are logged into the history window before being erased, regardless of whether enhanced history logging is enabled.

ASCII Host File Transfer

Setting Preferences

You can set up your Workstation to perform file transfers; some preferences need to be set first, as described in this section. Other facilities to simplify handling of transfers are also described.

Using XMODEM and YMODEM

Z and I Emulator for Windows allows you to transfer files to and from ASCII hosts that support the XMODEM and YMODEM protocols. In order to use XMODEM or YMODEM, you must have established a connection to an ASCII host.

Choosing a Protocol

You have four choices for protocols. The one you select will depend upon the protocols supported by your ASCII host and by your particular requirements. The following table shows the capabilities of the protocols:

	Downloading		Uploading	
	Single File	**Multiple Files**	**Single File**	**Multiple Files**
XMODEM	Yes	No	Yes	No
XMODEM1K	Yes	No	Yes	No
YMODEM	Yes	Yes	Yes	Yes
YMODEMG	Yes	Yes	Yes	Yes

XMODEM

The XMODEM protocol is a single-file half-duplex protocol that performs error checking. Data is transmitted in 128-byte packets. Error checking, either by CRC or by checksum, occurs automatically. The Z and I Emulator for Windows implementation of XMODEM first tries CRC. If the sender fails to acknowledge the first three requests for CRC, XMODEM shifts to the checksum mode.

XMODEM1K

The XMODEM1K protocol is the same as XMODEM, except that it always uses CRC and has a larger packet size of 1024 bytes. Because some hosts are not able to handle the 1024-byte packets, there is a need for both XMODEM and XMODEM1K

YMODEM

The YMODEM protocol is similar to XMODEM, but it allows you to send multiple files in a single transfer. You may use a set of unique file names, or you may specify groups of files.

YMODEMG

The YMODEMG protocol is the same as YMODEM, supporting multiple files, but it does not supply error checking. It assumes that the data always transfers correctly, and is only for use with error-correcting modems. For large amounts of data it can achieve much greater throughput than YMODEM because it does not wait for packet acknowledgment.

XMODEM and XMODEM1K

To use XMODEM, click **Edit → Preference → Transfer** in your Z and I Emulator for Windows session. The Transfer Preferences window appears. Select the **XMODEM** or **XMODEM1K** protocol, and optionally click on the tab for the selected modem protocol to define the **Transfer Type** or to change advanced settings.

When receiving a file, in the Receive File from Host dialog box, enter the file name in the **PC File** field or select a personal computer file name from the drop-down listbox. The transfer type is automatically generated according to the templates.

YMODEM and YMODEMG

To use YMODEM, click **Edit → Preference → Transfer** in your Z and I Emulator for Windows session. The Transfer Preferences window appears. Select the **YMODEM** or **YMODEMG** protocol, and optionally click on the tab for the selected modem protocol to define the **Transfer Type** or to change advanced settings.

When receiving a file, you cannot select the personal computer file name, but you can change the default transfer type, the drive, and the directory, if necessary.

File-Transfer Timeout

You can define the time the workstation waits for a response from the host system (in seconds). If the host system does not respond, the transfer is canceled, and an error message appears. A number in the range 20–65535 (or 0) can be specified. The default is 60 seconds for ASCII sessions. Specify an appropriate value such that the error message does not appear too early. If you specify 0, a timeout is not set.

If a packet or block size is relatively large for low-speed lines, such as COM port lines, it is recommended that 150 seconds or greater be specified.

Extension for List-Files

You can change the default extension (.SRL) of file-transfer list files.

Show Status Window

You can choose the method of displaying the file-transfer-progress status.

In Session

When file transfer starts, the status window appears. The name of the file being transferred and the transfer progress appear.

In Icon

When file transfer starts, the status icon appears on the screen. If the icon is restored, the status window appears.

Defining Transfer Types

Transfer types define the option information used for controlling file transfer. Up to 32 transfer types can be defined for each host system. The original default types are: **delete** (deletes a file on abort), **over** (overwrites existing files) and **none** (does not delete on abort, and does not overwrite).

To add or change transfer types:

1. Click **Edit → Preference → Transfer**.
2. Click the tab for the modem protocol you have selected. The items that appear depend on the selected host system.
3. Enter transfer-type names in the **Transfer-Type** text box, or select them from the drop-down list.
4. To add or replace a transfer type, click **Save**. To delete a transfer type, click **Delete**.
5. Depending on the transfer type, select one of the following file receive options:

   Delete File on Abort

   With this option, if a file transfer is aborted then the incompletely received file is automatically deleted.

   Overwrite Existing File

   With this option, any existing file with the same name as the incoming file is overwritten.
   If you do not select this option, then a new name is given to the incoming file, according to the following scheme:

Existing file:	EXAMPLE.TXT
First contender becomes:	EXAMPLE.TX1
Second contender:	EXAMPLE.TX2
Tenth contender:	EXAMPLE.T10
Hundredth contender:	EXAMPLE.100
999th contender:	EXAMPLE.999

6. Click **OK**.

These options are independent of each other.

File Transfer Templates

For sending ASCII files, Z and I Emulator for Windows automatically generates Host file names and transfer types. For receiving ASCII files via XMODEM and XMODEM1K, Z and I Emulator for Windows generates a transfer type. In both situations, templates define the rules for file name and transfer type generation.

Defining Templates

The templates are common for all sessions and are used for both sending and receiving files. For ASCII host file transfer, you can define up to three templates for each protocol.

To display the templates panels, click the **Templates** button in the **Send File to Host** or **Receive File from Host** panel.

You can add, delete, or replace templates; you can also test templates to see how Z and I Emulator for Windows generates the target file name and transfer type.

When defining templates, you can use * (asterisk) for the global searching of file names; for example, *.EXE for all files that have a file name extension of EXE.

Automatic Generation of File Names

The templates are numbered from 1 to 32; when Z and I Emulator for Windows generates file names, the templates are searched, starting from 1, and the first template that matches is used.

Example of ASCII Protocol Template

The following example shows the use of templates for ASCII host file transfer. When sending files, Z and I Emulator for Windows automatically generates a host file name from a personal computer file name, and vice versa. It also generates a transfer type. When receiving files, Z and I Emulator for Windows automatically generates only transfer types, and only for the XMODEM and XMODEM1K protocols.

For more information about templates, refer to *Administrator's Guide and Reference*.

Following are the definitions of the three default templates. The template is selected from the available choices by matching the name of the file being transmitted or received against each template's file specifications.

Template Number	Wildcard specification for PC File	Wildcard specification for host File	Type
1	*.exe	*.*	delete
2	*.txt	*.*	over
3	*.*	*.*	none

Send Example: If you enter `program.exe`, Z and I Emulator for Windows selects template 1, and displays `program.exe delete` in the list box.

Receive Example: (XMODEM AND XMODEM1K only) If you enter `program.exe`, Z and I Emulator for Windows selects template 1, and displays `program.exe delete` in the list box.

Working with Lists of Files

For transferring a group of files it is convenient to use a list. A list makes it easier to transfer the same groups of files frequently, with a single command. Even if you are transferring a group of files only once, a list can help prevent errors. A list of files is itself a file.

You can transfer multiple files at once by using the send/receive list; it is accessible from the Send Files to Host or Receive Files from Host windows. For either window, the files selected are displayed in a **Transfer List**. This list can be saved, and later retrieved and modified. For instructions on selecting a file, see Receiving Files from an ASCII Host on page 361 and Sending Files to an ASCII Host on page 362.

File Name Extension for List Files

By default, send/receive list files have a file name extension of .SRL. You can change this default on the property page with the **General** tab, by clicking **Preferences → Transfer** from the **Edit** menu.

Note: Z and I Emulator for Windows does not recognize a file as a send/receive list file unless its name has the specified extension.

Remove From List

By clicking the **Remove** button, you can delete the selected file from a send/receive list.

Open List File

If you click the **Open List** button, the Open File-Transfer List File dialog box appears, allowing you to manipulate the file names in the list.

Save List File

If you click the **Save** button, the Save File-Transfer List File As dialog-box appears and you can save the list of files.

Changing a List of Files

You can make changes to a list of files to be transferred:

Change the Personal Computer or Host File Name

When you select a file to send or receive, Z and I Emulator for Windows automatically generates a host or personal computer file name by using templates. To change the generated file name, just type over it.

Note: When receiving a file from an ASCII host, you specify the host file name on the host system.

For receiving files, you can select a personal computer file from the dialog obtained by clicking the **Browse** button.

Note: The browse function is not available when *receiving* files from an ASCII host; it is available when sending files, but only when using the YMODEM or YMODEMG protocols.

Delete File Names From List

To delete a file from the list, select it from the list and click the **Remove** button.

Add More File Names To List

To add more files to the list, select a file in the **PC File** list box with Ctrl + left mouse-button, or type a file name in the **PC File** entry field and press Enter.

Receiving Files from an ASCII Host

Receive File From Host allows you to receive files from a host system to your personal computer; with one command, you can receive a single file or several. If you often receive the same list of files, you can save the list of file names and receive all the files with one command.

For ASCII host file transfer, the host system must support one of two protocols, XMODEM or YMODEM.

Selecting a Workstation Directory

To receive files to a workstation directory, you can key in the directory information or click the **Browse** button to open the Browse dialog and select the directory; this can be done as part of setting preferences (setting the **Default PC Directory** field) or at the time of the file transfer.

Selecting Files to Receive

For ASCII host file transfer, select the file to receive on the host system.

Follow these steps to receive one or more files from an ASCII host:

1. Prepare the host system. The exact method of preparation, including selection of file names, depends on the kind of host system to which you are connected. Contact your host-system administrator for details.

   Note: The host system must support one of two protocols: XMODEM or YMODEM.

2. Click **Edit → Preference → Transfer** to display the Transfer Preferences window. Select the type of protocol you want to use from the drop-down list box on the property page with the **General** tab.
3. In the **Default PC Directory** field, type the workstation directory where the file or files should be sent; or, click the **Browse** button to open a dialog and select the directory.
4. To change the transfer parameter defaults for the protocol you selected, click the tab to display the property page for the selected modem protocol.
5. When all preferences have been set, click **OK**.
6. Click **Receive File from Host** from the **Actions** menu. The **Receive File from Host** window appears.
7. For XMODEM and XMODEM1K, click the **Browse** button to open a dialog and select a personal computer file name or names, or enter the names in the **PC File** entry field. The transfer type is automatically generated and appears in the **Transfer Type** entry-field.

8. For YMODEM and YMODEMG, select the transfer type and click the **Browse** button to open a dialog and change the directory, if you desire.
9. Click the **Receive** button to display the **Receive Files Status** window and start the transfer.

Sending Files to an ASCII Host

Send file to host allows you to send files from your personal computer to the host system; with one command, you can send a single file or several files. If you often send the same list of files, you can save the list of file names, and subsequently send all the files with one command.

Note: This is supported using the YMODEM and YMODEMG protocols only.

Selecting Files to Send

There are several ways to select files to send:

Basic Methods

Type a file name in the **PC File** field and press Tab; a host file name and a transfer type are generated automatically according to the templates.

Select files from the dialog obtained by clicking the **Browse** button.

Select from a Send/Receive List

If you have saved a list of file names in a send/receive list, click the **Open List** button and select the list you want to use; the file names saved in the list appear.

Note: For ASCII host file transfer, you can use the send/receive list only with the YMODEM and YMODEMG protocols (not with XMODEM or XMODEM1K).

Advanced Method

The Browse window, obtained by clicking the **Browse** button, displays all the files in the current directory; you can display only certain types of files if you want to.

For example, if the directory has many files and you want to display only files that have the extension .DOC, you can type `*.doc` in the **PC File** field and click the **Browse** button; the resulting dialog shows only files that have the extension .DOC.

Changing the Host File Name or the Transfer Type

When you select a file to send, Z and I Emulator for Windows automatically generates a host file name and selects a transfer type from the default templates. You can change the file name by typing over the text in the **Host File** field; you can change the transfer type by selecting a different one from the **Transfer Type** drop-down list.

Saving a List of Files to Send

If you frequently send the same set of files, it is a good idea to save the names in a list, called a send/receive list.

Note: For ASCII host file transfer, you can use the send/receive list only with the YMODEM or YMODEMG protocols (not with XMODEM or XMODEM1K).

Sending a List of Files

Select the list, then click the **Send** button.

PC Code Page

When a file is transferred, EBCDIC codes are converted to 1-byte workstation codes, and vice versa. A valid value is automatically selected from among the following values for SBCS sessions: 437, 737, 806, 813, 819, 833, 850, 852, 854, 857, 858, 860, 861, 862, 863, 864, 865, 866, 869, 874, 912, 915, 916, 920, 921, 922, 1008, 1089, 1124, 1125, 1127, 1129, 1131, 1133, 1153, 1155, 1156, 1157, 1158, 1160, 1164, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, and 1258; —according to the host code page specified when the workstation is configured. For an explanation of how to select host code pages, see the online help for the host code page.

Troubleshooting

Troubleshooting tips

1. Connecting to z/OS console via Z and I Emulator for Windows 62x160 screen size results in error IEE938I

Problem

IEE936I CONSOLE HHSP0141 INITIALIZATION ERROR - RC:01 - 3277-2 IS ASSUMED is logged by z/OS console when using HCL Z and I Emulator for Windows version 6.0.2 with a 62x160 screen size.

Cause

z/OS console connection is being made through the OSA-ICC telnet server. This server does not support the Terminal Type of IBM-DYNAMIC.

APAR IC71220 changed the default Terminal Type string passed to the telnet server for screen sizes greater than 27x132.

Resolution

To use 62x160 screen size display sessions with OSA-ICC telnet servers, take the following steps:

1. Edit the WS session profile.
2. Add the following statements under the Telnet3270 stanza to force HCL Z and I Emulator for Windows to use the original default string:

```
[Telnet3270]
TerminalTypeString=IBM-3278-2-E
```

2. Z and I Emulator for Windows TCP/IP Data Transfer fails with terminated transfer function

Problem

TCP/IP Data Transfer might fail with the error "Transfer function will be terminated". This message provides a checklist for iSeries or AS/400 host requirements.

Resolution

To resolve this problem, take the following steps to check the status of the iSeriesTM or AS/400® host requirements:

1. Confirm that the iSeries or AS/400 host has the following required licensed program(s) installed (type GO LICPGM and select Option 10 - Display Installed Licensed Programs:):

```
For V5Rx:
5722TC1 *BASE iSeries TCP/IP Connectivity Utilities/400
5722SS1 Option 12 OS/400 ® - Host Servers

For V4R5:
5769TC1 *BASE TCP/IP Connectivity Utilities for AS/400
5769SS1 Option 12 OS/400 - Host Servers
```

2. Verify that you have a Relational Database entry for YOURSYSTEM at REMOTE Location = *LOCAL (issue the command WRKRDBDIRE to "Work with Relational Database Directory Entries"):

```
RELATIONAL DATABASE . . . . . . : YOURSYSTEMNAME
REMOTE LOCATION:
REMOTE LOCATION . . . . . . . : *LOCAL
TEXT . . . . . . . . . . . . . :
```

3. Verify that user ID QUSER is enabled (issue the following command for profile QUSER):

```
WRKUSRPRF USRPRF(QUSER)
```

4. Verify that TCP/IP is active, using the following command (This is required before any TCP/IP processing):

   ```
   START TCP/IP (STRTCP) COMMAND
   ```

5. If you have not already done so, issue the following "Start Host Server" command:

   ```
   STRHOSTSVR SERVER(*ALL)
   RQDPCL(*TCP)
   ```

The following sets of instructions are to verify that the required server daemon and prestart server jobs are active for DDM, Database, and File Transfer functions.

1. For DDM, take the following steps:
   a. Verify that the DDM daemon is active. Use WRKACTJOB to find the QRWTLSTN job under QSYSWRK. This daemon is automatically started when the STRTCP command is run, if the AUTOSTART parameter is set to *YES on the CHGDDMTCPA command <F4>. If the daemon does not start when the STRTCP command is run, you can start the daemon by issuing the following command:

      ```
      STRTCPSVR SERVER(*DDM)
      ```

      Note: If the DDM daemon still fails to start with error message CPF3E30, refer to APAR SA81267.

   b. Verify that the DDM prestart server jobs are active. Use WRKACTJOB to find the server jobs named QRWTSRVR. For V5R2 and newer releases, the prestart jobs run in the QUSRWRK subsystem, but can be configured to run under other subsystems. They are automatically started with the subsystem. If these jobs are not active, you can issue the start prestart job command STRPJ <F4>. For host versions prior to V5R2, these jobs runs in the QSYSWRK subsystem.
2. For Database, take the following steps:
   a. Verify that the subsystem for the Database server daemon is active. The subsystem is QSERVER, and the daemon job is named QZDASRVSD.
   b. Verify that the Database prestart server jobs are active. For V5R1 and newer releases, the prestart jobs are QZDASOINIT and QZDASSINIT. Both jobs run in the QUSRWRK subsystem, but can be configured to run under other subsystems. As with the DDM prestart jobs, these jobs are automatically started with the subsystem, but if they are not active, they can be started with the STRPJ command.
   c. If port 8478 is not active, end and restart the Database server job QZDASRVSD using the following commands:

      ```
      ENDHOSTSVR *DATABASE
      STRHOSTSVR *DATABASE
      ```

3. For File Transfer, take the following steps:

a. Verify that the subsystem for the transfer function server is active. The subsystem is QSERVER, and the daemon job is named QZDASRVSD.
b. Verify that the transfer function prestart server jobs are active. The prestart job is QTFPJTCP. This job runs in the QSERVER subsystem.

3. Z and I Emulator for Windows Telnet connection timeout with error 657

Problem

Keyword for Telnet timeout.

Resolution

Z and I Emulator for Windows uses non-blocking sockets when connecting to a remote system. By default, Z and I Emulator for Windows waits for 3 seconds to establish the socket connection. When accessing remote terminals over a dialup network, this default wait time might not be sufficient and could lead to a connection failure and return the error 657.

The default time can be changed by adding the following keyword to the Telnet3270 or Telnet5250 stanza of the workstation profile:

```
[Telnet3270]
InactiveTimeout=xx
```

The value `xx` is the time in seconds and can take any positive numerical values.

4. PCSXFER041 timeout during Z and I Emulator for Windows file transfer TSO session

Problem

Three situations might cause a timeout at the Z and I Emulator for Windows client and a solution is provided for each case.

Cause

Resolution

CUT mode transfer

By default, Z and I Emulator for Windows uses DFT mode for transfers to and from a TSO host. When using Dial to connect to TSO, the host might be configured to do the transfer in CUT mode, while Z and I Emulator for Windows is still in DFT mode.

You can modify the workstation profile that you use to dial the TSO session. To manually change the DFT/CUT setting, add the following case-sensitive line to the Transfer stanza of the workstation file:

```
[Transfer]
CUTprotocol=Y
```

You can also change the LU description on the zSeries host. Change the VTAM BIND image so that the Write Structured Field query support bit is set in the PSERVIC. Refer to the VTAM documentation for more information.

Using \FT trigger in SuperSession Manager

If you encounter the timeout problem when using Supersession, you can use the \FT trigger before logon. This can be applied on an individual user and session basis.

The \FT trigger invokes the file transfer script KLSXFER, which enables query passthru. It also inhibits immediate broadcasts and session locking.

Document mode and Word Wrap mode

If Document mode or Word Wrap mode is enabled, the file transfer will fail. If you have a key mapped to toggle this feature on or off, turn it off before initiating the file transfer. If you want to disable this feature, change the following case-sensitive line in the ENTRYASSIST stanza of the workstation file:

```
[ENTRYASSIST]
DocmodeWordWrap=N
```

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

> HCL
> 330 Potrero Ave.
> Sunnyvale, CA 94085
> USA
> Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Admin Guide

Contents

HCL Z and I Emulator for Windows (ENGLISH)

About This Book

HCL Z and I Emulator for Windows reference books are comprised of this Administrator's Guide and Reference and an Emulator User's Reference. These volumes provide information for using HCL Z and I Emulator for Windows operating systems (hereafter called PC400).

Note:

1. *PC/3270* refers to the 3270 portion of the combined package.
2. *Workstation* refers to all supported personal computers.
3. *Windows®* refers to Windows® 8, Windows® 8.1, Windows® 10, Windows® 10 x64 and Windows® Server 2008. When information is applies only to a specific operating systems, this is indicated in the text.

Who Should Read This Book

This book is intended for administrators of Z and I Emulator for Windows.

How to Use This Book

This book contains reference information that you might need to refer to when installing or operating Z and I Emulator for Windows.

Z and I Emulator for Windows is designed to use various communication adapters and to work with other workstation and host system software. Refer to the appropriate documentation for the products you use.

Command Syntax Symbols

Parentheses, brackets, ellipses, and slashes have the following meanings or uses:

( )

Parentheses enclose operands that govern the action of certain command options.

[ ]

Brackets indicate an optional command argument. If you do not use the optional item, the program selects a default.

...

Ellipsis after an argument indicates that you can repeat the preceding item any number of times.

/

For 3270, a slash must precede the Time Sharing Option Extensions (TSO/E) password. A slash must also precede parameters of DOS commands entered from the command line. For 5250, a slash must precede parameters of IBM® DOS commands entered from the command line.

****

A backslash is included as part of any directory name. An initial backslash indicates the first-level directory, and an additional backslash is inserted in the directory name to indicate another level.

All directives, operands, and other syntax can be typed in either uppercase or lowercase, unless otherwise indicated.

Where to Find More Information

The following sections discuss getting help when you are installing, configuring, or using Z and I Emulator for Windows.

Information Center

You can find documentation and links to other resources at the Z and I Emulator for Windows Information Center, at the following address:

https://help.hcltechsw.com/zie/ziewin/3.0/index.html

The Z and I Emulator for Windows Information Center provides information in English.

Online Help

The help facility describes how to install, configure, and use Z and I Emulator for Windows. Online help is very extensive and includes information about every aspect of configuring and using Z and I Emulator for Windows.

Use help to obtain the following information:

- Menu choices
- Operation procedures
- Operations in windows
- Meanings of the terms displayed in windows
- Causes of errors and the corresponding actions to take

- Mouse-based operations
- Operation without a mouse
- Detailed explanations of specific terms
- Further technical information about Z and I Emulator for Windows
- Detailed explanations of operator information area (OIA) messages

To display online help, select choices from the **Help** menu or press **F1**.

You can use Z and I Emulator for Windows online help just as you use the online help for Windows®.

Messages and Alerts

Online messages are displayed by Z and I Emulator for Windows, but a message does not always mean an error occurred. For example, a message might tell you that an operation is in progress or has been completed. A message can also prompt you to wait for the completion of an operation.

Messages That Appear in Pop-Up Windows

While using Z and I Emulator for Windows, you may see messages appear in popup windows, but not necessarily as a direct result of your actions. These messages can appear for a number of reasons, outlined in the following sections.

System-Fault Messages

If a message does appear in a pop-up window, you can paste its contents into the Windows® clipboard. To do this:

1. Click **Details** on the pop-up window.
2. Mark the text that you want to copy.
3. Click the marked text with the right mouse button and then click **Copy**.
4. Start an editor, such as Notepad, and click **Paste** from the **Edit** menu.
5. Save the file in case an IBM® Service Representative needs this information to diagnose your problem.

Security-Related Messages

Z and I Emulator for Windows optionally utilizes Secure Sockets Layer (SSL) or Transport Layer Security (TLS) to establish sessions with servers; this may require input from you (for example, a password). See Configuring and Using Security for Z and I Emulator for Windows on page 395 for details.

System-Policy-Related Messages

Your Z and I Emulator for Windows workstation configuration can be controlled centrally using facilities for managing system policies. See System Policy Support on page 391 for details.

OIA Messages

Z and I Emulator for Windows displays messages in the operator information area (OIA) or in a pop-up window. Messages from Z and I Emulator for Windows are displayed in the message window; messages from the host system regarding the condition of the session are displayed in the OIA of the session window.

The OIA is the bottom line of the session window. An OIA message indicates the status of Z and I Emulator for Windows as well as information about the workstation, host system, and attachment method.

All of the OIA indicators, reminders, and messages are described in the online help. To view this information:

1. Click **Index** from the **Help** menu.
2. Select **The operator information area messages**.

   To look up a specific OIA message, select **Search**. When the Search window appears, type the letters that appear in the OIA. For example, `MACH` or `PROG`. If a lightning bolt appears, type `COMM`.
3. Double-click the index entry that matches your search.
4. Scroll through the window until you find the number that appears in your OIA.

Alerts

Alerts may be generated which correspond to specific Z and I Emulator for Windows messages. See Alerts on page 493 for more information.

Z and I Emulator for Windows Library

The Z and I Emulator for Windows library includes the following publications:

- *CD-ROM Guide to Installation*
- *Quick Beginnings*
- *Emulator User's Reference*
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Host Access Class Library*

In addition to the printed books, there are HTML documents provided with Z and I Emulator for Windows:

Host Access Class Library for Java

This HTML document describes how to write an ActiveX/OLE 2.0–compliant application to use Z and I Emulator for Windows as an embedded object.

General Information

Z and I Emulator for Windows Highlights

Z and I Emulator for Windows brings the power of personal networking to your workstation by providing a variety of connectivity options supporting local area network (LAN) and wide area network (WAN) environments. Whether you need host terminal emulation, client/server applications, or connectivity, Z and I Emulator for Windows offers a robust set of communications, networking, and administrative features.

Z and I Emulator for Windows is a full-function emulator package with an easy-to-use graphical interface, which includes many useful features such as file transfer and dynamic configuration, and emulator APIs including the IBM® Host Access Class Library.

Z and I Emulator for Windows provides the following functions:

- **zSeries Connections**

  LAN

  Telnet3270

  VT-over-Telnet (TCP/IP)
- **iSeries™ Connections**

  LAN

  Telnet5250 over TCP/IP

  VT over Telnet

  COM port

  VT over Telnet

  Telnet 5250
- **ASCII Emulator Connections**

  LAN

  VT over Telnet

  COM port

  VT over Telnet
- **Client/Server Connections**

  LAN

  COM port
- **Log Viewer**
  - View Message Log, Trace Log, and Merged Log files
  - Summary and Detail views
  - Set default Message Log size and location

    - Filter and search Log files
    - Message Log entries Help
- **Trace Capability**
    - 3270/5250 emulator data
    - Connectivity data, such as LAN
    - User services data, such as node initialization
- **Sample Programs**
    - Located in \ZIE for Windows\samples subdirectory
- **Installation and Configuration**
    - Partial installation option
    - Program sharing on a network server
    - Automatic detection of installed communication adapters
    - Dynamic change of communication configurations
    - Silent Installation
    - Verification of ASCII configuration
- **Host Session Function**
    - Up to 52 sessions
    - Variable screen size and automatic font scaling
    - Function settings (of the host code page, for example) for each session
- **Host Graphics Support**
    - Built-in vector graphics support for GDDM® and other graphics applications
- **File Transfer Function**
    - Easy operation through graphical user interface (GUI) windows
    - Batch transfer of multiple files
    - Concurrent file transfer through multiple sessions
    - Background file transfer
    - File transfer invocation by macro
    - OfficeVision/MVS™ Import/Export functions
    - VT File Transfer (XModem and YModem)
- **Edit (Cut and Paste) Function**

  You can use the clipboard to cut, copy, and paste a selected areaIn addition, you can paste data in other applications, such as spreadsheet programs, that support the PasteLink function.
    - Support of spreadsheet data format (Sylk, Biff3, Wk3 formats)
    - Copy Append
    - Paste Next
    - Paste to Trim Rectangle
    - Paste Stop at Protected Line
- **Graphical User Interface (GUI)**
    - Customizable 3D iconic tool bar
    - 3D-button hotspots
    - Pop-up keypad
    - Macro function, including record and play

  - VBScripts, including record and play
  - Keyboard-function setup and remapping
  - Mouse-button-function setup and remapping
  - Display setup (cursor type, graphics, sound, colors, for example)
  - Automatic font size adjustment or fixed font size
  - Window-appearance setup
  - Menu-bar customization
  - 3270 Light Pen emulation by using a mouse
  - Status bar with history
  - Page Setup
  - Revised Configuration Dialog
  - Online help
- **Print Function**
  - Printer session (for PC/3270:)
  - Graphics local print
  - Printing with the Windows printer drivers
  - Print function by printer definition table (PDT)
  - Multiple host-print functions in multiple sessions
  - PDF-to-PDT conversion tool
  - PC400 print function by OS/400® and i5/OS® Host Print Transform (HPT)
  - PC400 printing supported by the iSeries™, eServer™ i5, and System i5® Advanced Print Support Utility
  - ZipPrint
- **Programming Interfaces**
  - 16/32-bit Emulator High-Level Language Application Programming Interface (EHLLAPI)
  - 16/32-bit Dynamic Data Exchange (DDE)
  - 32-bit Node Operations Facility (NOF)
  - 16/32-bit Z and I Emulator for Windows API (PCSAPI)
  - 32-bit Common Programming Interface for Communications (CPI-C)
  - 32-bit Automation Object API
  - 32-bit ActiveX/OLE 2.0
  - ActiveX® Controls
- **PC400 Client Function**
  - Data transfer
  - PC Organizer
  - Text Assist
  - Enhanced Programmable Terminal User Interface (ENPTUI)

Problem Analysis

This chapter describes the information that will help you analyze problems with Z and I Emulator for Windows, and ways to report a problem to HCL. For detailed information about contacting HCL, refer to *Quick Beginnings*.

For information about Z and I Emulator for Windows and support, refer to the following Web sites:

- The HCL Software home page provides access to general product information, and download services. To view this page, go to the following Internet address:

  https://www.hcltech.com/software
- The HCL Z and I Emulator for Windows support page provides links to code fixes, tips, newsgroups, support options, and services. To view this page or to submit a software defect report, go to the following Internet address:

  https://hclpnpsupport.hcltech.com/csm

Z and I Emulator for Windows provides several utilities to help you with problem analysis. They can be invoked by selecting their icons from the **Programs → HCL Z and I Emulator for Windows → Administrative and PD Aids** subfolder on the Windows® **Start** menu.

The following sections describe these utilities and how to use them.

Log Viewer

The Z and I Emulator for Windows log viewer utility enables you to view, merge, sort, search, and filter information contained in message and trace logs. Use the log viewer during problem analysis to work with message and trace log entries. The default name of the message log output file is PCSMSG.MLG; its file extension must be .mlg. The file extension for trace logs must be .tlg.

To view message or trace logs:

1. From the **Administrative and PD Aids** subfolder, click **Log Viewer**; or, from an active session, click **Launch → Log Viewer** from the **Actions** menu.
2. From the list of logged messages, double-click a message to display the message text.

Note: Only one message log is created per machine. In simultaneous user environments such as WTS, all user messages are logged into that single instance of the log file. See Using Windows Terminal Services on page 402 for more information about terminal services.

For more information about log viewer functions, see Log Viewer Functions on page 408.

Trace Facility

The Z and I Emulator for Windows trace facility enables you to log trace information for certain Z and I Emulator for Windows functions.

To start a trace, perform the following steps:

1. From the **Administrative and PD Aids** folder, click **Trace Facility**; or, from an active session, select **Launch → Trace Facility** from the **Actions** menu. The trace status on the title bar displays the current state:

   Active

   Trace data is being collected by the trace facility.

   Inactive

   No trace data is being collected.
2. From the main dialog box, click **Set Up** to set the desired trace system parameters.
3. Click **OK** to return to the main trace dialog box.
4. From the main trace dialog box, select the type of data you want to trace from the **Function Name**, **Component Name**, and **Trace Option** list boxes.

   Function Name

   A specific set of Z and I Emulator for Windows features, such as 3270/5250 Emulator or User Services.

   Component Name

   The name of a specific part of a function, such as API data (for the 3270/5250 Emulator function) or Node Initialization (for the User Services function).

   Trace Options

   The options associated with a particular component, such as EHLLAPI (for the API component) or API trace (for the Node Initialization component).
5. Start tracing data by clicking **Start**, or apply changes to the trace options by clicking **Apply**.
6. Run the operation that you want to trace.
7. Optionally, stop the trace by clicking **Stop**.
8. Save the trace data to your hard disk by clicking **Save**.
9. Click **Format** to specify a formatted trace file name and to format the trace data. The Information Bundler utility should be used immediately after the trace is complete to ensure that the correct information is gathered.

   Note: If you have changed the default path setting for the formatted trace file, the Information Bundler will not find the trace information. Copy the trace files to the system-class application data directory.
10. Click **OK**.
11. Click **Clear** to clear the trace buffer where you saved a trace.
12. Use the log viewer to view the formatted trace log.

Note: Trace facility can capture only the application level (Ring 3) tracing, when logged on WTS environment, where the conso session ID is non-zero. If you want to trace the kernel level tracing (Ring 0), use the command line tracing options.

Information Bundler

The Z and I Emulator for Windows Information Bundler utility gathers system files, trace and log files, and registry information into a .ZIP file. This file can be sent to support personnel, using the Internet Service utility. The Information Bundler should be executed immediately after the trace is complete to ensure that the correct information is gathered.

Start Information Bundler using one of the following methods:

- Click **Administrative and PD Aids → Information Bundler** from the Z and I Emulator for Windows program menu.
- In an active emulator session, click **Actions → Launch → Information Bundler** from the menu bar.

The X12345.ZIP file is created in the Z and I Emulator for Windows system-class application data directory. This file contains system and Z and I Emulator for Windows information. Refer to the installation documentation for the location of the system-class application data directory for each Windows® operating system.

NOTE : Information Bundler utility requires dotnet version 4.6.1 to work.

Advanced Configuration, Management, and Operations

Advanced Configuration

This chapter describes facilities useful for deploying Z and I Emulator for Windows in large networks. Some of these facilities are handled by features of Z and I Emulator for Windows itself, while others are provided by external products, augmented with facilities provided by Z and I Emulator for Windows.

Configuration Files

The following sections describe the advanced configurations that you can make with the built-in files of Z and I Emulator for Windows. Advanced configurations enable you to easily configure and distribute common keywords and parameters to your client base, and include the following:

- Initial Configuration Definitions
- Configuration with Template and Update Files

Initial Configuration Definitions

Z and I Emulator for Windows enables network administrators to create an initial configuration definitions file that contains common configuration definitions for their clients. By using an initial configurations file, the administrator can distribute preconfigured definitions and have them automatically preloaded whenever a new configuration is created on a client.

The first step is to create a configuration using Start or Configure Sessions, or an ASCII editor. For detailed information on configuring sessions, refer to *Quick Beginnings*.

After you create the configurations file, rename the file to the appropriate reserved name. For workstation profiles (*.WS), the file name is PCSINIT.WS$.

After you rename the files, they can be distributed to client workstations. Put the files in the configuration files directory. The definitions in the files will be preloaded whenever a user creates a new configuration.

Note: The initial configuration file does not override parameter defaults for new definitions in new configurations, but preloads complete definitions into new configurations. Users can modify these definitions to get custom parameter values; however, the original initial configuration file remains unchanged.

Configuration File and Emulator Profile Directories

The default directory for configuration files is specified during installation. Configuration files can be used for all users or a specific user. Refer to *CD-ROM Guide to Installation* for details on specifying the initial default directory.

By default, Z and I Emulator for Windows searches for emulator profiles in the configuration files directory. You can use the **User Preference Manager** utility to indicate a different location for profiles.

Using Template and Update Files

When creating configurations for a large number of clients to implement, you can create a template configuration file that represents the common configuration elements for all clients. Using an update file with only those changes necessary for each client, you can distribute the template and update file and merge the two to create the target configuration.

The Z and I Emulator for Windows Server template and update files enable you to create or modify a configuration using an ASCII editor. You can configure all of the Z and I Emulator for Windows configuration keywords and parameters with update files.

Template files can ease the mass distribution of configurations to remote clients. A template file can specify the keywords which are common to several clients. For example, if you have multiple clients to configure, many of the keywords will be identical. You can create a template configuration file that reflects those common keywords.

You can use update files to add, modify, or delete keywords in a template file. The original template configuration file is left unchanged. An update file is merged into a template file by specifying the INCLUDE keyword at the end of the template file. For example, if an update file is named myconfig.chg, the last line of the template file that will use the update file is `INCLUDE=myconfig.chg`. When the template file and the update file are merged, you can give the resulting configuration file a name with the .ACG extension that distinguishes it from other .ACG files.

Key Fields

The key field is the parameter in a keyword that names the keyword and uniquely identifies it from other keywords of the same type.

Some keywords do not have key fields because they can only be specified once in a configuration file. An example of a keyword that can only be specified once is the NODE keyword.

Adding Keywords to a Template File

When using an update file to add a new keyword definition, you must provide the entire keyword. The key field must be provided along with a unique value. If any subfields are omitted from the keyword, the defaults for those fields are used. For example, to add a MODE keyword to the configuration, the update file might contain the following keyword:

```
MODE=(
     MODE_NAME=MYMODE
     COS_NAME=#INTER
     CRYPTOGRAPHY=NONE
     DEFAULT_RU_SIZE=1
     MAX_NEGOTIABLE_SESSION_LIMIT=128
     MAX_RU_SIZE_UPPER_BOUND=4096
     MIN_CONWINNERS_SOURCE=15
)
```

The content of the update file assumes that a MODE keyword with the parameter of MODE_NAME=MYMODE does not exist in the template. If it does, the parameters will be updated with the values provided in the update file.

If the MODE_NAME parameter is omitted from the update file, an error will occur during the configuration verification because the MODE_NAME parameter cannot be uniquely identified. Not all parameters available for the MODE keyword are specified in the update file. The remaining parameters use the defaults as specified in *Configuration File Reference*. The resulting addition to the configuration will look like this:

```
MODE=(
     MODE_NAME=MYMODE
     AUTO_ACT=0
     COMPRESSION=PROHIBITED
     COS_NAME=#INTER
     CRYPTOGRAPHY=NONE
     DEFAULT_RU_SIZE=1
     MAX_NEGOTIABLE_SESSION_LIMIT=128
     MAX_RU_SIZE_UPPER_BOUND=4096
     MIN_CONWINNERS_SOURCE=15
     PLU_MODE_SESSION_LIMIT=32
     RECEIVE_PACING_WINDOW=1
)
```

Modifying a Keyword in a Template File

When using the update file to modify an existing keyword definition, the original keyword should exist in the template file. If it does not exist in the template file, the update file adds an entry to the new configuration. You must specify the key parameter in the update file to identify the target keyword. Only those parameters specified in the update file keyword are updated in the template file's keyword. Parameters not specified in the update file are left unchanged. For example, if the following MODE keyword is in the template file:

```
MODE=(
     MODE_NAME=#INTER
     AUTO_ACT=0
     COMPRESSION=PROHIBITED
     COS_NAME=#INTER
     CRYPTOGRAPHY=NONE
     DEFAULT_RU_SIZE=1
     MAX_NEGOTIABLE_SESSION_LIMIT=256
     MAX_RU_SIZE_UPPER_BOUND=4096
     MIN_CONWINNERS_SOURCE=128
     PLU_MODE_SESSION_LIMIT=256
     RECEIVE_PACING_WINDOW=20
)
```

and the following keyword is specified in the update file:

```
MODE=(
     MODE_NAME=#INTER
     AUTO_ACT=10
)
```

the resulting configuration would have the following MODE keyword definition:

```
MODE=(
     MODE_NAME=#INTER
     AUTO_ACT=10
     COMPRESSION=PROHIBITED
     COS_NAME=#INTER
     CRYPTOGRAPHY=NONE
     DEFAULT_RU_SIZE=1
     MAX_NEGOTIABLE_SESSION_LIMIT=256
     MAX_RU_SIZE_UPPER_BOUND=4096
     MIN_CONWINNERS_SOURCE=128
     PLU_MODE_SESSION_LIMIT=256
     RECEIVE_PACING_WINDOW=20
)
```

Deleting a Keyword from a Template File

When using the update file to delete a keyword from the template, you must specify the key parameter and value that identify the keyword, along with the keyword DELETE. For example, if the template file specifies the following keyword:

```
MODE=(
     MODE_NAME=#INTER
     AUTO_ACT=0
     COMPRESSION=PROHIBITED
     COS_NAME=#INTER
     CRYPTOGRAPHY=NONE
     DEFAULT_RU_SIZE=1
     MAX_NEGOTIABLE_SESSION_LIMIT=256
     MAX_RU_SIZE_UPPER_BOUND=4096
     MIN_CONWINNERS_SOURCE=128
     PLU_MODE_SESSION_LIMIT=256
     RECEIVE_PACING_WINDOW=20
)
```

and the response file contains the following keyword:

```
MODE=(
     MODE_NAME=#INTER
     DELETE
)
```

the resulting configuration does not contain the #INTER mode definition.

The DELETE keyword can appear after a *parameter=value* specification or on a line by itself, either preceding or following the parameter. For example, the following uses of the DELETE keyword are valid:

```
MODE=(
     MODE_NAME=#INTER
     DELETE
)
MODE=(
     DELETE
     MODE_NAME=#INTER
)
```

The DELETE keyword cannot appear in front of a *parameter=value* specification on the same line. For example, the following uses of the DELETE keyword are not valid:

```
MODE=(
     DELETE MODE_NAME=#INTER
)

MODE=(
     MODE_NAME=#INTER DELETE
)
```

To delete all keywords of a particular type, or to delete one keyword that does not have a key field, only the keyword and the DELETE keyword are necessary. For example,

```
MODE=(
     DELETE
)
```

Automatic Device Name Generation (5250 Only)

The Telnet 5250 client function can generate a new and non-arbitrary DEVice NAME (DEVNAME) for a session without requiring per-session profile (.WS) customization or a user exit.

You can use keywords and special characters in the WorkStationID (WID) field (in the [5250] stanza of the Workstation profile) to cause some or all of the following information to be substituted into the DEVice NAME value that is sent to the TN5250 server:

- Computer name or user name
- Short session ID
- Session type ID
- Collision avoidance ID

When specified, the Collision Avoidance ID enables the generation of a new DEVice NAME if the Telnet server rejects a submitted name (which can occur when the old name is already in use on the iSeries™, eServer™ i5, or System i5™). The ability to have a variety of names generated allows multiple sessions to the same iSeries™, eServer™ i5, or System i5™ from one or more clients using just one WorkStation Profile (.WS) file. The definition of the existing .WS file parameter WorkStationID in the [5250] stanza is extended to accomplish this.

Substitution Characters

You can use special substitution characters in the WID field to control the placement of the generated information into the DEVNAME field. One substitution character is used in the WID for each generated character. This reserves space in the DEVNAME for each generated character and indicates where each generated character is to be placed. The three special substitution characters are:

Short Session ID

(value range: A-Z or a-z) The special character signifying this in the WID is the asterisk (*).

Example:

If the WorkstationID is configured as `123*` and the short ID of the first session is A, then the device names generated for the first three sessions will be `123A`, `123B`, and `123C`.

Session Type ID

(possible values: S for diSplay or P for Printer) The special character signifying this in the WID is the percent sign (%).

Example: If the Workstation ID is configured as `%123*` and the session type is Printer, then the first three device names generated would be `P123A`, `P123B`, and `P123C`.

Collision Avoidance ID

(value range: 1-9, A-Z or a-z) The Collision Avoidance ID (CAID) is used by the device name collision (DNC) function (see Device Name Collision Processing on page 385) to generate a new DEVice NAME when the old name is rejected by the Telnet server as already being in use. The special character signifying this in the WID is the equals sign (=).

Example: If the Workstation ID is configured as `%ABC=`, the session type is Display, and the device name `SABC1` is already in use on the iSeries™, eServer™ i5, or System i5™, then the first generated device name (SABC1) will be rejected by the server, but the second name (SABC2) will be accepted.

Client Naming Function

If you specify a Client Naming (CN) substitution keyword in the Workstation ID (WID) field, then an external name is retrieved and used when generating the DEVice NAME.

The CN keywords are prefixed with the ampersand character (&), followed by a five character identifier. Two keywords are supported:

&COMPN

Windows® COMPuter Name for the client

&USERN

USER Name specified during logon to the Windows® computer where the emulator executes

A name whose length exceeds the space remaining in the 10 character long DEVNAME field will have that excess trimmed from the left side by default. Excess characters can alternatively be trimmed from the right side by prefixing the CN keyword with a plus sign (+) character (for example, `+&COMPN`).

Note:

1. If the specified name cannot be obtained, then the message `Unable to get the local "x" name` (where "x" is COMPN or USERN) is displayed in the status bar.
2. If a client naming keyword is specified in the WID, then characters other than those defined for this feature are ignored.
3. A numeric character in the first position of a DEVNAME is invalid, and may be converted by the iSeries™, eServer™ i5, or System i5™ to the pound (or number) character (#).

Example A: If the Workstation ID is `&COMPN*` and the name of the local computer is `clientaccess1`, then the device names generated for the first three sessions would be `ntaccess1A`, `ntaccess1B`, and `ntaccess1C`.

Example B: If the Workstation ID is `+&COMPN*%` and the name of the USER logon for the local computer is `clientaccess1`, then the device names generated for the first three sessions would be `clientaccA`, `clientaccB`, and `clientaccC`.

Device Name Collision Processing

Device name collision occurs when a Telnet client sends the Telnet server a virtual device name, but that device name is already in use on the server. When this occurs, the Telnet server sends a request to the client asking it to send a different DEVNAME.

Device name collision (DNC) processing handles requests from the server for a different DEVNAME. If the collision avoidance ID (CAID) substitution character is present in the WID, the CAID is incremented and sent as part of the new DEVNAME to the server.

If the server requests a different DEVNAME and the CAID is not present in the WID, then the error message `Device Name "x" is invalid or already in use on the server` is displayed on the status bar and the session is disconnected.

Commands for Emulator Functions

Z and I Emulator for Windows provides the following commands for managing Z and I Emulator for Windows sessions:

PCOMSTRT

Start a Z and I Emulator for Windows session

PCOMSTOP

Stop a Z and I Emulator for Windows session

PCOMQRY

Query Z and I Emulator for Windows sessions

Returns: DOS Error level is set for use when this command is invoked by a program. When the command is directly entered, a message is displayed indicating the session is stopping.

Configure OneDrive

OneDrive is a Microsoft Cloud service that provides access to all your Z and I Emulator for Windows configuration files from anywhere. It also stores and protects your files and allows you to share them with other users. When a user uses a different account with OneDrive, all the configured files get synced with the OneDrive user account.

User can also configure OneDrive with Z and I Emulator for Windows profile directory, Macro directory and Tool Bar icons directory. For more details, refer to below sections.

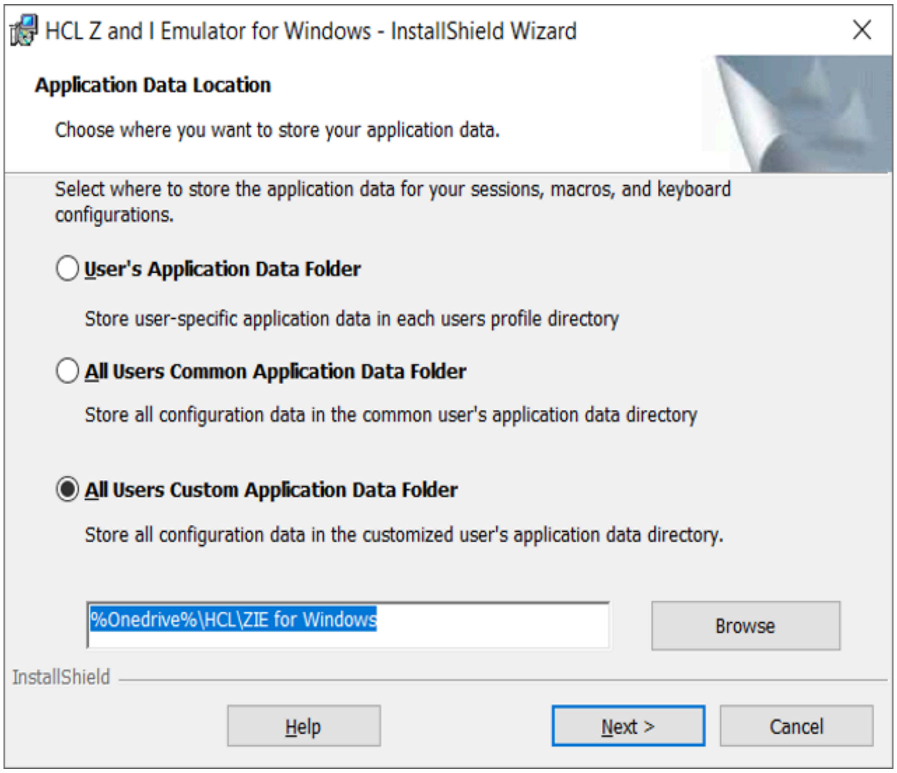
Configure Profile Directory with OneDrive

A user can configure profile directory with OneDrive either during Z and I Emulator for Windows installation using %OneDrive% variable or after installation through Preference Utility.

Execute the following steps to configure OneDrive while installing Z and I Emulator for Windows:

Note: Make sure that the user is configured with OneDrive and able to access %OneDrive% configured path.

1. Set %OneDrive% as profile directory during installation.
2. Select all users **“Custom Application Data Folder”** option and give the variable path.



HCL Z and I Emulator for Windows - InstallShield Wizard

Application Data Location

Choose where you want to store your application data.

Select where to store the application data for your sessions, macros, and keyboard configurations.

○ **User's Application Data Folder**

Store user-specific application data in each users profile directory

○ **All Users Common Application Data Folder**

Store all configuration data in the common user's application data directory

● **All Users Custom Application Data Folder**

Store all configuration data in the customized user's application data directory.

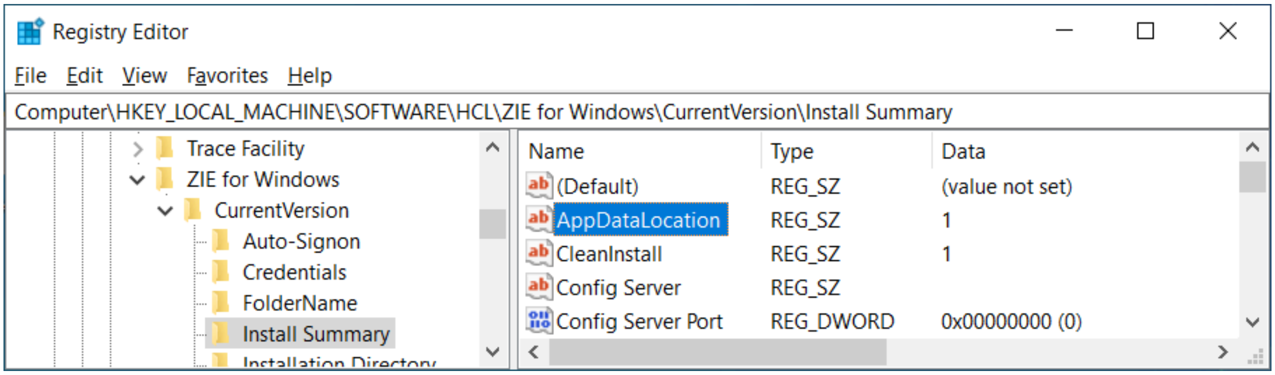
%Onedrive%\HCL\ZIE for Windows

3. After successful configuration, a user can load all session files from OneDrive.

Execute the following steps to migrate the files to OneDrive after installing Z and I Emulator for Windows:

Note: Make sure that **AppDataLocation** registry is set to 1.

Registry path: *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Personal Communications \CurrentVersion\Install Summary*



Registry Editor

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\HCL\ZIE for Windows\CurrentVersion\Install Summary

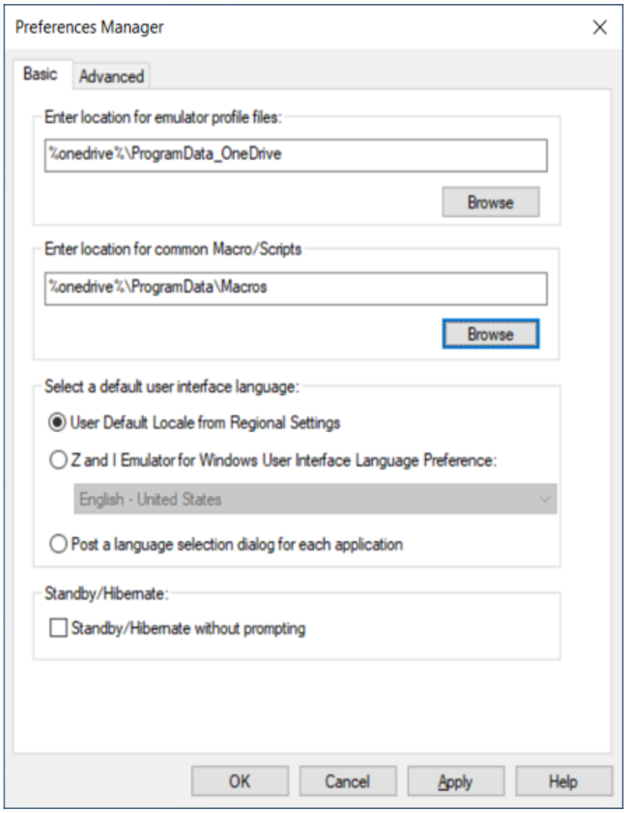
Name	Type	Data
(Default)	REG_SZ	(value not set)
AppDataLocation	REG_SZ	1
CleanInstall	REG_SZ	1
Config Server	REG_SZ	
Config Server Port	REG_DWORD	0x00000000 (0)

1. Open the Preference Utility.
2. Enter the **%OneDrive%** variable path under **Enter location for emulator profile files** field and click **OK**.
3. After successful configuration, a user can load all session files from OneDrive.

Configure Z and I Emulator for Windows Tool bar icons Directory with OneDrive

User can configure the “preferences” and load the customize icon/bitmap from “OneDrive”. Set the Keyword which needs be added under pcswin.ini as follows:

1. Open the **pcswin.ini** file.
2. Under the **[ToolBar]** section, set the **IconPath** to OneDrive path.



Enter location for emulator profile files: %onedrive%\ProgramData_OneDrive

Enter location for common Macro/Scripts: %onedrive%\ProgramData\Macros

Select a default user interface language:

● User Default Locale from Regional Settings

○ Z and I Emulator for Windows User Interface Language Preference: English - United States

○ Post a language selection dialog for each application

Standby/Hibernate:

□ Standby/Hibernate without prompting

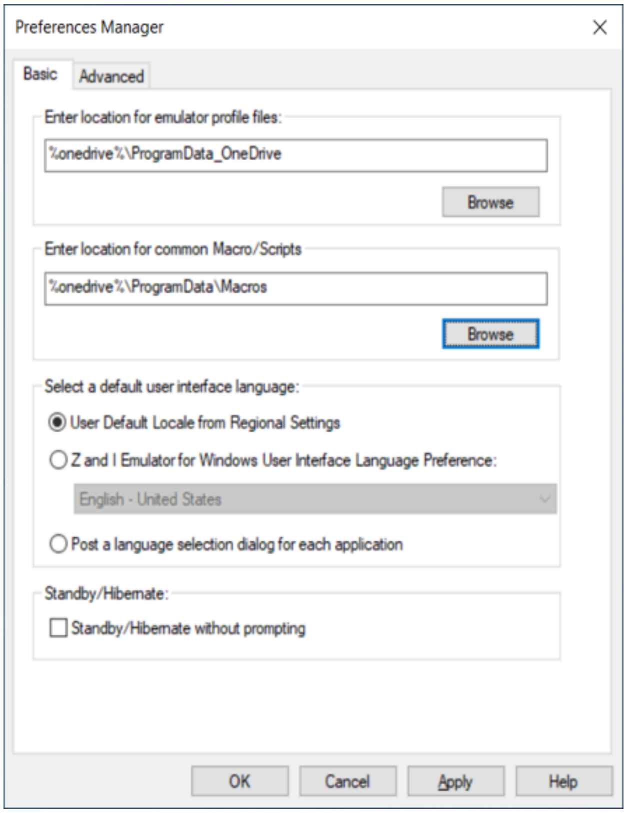
Once user configures the above keyword in pcswin.in, the tool bar files (*.bar) use the OneDrive path to load all the custom icons ignoring the local custom tool bar icon file path. The icon files are looked up in “IconPath” location with its name. If the file is not found, the user is informed with a message box and loads the default icon.

Configure Macro Directory with OneDrive

User can configure the preference and load the macros from “OneDrive”. Set the Keyword which needs be added under pcswin.ini as follows:

1. Open the **pcswin.ini** file.
2. Under the **[Macro]** section, set the **DIR** to OneDrive path.

Note: User can configure the same under “Enter location for common Macro/scripts” through **Preference Utility**.



Preferences Manager

Enter location for emulator profile files:

%onedrive%\ProgramData_OneDrive

Enter location for common Macro/Scripts

%onedrive%\ProgramData\Macros

Select a default user interface language:

● User Default Locale from Regional Settings

○ Z and I Emulator for Windows User Interface Language Preference:

English - United States

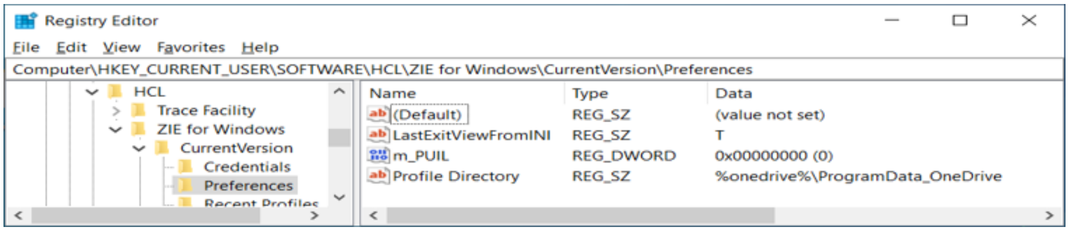
○ Post a language selection dialog for each application

Standby/Hibernate:

□ Standby/Hibernate without prompting

Limitations:

1. The given %onedrive% variable path will be expanded to full path in **Preferences** for a user**.** The variable path is stored in the following directory.
   Computer\HKEY_CURRENT_USER\SOFTWARE\HCL\ZIE for Windows\CurrentVersion\Preferences



Registry Editor

Computer\HKEY_CURRENT_USER\SOFTWARE\HCL\ZIE for Windows\CurrentVersion\Preferences

Name	Type	Data
(Default)	REG_SZ	(value not set)
LastExitViewFromINI	REG_SZ	T
m_PUIL	REG_DWORD	0x00000000 (0)
Profile Directory	REG_SZ	%onedrive%\ProgramData_OneDrive

2. All configuration files associated with session files can use the variable path, any change in session will rewrite the session file with full path.

Start a Z and I Emulator for Windows Session

The command PCOMSTRT has the following parameters:

/p

Name of workstation profile to start (required). The syntax is /p=*workstation-profile*. You can specify the workstation profile as either the path (drive, directory, and file name) or just the file name, in which case the location of the workstation profile file is the user-class application data directory.

Note: If multiple `/p` parameters are given, PCOMSTRT only uses the last one to start a profile (.WS file).

/s

Session letter of the session to start. The syntax is /s=*session-letter*. This is optional. If omitted, the first available session letter is used.

/w

Session window startup state. The syntax is /w={0|1|2|3}.

0

Hidden

1

Normal (default)

2

Minimized

3

Maximized

/q

Quiet mode. In quiet mode, PCOMSTRT does not write any messages to stdout.

/nowait

Do not wait for session to start. The /nowait option tells PCOMSTRT to complete execution without waiting until the emulator session is started. There is no /wait option; the default is to wait until the session is started.

/?

Displays help information.

Returns: DOS Error level is set for use when this command is invoked by a program. When the command is directly entered, a message indicating the session is starting is displayed.

Stop a Z and I Emulator for Windows Session

The command PCOMSTOP has the following parameters:

/s

Session letter of session to stop. The syntax is /s=*session-letter*. This is optional. If omitted, the first available session letter is used.

/all

Stops all sessions

/NCE

Stops all or specific session without confirmation, even if the confirmation on exit or exit all option are set.

/q

Quiet mode. In quiet mode, PCOMSTOP does not write any messages to stdout.

/?

Displays help information.

Query Z and I Emulator for Windows Sessions

The command PCOMQRY has the following parameters:

/s

Session letter of session to query, The syntax is /s=*session-letter*. This is optional. If omitted, the first available session letter is used.

/all

Queries all sessions.

/q

Quiet mode. In quiet mode, PCOMQRY does not write any messages to stdout.

/nowait

Do not wait for session to query. The /nowait option tells PCOMQRY to complete execution without waiting until the emulator session is queried. There is no /wait option; the default is to wait until the session is queried.

/?

Displays help information.

System Policy Support

System policies allow the user to control the actions that a user is permitted to perform.

Any application or component can define a policy. The policy appears in the administrator user interface; information that the user sets about the policy migrates to the local computer's registry. The application or component that defines a policy must check the registry to enforce its policy.

We provide support for both*.ADM* and*.ADMX* policy template files. These template files allow the users to enforce policy restrictions on various features—these files are package under the **Docs-Admin-Aids**.

Using ADM File Format:

1. Categories, policies, and parts are described in a policy template (*.ADM) file. The *Microsoft Resource Kit* includes three policy template files: WINNT.ADM, COMMON.ADM, and WINDOWS.ADM. Applications or components can also provide their own policy template files.

   Note: Z and I Emulator for Windows supplies a policy template for each language currently supported. For example, the policy template for the US English language is named ENUPOL.ADM and the policy template for the French language is named FRAPOL.ADM.

2. The user run the policy editor, which reads one or more policy templates and lists the available categories and policies. The user set up the desired policies, and the policy editor uses registry functions to save the work to a policy (*.POL) file. A group policy editor is provided with Windows®. Documentation about the use of Microsoft® policy editors is found at http://www.microsoft.com.
3. After the user logs on (and user profiles are reconciled if they are enabled), the policy downloader determines where to find the file on the network, opens the policy file, and merges the appropriate computer, user, and user group policies into the local registry.

Using ADMX File Format:

ADMX file is a Windows Group Policy settings XML-based file that serves as a replacement for the older ADM file type. ADMX files were introduced in Windows Vista and Windows Server 2008 and are referred to as “Administrative Template XML-Based files”.

Z and I Emulator for Windows provides a policy template files (***pol.ADMX and ***pol.ADML) for each language. The templates are on the **Docs-Admin-Aids** in the admin directory. The admin users configure the desired policies to enforce policy restrictions on various features.

ADMX files must be copied under the “**C:\Windows\PolicyDefinitions”.** To import Z and I Emulator for Windows policy ***pol.ADMX template files, copy them to the “**PolicyDefinitions”** folder.

Each ADMX has a corresponding language ADML policy template file and must be copied under the respective language folder under “**PolicyDefinitions”.**

For example: For US English language, please copy the enupol.ADML files under

“**en-US**” is the subfolder under “PolicyDefinitions”.

Run the windows group policy editor which reads the imported ADMX/ADML file and a new entry of “HCL Z and I Emulator for Windows” is populated under the “Administrative Templates”.

The users can navigate to the “HCL Z and I Emulator for Windows” policy categories and modify configuration settings. Find the following **Group Policy** options:

- **Not Configured**: This option does not enforce the policy restriction.
- **Enabled**: This option does not enforce the policy restriction.
- **Disabled**: This option enforces the policy restriction.

Note: When using the group policy editor provided with Windows 2000, the Not Configured setting allows the same permission or access to features as the Enabled setting. HCL Z and I Emulator for Windows provides its policy template file (PCSPOL.ADM).
It contains one category of type USER. Within the HCL Z and I Emulator for Windows category are the following policies:

- **Configuration**: Contains policy information related to configuration.
- **Execution**: Contains policy information related to the execution.
- **Installation**: Contains policy information related to removing HCL Z and I Emulator for Windows.
- **View**: Contains policy information related to changing the session window view.

Disable Licensing:

This policy controls the user's ability to disable the licensing.

The group policy editor has the following options to enforce ***"Disable Licensing"***:

- **Not Configured**: This option does not enforce the policy restriction for ***"Disable Licensing"***.
- **Enabled**: This option enforces the policy restriction for ***"Disable Licensing"***.
- **Disabled**: This option does not enforce the policy restriction for ***"Disable Licensing"***.

Note: If a user enforces the ***"Disable Licensing"*** through the group policy editor, it won't allow a user to enable ***"Disable Licensing"*** through Preference Manager.

Execution Policy

This category contains policy information related to execution.

Dynamic Menu Modification

This policy controls whether or not DDE applications executed by the user are permitted to dynamically add themselves to the menu of an active session.

Java™ Applet

This policy controls the user's ability to execute Java™ applets from the **Actions → Run Java™ Applet** menu.

Macro Play/Record

This policy controls the user's ability to play and record macros.

The “Not Configured” and “Disabled” options under the “Policy Editor” it does not impose any policy restriction.

The Macro Play/Record drop-down list has the following options:

No Access

The user has no access to macros; that is, the user cannot play or record macros. This is the most restrictive level of control.

Play

The user can play macros.

Record

The user can play and record macros.

Start Session

This policy controls the user's ability to minimize an emulator session window.

Minimize Session

This policy controls the user's ability to minimize an emulator session window.

Maximize Session

This policy controls the user's ability to maximize an emulator session window.

Close Session

This policy controls the user's ability to close an emulator session window.

Delete Session

This policy controls the user's ability to delete an emulator session from the Session Manager window.

Product Update

This policy controls the user's ability to start the Product Update Tool.

Detect and Repair

This policy controls the user's ability to utilize the Help Detect and Repair function.

File Transfer

This policy controls the user's ability to send or receive files from the host.

Installation Policy

This category contains policy information related to installing or removing Z and I Emulator for Windows.

Uninstall

This policy controls the user's ability to uninstall HCL Z and I Emulator for Windows.

View Policy

This category contains policy information related to changing the session window.

Menu Bar

This policy controls whether the user can view the menu bar.

Select Disable to hide the menu bar. The Not Configured or Enabled settings retain the menu bar in the session window.

Tool Bar

This policy controls whether the user can view the tool bar.

Select Disable to hide the tool bar. The Not Configured or Enabled settings retain the tool bar in the session window.

Status Bar

This policy controls whether the user can view the status bar.

Select Disable to hide the status bar. The Not Configured or Enabled settings retain the status bar in the session window.

Expanded OIA

This policy controls whether the user can view the expanded OIA.

Select Disable to hide the status bar. The Not Configured or Enabled settings retain the expanded OIA in the session window.

Quick Connect Bar

This policy controls whether the user can view the Quick Connect Bar. Select Disable to hide the Quick Connect Bar. The Not Configured or Enabled settings retain the Quick Connect Bar in the session window.

Configuring and Using Security for Z and I Emulator for Windows

Z and I Emulator for Windows provides session security using Microsoft CryptoAPI (MSCAPI). These packages enable use of the Transport Layer Security (TLS) security protocols.

The configuration information in this chapter usually applies to TLS. See Using Transport Layer Security on page 401 for more information.

You can display information about the security aspects of your session by clicking **Communication → Security Information** from the session menu bar. This provides details about the certificates exchanged during TLS negotiations between client and server.

A TLS session is established in the following sequence:

1. The client and the server exchange hello messages to negotiate the encryption algorithm and hashing function (for message integrity) to be used for the session.
2. The client requests an X.509 certificate from the server to verify the identity of the server. Optionally, the server can request a certificate from the client (known as *Client Authentication*).

   The digital signature of the certificate authority (CA) is authenticated using a published *root* certificate of the issuing CA. The client automatically decrypts certain information on the presented certificate using a *public* key on the CA's root certificate. This step is successful only when the presented certificate was encrypted using a well-guarded, unique, and corresponding *private* key, known only to the CA. This process can detect (and reject) intentional alterations (forgeries) and the rare garbling that can occur over data circuits. Z and I Emulator for Windows also allows users to use self-signed certificates for this purpose.
3. Once the certificate-issuer authentication step succeeds, the client and server negotiate for an encryption key to be used during the ensuing data exchange session. The client randomly generates a set of keys to be used for encryption. The keys are encrypted with the server's public key and are securely communicated to the server.

When a secure connection is established, a padlock icon is displayed in the Z and I Emulator for Windows status bar. Depending on the level of encryption, the icon is accompanied by a number (0, 40, 56, 128, 168, 256). If the session is not TLS-based, the icon shows as unlocked.

Certificates

Security is controlled by digital certificates that act as electronic ID cards. The purpose of a certificate is to assure a program or a user that it is safe to allow the proposed connection and (if encryption is involved) to provide the necessary encryption/decryption keys. They are usually issued by Certificate Authorities (CAs), which are organizations that are trusted by the industry as a whole and who are in the business of issuing of Internet certificates. A CA's certificate, which is also known as a root certificate, includes the CA's signature and a validity period, among other things.

Encryption and authentication are performed by means of a pair of keys, one public and one private. The public key is embedded in a certificate, known as a site or server certificate. The certificate contains several items of information, including the name of the Certificate Authority (CA) that issued the certificate, the name and public key of the server or client, the CA's signature, and the date and serial number of the certificate. The private key is created when you create a self-signed certificate or a CA certificate request and is used to decrypt messages from clients.

Managing Certificates in the Microsoft Certificate Stores

In order to connect secure sessions using the Microsoft CryptoAPI (MSCAPI) security package, the appropriate certificates must exist in the Microsoft Certificate Stores. To connect to a secure host, the root certificate of the host certificate's verification chain must be in the Trusted Root Certification Authorities store. To connect a secure client authentication session, the client certificate must be in the Personal store.

To add, remove, and view certificates in the Microsoft Certificate Stores, select **Internet Options** in the Windows Control Panel. On the **Content** tab, click **Certificates**. The tabs represent the different Microsoft Certificates Stores. Each tab shows the certificates that exist within each store.

To add a certificate to a store, click **Import**; the **Certificate Import** wizard helps you import certificates from a file. The Import wizard can import certificates from several types of certificate files, including the ARM, DER, and P12 formats that can be extracted or exported from the Certificate Management utility.

Configuring and Using Secure Sockets Layer

The purpose of basing communications on TLS is to provide privacy and integrity during communication over an unsecured TCP/IP connection between a client and a target server. This section briefly describes how to configure the Z and I Emulator for Windows client to use this mode.

Preparation for TLS Communication

There is a division of labor for TLS configuration tasks. The configurations of the client and the server are coordinated to achieve the required compatibility. The following sections describe the preparation tasks required for client configuration and server configuration.

Client Configuration

The following elements must be configured on the client side to enable TLS:

- Security must be enabled in order to operate in TLS mode. A client operating in TLS mode cannot establish a connection with a server that is operating in ordinary Telnet mode. Likewise, a client operating in ordinary Telnet mode cannot establish a connection with a server operating in TLS mode. See Configuring Z and I Emulator for Windows Session Security on page 398 for information about enabling security.
- Select **Send Personal Certificate to Server if Requested** on the **Security Setup** property page for client authentication. If this option is not selected, only server-side authentication is performed. If the server requests a client certificate and this option is not selected, there will be no active connection. See Opening a Key Database and Adding a Root Certificate for details.

To add, remove, and view certificates in the Microsoft Certificate Stores, select **Internet Options** in the Windows Control Panel. On the **Content** tab, click **Certificates**. The tabs represent the different Microsoft Certificates Stores. Each tab shows the certificates that exist within each store.

Establishing a Secure Session

Upon establishing a preliminary connection with a target server, the Z and I Emulator for Windows client is presented a certificate by that server; if you have enabled client certificate authentication, your certificate is likewise presented to the server. The digital signature of the CA is authenticated using a published *root* certificate of the issuing CA. The client automatically decrypts certain information on the presented certificate using a *public* key on the CA's root certificate. This step is successful only when the presented certificate was encrypted using a well-guarded, unique, and corresponding *private* key, known only to the CA. This process can detect (and reject) intentional alterations (forgeries) and the rare garbling that can occur over data circuits.

Z and I Emulator for Windows also allows users to use self-signed certificates for this purpose.

Note:

Once this certificate-issuer authentication step succeeds, the client and server negotiate to agree on an encryption key to be used during the ensuing data exchange session.

Configuring Z and I Emulator for Windows Session Security

Whether you are configuring a TN3270, TN5250, or VT session, the underlying protocol must be TCP/IP. Use the following procedure to enable security:

1. Start a workstation profile from the Session Manager; or, from an active session, click **Configure** from the **Communication** menu. When the dialog box opens, click **Configure**.
2. In the Customize Communication panel, choose the appropriate Type of Host, Interface, and Attachment values for the desired Telnet host.
3. Click **Link Parameters**.
4. On the **Host Definition** property page, do the following:
    a. Specify the normal host name and LU parameters under **Primary**.
    b. Specify the **Port Number** under **Primary**. It is likely that it will not be the default port value for Telnet. The administrator of the destination server might have set up a specific port number to handle TLS/SSL service.
5. On the **Security Setup** property page, check **Enable Security**.

    For server authentication only, no additional setup is required. For client authentication, proceed to the next step.
6. For 3270 sessions, select the **Telnet-negotiated** option to have Z and I Emulator for Windows negotiate security with the Telnet 3270 server. See Negotiated Telnet Security on page 402 for details. If Enable Security is unchecked, the Telnet-negotiated option cannot be selected.
7. On the **Security Setup** property page, select the **Microsoft CryptoAPI (MSCAPI)** security package.

    Note: To avoid the need of manually adding host certificate into the Microsoft Certificate Store, refer to **Pass Through Certificate Validation**.
8. To protect against security vulnerability in RC4 stream cipher, the FIPS (Federal Information Processing Standard) mode has been made mandatory.

    For MSCAPI, refer to the vedor documentation for the latest information.

> **Note:** Follow the below steps to enable AES support with MSCAPI on Windows® 8, Windows® 8.1, Windows® 10, Windows® Server 2008, and Windows® Server 2012.
>
> a. From an administrator account, open **Group Policy Editor** (gpedit.msc).
> b. Choose **Computer Configuration**->**Administrative Templates**->**Network**->**SSL Configuration Settings**.
> c. Open **SSL Cipher Suite Order** and select **Enabled**.
> d. Alter the cipher order as per you organization's needs, save the changes, and REBOOT the system for the above changes to apply.
>
> It is important to note that the client can only present the server a prioritized cipher list. The host has the final say on what gets selected as the cipher for the session. When choosing an algorithm with a specific a bit length, one important consideration is to remember that encryption and decryption are CPU intensive operations which take time depending upon the key size. In almost all cases, a 128-bit key is more than sufficient to protect the information you are exchanging over your telnet connections.

9. Enable **Check for Server Name and Certificate Name Match** to have the session authenticate the server by matching the server name to the host or server certificate name. The server and certificate names must match exactly. For MSCAPI sessions, if the certificate name and server name do not match, an error is returned.
10. In the **Client Authentication** group box, you determine when and how the client certificate will be chosen for sending to the server.

    If you want to enable client authentication and have the personal client certificate from the key database file sent to the server when requested, check **Send Personal Certificate to Server if Requested**.

    Send Personal Certificate Trusted by Server

    Select this option if you do not want to be prompted to select a personal client certificate from a key database file. Z and I Emulator for Windows will send the personal client certificate trusted by the server.

    Send Personal Certificate based on Key Usage

    Use this option to select one or more key usages. Click **Key Usage** to select the defined Object ID (OID) key usages. Go to the **Extended Key Usage** panel to add a new OID and description to the list.

    At authentication time, Z and I Emulator for Windows chooses certificates for client authentication, based on the key usage that you select. If a certificate's Enhanced Key Usage attribute contains one or more of the OIDs that you specify, the certificate is eligible for use.

    If no eligible certificates are found, the authentication fails. If one eligible certificate is found, it is automatically used. If two or more eligible certificates are found, you will be prompted to select a personal client certificate.

Select or Prompt for Personal Client Certificate

Use this option if you want to choose the personal client certificate. You will be prompted to select a personal client certificate during session establishment, when the server requests the client certificate.

To preselect a personal client certificate during configuration, click **Select now** and choose the **Personal Certificate Label**.

Pass Through Host Certificate Validation

Use this option to disable the default certificate validation process during TLS handshake. Applicable only for Microsoft schannel provider.

Note: By default, schannel (MSCAPI) is responsible for validating the host certificate chain received during TLS handshake. Schannel runs several checks on the received certificate chain one of which is verifying that the signature affixed to the certificate valid, that is, the hash value computed on the certificate contents matches the value that results from decrypting the signature field using the public component of the issuer. In order to perform this operation, the user must possess the public component of the iss either through some integrity-assured channel, or by extracting it from another (validated) certificate. The default certificate valid process is exhaustive and runs several checks on the host certificate chain in order to successfully validate it. By enabling this option the user would effectively suppress the default validation done by schannel and the identity of the host is not verified. The use of this option is not recommended.

Problem Determination

Following is some information to help you avoid problems that might be related to TLS configuration.

- With server-side authentication, the common name in the sever's certificate is always compared to the name you type in the Host Name field on the client. These names must match exactly. You cannot:
  - Type the IP address in one place and the host name in the other
  - Type wrt05306 in one place and WTR5306 in the other

  Note: This information is available only from the target server administrator.
- Make sure that TLS is enabled on both the Z and I Emulator for Windows client and the TLS server.
- Makes sure that the port number in the Advanced configuration panel on the client matches the port number defined in the server.
- For each different server using a self-signed certificate, you must add a copy of each of the server certificates to your keyring.

- Be sure there is a root certificate of the proper class to correspond with the class and issuer of the certificate on the server.
- Make sure that the password to your key database has not expired.

Note: Notify your server administrator of any problems prior to contacting HCL Service.

Using Transport Layer Security

Z and I Emulator for Windows allows you to negotiate the Transport Layer Security 1.0 protocol. The TLS protocol is based on the SSL protocol. TLS differs from SSL mainly in the initial handshake protocol for establishing client/server authentication and encryption. TLS also allows you to use **FIPS** (Federal Information Processing Standard) mode. Although TLS and SSL do not operate with each other, TLS provides a mechanism by which a TLS 1.0 implementation can revert to SSLv3.

The TLS protocol uses public-key and symmetric-key cryptographic technology. Public-key cryptography uses a pair of keys, one public and one private. Information encrypted with one key can only be decrypted with the other key. For example, information encrypted with the public key can be decrypted only with the private key. Each server's public key is published, while the private key is confidential. To send a secure message to the server, the client encrypts the message by using the server's public key. When the server receives the message, it decrypts the message with its private key.

Symmetric-key cryptography uses the same key to encrypt and decrypt messages. The client randomly generates a symmetric key to be used for encrypting all session data. The key is then encrypted with the server's public key and sent to the server.

TLS provides three basic security services:

Message privacy

Achieved through a combination of public-key and symmetric-key encryption. All traffic between a client and a server is encrypted using a key and an encryption algorithm negotiated during session setup.

Message integrity

Ensures that session traffic does not change while in route to its final destination. TLS and SSL use a combination of public/private keys and hash functions to ensure message integrity.

Mutual authentication

Exchange of identification through public-key certificates. The client and server identities are encoded in public-key certificates, which contain the following components:

- Subject's distinguished name
- Issuer's distinguished name
- Subject's public key
- Issuer's signature

- Validity period
- Serial number

Negotiated Telnet Security

Typically, a secure channel is established before the Telnet 3270 server and client negotiate for a session. For Z and I Emulator for Windows 3270 sessions, you can use negotiated Telnet security. This option enables Z and I Emulator for Windows to establish security with the Telnet 3270 server during the Telnet negotiation. The security protocol used is TLS 1.0, TLS 1.1, and TLS 1.2.

If you select the **Telnet-negotiated** option on the Security Setup panel, the Telnet connection is established. Z and I Emulator for Windows then uses the *TLS-based Telnet Security* protocol (as defined by IETF) to negotiate the TLS security. If the connection is successful, a status message is returned.

This support is only applicable with a Telnet server which supports the *TLS-based Telnet Security* protocol. By default, this option is not enabled.

Using Windows Terminal Services

Windows® Terminal Services (WTS) is a feature that allows more than one user to log onto a Windows machine. This functionality is also referred to as *simultaneous user environments* or *terminal services*.

Users can logon to the Windows® machine from the console (the screen attached to the Windows® machine) or from a remote desktop client.

On Windows® 8, Windows® 8.1, and Windows® 10, more than one user can logon at the console, although only one of the logged-on users will be able to see his desktop at any time. This terminal services function is called *fast user switching*.

Citrix MetaFrame allows administrators to configure their WTS servers to run individual applications, and to configure each user to run different applications on different servers. Thus, instead logging on to a WTS server, a user launches preconfigured applications from his Citrix environment running on their client machine. Citrix then logs onto the appropriate WTS server and runs the application. When the user closes the application, Citrix shuts down the application, waits for all processes that were started by the application to terminate, and then logs off from the WTS server.

Note:

1. A message file log is not maintained for each WTS logon session. Only one message log file is maintained for the WTS server.
2. When Z and I Emulator for Windows is used in a WTS environment, the maximum of 52 sessions (A-Z or a-z) applies to each WTS logon session. There is not a limit of 52sessions per WTS server.

Session IDs

In simultaneous user environments such as WTS, each time a user logs on, that specific logon session is assigned a session ID. When a user logs on at the console, that session is assigned the session ID **0** (zero)—this is also called the **server console** session. When a user logs on from a remote desktop, the assigned session IDs begin with session 1.

Trace Facility

For simultaneous user environments such as WTS, the **Connectivity** functions are only available for users logged in as session 0.

Each user can run his own trace facility, which gives information from that user's specific WTS logon session. However, there are trace options that enable tracing from device drivers, which are not associated with any specific WTS logon session. Thus, those options only appear on the trace facility that is started in the WTS console session (session 0).

Azure Virtual Desktop

Microsoft Azure, often referred to as Azure, is a cloud computing service operated by Microsoft for application management via Microsoft-managed data centres.

“Azure Virtual Desktop” (Windows Virtual Desktop) is a desktop and app virtualization service that runs on the cloud. Virtual machines (VMs) for multiple users and desktops get hosted on the Azure platform in an Azure Virtual Desktop environment. Organizations can use Azure Virtual Desktop service to deliver virtual applications and desktops to their employees via Azure’s cloud infrastructure.

Azure Virtual Desktop provides an application virtualization to any personal device with internet connection.

We can provide right access controls to the users and devices with Azure Active Directory Conditional access. On providing the access, users can launch any Azure Virtual Desktop client to connect to their published Windows applications.

Z and I Emulator for Windows supports the Windows Virtual Desktop. ZIEWin application can be published as a windows application using the WVD setup and we can access all the features of ZIEWin.

For more information on Azure Virtual Desktop : https://docs.microsoft.com/en-us/azure/virtual-desktop/overview.

Express Logon

Bypass Signon Using Password Substitute (5250)

This option enables the user to bypass iSeries™ login screen by sending a SHA1 password substitute.

Note: This option works only when the `QPWDLVL` system value at the iSeries™ is either 2 or 3. A change to this system value takes effect at the next IPL. To see the current and pending password level values, use the **Display Security Attributes** (DSPSECA) command. The `QRMTSIGN` system value, which specifies how the system handles remote sign-on requests, als needs to be set to `*VERIFY`.

The credentials are encrypted and saved in the current user's registry hive on the local computer. The user is prompted for a password in case the password stored in the registry is no longer valid. The newly entered password shall be stored in the registry and used for subsequent bypass logins.

Users can update the existing password in the registry or add a new password by using **Update registry with bypass login credentials...** menu item present in the Actions menu. This option is used whenever an user changes the password on the host; for example, at the time of password expiry.

If bypass login is enabled, ZIEWin prompts the user for a password in case the password stored in the registry is incorrect. The newly entered password is to be stored in the registry and used for subsequent bypass logins.

Also, there is a new menu option added to Actions menu called **Update registry with bypass login credentials...**, which allows users to update the existing password in the registry or add a new password corresponding to a particular hostname or IP address.

In case the password stored in the registry is expired and bypass login is enabled and the user logs in, the password change screen displays so that the user can set a new password.

When the user change the password successfully (registry still contains the old or expired password) , log out, disconnect, and try bypass login again, the old or expired password is still taken from the registry and used for the bypass login. Due to the invalidity of the old password from the registry, the login fails for the first time and ZIEWin prompts the user for new password. When the user enter the new password that has been created before reglogin, the correct password is to be stored in the registry and ZIEWin reconnects using the new password.

Kerberos Services Ticket Auto-Signon

For 5250 emulator sessions, the **Bypass signon using Kerberos principal** option enables Kerberos authentication.

- If the “Kerberos auto-signon” is disabled during the “Custom” installation, “Bypass signon using Kerberos principal” is disabled.
- If the “Kerberos auto-signon” is enabled during the “Custom” installation, “Bypass signon using Kerberos principal” is enabled.

A ticket is generated and passed to the iSeries™, eServer™ i5, or System i5™ host during TN5250 negotiation.

If the ticket is valid, authentication is completed and the user is logged onto the host. If authentication fails, a host login screen get displayed.

Note: The user must log into a Windows™ domain in order to use Kerberos authentication. Refer to the relevant Microsoft™ documentation for specific details.

For the Data Transfer utility, the user can set the **Use Kerberos principal, no prompting** option (from **Setup → Signon Options**).

- If the “Kerberos auto-signon” is disabled during the “Custom” installation, “Use Kerberos principal, no prompting” is not listed in the signon options of the Data Transfer utility.
- If the “Kerberos auto-signon” is enabled during the “Custom” installation, “Use Kerberos principal, no prompting” is listed in the signon options of the Data Transfer utility.

This function enables Kerberos authentication, using the ticket generated by the Windows™ user credentials.

Certificate Express Logon

Certificate Express Logon (formerly known as Express Logon Feature or ELF) enables a Z and I Emulator for Windows Telnet 3270 user to securely logon to a host application without sending the User ID and password. One advantage of using this function is that it reduces the time you spend maintaining host user IDs and passwords. It also reduces the number of user IDs and passwords that the users have to remember.

To use Certificate Express Logon, the host session must be configured for SSL and client authentication. This means the client must have a valid client certificate. The SSL connection must be made to one of the supported Telnet 3270 servers.

Using Certificate Express Logon

When starting a session using Certificate Express Logon, Z and I Emulator for Windows establishes an SSL client authentication session with the Telnet 3270 server. During the logon process, a macro with the Certificate Express Logon information is played. Once the session is established, Z and I Emulator for Windows sends the application ID for the application that the user is accessing to the Telnet 3270 server. This information is contained in the logon macro. The Telnet 3270 server uses certificate information from the SSL connection and the application ID received from Z and I Emulator for Windows, and requests the user ID and passticket (a temporary password) from the host access control program (such as RACF®).

Z and I Emulator for Windows uses the macro function to put predefined substitute strings in the user ID and password fields. The Telnet 3270 server substitutes the user ID and passticket in the appropriate place in the 3270 datastream. The logon is completed.

After a Certificate Express Logon macro is recorded, it can be distributed to multiple users for playback without further modification.

Preparing to Configure Certificate Express Logon

Before you configure an Certificate Express Logon macro, you need to have the following information.

- Host application name

  Name of the host application the user is logging onto. For example, the name entered on the USSMSG10 screen.
- Host access application ID

  This name must match the RACF® PTKTDATA (Passticket Data Profile) application name that is configured on the OS/390® (V2R10 or later) or z/OS™ host. This name could be the same as the application name that the user is logging onto (for example, the name on USSMSG10). When creating PTKTDATA profiles for applications such as TSO, the application name portion of the profile will most likely not be the same. For example, RACF® requires that the application ID portion of the profile name be TSO+SID. Refer to *OS/390 V2R10.0 SecureWay Security Server RACF Security Administrator's Guide* or *z/OS V1R1.0 SecureWay Security Server RACF Security Administrator's Guide* to determine the correct profile naming.
- User ID and password for the application that you are logging on to.

  During macro recording, the actual user ID and password are used. They are not recorded in the macro; only the predefined substitute strings are recorded in the macro. The Telnet 3270 server replaces the predefined substitute strings with the actual user ID and password during the logon process.
- Client Security Certificate
  The security certificate for the client must be stored in RACF® using the RACF® RACDCERT command.
  - For information about using digital certificates with RACF®, refer to the following books:
    - For OS/390®, refer to *OS/390 V2R10.0 SecureWay Security Server RACF Security Administrator's Guide* and *OS/390 V2R10.0 SecureWay Security Server RACF Command Language Reference*.
    - For z/OS™, refer to *z/OS V1R1.0 SecureWay Security Server RACF Security Administrator's Guide* and *z/OS V1R1.0 SecureWay Security Server RACF Command Language Reference*.
  - For information about configuring DCAS to use RACF® certificates, refer to the following books:
    - For OS/390®, refer to *OS/390 V2R10.0 IBM Communication Server IP Migration*.
    - For z/OS™, refer to *z/OS V1R1.0 IBM Communication Server IP Migration*.

Configuring Certificate Express Logon

Recording the Macro

You must record a macro for each host application that you want to access. You cannot log on to multiple applications with one macro. You do not have to configure SSL, and client authentication is not required on the telnet servers and OS/390® or z/OS™ before recording the logon macro, but you must do this before you can play the macro.

Manual Configuration of a Certificate Express Logon Macro

You can manually configure an existing Macro format file for Certificate Express Logon use. The procedure is as follows:

1. From the Action Bar, open the macro file containing the recorded keystrokes by selecting **Edit → Preferences → Macro/Script**.
2. Select the macro file you just recorded and then select **Customize**.
3. Replace the UserID recorded in the macro with two tags: the Certificate Express Logon Application ID and the UserID placeholder. The Application ID tag consists of three words, each separated by a blank character: `elf`, `applid`, and the identifier of the host application that will be logged onto. The UserID placeholder is `)USR.ID(`.

   For example, replace `"myUserID` with `")USR.ID(`.
4. Replace the Password recorded in the macro with the Certificate Express Logon Password placeholder tag `)PSS.WD(`.

   For example, replace `"myPassword` with `")PSS.WD(`.

Limitations of the Logon Macro

- Automatic insertion of the user ID and password placeholders into the recorded macro requires that the password be typed into the first non-display input field. The user ID is assumed to have been entered just prior to the password. The Modified Data Tag attribute of each formatted input field is assumed to be **ON** only when the field has been modified by the operator.
- There is a short delay while the Telnet 3270 server acquires the passticket from the host access control facility. The amount of time is probably less than the usual delay incurred when the user enters a user ID and password. However, the user can see the macro proceed through the screens during the logon process.
- Logon Express recording requires that the host program uses 3270 field attributes to define the password field on the screen.

Problem Determination

If the client logon fails and displays the messages `)USR.ID( NOT IN CP DIRECTORY`, `INVALID USERID, )USR.ID(`, `PASSWORD NOT AUTHORIZED` or any similar messages, check the Telnet 3270 server log for details.

Possible reasons for failures are:

- The application ID defined in the macro is not valid.
- The Telnet 3270 server could not connect to DCAS. The host might be down.
- The client certificate is not defined in RACF® or it is not valid.
- The passticket has expired and could not be used to log on.
- The Telnet 3270 server completed scanning of data stream without replacing the user ID or password.
- The Telnet 3270 server or the host does not support Certificate Express Logon.

Log Viewer Functions

The Z and I Emulator for Windows log viewer utility enables you to view, merge, sort, search, and filter information contained in message and trace logs. You can use the viewer during problem determination to work with message and trace log entries. The default name of the message log output file is PCSMSG.MLG; its file extension must be .MLG. The file extension for trace logs must be .TLG. Note that the Help per Message Log Item functionality is available only for message logs.

Viewing Message and Trace Logs

To view message or trace logs:

1. From the **Administrative and PD Aids** folder, click **Log Viewer**; or, from an active session window, click **Actions → Launch → Log Viewer**.
2. From the list of logged messages, click on one of the details on the log entry in the bottom pane.

Note: If the logger device driver determines that the product kernel driver-generated log is full and cannot log a message, it will create an entry in the Windows® log. The information logged may include which log failed, as well as the location and reason for the failure.

Changing Message Log Size and Location

The Z and I Emulator for Windows log viewer utility allows you to modify the size and location of message log files, and change the name of the default message log file. The size of a log file is counted in kilobytes and can range from a minimum of 4Kb to a maximum limited only by available hard disk space.

To modify the location and size of the log:

1. From the **Log Viewer** main menu, click **Options** and then click **Configure Message Log Settings...** A Windows® common dialog box is displayed.
2. From the dialog box, browse the directory structure and choose the destination directory and file name for the message log.
3. Using the spin control counter field, use the up and down arrows to increase or decrease the log file size (in kilobytes).
4. Select **OK** to save settings and exit the window.

Merging Message and Trace Logs

The Z and I Emulator for Windows log viewer utility allows you to open and merge message and trace log entries in the same log viewer window. You can merge any combination of message and trace log files.

- .MLG into .TLG
- .TLG into .TLG
- .TLG into .MLG
- .MLG into .MLG

To merge message and trace log files:

1. From the Log Viewer window, select the message or trace log file window where the files will be merged.
2. Click the **File** menu, then click **Merge** and choose a file to be merged.

Sorting Message and Trace Logs

The Z and I Emulator for Windows log viewer utility allows you to sort message and log files in ascending and descending order. To sort files in ascending order, click the column header one time, or right mouse click the data. Click the column header, or right mouse click the data a second time to sort in descending order.

Message and logs can also be sorted by selecting the data to be sorted, and right mouse clicking to display the pop-up menu. Click **Sort**.

Searching Z and I Emulator for Windows Logs

To search the log files, click **Edit → Find** on the **Main** menu.

Type your search string in the provided box. You can refine your search by checking the **Match case** check box if your search is to be case-specific. If you want to limit your search to only complete words, select the **Match whole word only** check box.

Clicking **Find Next** takes you to the next instance where your search string appears highlighted in the log.

Clicking **Cancel** will stop the search.

Filtering Z and I Emulator for Windows Logs

Messages can be filtered by component only. Traces can be filtered by component, by process ID, and by thread ID.

To filter the message or trace record list, do the following:

1. Decide what you want to filter by. For example, in the message log, you may want to filter your view so that it contains only messages issued by a particular component. (Filtering by component is the default.)
2. Click the left mouse button to highlight the item that has the value that you want to filter by.

3. Right click in the appropriate column list area.
4. Click **Filter In**, **Filter Out** or **Sort**.

   Filter In allows only those items selected to be in the resulting view. **Filter Out** removes the selected items from the resulting view. **Sort** allows you to sort entries in ascending or descending order based upon the entry selected. This function works much like sorting by column header. Filters are cumulative, so you can filter the results of your first filter.

To restore your original view, click **View** and then **Refresh**, or you can press **F5**.

Building a Printer Definition Table (PDT)

Building a Printer Definition Table (PDT) for PC/3270

This chapter, in combination with Building a Printer Definition Table (PDT) for PC400 on page 418, explains how to customize a printer definition table (PDT file) for PC/3270. Building a Printer Definition Table (PDT) for PC400 on page 418 contains basic information about creating and changing PDTs; this chapter contains specific information about creating PC/3270 PDTs. PDTs for PC/3270 and PC400 differ only slightly; if you use a common PDT (used for both PC/3270 and PC400), the additional statements for PC400 are ignored for PC/3270 processing. Similarly, statements unique to PC/3270 are ignored for PC400 processing.

ASCII_PASSTHRU? and EBCDIC_PASSTHRU?

The ASCII_PASSTHRU? and EBCDIC_PASSTHRU? PDF statements are new options available for PC/3270. See Transparent Print Capability on page 423 for details.

Supplemental Explanation of PDF Statements for PC/3270

The following PDF statements have functions that differ from those for PC400. All statements are listed in Field Names of Printer Definition Files on page 436.

MAXIMUM_PAGE_LENGTH

Printed lines per page. If you change this value, you must change the value in the SET_PAGE_LENGTH=SFL value statement (see Session parameters on page 421) to be the same.

MAXIMUM_PRINT_POSITION

Printed characters per line.

COMPRESS_LINE_SPACING?

Whether blank or null lines are to be printed if all characters on that line are nulls (for LU 3 only).

FORM_FEED_ANY_POSITION?

Whether a form feed is to be valid in any position. If NO, a form feed will be valid only in the following positions:

- First print-position of the buffer
- After a valid new line operation
- First print-position of a line

OVERRIDE_FORMATTED_PRINT?

Whether nulls are to be printed as blanks.

INTERV_REQ_TIMER

This statement is ignored.

INTERV_TIMER_ON_PE_ONLY?

This statement is ignored.

RESELECT_TIME_EXCPT_5204

This statement is ignored.

ESC/P_LINE_FEED?

If YES, the line feed (LF) function is emulated when the line feed command is received. This is useful when you do not want a line feed accompanied by a carriage return (CR) on a printer using the ESC/P printer language. If NO, the value defined in the LINE_FEED statement is sent to the printer.

IGNORE_FORM_FEED_AT_FIRST_POS?

If YES, the form feed (FF) function is ignored at the first position (for LU 2, LU 3) or at the beginning of the print job (for LU 1 sessions). Using this option eliminates extra blank pages at the beginning of each print job.

FORM_FEED_TAKES_POSITION?

If YES, the form feed (FF) function is effective if followed by data (LU 2, LU 3 only).

ZENKAKU_SPACE

The size (adjustment unit) of a user-defined character and a HANKAKU character. This value cannot be changed.

SBCS_FONT_LOAD

Registration of a HANKAKU GAIJI. This value cannot be changed.

SET_LOCAL_FONT

Set a font set of user-defined characters. Remove it when user-defined characters are not loaded to a printer.

RESET_LOCAL_FONT

Reset a font set of user-defined characters. Remove it when user-defined characters are not loaded to a printer.

ATTRIBUTE_GRID_LINE

Set to grid-line print. This value cannot be changed.

START_DOUBLE_WIDTH_CHARACTER

Set a double-width character.

END_DOUBLE_WIDTH_CHARACTER

Reset a double-width character.

Note:

1. When using IBM5577.PDF, change FORM_FEED=EJC to FORM_FEED in the file when a continuous form job does not feed correctly.

SCS TAB Setting

A PC/3270 printer session LU type 1 can accept any number of tab positions, and the host printer session can send any number of tabs to the printer session. However, the workstation printer you are using might support fewer tab positions than the host application sets; for example, the IBM® Proprinter supports 27 tab positions.

If the number of tab positions that the host application sets exceeds the maximum number of positions that the printer supports, your printed output will not look as you expect it to. You can avoid this situation by modifying the PDF file and reconfiguring PC/3270 as follows:

1. Modify the SET_HORIZONTAL_TABS statement as follows:

   ```
   SET_HORIZONTAL_TABS=number
   ```
2. Save the file under a new name.
3. Convert the PDF file (with the procedure described in Building a Printer Definition Table (PDT) for PC400 on page 418).
4. Select the new PDT file created in Step 2 on page 412.

Printer Color Mixing

Some printers, such as the IBM® 5182, compose certain colors by mixing colors. Colors are mixed by printing the text in one color and then printing over the same text in another color on a second pass.

PC/3270 will compose a color if the color is not defined in the printer definition table of a color printer that is capable of mixing colors. Therefore, if you are using a printer that composes some colors by mixing two colors, leave the definition of the composed colors blank in the printer definition file. Only the composite colors defined in Table 49: Color Mixes on page 413 are created by double-printing the primary colors.

Table 49. Color Mixes

Composite Color	Primary Colors
Red	yellow, magenta
Green	yellow, cyan
Blue	magenta, cyan

For example, to create red, you must define yellow and magenta. The primary colors must be defined in the printer definition table.

Printer Session Data Stream Support

3270 Data Stream

The 3270 data stream is a buffer-oriented data stream. The print data is formatted as if it were going to be displayed on a screen. The host system sends commands to format the presentation space. These commands can change the presentation space in any location at any time. Once the host system completes formatting the presentation space, it issues a START PRINT command and the presentation space is printed as accurately as the printer hardware allows.

Table 50: 3270 Data Stream Commands on page 413 lists the commands that can be sent in the 3270 data stream.

Table 50. 3270 Data Stream Commands

Command	Meaning
W	Write
EW	Erase/Write
EWA	Erase/Write Alternate
RB	Read Buffer
RM	Read Modified
RMA	Read Modified All
EAU	Erase All Unprotected
WSF	Write Structured Field

Table 51: 3270 Data Stream Orders on page 414 lists the orders that can be sent in the 3270 data stream.

Table 51. 3270 Data Stream Orders

Order	Meaning
SBA	Start Buffer Address
SF	Start Field
IC	Insert Cursor
PT	Program Tab
RA	Repeat to Address
EUA	Erase Unprotected to Address
SFE	Start Field Extended
SA	Set Attribute
MF	Modify Field

The last three orders in the preceding table manage the color, extended highlighting, and programmed symbols attributes for fields and individual characters. The programmed symbols attribute is not supported by PC/3270.

In addition to the commands and orders in the two preceding tables, there are special printer formatting control codes that can be included in the 3270 data stream.

The following table lists the control codes that can be sent in the 3270 data stream.

Table 52. 3270 Data Stream Format Control Codes

Code	Description
NL	New Line control code moves the print position to the left margin and down one line.
CR	Carriage Return control code moves the print position to the left margin.
EM	End of Message control code ends the print operation.
FF	Form Feed control code moves the print position to the left margin at the top of the next page.

Note: NL, CR, and EM are valid only if a line-length format specified by the WCC is not used. The FF code is valid in any buffer position.

PC/3270 printer support interprets each 3270 attribute and printer control code and translates them into a sequence of one or more workstation printer control codes. For more information about the 3270 data stream, refer to *IBM 3270 Information Display Data Stream Programmer's Reference*.

Delimiting Print Jobs

Many print jobs can be sent over a single PC/3270 printer session. PC/3270 allows multiple sessions and applications to share a single workstation printer on a between-jobs basis. PC/3270 needs to know when each print job starts and ends so that printers can be shared properly and begin and end job strings can be sent at the appropriate times. The emulator recognizes a number of different methods of delimiting print jobs:

By Session

PC/3270 printer support assumes, by default, that all print jobs are delimited by sessions. That is, in the absence of all other indicators, PC/3270 assumes that a print job begins when a printer session is started and ends when it is reset.

Time-Out Interval

On DFT sessions, print jobs can be delimited by a user-specified timeout interval. A print job on a DFT printer session begins when the first host-outbound data for that job is received, and ends when no host-outbound data is received for a period of time exceeding the user-specified DFT timeout interval. For PC/3270, this interval is specified during configuration.

Structured Fields

The host can use structured fields to indicate to the device that a new file is beginning or that the current file is completed. PC/3270 delimits print jobs with Begin of File and End of File structured fields (SF) to perform host-directed printing. Structured fields are described in Structured Fields on page 415.

Structured Fields

The host uses Begin of File and End of File structured fields to indicate to a device that a file is beginning or ending.

Begin/End of File Query Reply

The Begin/End of File query reply indicates that a device supports Begin of File and End of File to delineate print jobs. The PC/3270 sends a query reply, as shown in Table 53: Begin/End of File Query Reply Format on page 415, to the host in response to a Read Partition General query.

Table 53. Begin/End of File Query Reply Format

Byte	Contents	Description
0–1	X'0005'	The length of this structure
2	X'81'	Query reply
3	QCODE X'9F'	Begin/End of File
4	FLAGS	Reserved; must be set to 0’s

Begin/End of File Structured Fields

Begin/End of File structured fields are accepted on either LU 1 or LU 3 sessions. Table 54: Begin/End of File Structured Field Format on page 416 shows the format of the Begin/End of File structured fields.

Table 54. Begin/End of File Structured Field Format

Byte	Bit	Contents	Description
0–1		X'0007'	The length of this structure
2–3		X'0F85'	Begin/End of File
4		PID	Partition ID
5		FLAG1[1] B'00' B'01' B'10'	
	0–1	B'11'	Reserved
			End of File is being sent
			Begin of File is being sent
			Reserved
	2–7		Reserved
6		FLAG2	Reserved; must be set to 0's.

:

[1] This byte indicates whether Begin of File or End of File is being sent

Processing Begin or End of File Structured Fields

When the Begin or End of File structured fields are used with brackets or timeout intervals, the Begin or End of File SFs take precedence over the brackets or timeout intervals in determining when a print job begins or ends. See the following examples:

- Begin or End of File structured fields overriding brackets:

  ```
  Begin Bracket, Begin of File Structured Field, ...Data...,
  End Bracket
  ```

  The device will wait indefinitely until the End of File structured field is received before ending the print job.
- Begin or End of File structured fields overriding timeout intervals:

  ```
  Begin of File structured fields, ...Data..., pause > timeout value
  ```

  The device will wait indefinitely until the End of File structured fields is received before ending the print job.
- Inconsistent use of Begin or End of File structured fields and brackets:

  ```
  Begin Bracket, ...Data1...,
  Begin of File Structured Fields,...Data2..,
  End of File Structured Fields, ...Data3...,
  End Bracket
  ```

  When you use the Begin of File and End of File structured fields inconsistently with brackets, the results are unpredictable. In the preceding example, the device might process Data1, Data2, and Data3 as separate jobs or combine two or more of them into one file.

For predictable results, each data block must be enclosed by a Begin of File structured field and an End of File structured field. The following example shows three print jobs all delimited by Begin or End of File structured fields:

```
Begin Bracket, Begin of File Structured Field, ...Data1...,
End of File Structured Field,(job1)
Begin of File Structured Field, ...Data2..., End of File Structured Field, (job2)
Begin of File Structured Field, ...Data3..., End of File Structured Field,
End Bracket(job3)
```

PC/3270 always keeps track of brackets and timeout intervals. After the emulator receives a Begin of File structured field, it takes no action on Begin Brackets, End Brackets, or timeout until it receives an End of File structured field. After a valid End of File SF is processed, the emulator defaults to delimiting jobs by brackets or timeout intervals until it receives the next Begin of File structured field.

Begin or End of File Structured Field Error Conditions

PC/3270 does not accept transmission of data belonging to two separate print jobs in the same chain. To be accepted by the emulator, Begin of File structured fields must be the first structured field of a chain and End of File structured fields must be the last structured field of a chain.

PC/3270 rejects transmission in the following instances:

- The emulator receives an End of File structured field without first receiving a Begin of File structured field.
- The emulator receives a second Begin of File structured field without receiving an intervening End of File structured field.
- The emulator receives a Begin of File structured field that is not the first structured field following a **write structured field** command (LU 2, LU 3) or a Function Management Header 1 (LU 1 sessions).
- The emulator receives an End of File structured field that is not the last structured field following a **write structured field** command (LU 2, LU 3) or a Function Management Header 1 (LU 1 sessions).

Processing SCS Data Streams

When processing an SCS data stream, PC/3270 treats Begin or End of File structured fields as follows:

- A Begin of File structured field indicates that all SCS data in the same transmission until an End of File structured field is received is part of a new print job.
- An End of File structured field indicates that any SCS data received in the same chain as the End of File structured field is the last data of the current print job.

Processing 3270 Data Streams

When processing a 3270 data stream, PC/3270 treats Begin or End of File structured fields as follows:

- A Begin of File structured field indicates that the next presentation space print (initiated by a **write type** command with the start print bit turned on in the write control character) is the first in a print job.
- An End of File structured field indicates that the last presentation space print was the last of the current print job. The emulator immediately sends a terminate string to the printer to close the printer session.

If PC/3270 receives a Begin of File structured field and an End of File structured field without at least one presentation space separating them, it ignores the structured fields.

Building a Printer Definition Table (PDT) for PC400

This chapter explains how to create and change the printer definition table (PDT file) used for PC400. Building a Printer Definition Table (PDT) for PC/3270 on page 410 contains specific information about creating PC/3270 PDTs. PDTs for PC/3270 and PC400 differ only slightly; if you use a common PDT (used for both PC/3270 and PC400), the additional statements for PC400 are ignored for PC/3270 processing. Similarly, statements unique to PC/3270 are ignored for PC400 processing.

The PDT file is created by converting the printer definition file (PDF file). The PDF and PDT define the transmission of characters and control codes to the printer and the format of printer output. To change an existing PDF (the recommended method) or create a new one, use a text editor that can produce or update an ASCII file.

Using the Printer Definition Table (PDT) File

To use the PDT file:

1. Select **File** from the menu bar of the workstation window.
2. Select **Printer Setup** from the **File** menu.

   The Printer Setup window appears.
3. Select the **Use PDT file** check box and **Select PDT**.

   The Select PDT file window appears.

To build the PDT file (required only if the PDF has been changed or created):

1. Select **Convert PDF**. Select the PDF file to be converted from the list in the Convert PDF to PDT window; then select **Convert**. The PDF File Converter window appears. After the file has been converted, click on **Close**, then click **Close** in the Convert PDF to PDT window.
2. Click **OK** in the Select PDT file window.
3. Click **OK** on the Printer Setup window.

   After printer setup is complete, the Printer Setup window is closed.

Printer Definition File (PDF File) Format

A PDF contains 3 main sections:

- Macro definitions
- Formatting controls
- Character definitions using EBCDIC_xx keywords

Macro Definitions

This section of a PDF contains user-defined macros. A macro is a single mnemonic that stands for a control code or a sequence of control codes. A mnemonic simplifies defining control sequences for PC printers and makes it easier to read the information in the PDF.

The following table shows the structure of a macro definition statement. A macro definition is composed of four parts:

1	**2**	**3**	**4**
name	EQU	PC Printer Control Codes	Comments

For example:

```
  FFF    EQU   0C                /* Form Feed */
```

The first part is the user-defined mnemonic or macro name. This name must be exactly three characters long and must not begin with a number. It is helpful to define a meaningful mnemonic, such as P17 for 17.1 pitch.

The second part, EQU, stands for equate and must be coded as EQU.

The third part is the hexadecimal control code, which is specific to a PC printer. Each PC printer manufacturer can define different control codes for the same printer function. For example, the IBM® 4019 LaserPrinter uses control codes defined by the IBM® Personal Printer Data Stream (PPDS). Hewlett-Packard printers use control codes defined by the Hewlett-Packard Printer Control Language (PCL). These control codes are usually defined in the manual that comes with the printer.

Some PC printer manuals describe control sequences as a string of ASCII symbols, such as ESC J 1 K: others use hexadecimal numbers, such as 1B 57 01; while others use decimal values, such as 27 28 1. The printer definition table compiler accepts any of these formats.

The control codes in the macro definitions section can be any of the following:

- Single characters that are interpreted as their ASCII value
- Two-digit numbers that are interpreted as hexadecimal values
- Three-digit numbers that are interpreted as decimal values

If you leave the control code section blank or if you delete it, the character or control code is interpreted as a null string. If a character or control code is defined more than once in the file, the last definition is used.

The fourth part is the comment section. The symbols /* indicate the beginning of a comment and the symbols */ indicate the end of a comment. Comments can be coded at any point in the printer definition file and are ignored by the printer definition table compiler.

The following is an example macro definition statements that are specific to the IBM® 4019 LaserPrinter.

```
LFF EQU 0A                         /* Line Feed                  */
VTB EQU 0B                         /* Vertical Tab                */
FFF EQU 0C                         /* Form Feed                   */
CRR EQU 0D                         /* Carriage Return             */
P05 EQU 1B 57 01                   /* 5 Pitch-Characters/inch     */
P10 EQU 12                         /* 10 Pitch-Characters/inch    */
CDW EQU 1B 57 00                   /* Cancel Double Wide contin.  */
P17 EQU 12 0F                      /* 17.1 Pitch-Character/inch   */
LL8 EQU 1B 41 09 1B 32             /* Set line length 8 lines/inch*/
P12 EQU 1B 3A                      /* 12 Pitch-characters/inch    */
RES EQU 1B 5B 4B 07 00 05 31 01 A4 00 00 90
                                   /* The above macro resets the  */
                                   /* printer using the factory   */
                                   /* defaults.  See the IBM Laser*/
                                   /* Printer Technical Reference */
                                   /* manual.                     */
```

To illustrate how macros are coded, consider this example. To have the host print job printed in double-wide characters, you must know what control code turns on double-wide printing on your PC printer. On the IBM® 4019 LaserPrinter, the control code to turn on double-wide printing is X'1B5701'. This would be coded as:

```
BDW EQU 1B 57 01
```

where BDW stands for Begin Double Wide.

This alone would not cause 3270 host print to send this control to the printer. The mnemonic would have to be included in a control code statement, such as START_JOB which is described in the control codes section of the file.

Note: The IBM® 4019 LaserPrinter printer definition file that comes with HCL Z and I Emulator for Windows already has this control code defined as the P05 macro. See line 9 of the sample IBM® 4019 LaserPrinter Macro Definition Statements (IBM4019.PDF File Contents on page 424).

Macro Name Examples

The following are example mnemonics or macros. The control codes are for the IBM® 4019 LaserPrinter:

```
   LND      EQU 1B 26 6C 31 4F                /* LANDSCAPE    */
   POR      EQU 1B 26 6C 30 4F                /* PORTRAIT     */
   P12      EQU 1B 28 73 31 32 2E 30 30 48    /* PITCH_12.00  */
   T10      EQU 1B 28 73 31 30 2E 30 56       /* POINT_10.0   */
   LTR      EQU 1B 26 6C 32 41                /* LETT_PAPER   */
   G66      EQU 1B 26 6C 36 36 50             /* PG_LENGTH_66 */
```

Formatting Controls

The controls section of a PDF contains the following:

- Session parameters
- Control codes
- Color specifications
- Highlight specifications

Session parameters

MAXIMUM_PAGE_LENGTH, MAXIMUM_PRINT_POSITION, and SET_PAGE_LENGTH

The parameter to the left of the equal sign is a keyword and *must* be coded exactly as shown. The keyword is used to define a statement. The values to the right of the equal sign are macros or control codes. Because the values to the right of the equal sign can be both macros and control codes, they will sometimes be referred to as parameters.

The MAXIMUM_PAGE_LENGTH, MAXIMUM_PRINT_POSITION, and SET_PAGE_LENGTH parameters specify the dimensions of the output job. The number specified in the MAXIMUM_PAGE_LENGTH parameter is used in the SET_PAGE_LENGTH parameter and is substituted for the value keyword. In other words, if SET_PAGE_LENGTH and MAXIMUM_PAGE_LENGTH are coded as follows:

```
MAXIMUM_PAGE_LENGTH=066 /* Printed lines per page */
SET_PAGE_LENGTH=SFL 066
```

The results would be:

SET_PAGE_LENGTH=SFL 066

Because SFL is coded as X'1B 43' in the macro section, the actual control code that would be sent to the printer to set the maximum page length is:

X'1B 43 42'

where X'42' is decimal 66.

Note: Setting MPL=255 causes suppression of form feeds (FFs).

Control Codes

The control codes section of a PDF is used by the PDT function to determine what specific PC printer control code is to be sent to the PC printer when an SCS control code is received. Some of the statements used by Communications Manager are shown in the following example.

```
START_JOB=SEL CDW CDL CUL CDS CP8 CS2
END_JOB=CAT CDW CDL CUL CDS CP4 FFF
SET_PAGE_LENGTH=SFL value
SET_VARIABLE_LINE_DENSITY=ESC A value ESC 2
SET_10_CHARACTERS_PER_INCH=P10
```

Note: The above example lines come from the IBM® 4019 LaserPrinter PDF.

START_JOB and END_JOB

The control codes associated with START_JOB are sent at the start of each host print job. It is best to set all printer options to a known or desired state at the beginning of each host print job. The PC printer changes its state or changes the options selected only when instructed to do so by control codes. Therefore, the previous PC application could have left the printer in portrait mode with a Courier font selected, and unless your job changed them, these would be the printer options used.

Even though the START_JOB and END_JOB control codes are the ones shipped with the IBM4019.PDF, many users change them to be more like the ones below:

```
    START_JOB=RES P12 LL8
    END_JOB=RES
```

In this example, the RES, P12 and LL8 macros are used on the START_JOB statement (these macros are defined in IBM4019.PDF File Contents on page 424). This translates into the following control codes being sent to the printer at the beginning of your print job:

X'1B 5B 4B 07 00 05 31 01 A4 00 00 90 1B 3A 1B 41 09 1B 32'

This sequence of control codes:

- Resets the printer to the IBM® PPDS factory default settings (RES)
- Begins printing in 12 pitch (P12)
- Begins printing at 8 lines per inch (LL8)

The END-JOB statement sends the following control code to the printer at the end of your job:

X'1B 5B 4B 07 00 05 31 01 A4 00 00 90'

This control code resets the printer to IBM® PPDS factory default settings (RES).

SET_VARIABLE_LINE_DENSITY

The SET_VARIABLE_LINE_DENSITY statement is used in combination with the panel where you can select the number of lines per inch (LPI), which can be either 6 or 8.

The PDT process uses whatever is selected in the lines per inch field, and substitutes this number for the value keyword in the SET_VARIABLE_LINE_DENSITY parameter. In other words, if lines per inch is set to 8, the SET_VARIABLE_LINE_DENSITY parameter is

```
SET_VARIABLE_LINE_DENSITY=ESC A 09 ESC 2
```

where the 09 comes from 72/8. The number of typographic points in 1 inch is 42; thus a value of 12 points would indicate six lines to an inch.

The control code that is sent to the printer to set the line density is:

```
X'1B 41 09 1B 32'
SET_10_CHARACTERS_PER_INCH
```

On most printers, the default pitch is 10 characters per inch. In most PDTs, Z and I Emulator for Windows uses this same convention and sends the control code found on the SET_10_CHARACTERS_PER_INCH statement. This is usually coded in the PDF as:

```
SET_10_CHARACTERS_PER_INCH=P10
```

where P10 is coded in the macro definition section as:

```
P10 EQU 12
```

Printing More than One Screen on a Page

If you want to print two or more screens per page, use the BEL command in the PDF to specify the number of blank lines to insert (instead of a Form Feed between two successive screens). You must coordinate this modification with the usage of the LPI and MPL parameters in the PDF (see Field Names of Printer Definition Files on page 436).

Note: This specific use of the BEL command is applicable only for printing screens using the **Print Screen Collection** function.

Setting the value `BEL=FF` will send a Form Feed, while the value `BEL=00` does not insert a Form Feed or a Line Feed.

Values between 00 and FF send that number of blank lines between successive screens. For example, `BEL=02` inserts two blank lines between two successive screens. Thus, more than one screen can be printed on a single page.

Transparent Print Capability

ASCII_PASSTHRU? Keyword Support

If you add the following line to your PDF, Z and I Emulator for Windows will send host data without any translation:

```
    ASCII_PASSTHRU? = YES
```

This option is for special host applications that generate PC printer control codes directly.

Even if ASCII_PASSTHRU? is set, control codes defined START_JOB and END_JOB are sent to the printer at the start and the end of a print job respectively. To remove those commands, you need to rebuild the PDT file after removing the PDF keyword definitions for START_JOB and END_JOB.

If ASCII_PASSTHRU? is set, all character definition lines, for example, EBCDIC_xx, are ignored.

If both ASCII_PASSTHRU? and EBCDIC_PASSTHRU? are set, ASCII_PASSTHRU? has priority.

EBCDIC_PASSTHRU? Keyword Support

If you add the following line to your PDF, Z and I Emulator for Windows will ignore all SCS commands and send data to the printer after EBCDIC-to-ASCII translation:

```
    EBCDIC_PASSTHRU? = YES
```

For example, the default EBCDIC-ASCII translation table used for U.S. English host code page 037 is as follows:

```
    Hex |  0 1  2 3  4 5  6 7  8 9  A B  C D  E F
   -----+-------------------------------------------
      0 | 2020 2020 2020 2020 2020 2020 2020 2020
     10 | 2020 2020 2020 2020 2020 2020 2020 2020
     20 | 2020 2020 2020 2020 2020 2020 2020 2020
     30 | 2020 2020 2020 2020 2020 2020 2020 2020
     40 | 20FF 8384 85A0 C686 87A4 BD2E 3C28 2B7C
     50 | 2682 8889 8AA1 8C8B 8DE1 2124 2A29 3BAA
     60 | 2D2F B68E B7B5 C78F 80A5 DD2C 255F 3E3F
     70 | 9B90 D2D3 D4D6 D7D8 DE60 3A23 4027 3D22
```

```
      80 | 9D61 6263 6465 6667 6869 AEAF D0EC E7F1
      90 | F86A 6B6C 6D6E 6F70 7172 A6A7 91F7 92CF
      A0 | E67E 7374 7576 7778 797A ADA8 D1ED E8A9
      B0 | 5E9C BEFA B8F5 F4AC ABF3 5B5D EEF9 EF9E
      C0 | 7B41 4243 4445 4647 4849 F093 9495 A2E4
      D0 | 7D4A 4B4C 4D4E 4F50 5152 FB96 8197 A398
      E0 | 5CF6 5354 5556 5758 595A FDE2 99E3 E0E5
      F0 | 3031 3233 3435 3637 3839 FCEA 9AEB E9FF
```

You can modify this code page using EBCDIC_xx keywords. Note that the EBCDIC_PASSTHRU? line precedes any EBCDIC_xx lines in your PDF file because Z and I Emulator for Windows reinitializes the EBCDIC-to-ASCII translation table when it finds that EBCDIC_PASSTHRU? is set.

Even if EBCDIC_PASSTHRU? is set, control codes defined START_JOB and END_JOB are sent to the printer at the start and the end of a print job respectively. To remove those commands, you need to rebuild the PDT file after removing PDF keyword definitions for START_JOB and END_JOB.

Printer Definition Tables

Standard printer definition table file names are of the form `IBMnnnnn.PDT`, and PDT's ASCII-to-ASCII character definitions are of the form `PRNnnnnn.PDT`, where `nnnnn` is a machine type. See character definition descriptions for more details.

See the help panel or the specific 5250, 3270, or VT emulator user's reference for a list of the PDT files provided by Z and I Emulator for Windows.

The PDT files contained in the PC400 installation diskette can be used as is. However, you might want to do special formatting by changing the definitions of some fields. To do so, copy an existing PDF file, modify it, and then convert it to a new PDT file.

Example PDF files are shown in "Example Printer Definition Files". Do not attempt to change the statements in a field for which modification is specifically prohibited. If you use a PDT file created from a changed PDF file, the results of printing cannot be guaranteed.

Example Printer Definition Files

The following examples are annotated versions of printer definition files for the IBM® LaserPrinter 4019 (for SBCS sessions). These are examples only; the actual files may differ.

IBM4019.PDF File Contents

```
/*******************************************************************/
/*                                                                 */
/*    PRINTER SESSION DEFINITION FILE FOR: LaserPrinter 4019/4019-E   */
/*                                                                 */
/*******************************************************************/
/*******************************************************************/
/*                        Macro Definitions                        */
/* Define values here that will be used commonly throughout your   */
/* definitions.  Then use the left hand side of the equate as you  */
/* define your characters and control strings.  The printer compiler  */
/* will substitute the right hand side of the equate for each         */
/* occurrence of the left hand side throughout the file.              */
/*                                                                    */
/* Macro names must be at least three characters long and may not     */
/* begin with a number.                                               */
/*                                                                    */
/* Format                                                             */
/* A Macro Name is associated with a value or string of values by the */
/* EQU statement.  The right hand side of an EQU statement must be a  */
/* string of zero or more two digit hexadecimal numbers.  If a macro  */
/* definition is more than one line long, you may extend it to the    */
/* next line by ending the first line with a comma.  In this manner   */
/* you may define a macro which is many lines long by terminating each*/
/* line except the last with a comma.  No macro names are allowed on  */
/* right hand side.                                                   */
/**********************************************************************/
BEGIN_MACROS
/* The following values are standard for most printers.  Check your   */
/* printer manual to verify that these are correct for your printer   */
NUL EQU 00             /* Nul character                               */
BEL EQU 07             /* Beeper                                      */
BAK EQU 08             /* Back Space                                  */
TAB EQU 09             /* Tab                                         */
LFF EQU 0A             /* Line Feed                                   */
VTB EQU 0B             /* Vertical Tab                                */
FFF EQU 0C             /* Form Feed                                   */
CRR EQU 0D             /* Carriage Return                             */
P05 EQU 1B 57 01       /* 5 Pitch-Characters/inch                     */
                       /* Same as Double Wide                         */
SEL EQU 11             /* Select Printer                              */
P10 EQU 12             /* 10 Pitch-Characters/inch                    */
CDW EQU 1B 57 00       /* Cancel Double Wide contin.                  */
CDL EQU 14             /* Cancel Double Wide line                     */
ESC EQU 1B             /* Escape                                      */
CAN EQU 18             /* Cancel Data                                 */
SPA EQU 20             /* Space                                       */
P17 EQU 12 0F          /* 17.1 Pitch-Characters/inch                  */
CS2 EQU 1B 36          /* Select Character Set 2                      */
CS1 EQU 1B 37          /* Select Character Set 1                      */
P12 EQU 1B 3A          /* 12 Pitch-characters/inch                    */
SVT EQU 1B 42          /* Set Vertical Tabs                           */
SFL EQU 1B 43 00       /* Set Form Length                             */
SHT EQU 1B 44          /* Set Horizontal Tabs                         */
SDS EQU 1B 47          /* Start Double Strike                         */
CDS EQU 1B 48          /* Cancel Double Strike                        */
SSP EQU 1B 4E          /* Set skip perforation                        */
CSP EQU 1B 4F          /* Cancel skip perforation                     */
CAT EQU 1B 52          /* Cancel all tabs Clears VT                   */
                       /* and sets HT every 8 position                */
CSS EQU 1B 54          /* Cancel Subscript or Superscript             */
SSO EQU 1B 53 00       /* Set Superscript over                        */
SSU EQU 1B 53 01       /* Set Subscript under                         */
SUL EQU 1B 2D 01       /* Start Underline                             */
CUL EQU 1B 2D 00       /* Cancel Underline                            */
SCP EQU 1B 5B 54 04 00 00 00 /* ESC T - select code page              */
CP8 EQU 1B 5B 54 04 00 00 00 03 52     /* select code page 850        */
CP4 EQU 1B 5B 54 04 00 00 00 01 B5     /* select code page 437        */
LL2 EQU 1B 41 24 1B 32 /* Setline length 2 lines/inch                 */
```

```
LL3 EQU 1B 41 18 1B 32 /* Setline length 3 lines/inch                */
LL4 EQU 1B 41 12 1B 32 /* Setline length 4 lines/inch                */
LL6 EQU 1B 41 0C 1B 32 /* Set line length 6 lines/inch               */
LL8 EQU 1B 41 09 1B 32 /* Set line length 8 lines/inch               */
LL0 EQU 1B 41 07 1B 32 /* Set line length 10 lines/inch              */
                       /* actually 7/72 inch                          */
SD1 EQU 1B 5B 46 05 00 00 01 01 00 00 /* Select Drawer 1              */
SD2 EQU 1B 5B 46 05 00 00 01 02 00 00 /* Select Drawer 2              */
ENV EQU 1B 5B 46 05 00 00 02 00 00 00 /* Select Envelope              */
FRM EQU 1B 64          /* Forward Relative Movement                   */
VLF EQU 1B 4A          /* Vertical Line Feed 1/216 inch units         */
SPO EQU 1B 6B          /* Set Portrait Orientation                    */
SLO EQU 1B 6C          /* Set Landscape Orientation                   */
SFG EQU 1B 5B 49 08 00 /* Set Font Global                             */
END_MACROS
/*                     Session Parameters                             */
/* These parameters determine the way in which output will be         */
/* formatted for your printer.                                        */
/* Numeric Parameters                                                 */
/* These parameters should be defined with a two digit hex number     */
/* or a three digit decimal number.  The range of the number is zero  */
/* to 255 (decimal).                                                  */
MAXIMUM_PAGE_LENGTH=066              /* Printed lines per page        */
MAXIMUM_PRINT_POSITION=080           /* Printed characters per line   */
INTERV_REQ_TIMER=001
RESELECT_TIME_EXCPT_5204=001
INTERV_TIMER_ON_PE_ONLY?=NO
HORIZONTAL_PEL=120
VERTICAL_PEL=216
LINE_SPACING_RATIO=072
PAGE_LENGTH_TYPE?=INCH               /* SET_PAGE_LENGTH "value" is    */
                                       /* values                      */
/* YES/NO Parameters                                                  */
/* These parameters should be defined with either "YES" or "NO" on the*/
/* right hand side of the '='                                         */
COMPRESS_LINE_SPACING?=NO            /* Should blank or null lines    */
                                       /* be printed?                 */
FORM_FEED_ANY_POSITION?=YES          /* Should the form feed be       */
                                       /* valid in any position?      */
OVERRIDE_FORMATTED_PRINT?=YES        /* Should nulls be printed as    */
                                       /* blanks?                     */
AUTO_NEWLINE_AT_MAX_POS?=NO
/*                         Control Codes                              */
/* These definitions tell the emulator what control strings to send to*/
/* your printer to issue control commands.                            */
/*                                                                    */
/* Format                                                             */
/* The name of the control command should always be at the beginning  */
/* of a line followed by a '=' and then a definition string.          */
/* A Definition String is any combination of macro names, hexadecimal */
/* numbers, and characters separated by blanks.  A macro must have    */
/* previously defined in the macro definitions section above. A       */
/* hexadecimal number must be two digits (0,..,F) long. and a         */
/* character must be preceded and followed by a blank.  If a          */
/* definition string will not fit on a line, it may be continued      */
/* as many lines as you wish by ending each line except the last with */
/* a comma;  ','.   You made add any comments you wish to by including*/
/* them between a slash*  and a *slash where slash is the symbol /.   */
/* START_JOB is the control string which will be sent to your printer */
/* at the beginning of each print job.                                */
START_JOB=SEL CDW CDL CUL CDS CP8 CS2
/* END_JOB is the string which will be sent to your printer at the end*/
/* of each print job.                                                 */
END_JOB=CAT CDW CDL CUL CDS CP4 FFF
BACKSPACE=BAK
BEL=BEL
CARRIAGE_RETURN=CRR
NEW_LINE=CRR LFF
LINE_FEED=LFF
FORM_FEED=FFF
HORIZONTAL_TAB=TAB
VERTICAL_TAB=VTB
START_SUBSCRIPT=SSU
END_SUBSCRIPT=CSS
START_SUPERSCRIPT=SSO
END_SUPERSCRIPT=CSS
DUP=*
FIELD_MARK=;
/* The following commands specify control codes for which most PC     */
/* printers require command strings which contain a variable value    */
/* or values somewhere in the middle of the string.                   */
/* Place the word "value(s)" in the position of your definition       */
/* string where the Z and I Emulator for Windows 5250 should fill in       */
/* the hexadecimal value(s) indicated.                                */
/* For example, on the IBM Proprinter, the SET_HORIZONTAL_TABS        */
/* definition is:                                                     */
/* SET_HORIZONTAL_TABS=ESC D values NUL                               */
SET_HORIZONTAL_TABS=SHT values NUL   /* "values" are the tab stops    */
                                       /* in column numbers           */
SET_VERTICAL_TABS=SVT values NUL     /* "values" are the tab stops    */
                                       /* in line numbers             */
SET_HORIZONTAL_MARGINS=
SET_PAGE_LENGTH=SFL value            /* "value"=inch of the page      */
SET_AUTO_PERFORATION_SKIP=SSP value
                                       /* "value"=number of lines to  */
                                       /* skip over the perforation   */
                                       /* between pages.  Used to set */
                                       /* top and bottom margins.     */
SET_VARIABLE_LINE_DENSITY=ESC A value ESC 2
                                       /* "value"=number of points.   */
                                       /* A point is                  */
                                       /* 1/(LINE_SPACING_RATIO) inch.*/
SET_CHARACTER_SET=
/*SET_CHARACTER_SET=ESC I NULL  selects the normal font               */
/*SET_CHARACTER_SET=ESC I 02  selects the NLQ (near letter quality)   */
/*SET_CHARACTER_SET=ESC I 04  selects the normal downloaded font      */
/*SET_CHARACTER_SET=ESC I 06  selects the NLQ downloaded font         */
/*SET_CHARACTER_SET=CS1       selects the Character set 1             */
/*SET_CHARACTER_SET=CS2       selects the Character set 2             */

/* These control codes set the printer lines per inch and characters  */
/* per inch to fixed amounts.                                         */
/* If your printer does not support setting the line density in points*/
/* then you can enter control strings for the following commands.     */
/* When Z and I Emulator for Windows 5250 gets a command from the host to  */
/* set the lines per  inch, it will round it to the closest line per  */
/* inch setting that you provide.  Note that if you provide a command */
/* for the SET_VARIABLE_LINE_DENSITY command above that it will be    */
/* used and any control strings you provide for the set lines per inch*/
/* commands below will not be used.                                   */

SET_2_LINES_PER_INCH=LL2
SET_3_LINES_PER_INCH=LL3
SET_4_LINES_PER_INCH=LL4
SET_6_LINES_PER_INCH=LL6
SET_8_LINES_PER_INCH=LL8
SET_10_LINES_PER_INCH=LL0            /* 7/72 inch or 9/96 inch        */
SET_10_CHARACTERS_PER_INCH=P10
SET_12_CHARACTERS_PER_INCH=P12
SET_13_CHARACTERS_PER_INCH=          /*                               */
SET_15_CHARACTERS_PER_INCH=          /* The proprinter does not       */
                                       /* support 15 pitch except in  */
                                       /* graphic mode                */
SET_17_CHARACTERS_PER_INCH=P17       /* Condensed mode                */
SET_20_CHARACTERS_PER_INCH=
START_DOUBLE_WIDTH_CHARACTERS=P05
END_DOUBLE_WIDTH_CHARACTERS=CDW

/* These control codes are used to select the source drawer number    */
/* when your printer has the dual drawer sheetfeed option.            */
SELECT_DRAWER1=SD1
SELECT_DRAWER2=SD2
SELECT_DRAWER3=
SELECT_ENVELOPE=ENV                  /* Envelope                      */

/* These control codes select the print mode (quality of print).      */
SELECT_DRAFT_QUALITY=
SELECT_LETTER_QUALITY=
SELECT_ENHANCED_QUALITY=
SELECT_SETUP_QUALITY=

/* These control codes                                                */
SET_DUPLEX=
SET_DUPLEX_TUMBLE=
RESET_DUPLEX=

/* These control codes set page orientation                           */
SET_PORTRAIT_ORIENT=SPO
SET_LANDSCAPELEFT_ORIENT=SLO
SET_PORTRAITUPDWN_ORIENT=SPO
SET_LANDSCAPERGHT_ORIENT=SLO

/* These control codes move the print position (Horizontal/Vertical)  */
FORWARD_HORIZONTAL_SKIP=FRM word-value(LH)
FORWARD_VERTICAL_STEP_FEED=VLF byte-value

/* These control codes select the printer font via global font ID     */
SET_FONT_GLOBAL=
SET_GFID_0003=SFG 00 03 00 90 01 01 03 52 CDW   /* OCR-B.10           */
SET_GFID_0005=SFG 00 05 00 90 01 01 03 52 CDW   /* Orator.10          */
SET_GFID_0011=SFG 00 0B 00 90 01 01 03 52 CDW   /* Courier.10         */
SET_GFID_0012=SFG 00 0C 00 90 01 01 03 52 CDW   /* Prestige.10        */
SET_GFID_0013=SFG 00 0B 00 90 01 01 03 52 CDW   /* Artisan.10         */
```

```
SET_GFID_0018=SFG 00 12 00 90 01 01 03 52 CDW   /* Courier.Italic.10  */
SET_GFID_0019=SFG 00 13 00 90 01 01 03 52 CDW   /* OCR-A.10           */
SET_GFID_0020=SFG 00 14 00 90 01 01 03 52 CDW   /* Pica.10            */
SET_GFID_0030=SFG 00 1E 00 90 01 01 03 52 CDW   /* Math-Symbol.10     */
SET_GFID_0038=SFG 00 26 00 90 01 01 03 52 CDW   /* Orator.Bold.10     */
SET_GFID_0039=SFG 00 27 00 90 01 01 03 52 CDW   /* Gothic.Bold.10     */
SET_GFID_0040=SFG 00 28 00 90 01 01 03 52 CDW   /* Gothic-Text.10     */
SET_GFID_0041=SFG 00 29 00 90 01 01 03 52 CDW   /* Roman-text.10      */
SET_GFID_0042=SFG 00 2A 00 90 01 01 03 52 CDW   /* Serif-text.10      */
SET_GFID_0043=SFG 00 2B 00 90 01 01 03 52 CDW   /* Serif-text.Italic.10*/
SET_GFID_0044=SFG 00 2C 00 90 01 01 03 52 CDW   /* Katakana-gothic.10 */
SET_GFID_0045=SFG 00 2D 00 90 01 01 03 52 CDW   /* APL.10             */
SET_GFID_0046=SFG 00 2E 00 90 01 01 03 52 CDW   /* Courier.Bold.10    */
SET_GFID_0050=SFG 00 32 00 90 01 01 03 52 CDW   /* Shalom.10          */
SET_GFID_0066=SFG 00 42 00 78 01 01 03 52 CDW   /* Gothic-text.12     */
SET_GFID_0068=SFG 00 44 00 78 01 01 03 52 CDW   /* Gothic-text.Italic.12*/
SET_GFID_0069=SFG 00 45 00 78 01 01 03 52 CDW   /* Gothic.Bold.12     */
SET_GFID_0070=SFG 00 46 00 78 01 01 03 52 CDW   /* Serif-text.12      */
SET_GFID_0071=SFG 00 47 00 78 01 01 03 52 CDW   /* Serif-text.Italic.12*/
SET_GFID_0072=SFG 00 48 00 78 01 01 03 52 CDW   /* Serif.Bold.12      */
SET_GFID_0080=SFG 00 73 00 78 01 01 03 52 CDW   /* Math-Symbol.12     */
SET_GFID_0084=SFG 00 54 00 78 01 01 03 52 CDW   /* Script.12          */
SET_GFID_0085=SFG 00 55 00 78 01 01 03 52 CDW   /* Courier.12         */
SET_GFID_0086=SFG 00 56 00 78 01 01 03 52 CDW   /* Prestige.12        */
SET_GFID_0087=SFG 00 57 00 78 01 01 03 52 CDW   /* Letter-gothic.12   */
SET_GFID_0091=SFG 00 70 00 78 01 01 03 52 CDW   /* Light.Italic.12    */
SET_GFID_0107=SFG 00 55 00 78 01 01 03 52 CDW   /* Courier.12         */
SET_GFID_0110=SFG 00 6E 00 78 01 01 03 52 CDW   /* Letter-Gothic.Bold.12*/
SET_GFID_0111=SFG 00 6F 00 78 01 01 03 52 CDW   /* Prestige-Elite.Bold.12*/
SET_GFID_0112=SFG 00 70 00 78 01 01 03 52 CDW   /* Prestige.Italic.12 */
SET_GFID_0115=SFG 00 73 00 78 01 01 03 52 CDW   /* Math-Symbol.12     */
SET_GFID_0155=SFG 00 9B 00 78 02 01 03 52 CDW   /* Boldface.Italic.PSM*/
SET_GFID_0158=SFG 00 9E 00 78 02 01 03 52 CDW   /* Modern.PSM         */
SET_GFID_0159=SFG 00 9F 00 78 02 01 03 52 CDW   /* Document.PSM       */
SET_GFID_0160=SFG 00 A0 00 78 02 01 03 52 CDW   /* Essay.PSM          */
SET_GFID_0162=SFG 00 A2 00 78 02 01 03 52 CDW   /* Essay.Italic.PSM   */
SET_GFID_0163=SFG 00 A3 00 78 02 01 03 52 CDW   /* Essay.Bold.PSM     */
SET_GFID_0168=SFG 00 A8 00 78 02 01 03 52 CDW   /* Barak.PSM          */
SET_GFID_0173=SFG 00 AD 00 78 02 01 03 52 CDW   /* Essay.Light.PSM    */
SET_GFID_0175=SFG 00 AF 00 78 02 01 03 52 CDW   /* Document.PSM       */
SET_GFID_0176=SFG 00 B0 00 78 02 01 03 52 CDW   /* Boldface.PSM       */
SET_GFID_0177=SFG 00 9B 00 78 02 01 03 52 CDW   /* Boldface.Italic.PSM*/
SET_GFID_0193=SFG 00 73 00 78 01 01 03 52 CDW   /* Math-Symbol.12     */
SET_GFID_0198=SFG 00 1E 00 90 01 01 03 52 CDW   /* Math-Symbol.10     */
SET_GFID_0204=SFG 00 CC 00 6C 01 01 03 52 CDW   /* Gothic-text.13     */
SET_GFID_0221=SFG 00 DD 00 60 01 01 03 52 CDW   /* Prestige.15        */
SET_GFID_0222=SFG 00 E6 00 60 01 01 03 52 CDW   /* Gothic-text.15     */
SET_GFID_0223=SFG 00 DF 00 60 01 01 03 52 CDW   /* Courier.15         */
SET_GFID_0225=SFG 00 E1 00 60 01 01 03 52 CDW   /* Math-symbol.15     */
SET_GFID_0229=SFG 00 E5 00 60 01 01 03 52 CDW   /* Serif-text.15      */
SET_GFID_0230=SFG 00 E6 00 60 01 01 03 52 CDW   /* Gothic-text.15     */
SET_GFID_0245=SFG 00 2E 00 90 01 01 03 52 P05   /* Courier.Bold.5     */
SET_GFID_0252=SFG 00 FC 00 54 01 01 03 52 CDW   /* Courier.15         */
SET_GFID_0253=SFG 00 FD 00 54 01 01 03 52 CDW   /* Courier.Bold.17    */
SET_GFID_0254=SFG 00 FE 00 55 01 01 03 52 CDW   /* Courier.17         */
SET_GFID_0280=SFG 01 18 00 48 01 01 03 52 CDW   /* APL.20             */
SET_GFID_0281=SFG 01 19 00 48 01 01 03 52 CDW   /* Gothic-text.20     */
SET_GFID_0290=SFG 01 22 00 36 01 01 03 52 CDW   /* Gothic-text.27     */
```

```
SET_GFID_0751=SFG 11 37 00 A0 01 03 03 52 CDW   /* Sonoran-serif.8pt  */
SET_GFID_1051=SFG 11 37 00 C8 01 03 03 52 CDW   /* Sonoran-serif.10pt */
SET_GFID_1053=SFG 11 4B 00 C8 01 03 03 52 CDW   /* Sonoran-serif.bold.10pt*/
SET_GFID_1056=SFG 11 B7 00 C8 01 03 03 52 CDW   /* Sonoran-serif.italic.10pt*/
SET_GFID_1351=SFG 11 37 00 F0 01 03 03 52 CDW   /* Sonoran-serif.12pt  */
SET_GFID_1653=SFG 11 4B 01 40 01 03 03 52 CDW   /* Sonoran-serif.Bold.16pt*/
SET_GFID_2103=SFG 11 4B 01 E0 01 03 03 52 CDW   /* Sonoran-serif.Bold.24pt*/

/*                      Color Specifications                          */
START_COLOR_BLUE=
END_COLOR_BLUE=
START_COLOR_GREEN=
END_COLOR_GREEN=
START_COLOR_CYAN=
END_COLOR_CYAN=
START_COLOR_RED=
END_COLOR_RED=
START_COLOR_MAGENTA=
END_COLOR_MAGENTA=
START_COLOR_YELLOW=
END_COLOR_YELLOW=
START_COLOR_BLACK=
END_COLOR_BLACK=
START_COLOR_WHITE=
END_COLOR_WHITE=

/*                        Highlight Specifications                    */
/* These definitions will determine how things which are sent by the  */
/* host to be displayed or printed as underlined, reverse video, or   */
/* blinking will be highlighted on your printer.                      */

START_HIGHLIGHT_INTENSE=SDS           /* This is double strike        */
END_HIGHLIGHT_INTENSE=CDS
START_HIGHLIGHT_UNDERLINE=SUL
END_HIGHLIGHT_UNDERLINE=CUL
START_HIGHLIGHT_REVERSE_VIDEO=
END_HIGHLIGHT_REVERSE_VIDEO=
START_HIGHLIGHT_BLINK=
END_HIGHLIGHT_BLINK=

/*                      Character Definitions                         */
SPACE=SPA
EXCLAMATION_POINT=21
QUOTATION_MARKS=22
NUMBER_SIGN=23
DOLLAR_SIGN=24
PERCENT_SIGN=25
AMPERSAND=26
APOSTROPHE=27
LEFT_PARENTHESIS=28
RIGHT_PARENTHESIS=29
ASTERISK=2A
PLUS_SIGN=2B
COMMA=2C
HYPHEN=2D
PERIOD=2E
SLASH=2F
ZERO=0
```

```
ONE=1
TWO=2
THREE=3
FOUR=4
FIVE=5
SIX=6
SEVEN=7
EIGHT=8
NINE=9
COLON=3A
SEMICOLON=3B
LESS_THAN_SIGN=3C
EQUAL_SIGN=3D
GREATER_THAN_SIGN=3E
QUESTION_MARK=3F
AT_SIGN=40
A_CAPITAL=A
B_CAPITAL=B
C_CAPITAL=C
D_CAPITAL=D
E_CAPITAL=E
F_CAPITAL=F
G_CAPITAL=G
H_CAPITAL=H
I_CAPITAL=I
J_CAPITAL=J
K_CAPITAL=K
L_CAPITAL=L
M_CAPITAL=M
N_CAPITAL=N
O_CAPITAL=O
P_CAPITAL=P
Q_CAPITAL=Q
R_CAPITAL=R
S_CAPITAL=S
T_CAPITAL=T
U_CAPITAL=U
V_CAPITAL=V
W_CAPITAL=W
X_CAPITAL=X
Y_CAPITAL=Y
Z_CAPITAL=Z
LEFT_BRACKET=5B
BACKSLASH=5C
RIGHT_BRACKET=5D
CIRCUMFLEX_ACCENT=5E
UNDERLINE=5F
GRAVE_ACCENT=60
A_SMALL=a
B_SMALL=b
C_SMALL=c
D_SMALL=d
E_SMALL=e
F_SMALL=f
G_SMALL=g
H_SMALL=h
I_SMALL=i
J_SMALL=j
```

```
K_SMALL=k
L_SMALL=l
M_SMALL=m
N_SMALL=n
O_SMALL=o
P_SMALL=p
Q_SMALL=q
R_SMALL=r
S_SMALL=s
T_SMALL=t
U_SMALL=u
V_SMALL=v
W_SMALL=w
X_SMALL=x
Y_SMALL=y
Z_SMALL=z
LEFT_BRACE=7B
VERTICAL_BAR=7C
RIGHT_BRACE=7D
TILDE_ACCENT=7E
C_CEDILLA_CAPITAL=80
U_DIAERESIS_SMALL=81
E_ACUTE_SMALL=82
A_CIRCUMFLEX_SMALL=83
A_DIAERESIS_SMALL=84
A_GRAVE_SMALL=85
A_OVERCIRCLE_SMALL=86
C_CEDILLA_SMALL=87
E_CIRCUMFLEX_SMALL=88
E_DIAERESIS_SMALL=89
E_GRAVE_SMALL=8A
I_DIAERESIS_SMALL=8B
I_CIRCUMFLEX_SMALL=8C
I_GRAVE_SMALL=8D
A_DIAERESIS_CAPITAL=8E
A_OVERCIRCLE_CAPITAL=8F
E_ACUTE_CAPITAL=90
AE_DIPTHONG_SMALL=91
AE_DIPTHONG_CAPITAL=92
O_CIRCUMFLEX_SMALL=93
O_DIAERESIS_SMALL=94
O_GRAVE_SMALL=95
U_CIRCUMFLEX_SMALL=96
U_GRAVE_SMALL=97
Y_DIAERESIS_SMALL=98
O_DIAERESIS_CAPITAL=99
U_DIAERESIS_CAPITAL=9A
O_SLASH_SMALL=9B
POUND_SIGN=9C
O_SLASH_CAPITAL=9D
MULTIPLY_SIGN=9E
A_ACUTE_SMALL=A0
I_ACUTE_SMALL=A1
O_ACUTE_SMALL=A2
U_ACUTE_SMALL=A3
N_TILDE_SMALL=A4
N_TILDE_CAPITAL=A5
ORDINAL_INDICATOR_FEMININE=A6
```

```
ORDINAL_INDICATOR_MASCULINE=A7
QUESTION_MARK_INVERTED=A8
REGISTERED_TRADEMARK_SYMBOL=A9
LOGICAL_NOT=AA
ONE_HALF=AB
ONE_QUARTER=AC
EXCLAMATION_POINT_INVERTED=AD
LEFT_ANGLE_QUOTES=AE
RIGHT_ANGLE_QUOTES=AF
A_ACUTE_CAPITAL=B5
A_CIRCUMFLEX_CAPITAL=B6
A_GRAVE_CAPITAL=B7
COPYRIGHT_SYMBOL=B8
CENT_SIGN=BD
YEN_SIGN=BE
A_TILDE_SMALL=C6
A_TILDE_CAPITAL=C7
INTERNATIONAL_CURRENCY_SYMBOL=CF
ETH_ICELANDIC_SMALL=D0
ETH_ICELANDIC_CAPITAL=D1
E_CIRCUMFLEX_CAPITAL=D2
E_DIAERESIS_CAPITAL=D3
E_GRAVE_CAPITAL=D4
I_DOTLESS_SMALL=D5
I_ACUTE_CAPITAL=D6
I_CIRCUMFLEX_CAPITAL=D7
I_DIAERESIS_CAPITAL=D8
VERTICAL_LINE_BROKEN=DD
I_GRAVE_CAPITAL=DE
O_ACUTE_CAPITAL=E0
SHARP_S_SMALL=E1
O_CIRCUMFLEX_CAPITAL=E2
O_GRAVE_CAPITAL=E3
O_TILDE_SMALL=E4
O_TILDE_CAPITAL=E5
MICRO_SYMBOL=E6
THORN_ICELANDIC_SMALL=E7
THORN_ICELANDIC_CAPITAL=E8
U_ACUTE_CAPITAL=E9
U_CIRCUMFLEX_CAPITAL=EA
U_GRAVE_CAPITAL=EB
Y_ACUTE_SMALL=EC
Y_ACUTE_CAPITAL=ED
OVERLINE=EE
ACUTE_ACCENT=EF
SYLLABLE_HYPHEN=F0
PLUS_OR_MINUS_SIGN=F1
THREE_QUARTERS=F3
PARAGRAPH_SYMBOL=F4
SECTION_SYMBOL=F5
DIVIDE_SIGN=F6
CEDILLA=F7
DEGREE_SYMBOL=F8
DIAERESIS=F9
MIDDLE_DOT_ACCENT=FA
ONE_SUPERSCRIPT=FB
THREE_SUPERSCRIPT=FC
TWO_SUPERSCRIPT=FD
```

```
REQUIRED_SPACE=SPA
/**********************************************************************/
/*                PC5250 Internal Data Area.                          */
/* Do not change these statements.                                    */
/**********************************************************************/
TOP_MARGIN=
LEFT_MARGIN=
DYNAMIC_START_JOB=00 00 00 00 00 00 00 00 00 00 00
DYNAMIC_END_JOB=00 00
DYNAMIC_SET_PAGE_LENGTH=00 00 00
PRINTER_ID=40 19
/*                      End of Definition File                        */
```

IBM5577.PDF File Contents

```
/**********************************************************************/
/*    PDF FILE (PRINTER DEFINITION FILE) FOR: PS/55 Printer           */
/**********************************************************************/
BEGIN_MACROS
NUL EQU 00
BEL EQU 07
BAK EQU 08
TAB EQU 09
LFF EQU 0A
VTB EQU 0B
FFF EQU 0C
CRR EQU 0D
SEL EQU 11
DC3 EQU 13
ESC EQU 1B
CAN EQU 18
SPA EQU 20
P10 EQU 1B 7E 02 00 01 32
P12 EQU 1B 7E 02 00 01 3C
P13 EQU 1B 7E 02 00 01 43
P15 EQU 1B 7E 02 00 01 4B
SDW EQU 1B 7E 0E 00 01 09
EDW EQU 1B 7E 0E 00 01 0A
SVT EQU 1B 7E 19
SHT EQU 1B 7E 18
CSS EQU 1B 7E 0E 00 01 0F
SSO EQU 1B 7E 0E 00 01 0D
SSU EQU 1B 7E 0E 00 01 0E
SUL EQU 1B 7E 11 00 01 01
CUL EQU 1B 7E 11 00 01 00
LL2 EQU 1B 7E 03 00 01 14
LL3 EQU 1B 7E 03 00 01 1E
LL4 EQU 1B 7E 03 00 01 28
LL6 EQU 1B 7E 03 00 01 3C
LL7 EQU 1B 7E 03 00 01 4B
LL8 EQU 1B 7E 03 00 01 50
SPL EQU 1B 7E 04 00 03 00
INZ EQU 1B 7E 01 00 00
EJC EQU 1B 7E 0E 00 01 06
END_MACROS
/**********************************************************************/
/*                     Session Parameters                             */
/**********************************************************************/
```

```
MAXIMUM_PAGE_LENGTH=066
MAXIMUM_PRINT_POSITION=132
DEFAULT_CPI?=010
DEFAULT_LPI?=006
COMPRESS_LINE_SPACING?=NO
FORM_FEED_ANY_POSITION?=YES
OVERRIDE_FORMATTED_PRINT?=YES
HORIZONTAL_PEL=180
VERTICAL_PEL=120
UNITS_OF_DRAW_LINE=
KANJI_CODE?=SHIFT_JIS
ZENKAKU_SPACE=
PAGE_LENGTH_TYPE?=6INCH
/*******************************************************************/
/*                       Control Codes                             */
/*******************************************************************/
START_JOB=INZ SEL LL6 P10
END_JOB=INZ
BACKSPACE=BAK
BEL=BEL
CARRIAGE_RETURN=CRR
NEW_LINE=CRR LFF
LINE_FEED=LFF
FORM_FEED=EJC
HORIZONTAL_TAB=TAB
VERTICAL_TAB=VTB
DESELECT=DC3
START_SUBSCRIPT=SSU
END_SUBSCRIPT=CSS
START_SUPERSCRIPT=SSO
END_SUPERSCRIPT=CSS
DUP=*
FIELD_MARK=;
SET_HORIZONTAL_TABS=SHT length(HL)-values
SET_VERTICAL_TABS=SVT length(HL)-values
SET_HORIZONTAL_MARGINS=
SET_PAGE_LENGTH=SPL word-value(HL)
SET_2_LINES_PER_INCH=LL2
SET_3_LINES_PER_INCH=LL3
SET_4_LINES_PER_INCH=LL4
SET_6_LINES_PER_INCH=LL6
SET_7.5_LINES_PER_INCH=LL7
SET_8_LINES_PER_INCH=LL8
SET_10_CHARACTERS_PER_INCH=P10
SET_12_CHARACTERS_PER_INCH=P12
SET_13.4_CHARACTERS_PER_INCH=P13
SET_15_CHARACTERS_PER_INCH=P15
START_DOUBLE_WIDTH_CHARACTERS=SDW
END_DOUBLE_WIDTH_CHARACTERS=EDW
IMAGE_TRANSMISSION=1B 25 31 length(HL)-images
FORWARD_HORIZONTAL_SKIP=1B 25 33 word-value(HL)
FORWARD_VERTICAL_STEP_FEED=1B 25 35 word-value(HL)
SET_FONT_SIZE=1B 7E 20 00 03 word-value(HL) 02
SET_TATEGAKI_MODE=1B 7E 0E 00 01 0B
RESET_TATEGAKI_MODE=1B 7E 0E 00 01 0C
SBCS_FONT_LOAD=1B 7E 81 00 28 F0 40 00 18 byte-values F0 40
SELECT_DRAWER=
SET_LOCAL_FONT=
RESET_LOCAL_FONT=
ABS_HORIZONTAL_COLUMN_SKIP=1B 7E 1C 00 02 00 byte-value
REL_HOR_COLUMN_SKIP_TO_RIGHT=1B 7E 1C 00 02 01 byte-value
SET_SOLID_LINE_TYPE=
SET_DOTTED_LINE_TYPE=
SET_LINE_WIDTH_THIN=
SET_LINE_WIDTH_BOLD=
DRAW_LINE=
KANJI_ON=
KANJI_OFF=
ATTRIBUTE_GRID_LINE=1B 7E 16 length(HL)-values
/**********************************************************************/
/*                     Highlight Specifications                       */
/**********************************************************************/
START_HIGHLIGHT_INTENSE=ESC 7E 0E 00 01 17
END_HIGHLIGHT_INTENSE=ESC 7E 0E 00 01 18
START_HIGHLIGHT_UNDERLINE=1B 7E 11 00 01 01
END_HIGHLIGHT_UNDERLINE=1B 7E 11 00 01 00
/**********************************************************************/
/*                     Internal Data Area.                            */
/* Do not change these statement.                                     */
/**********************************************************************/
TOP_MARGIN=
LEFT_MARGIN=
DYNAMIC_START_JOB=00 00 00 00 00 00 00 00 00 00 00
DYNAMIC_END_JOB=00 00
DYNAMIC_SET_PAGE_LENGTH=00 00 00
PRINTER_ID=55 77
/*                      End of Definition File                        */
```

Field Names of Printer Definition Files

Table 55: Field Names of PDF Files on page 436 lists the field names of the printer definition files (PDF files) and their meanings:

Table 55. Field Names of PDF Files

Field Name	Meaning	Remarks
MAXIMUM_-PAGE_LENGTH	Default MPL	Default is 66
MAXIMUM_-PRINT_-POSITION	Default MPP	Default is 132
DEFAULT_CPI?	Default CPI	Default is 10
DEFAULT_LPI?	Default LPI	Default is 6
COMPRESS_-LINE_S-PACING?	Specifies whether to print a line containing only space characters or non-print characters.	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
FORM_-FEED_ANY_-POSITION?	Specifies whether to validate the FF control code on the first line.	Do not change this field.
OVERRIDE_-FORMATTED_-PRINT?	Specifies whether to print NULL characters as blanks.	Do not change this field.
HORIZONTAL_-PEL	FORWARD_HORIZONTAL_SKIP length unit	
VERTICAL_PEL	FORWARD_VERTICAL_SKIP length unit	
IMAGE_HORIZONTAL_PEL	IMAGE_TRANSMISSION horizontal length unit	
IMAGE_VERTICAL_PEL	IMAGE_TRANSMISSION vertical length unit	
LINE_S-PACING_RATIO	SET_VARIABLE_LINE_DENSITY length unit	
PAGE_-LENGTH_TYPE?	SET_PAGE _LENGTH page length parameter type	
FIRST_LEFT_-POSITION	Distance from left paper edge	
FIRST_TOP_-POSITION	Distance from top paper edge	
DRAWER1_-ORIENTATION	Default page orientation for drawer 1	Default is `COR`
DRAWER2_-ORIENTATION	Default page orientation for drawer 2	Default is `COR`
AUTOMATIC_-ORIENTATION	Specifies whether to calculate the page orientation.	
START_JOB	Printer control code sent to a printer to start printing	If the control code specifying LPI/CPI is defined, also change DEFAULT_CPI, DEFAULT_LPI.
END_JOB	Printer control code sent to a printer when printing ends	
BACKSPACE	Backspace control code	
BEL	Bell control code	Specifies the number of blank lines to send in Print Screen Collection mode
CARRIAGE_RETURN	Carriage return control code	
NEW_LINE	New line (CR/LF) control code	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
LINE_FEED	New line control code	
FORM_FEED	Form feed (FF) control code	
HORIZONTAL_TAB	Horizontal tab control code	
VERTICAL_TAB	Vertical tab control code	
DESELECT	Device control 3 control code	
START_SUBSCRIPT	Subscript character specification	
END_SUBSCRIPT	Subscript character specification release	
START_SUPERSCRIPT	Superscript character specification	
END_SUPERSCRIPT	Superscript character specification release	
DUP	Character used for printing DUP codes	
FIELD_MARK	Character used for printing FIELD MARK characters	
SET_HORIZONTAL_TABS	Horizontal tab setup	Do not change this field.
SET_VERTICAL_TABS	Vertical tab setup	Do not change this field.
SET_HORIZONTAL_MARGINS	Right and left margin setup	Do not change this field.
SET_PAGE_LENGTH	Page length setup	Assign the unit used for the defined control code page length to the PAGE_LENGTH_TYPE field. When single sheets are to be used, delete this field.
SET_VARIABLE_LINE_DENSITY	Line density setup	Assign the unit used for the control code length defined to LINE_SPACING_RATIO field.
SET_2_LINES_PER_INCH	New line pitch (2LPI) setup	
SET_3_LINES_PER_INCH	New line pitch (3LPI) setup	
SET_4_LINES_PER_INCH	New line pitch (4LPI) setup	
SET_6_LINES_PER_INCH	New line pitch (6LPI) setup	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
SET_8_LINES_PER_INCH	New line pitch (8LPI) setup	
SET_10_LINES_PER_INCH	New line pitch (10LPI) setup	
SET_10_CHARACTERS_PER_INCH	Character pitch (10CPI) setup	
SET_12_CHARACTERS_PER_INCH	Character pitch (12CPI) setup	
SET_13_CHARACTERS_PER_INCH	Character pitch (13CPI) setup	
SET_15_CHARACTERS_PER_INCH	Character pitch (15CPI) setup	
SET_17_CHARACTERS_PER_INCH	Character pitch (17CPI) setup	
SET_20_CHARACTERS_PER_INCH	Character pitch (20CPI) setup	
IMAGE_TRANSMISSION	Image data setup (vertical 24-dot image)	
SELECT_DRAWER1	Page tray (Primary) setup	
SELECT_DRAWER2	Page tray (Alternate) setup	
SELECT_ENVELOPE	Envelope tray setup	
SELECT_DRAFT_QUALITY	Draft print quality setup	
SELECT_LETTER_QUALITY	Letter print quality setup	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
SELECT_ENHANCED_QUALITY	Enhanced print quality setup	
SET_DUPLEX	Duplex printing setup	
SET_DUPLEX_TUMBLE	Duplex (tumble) printing setup	
RESET_DUPLEX	Duplex printing release	
SET_PORTRAIT_ORIENT	Page orientation (Normal portrait (upright)) setup	
SET_LANDSCAPELEFT_ORIENT	Page orientation (Landscape left (270 degree clockwise rotation of text)) setup	
SET_PORTRAITUPDWN_ORIENT	Page orientation (Portrait upside down (180 degree clockwise rotation of text)) setup	
SET_LANDSCAPERGHT_ORIENT	Page orientation (Landscape right (90 degree clockwise rotation of text)) setup	
FORWARD_HORIZONTAL_SKIP	Variable skip (relative position/dot unit)	Assign the unit of the defined control code length to the HORIZONTAL_PEL field.
FORWARD_VERTICAL_STEP_FEED	Variable line feed (relative position/dot unit)	Assign the unit of the defined control code length to the VERTICAL_PEL field.
SET_FONT_GLOBAL	Global font ID setup	Do not change this field.
SET_GFID_0003	GFID 3 (OCR-B) setup	
SET_GFID_0005	GFID 5 (Orator) setup	
SET_GFID_0011	GFID 11 (Courier 10) setup	
SET_GFID_0012	GFID 12 (Prestige Pica) setup	
SET_GFID_0013	GFID 13 (Artisan 10) setup	
SET_GFID_0018	GFID 18 (Courier Italic 10) setup	
SET_GFID_0019	GFID 19 (OCR-A) setup	
SET_GFID_0020	GFID 20 (Pica) setup	
SET_GFID_0030	GFID 30 (Math Symbol 10) setup	
SET_GFID_0038	GFID 38 (Orator Bold) setup	
SET_GFID_0039	GFID 39 (Gothic Bold 10) setup	
SET_GFID_0040	GFID 40 (Gothic Text 10) setup	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
SET_GFID_0041	GFID 41 (Roman Text 10) setup	
SET_GFID_0042	GFID 42 (Serif Text 10) setup	
SET_GFID_0043	GFID 43 (Serif Italic 10) setup	
SET_GFID_0044	GFID 44 (Katakana 10) setup	
SET_GFID_0045	GFID 45 (APL 10) setup	
SET_GFID_0046	GFID 46 (Courier Bold 10) setup	
SET_GFID_0050	GFID 50 (Shalom 10) setup	
SET_GFID_0066	GFID 66 (Gothic Text 12) setup	
SET_GFID_0068	GFID 68 (Gothic Italic 12) setup	
SET_GFID_0069	GFID 69 (Gothic Bold 12) setup	
SET_GFID_0070	GFID 70 (Serif Text 12) setup	
SET_GFID_0071	GFID 71 (Serif Italic 12) setup	
SET_GFID_0072	GFID 72 (Serif Bold 12) setup	
SET_GFID_0080	GFID 80 (Math Symbol 12) setup	
SET_GFID_0084	GFID 84 (Script 12) setup	
SET_GFID_0085	GFID 85 (Courier 12) setup	
SET_GFID_0086	GFID 86 (Prestige Elite) setup	
SET_GFID_0087	GFID 87 (Letter Gothic 12) setup	
SET_GFID_0091	GFID 91 (Light Italic 12) setup	
SET_GFID_0110	GFID 110 (Letter Gothic Bold 12) setup	
SET_GFID_0111	GFID 111 (Prestige Elite Bold) setup	
SET_GFID_0112	GFID 112 (Prestige Elite Italic) setup	
SET_GFID_0115	GFID 115 (Math Symbol 12) setup	
SET_GFID_0155	GFID 155 (Boldface Italic) setup	
SET_GFID_0158	GFID 158 (Modern) setup	
SET_GFID_0159	GFID 159 (Boldface) setup	
SET_GFID_0160	GFID 160 (Essay) setup	
SET_GFID_0162	GFID 162 (Essay Italic) setup	
SET_GFID_0163	GFID 163 (Essay Bold) setup	
SET_GFID_0168	GFID 168 (Barak PSM) setup	
SET_GFID_0173	GFID 173 (Essay Light) setup	
SET_GFID_0175	GFID 175 (Document) setup	
SET_GFID_0176	GFID 176 (Boldface) setup	
SET_GFID_0177	GFID 177 (Boldface Italic) setup	
SET_GFID_0193	GFID 193 (Math Symbol 12) setup	
SET_GFID_0198	GFID 198 (Math Symbol 10) setup	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
SET_GFID_0204	GFID 204 (Gothic Text 13) setup	
SET_GFID_0221	GFID 221 (Prestige 15) setup	
SET_GFID_0222	GFID 222 (Gothic Text 15) setup	
SET_GFID_0223	GFID 223 (Courier 15) setup	
SET_GFID_0225	GFID 225 (Math Symbol 15) setup	
SET_GFID_0229	GFID 229 (Serif Text 15) setup	
SET_GFID_0230	GFID 230 (Gothic Text 15) setup	
SET_GFID_0245	GFID 245 (Courier Bold 5) setup	
SET_GFID_0252	GFID 252 (Courier 17) setup	
SET_GFID_0253	GFID 253 (Courier Bold 17) setup	
SET_GFID_0254	GFID 254 (Courier 17 (sub/super)) setup	
SET_GFID_0280	GFID 280 (APL 20) setup	
SET_GFID_0281	GFID 281 (Gothic Text 20) setup	
SET_GFID_0290	GFID 290 (Gothic Text 27) setup	
SET_GFID_0751	GFID 751 (Sonoran-Serif 8-pt Roman Medium) setup	
SET_GFID_1051	GFID 1051 (Sonoran-Serif 10-pt Roman Medium) setup	
SET_GFID_1053	GFID 1053 (Sonoran-Serif 10-pt Roman Bold) setup	
SET_GFID_1056	GFID 1056 (Sonoran-Serif 10-pt Italic Medium) setup	
SET_GFID_1351	GFID 1351 (Sonoran-Serif 12-pt Roman Medium) setup	
SET_GFID_1653	GFID 1653 (Sonoran-Serif 16-pt Roman Bold) setup	
SET_GFID_2103	GFID 2103 (Sonoran-Serif 24-pt Roman Bold) setup	
START_HIGHLIGHT_INTENSE	Highlight printing setup	
END_HIGHLIGHT_INTENSE	Highlight printing release	
START_HIGHLIGHT_UNDERLINE	Underline setup	

Table 55. Field Names of PDF Files (continued)

Field Name	Meaning	Remarks
END_HIGHLIGHT_UNDERLINE	Underline release	
TOP_MARGIN	Default top margin	Do not change this field.
LEFT_MARGIN	Default left margin	Do not change this field.
DYNAMIC_START_JOB	Printer control code sent to a printer to start printing (internal use)	Do not change this field.
DYNAMIC_END_JOB	Printer control code sent to a printer to stop printing (internal use)	Do not change this field.
DYNAMIC_SET_PAGE_LENGTH	Page length setup control code sent to a printer at the start of printing (internal use)	Do not change this field.
PRINTER_ID	Printer ID	Do not change this field.
ZENKAKU_SPACE	The size (adjustment unit) of a user-defined character and a HANKAKU character	Do not change this field.
SBCS_FONT_LOAD	Registration of a HANKAKU GAIJI	Do not change this field.
SET_LOCAL_FONT	Set a font set of user-defined characters	Remove this field when user-defined characters are not loaded to a printer.
RESET_LOCAL_FONT	Reset a font set of user-defined characters	Remove this field when user-defined characters are not loaded to a printer.
ATTRIBUTE_GRID_LINE	Grid-line print	Do not change this field.
START_DOUBLE_WIDTH_CHARACTER	Set a double-width character	
END_DOUBLE_WIDTH_CHARACTER	Reset a double-width character	

Note:

1. When using IBM5577.PDF, change FORM_FEED=EJC to FORM_FEED in the file when a continuous form job does not feed correctly.

The following table lists the session parameter field names and their effective values:

Table 56. Effective Values for PDF File Field Names

Field Name	Effective Value
MAXIMUM_PAGE_LENGTH	001 to 255
MAXIMUM_PRINT_POSITION	001 to 255
DEFAULT_CPI?	010/012/015
DEFAULT_LPI?	004/006/008
COMPRESS_LINE_SPACING?	YES/NO
FORM_FEED_ANY_POSITION?	YES/NO
OVERRIDE_FORMATTED_PRINT?	YES/NO
HORIZONTAL_PEL	FORWARD_HORIZONTAL_SKIP length unit
VERTICAL_PEL	FORWARD_VERTICAL_ STEP_FEED length unit
IMAGE_HORIZONTAL_PEL	IMAGE_TRANSMISSION horizontal unit
IMAGE_VERTICAL_PEL	IMAGE_TRANSMISSION vertical unit
LINE_SPACING_RATIO	SET_VARIABLE_LINE_DENSITY length unit
PAGE_LENGTH_TYPE?	LINE/INCH/6INCH*
FIRST_LEFT_POSITION	000 to 1440 in units of 1/1440 inch
FIRST_TOP_POSITION	000 to 1440 in units of 1/1440 inch
DRAWER1_ORIENTATION	LANDSCAPE/PORTRAIT/COR
DRAWER2_ORIENTATION	LANDSCAPE/PORTRAIT/COR
AUTOMATIC_ORIENTATION	YES/NO
* 6/INCH indicates that page length should be specified in units of 1/6 inch.	

Note:

1. If one of the desired CPI/LPI settings is not exactly supported by the printer, set the nearest value. The results of printing might not be as desired.
2. If the units used to specify the control code length defined in FORWARD_HORIZONTAL _SKIP and FORWARD_VERTICAL _STEP_FEED are not the same as the units used to specify the HORIZONTAL_PEL and VERTICAL_PEL, the desired output will not be obtained.
3. If the units used to specify the control code length defined in SET_VARIABLE_LINE_DENSITY are not the same as the units used to specify the LINE_SPACING_RATIO, the desired output will not be obtained.
4. When FIRST_LEFT_POSITION and FIRST_TOP_POSITION are specified, their values are regarded as specifying the unprintable area in the page of the printer. These values are included in the top margin and the left margin specified by the iSeries™, eServer™ i5, or System i5™ printer control code.

Symbols of Printer Definition Files

The following table lists the symbols that are defined for printer definition files.

Table 57. Printer Symbol Definitions

Field Name	Symbol
SPACE	
EXCLAMATION_POINT	!
QUOTATION_MARKS	"
NUMBER_SIGN	#
DOLLAR_SIGN	$
PERCENT_SIGN	%
AMPERSAND	&
APOSTROPHE	'
LEFT_PARENTHESIS	(
RIGHT_PARENTHESIS	)
ASTERISK	*
PLUS_SIGN	+
COMMA	,
HYPHEN	-
PERIOD	.
SLASH	/
ZERO	0
ONE	1
TWO	2
THREE	3
FOUR	4
FIVE	5
SIX	6
SEVEN	7
EIGHT	8
NINE	9
COLON	:
SEMICOLON	;
LESS_THAN_SIGN	<
EQUAL_SIGN	=
GREATER_THAN_SIGN	>
QUESTION_MARK	?
AT_SIGN	@
A_CAPITAL	A

Table 57. Printer Symbol Definitions (continued)

Field Name	Symbol
B_CAPITAL	B
C_CAPITAL	C
D_CAPITAL	D
E_CAPITAL	E
F_CAPITAL	F
G_CAPITAL	G
H_CAPITAL	H
I_CAPITAL	I
J_CAPITAL	J
K_CAPITAL	K
L_CAPITAL	L
M_CAPITAL	M
N_CAPITAL	N
O_CAPITAL	O
P_CAPITAL	P
Q_CAPITAL	Q
R_CAPITAL	R
S_CAPITAL	S
T_CAPITAL	T
U_CAPITAL	U
V_CAPITAL	V
W_CAPITAL	W
X_CAPITAL	X
Y_CAPITAL	Y
Z_CAPITAL	Z
LEFT_BRACKET	[
BACKSLASH	\
RIGHT_BRACKET	]
CIRCUMFLEX_ACCENT	^
UNDERLINE	_
GRAVE_ACCENT	`
A_SMALL	a
B_SMALL	b
C_SMALL	c

Table 57. Printer Symbol Definitions (continued)

Field Name	Symbol
D_SMALL	d
E_SMALL	e
F_SMALL	f
G_SMALL	g
H_SMALL	h
I_SMALL	i
J_SMALL	j
K_SMALL	k
L_SMALL	l
M_SMALL	m
N_SMALL	n
O_SMALL	o
P_SMALL	p
Q_SMALL	q
R_SMALL	r
S_SMALL	s
T_SMALL	t
U_SMALL	u
V_SMALL	v
W_SMALL	w
X_SMALL	x
Y_SMALL	y
Z_SMALL	z
LEFT_BRACE	{
VERTICAL_BAR	|
RIGHT_BRACE	}
TILDE_ACCENT	~
C_CEDILLA_CAPITAL	Ç
U_DIAERESIS_SMALL	ü
E_ACUTE_SMALL	é
A_CIRCUMFLEX_SMALL	â
A_DIAERESIS_SMALL	ä
A_GRAVE_SMALL	à
A_OVERCIRCLE_SMALL	å
C_CEDILLA_SMALL	ç
E_CIRCUMFLEX_SMALL	ê

Table 57. Printer Symbol Definitions (continued)

Field Name	Symbol
E_DIAERESIS_SMALL	ë
E_GRAVE_SMALL	è
I_DIAERESIS_SMALL	ï
I_CIRCUMFLEX_SMALL	î
I_GRAVE_SMALL	ì
A_DIAERESIS_CAPITAL	Ä
A_OVERCIRCLE_CAPITAL	Å
E_ACUTE_CAPITAL	É
AE_DIPTHONG_SMALL	æ
AE_DIPTHONG_CAPITAL	Æ
O_CIRCUMFLEX_SMALL	ô
O_DIAERESIS_SMALL	ö
O_GRAVE_SMALL	ò
U_CIRCUMFLEX_SMALL	û
U_GRAVE_SMALL	ù
Y_DIAERESIS_SMALL	ÿ
O_DIAERESIS_CAPITAL	Ö
U_DIAERESIS_CAPITAL	Ü
O_SLASH_SMALL	ø
POUND_SIGN	£
O_SLASH_CAPITAL	Ø
MULTIPLY_SIGN	×
A_ACUTE_SMALL	á
I_ACUTE_SMALL	í
O_ACUTE_SMALL	ó
U_ACUTE_SMALL	ú
N_TILDE_SMALL	ñ
N_TILDE_CAPITAL	Ñ
ORDINAL_INDICATOR_FEMININE	ª
ORDINAL_INDICATOR_MASCULINE	º
QUESTION_MARK_INVERTED	¿
REGISTERED_TRADEMARK_SYMBOL	®
LOGICAL_NOT	¬
ONE_HALF	½

Table 57. Printer Symbol Definitions (continued)

Field Name	Symbol
ONE_QUARTER	¼
EXCLAMATION_POINT_INVERTED	¡
LEFT_ANGLE_QUOTES	«
RIGHT_ANGLE_QUOTES	»
A_ACUTE_CAPITAL	Á
A_CIRCUMFLEX_CAPITAL	Â
A_GRAVE_CAPITAL	À
COPYRIGHT_SYMBOL	©
CENT_SIGN	¢
YEN_SIGN	¥
A_TILDE_SMALL	ã
A_TILDE_CAPITAL	Ã
E_CIRCUMFLEX_CAPITAL	Ê
E_DIAERESIS_CAPITAL	Ë
I_ACUTE_CAPITAL	Í
I_CIRCUMFLEX_CAPITAL	Î
I_DIAERESIS_CAPITAL	Ï
VERTICAL_LINE_BROKEN	¦
I_GRAVE_CAPITAL	Ì
O_ACUTE_CAPITAL	Ó
O_CIRCUMFLEX_CAPITAL	Ô
O_GRAVE_CAPITAL	Ò
O_TILDE_SMALL	õ
O_TILDE_CAPITAL	Õ
MICRO_SYMBOL	µ
U_ACUTE_CAPITAL	Ú
U_CIRCUMFLEX_CAPITAL	Û
U_GRAVE_CAPITAL	Ù
ACUTE_ACCENT	´
SYLLABLE_HYPHEN	-
PLUS_OR_MINUS_SIGN	±
THREE_QUARTERS	¾

Table 57. Printer Symbol Definitions (continued)

Field Name	Symbol
PARAGRAPH_SYMBOL	¶
SECTION_SYMBOL	§
DIVIDE_SIGN	÷
DEGREE_SYMBOL	°
ONE_SUPERSCRIPT	¹
THREE_SUPERSCRIPT	³
TWO_SUPERSCRIPT	²
REQUIRED_SPACE	
INTERNATIONAL_CURRENCY_SYMBOL	¤
ETH_ICELANDIC_SMALL	ð
ETH_ICELANDIC_CAPITAL	Đ
SHARP_S_SMALL	ß
THORN_ICELANDIC_SMALL	þ
THORN_ICELANDIC_CAPITAL	Þ
Y_ACUTE_SMALL	ý
Y_ACUTE_CAPITAL	Ý
OVERLINE	—
CEDILLA	¸
DIAERESIS	..
MIDDLE_DOT_ACCENT	.

Using Printer Control Codes

This section explains the String (SCS) control codes, or Final Form Text: Document Content Architecture (FFT DCA).

For details of iSeries™, eServer™ i5, or System i5™ printer control codes, refer to *AS/400 Guide to Programming for Printing*.

This table matches the iSeries™, eServer™ i5, or System i5™ table of fonts. PC400 builds a PC spool file with the selected font in it. The printer driver picks up the spool file and the font and sends it to the printer where the expected font is used. Refer to *Printer Device Programming* for additional information on other useful tables.

The following factors can produce unexpected results:

- Not all fonts are available on a PC or printer device.
  If the font that was selected from the table and incorporated into the spool file cannot be found on a PC or on a printer, the printer driver determines how to present the data on a printer. For example, HP printer drivers have the following order of considerations:
  - HP Font Priority Considerations:
    1. Symbol Set
    2. Spacing
    3. Pitch
    4. Height
    5. Style
    6. Stroke Weight
    7. Typeface Family
    8. Resolution
    9. Orientation
  - Location: printer ROM, SIMM module ROM, cartridge ROM, printer RAM
    Priority of locations:
    1. Soft font (lowest ID first)
    2. Cartridge Font
    3. SIMM Font
    4. Internal Font
  - 600 dpi has priority over 300 dpi

  To avoid this uncertainty, it is recommended that you update the table so that only the fonts that are available in the given environment are used.
- NLS

  The iSeries™, eServer™ i5, or System i5™ font has NLS characters in it, which are not part of a corresponding PC font. The IBM-supplied table does not support character sets other than ANSI, although it provides a field for them. In this case, PC400 builds a PC spool file with a font that doesn't recognize NLS.

  To fix the problem, it is recommended that you either change the font names in the table to the NLS enabled on a PC/printer, or if the font has the same name as the one in the table, update a Character Set value.
- After you decide to scale a printout which may be a result of CORig or BesFitting, you usually decrease the distance horizontally as well as vertically between characters. This can result in overlapping. PC400 attempts to adjust the given character size to a new one. A problem may occur when a font defined in the table is not a scalable font. Like GFID011, the most heavily used iSeries™, eServer™ i5, or System i5™ font is mapped to Courier. Courier is a non-scalable font which has only a limited number of character box sizes. To avoid possible problems it is recommended that you use Courier New instead, which is a scalable TTF font.

Printer Control Code Format

Some printer control codes perform single, specific functions by themselves, while others perform multiple functions according to the parameters specified after the control code.

A printer control code with parameters has the following format:

Control Code	Count	Parameter 1	Parameter 2...
1 or 2 bytes	1 byte	1 or 2 bytes	1 or 2 bytes

A count consists of 1 byte, and indicates the length of the parameters (including the count) after the control code, in bytes. For example, a count and two 1-byte parameters is shown as X'03', because the count itself is included. Some control codes, such as the Printing Position (PP) control code, do not have counts.

A parameter can be 1 or 2 bytes in length. The number of bytes depends on the control codes. Not all control codes have parameters.

Note: In this manual, counts and parameters that are actually processed in binary are all expressed in hexadecimal (0–F) to improve readability.

Parameter Definition of Printer Control Codes

Some printer control codes require that parameters be specified. Pay particular attention when defining a parameter because how this is done depends on the parameter type. If the definition method for another type is used, the desired output will not be obtained.

The following table lists printer control code parameter types and their meanings:

Table 58. Printer Control Code Parameter Types

Parameter Type	Meaning
byte-value	One-byte parameter.
byte-values	Multibyte parameter. Used if operands are fixed.
word-value(HL)	One-word parameter (higher and lower bytes).
word-value(LH)	One-word parameter (lower and higher bytes).
length(HL)-values	Multibyte parameter requiring operands. An operand consists of a higher and lower byte. Used if operands are variable.
length(LH)-values	Multibyte parameter requiring operands. An operand consists of a lower and higher byte. Used if operands are variable.
length(HL)-images	Image data requiring operands. An operand consists of a higher and lower byte.
length(LH)-images	Image data requiring operands. An operand consists of a lower and higher byte.
decimal-characters	Decimal characters parameter.

Supported Control Codes

Z and I Emulator for Windows supports all control codes for the 3812 printer.

Programming Notes

This section briefly explains how a printer reacts if a partial control code is received, or if an incomplete control code is sent.

If the transmission of a control code is interrupted, the printer waits for the remaining part of the code. If the data stream (the series of data units and control codes) sent after the interruption is consistent with the data stream sent before the interruption, (that is, if one complete printer control code is restored by chaining), the control code is processed as is.

If the two parts of the data stream are inconsistent, an error occurs. A negative response to an "Invalid Printer Parameter" is sent to the host system, or treated as a no-op (no operation; ignored because of a meaningless code). Detailed information is not sent to the host system if an error occurs in a control code. After programming, the data stream must be checked thoroughly by repeating the printing test.

Restrictions and Notes for iSeries, eServer i5, or System i5 Commands and Printer Setup

This section provides supplementary notes and explains restrictions for printing.

Printer Control Codes

Table 59. Printer Control Codes

Printer Control Code	**If *Use PDT file* is Selected:**	**If a Windows® Printer Driver is Used:**
SCD — Set the Character Density	Select the GFID for the valid character distance (CD) parameter as shown in Table 60: SCD Parameter Values on page 454.	
	If the specified font is not supported, a substitution is provided. For example, when 15 CPI font is specified but the font is not supported, the supported 17 CPI font is substituted.	See How to Determine PC400 Font on page 455.
SFG — Set Font ID through GFID	Recognizable GFIDs are restricted. (See Field Names of Printer Definition Files on page 436.) If the specified GFID is not supported and it is out of the range from 154 through 200, the closest font width from the fonts shown in Table 61: Commonly Used SFG GFID Values on page 454 is substituted.	

Table 59. Printer Control Codes

(continued)

Printer Control Code	**If *Use PDT file* is Selected:**	**If a Windows® Printer Driver is Used:**
	If the specified GFID is not supported and it is between 154 and 200, the following font is substituted: **Font name:** Document; **GFID value:**175.	
		See How to Determine PC400 Font on page 455.
BUS — Begin Underscore		The selected font might not support the underscore.
BES — Begin Emphasis		The selected font might not support the emphasis.
STO — Set Text Orientation		The page orientation can be changed to portrait or landscape. The direction, which is up, down, left, or right on the paper, depends on the Microsoft® Windows® printer driver. If the page orientation is changed, the current paper is ejected.

Table 60. SCD Parameter Values

CD Parameter	**Character Pitch (normal)**	**GFID Value (COR)**
000A	10 CPI	13 CPI 204
000B	Proportional	13 CPI 175
000C	12 CPI	15 CPI 86
000F	15 CPI	20 CPI 230
00FF	10 CPI	13 CPI 204

The following table lists only the most commonly used GFIDs. See Table 62: iSeries, eServer i5, or System i5 Font Parameters on page 455 for the complete list.

Table 61. Commonly Used SFG GFID Values

Font Name	**GFID value**
Courier Bold 5	245
Courier 10	11
Prestige Elite 12	86
Gothic-text 13	204
Gothic-text 15	230
Courier 17	252
Gothic-text 20	281
Gothic-text 27	290

When you use the Windows® printer driver, the spooler must be on.

How to Determine PC400 Font

When a print job is created on iSeries™, eServer™ i5, or System i5™, a certain font, identified by font ID (GFID), is associated with it. Such a font can be specified by the following parameters:

- Font family
- Pitch and family
- Character set
- Width
- Height
- Weight
- Style

System fonts are available with all print drivers, and more flexible device fonts are unique to each printer and printer driver. These fonts are also more fixed as to CPI, weight, code pages, and other criteria that can preclude use of it when matching to the host specified needs.

There are problems associated with mapping an iSeries™, eServer™ i5, or System i5™ font to a PC font. When your PC has all of the fonts that you need, use the following table to determine the best font to use.

Table 62. iSeries, eServer i5, or System i5 Font Parameters

Entry ID	Font Family	GFID	Pitch & Family	Character Set	Width	Height	Weight	Style
GFID0003	OCR-B	3	49	0	144	240	400	0
GFID0005	Orator	5	49	0	144	240	400	0
GFID0011	Courier	11	49	0	144	240	400	0
GFID0012	Prestige	12	49	0	144	240	400	0
GFID0013	Artisan	13	49	0	144	240	400	0
GFID0018	Courier Italic	18	49	0	144	240	400	255
GFID0019	OCR-A	19	49	0	144	240	400	0
GFID0020	Pica	20	49	0	144	240	400	0
GFID0030	Symbol	30	49	2	144	240	400	0
GFID0038	Orator	38	49	0	144	240	400	0
GFID0039	Gothic	39	49	0	144	240	800	0
GFID0040	Gothic	40	49	0	144	240	800	0
GFID0041	Roman	41	49	0	144	240	400	0
GFID0042	Serif	42	49	0	144	240	400	0
GFID0043	Serif	43	49	0	144	240	400	255
GFID0044	Katakana	44	49	0	144	240	400	0

Table 62. iSeries, eServer i5, or System i5 Font Parameters

(continued)

Entry ID	Font Family	GFID	Pitch & Family	Character Set	Width	Height	Weight	Style
GFID0045	APL	45	49	0	144	240	400	0
GFID0046	Courier Bold	46	49	0	144	240	800	0
GFID0050	Shalom	50	49	0	144	240	400	0
GFID0066	Gothic	66	49	0	144	240	400	0
GFID0068	Gothic	68	49	0	120	240	400	255
GFID0069	Gothic	69	49	0	120	240	800	0
GFID0070	Serif	70	49	0	120	240	400	0
GFID0071	Serif	71	49	0	120	240	400	255
GFID0072	Serif	72	49	0	120	240	800	0
GFID0080	Symbol	80	49	2	120	240	400	0
GFID0084	Script	84	49	0	120	240	400	0
GFID0085	Courier	85	49	0	120	240	400	0
GFID0086	Prestige	86	49	0	120	240	400	0
GFID0087	Letter-Gothic	87	49	0	120	240	400	0
GFID0091	Light	91	49	0	120	240	400	255
GFID0107	Courier	107	49	0	120	240	400	0
GFID0110	Letter-Gothic	110	49	0	120	240	800	0
GFID0111	Prestige	111	49	0	120	240	800	0
GFID0112	Prestige	112	49	0	120	240	400	255
GFID0115	Symbol	115	49	2	120	240	400	0
GFID0155	Boldface	155	18	0	120	240	400	0
GFID0158	Document	158	18	0	120	240	400	0
GFID0159	Boldface	159	18	0	120	240	800	0
GFID0160	Essay	160	34	0	120	240	800	0
GFID0162	Essay	162	34	0	120	240	800	255
GFID0163	Essay	163	34	0	120	240	800	0
GFID0168	Barak	168	18	0	120	240	400	0
GFID0173	Essay	173	34	0	120	240	400	0
GFID0175	Document	175	18	0	120	240	400	0
GFID0176	Boldface	176	18	0	120	240	800	0
GFID0177	Boldface	177	18	0	120	240	800	255
GFID0193	Symbol	193	49	2	120	240	400	0
GFID0198	Symbol	198	49	2	144	240	400	0

Table 62. iSeries, eServer i5, or System i5 Font Parameters

(continued)

Entry ID	Font Family	GFID	Pitch & Family	Character Set	Width	Height	Weight	Style
GFID0204	Gothic	204	49	0	108	210	400	0
GFID0221	Prestige	221	49	0	96	210	400	0
GFID0222	Gothic	222	49	0	96	210	400	0
GFID0223	Courier	223	49	0	96	210	400	0
GFID0225	Symbol	225	49	2	96	240	400	0
GFID0229	Serif	229	49	0	96	210	400	0
GFID0230	Gothic	230	49	0	96	210	400	0
GFID0245	Courier Bold	245	49	0	288	240	800	0
GFID0252	Courier	252	49	0	84	240	400	0
GFID0253	Courier Bold	253	49	0	84	240	800	0
GFID0254	Courier	254	49	0	84	120	400	0
GFID0280	APL	280	49	0	72	120	400	0
GFID0281	Gothic	281	49	0	72	120	400	0
GFID0290	Gothic	290	49	0	54	120	400	0
GFID0751	Sono-ran-serif	751	18	0	54	162	400	0
GFID1051	Sono-ran-serif	1051	18	0	66	198	400	0
GFID1053	Sono-ran-serif	1053	18	0	66	198	800	0
GFID1056	Sono-ran-serif	1056	18	0	66	198	400	255
GFID1351	Sono-ran-serif	1351	18	0	84	240	400	0
GFID1653	Sono-ran-serif	1653	18	0	108	312	800	0
GFID2103	Sono-ran-serif	2103	18	0	162	480	800	0

Note:

1. Default GFID from host is 011, we use Courier 10 CPI.
2. A print driver will change the font to its default if you ask for a font name that it does not recognize. Some drivers recognize Gothic, but the DeskJet drivers do not.

Table 62. iSeries, eServer i5, or System i5 Font Parameters

(continued)

Entry ID	Font Family	GFID	Pitch & Family	Character Set	Width	Height	Weight	Style
3. Most print drivers default to Courier New, instead of Courier. So Courier switched to Courier New works, but Gothic switched to Courier New changes the font family. Preferably, you should explicitly select Courier New. 4. Special fonts like CourHEB and GRCOUR869 (for Greek) has required them to be added to PCSPD.DAT to work. Note if the operating system properly or fully supports a language, that font could be the default font instead of Courier New.								

Avoiding iSeries System Dump

If you are running OS/400® Version 3 Release 1, and you attempt to perform a Telnet 5250 mode host print operation, you may experience an iSeries™ system dump under certain conditions. To prevent this from occurring you should apply PTF SF35327 on OS/400®.

PFT Migration Utility

The PFT migration utility converts the printer function table (PFT) for the PC Support/400 workstation feature to a printer definition file (PDF) for PC400.

This section describes the operator interface of the PFT migration utility.

For details about PFT, refer to *AS/400 PC Support: DOS and OS/2 Technical Reference*.

Using the PFT Migration Utility

The file name of the PFT Migration Utility program is PCSPFC.EXE. It is a Windows® application, and you can execute it by doing the following:

1. Double-click the program name using the Windows® Explorer utility.
2. Specify the program name (and parameters) in the Windows® Run utility as follows:

   ```
       PCSPFC [ [drive:] [path] PFT-file-name[.extension] ]
   ```

   If no parameter is specified, PCSPFC.EXE displays the Convert PFT to PDF dialog box.
   If you omit a drive name and a directory name, PCSPFC.EXE uses the current drive and the current directory. If you omit an extension, PCSPFC.EXE adds .PFT to the PFT file name.
3. When you execute the PFT Migration Utility, the Convert PFT to PDF dialog box appears. On the Convert PFT to PDF dialog box, select a PFT file from the list box or type a specific PFT file name, and click **OK**. The PFT Migration Utility starts the conversion and displays the PFT File Converter dialog box to show the conversion status.

After the conversion, if you click **Save List** on the PFT File Converter dialog box, conversion messages in the dialog box are saved into a list file. The list file is created in the same directory and with the same name as the PFT file, except the extension. The extension of the list file is .LS2.

If the conversion was completed successfully, you can click **Convert PDF to PDT** from the PFT File Converter dialog box to convert the PDF file to a PDT file. You can also create a PDT file by selecting **Printer Setup** from the File pull-down menu as explained in Using the Printer Definition Table (PDT) File on page 418.

Migration Considerations

When the base PDF file already exists, the converted PDF fields are appended to the end of the base PDF file. The name of the base PDF file is decided as follows:

Table 63. PDF File Name

PFT File Name	Base PDF File Name
`xxxxxxxx.PFT`	`xxxxxxxx.PDF`
`xxxxx.MNL`	`MNLxxxxx.PDF`
`zzzxxxxx.MNL`*	`zzzxxxxx.PDF`*
: * “`zzz`” is not “IBM”.	

Even if the same fields are already defined in the PDF file, the appended fields are effective because the last definition is always effective in a PDF file.

If the base PDF file does not exist in the directory, the PFT Migration Utilitycreates a new PDF file that has only the converted fields from the PFT file. In this case, you should append this file to an appropriate base file manually, because the fields converted from the PFT file do not cover all of the necessary PDF fields.

Therefore, it is recommended that you prepare both the PFT file and its base PDF file in the same directory before the conversion.

Details of Migration

This section describes how the PFT Migration Utility migrates the printer function table (PFT) to the printer definition file (PDF).

Migration from the Printer Function Table

The following table shows the target fields of the PDF for the data in the PFT.

Table 64. Migration from the Printer Function Table to the Printer Definition File

PFT Field Name	PDF Field Name
	Description
INITIALIZATION AND RESET	
Initialization	
Initialization	START_JOB=
	When the data is defined in the Initialization field of PFT, the data is appended to the string START_JOB=.
Filename	
Filename	None
	This field is ignored.
Reset	
Reset	END_JOB=
	When the data is defined in the Reset field of PFT, the data is appended to the string END_JOB=.
VERTICAL LINE SPACING	
6 lines per inch	
6 lines per inch	SET_6_LINES_PER_INCH=
	When the data is defined in the 6 lines per inch field of PFT, the data is appended to the string SET_6_LINES_PER_INCH=.
8 lines per inch	
8 lines per inch	SET_8_LINES_PER_INCH=
	When the data is defined in the 8 lines per inch field of PFT, the data is appended to the string SET_8_LINES_PER_INCH=.
Variable line spacing	
Control Sequence	SET_VARIABLE_DENSITY=
	When the data is defined in the control sequence field of the PFT for the variable line spacing, the data is appended to the string SET_VARIABLE_DENSITY=. The parameter `n` in the control sequence is replaced with the PDF parameter type.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
X/Y	LINE_SPACING_RATIO=
	When the data is defined in the X/Y field of the PFT for the variable line spacing, the value Y/X is appended to the string LINE_SPACING_RATIO= as three-digit or four-digit numbers to indicate the decimal number. For example, when Y/X is 72, the **`072`** is appended to the string LINE_SPACING_RATIO= and **`LINE_SPACING_RATIO=072`** is written to the output file. When the value is greater than 255, the four-

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	digit number is migrated. When the value is less than 255, the three-digit number is migrated.
Indexing Functions	
Begin Superscript	START_SUPERSCRIPT=
	When the data is defined in the Begin Superscript field and End Superscript field of the PFT for the indexing functions, the data is appended to the string START_SUPERSCRIPT=. If the data for the End Superscript is not defined, the data for the Begin Superscript is ignored.
End Superscript	END_SUPERSCRIPT=
	When the data is defined in the End Superscript field and Begin Superscript field of the PFT for the indexing functions, the data is appended to the string END_SUPERSCRIPT=. If the data for the Begin Superscript is not defined, the data for the End Superscript is ignored.
Begin Subscript	START_SUBSCRIPT=
	When the data is defined in the Begin Subscript field and End Subscript field of the PFT for the indexing functions, the data is appended to the string START_SUBSCRIPT=. If the data for the End Subscript is not defined, the data for the Begin Subscript is ignored.
End Subscript	END_SUBSCRIPT=
	When the data is defined in the End Subscript field and Begin Subscript field of the PFT for the indexing functions, the data is appended to the string END_SUBSCRIPT=. If the data for the Begin Subscript is not defined, the data for the End Subscript is ignored.
Reverse 1/2 Index	START_SUPERSCRIPT= END_SUBSCRIPT=
	The data is appended to the START_SUPERSCRIPT= for all of the following conditions: • When the data is not defined in the Begin Superscript field in the PFT or when the data is not defined in the End Superscript field in the PFT. • When the data is defined in the Reverse 1/2 index and Forward 1/2 index field in the PFT. The data is appended to END_SUBSCRIPT= for all of the following conditions:

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	• When the data is not defined in the Begin Subscript field in the PFT or when the data is not defined in the End Subscript field in the PFT. • When the data is defined in the Reverse 1/2 index and Forward 1/2 index field in the PFT.
Forward 1/2 Index	END_SUPERSCRIPT= START_SUBSCRIPT=
	The data is appended to END_SUPERSCRIPT= for all of the following conditions: • When the data is not defined in Begin Superscript field in the PFT or when the data is not defined in End Superscript field in the PFT. • When the data is defined in the Reverse 1/2 index and Forward 1/2 index field in the PFT. The data is appended to START_SUBSCRIPT= for all of the following conditions: • When the data is not defined in the Begin Subscript field in the PFT or when the data is not defined in the End Subscript field in the PFT. • When the data is defined in the Reverse 1/2 index and Forward 1/2 index field in the PFT.
Reverse Index	None
	This field is ignored.
HORIZONTAL LINE SPACING	
5 pitch	
5 pitch	None
	This field is ignored.
8.55 pitch	
8.55 pitch	None
	This field is ignored.
10 pitch	
10 pitch	SET_10_CHARACTERS_PER_INCH=
	When the data is defined in the 10 pitch field of the PFT for the horizontal character spacing, the data is appended to the string SET_10_CHARACTERS_PER_INCH=.
12 pitch	
12 pitch	SET_12_CHARACTERS_PER_INCH=
	When the data is defined in the 12 pitch field of the PFT for the horizontal character spacing, the data is appended to the string SET_12_CHARACTERS_PER_INCH=.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
15 pitch	
15 pitch	SET_15_CHARACTERS_PER_INCH=
	When the data is defined in the 15 pitch field of the PFT for the horizontal character spacing, the data is appended to the string SET_15_CHARACTERS_PER_INCH=.
17.1 pitch	
17.1 pitch	SET_17_CHARACTERS_PER_INCH=
	When the data is defined in the 17.1 pitch field of the PFT for the horizontal character spacing, the data is appended to the string SET_17_CHARACTERS_PER_INCH=.
Horizontal Motion Index	
Control Sequence	None
	This field is ignored.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
X/Y	None
	This field is ignored.
HORIZONTAL RELATIVE MOVEMENT	
Forward Relative Movement	
Control Sequence	FORWARD_HORIZONTAL_SKIP=
	When the data is defined in the Forward Relative Movement field of PFT, the data is appended to the string FORWARD_HORIZONTAL_SKIP=. The parameter `n` in the control sequence is replaced with the PDF parameter type.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
X/Y	HORIZONTAL_PEL=
	When the data is defined in the X/Y field of the PFT for the forward relative movement, the value Y/X is appended to the string HORIZONTAL_PEL= as three-digit or four-digit number to indicate the decimal number. For example, when Y/X is 120, **`120`** is appended to the string HORIZONTAL_PEL= and **`HORIZONTAL_PEL=120`** is written to the output file. When the value is greater than 255, the four-digit number is migrated. When the value is less than 255, the three-digit number is migrated.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	**PDF Field Name** **Description**
Backward Relative Movement	
Control Sequence	None This field is ignored.
Maximum	None This field is ignored.
Offset	None This field is ignored.
X/Y	None This field is ignored.
HIGHLIGHTING	
Begin Emphasis	
Begin Emphasis	START_HIGHLIGHT_INTENSE= When the data is defined in the Begin Emphasis of PFT, the data is appended to the string START_HIGHLIGHT_INTENSE=.
End Emphasis	
End Emphasis	END_HIGHLIGHT_INTENSE= When the data is defined in the End Emphasis of PFT, the data is appended to the string END_HIGHLIGHT_INTENSE=.
Begin Underline	
Begin Underline	START_HIGHLIGHT_UNDERLINE= When the data is defined in the Begin Underline of PFT, the data is appended to the string START_HIGHLIGHT_UNDERLINE=.
End Underline	
End Underline	END_HIGHLIGHT_UNDERLINE= When the data is defined in the End Underline of PFT, the data is appended to the string END_HIGHLIGHT_UNDERLINE=.
Begin Quality Print	
Begin Quality Print	None This field is ignored.
End Quality Print	
End Quality Print	None This field is ignored.
PAPER HANDLING	
Bottom Tray Feed	
Bottom Tray Feed	SELECT_DRAWER2=

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name
	Description
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control of the portrait orientation. For this, the PFT Migration Utility divides this control sequence into two parts and migrates the control sequence for the drawer select and paper feed. The control sequence of the portrait orientation is migrated to SET_PORTRAIT_ORIENT=. If no data is defined in any fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility migrates this control sequence to the SELECT_DRAWER2=. (See Definition of PAPER HANDLING Migration on page 481.)
Top Tray Feed	
Top Tray Feed	SELECT_DRAWER1=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control of the portrait orientation. For this, the PFT Migration Utility divides this control sequence into two parts and migrates the control sequence for the drawer select and paper feed. The control sequence of the portrait orientation is migrated to SET_PORTRAIT_ORIENT=. If no data is defined in any fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility migrates this control sequence to SELECT_DRAWER1=. (See Definition of PAPER HANDLING Migration on page 481.)
Envelope Feed	
Envelope Feed	SELECT_ENVELOPE=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control of the portrait orientation. For this, the PFT Migration Utility divides this control sequence into two parts and migrates the control sequence for the drawer select and paper feed. The control sequence of the portrait orientation is migrated to SET_PORTRAIT_ORIENT=. If no data is defined in any fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility migrates this control sequence to SELECT_ENVELOPE=. (See Definition of PAPER HANDLING Migration on page 481.)
Manual Feed	
Manual Feed	SELECT_DRAWER3=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control of the portrait orientation. For this, the PFT Migration Utility divides this control

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name Description
	sequence into two parts and migrates the control sequence for the drawer select and paper feed. The control sequence of the portrait orientation is migrated to SET_PORTRAIT_ORIENT=. If no data is defined in any fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility migrates this control sequence to SELECT_DRAWER3=. (See Definition of PAPER HANDLING Migration on page 481.)
Continuous Feed	
Continuous Feed	None This field is ignored.
Ignore Paper End Sensor	
Ignore Paper End Sensor	None This field is ignored.
Enable Paper End Sensor	
Enable Paper End Sensor	None This field is ignored.
Eject Automatic Cut Sheet	
Eject automatic Cut Sheet	None This field is ignored.
Eject Manual Cut Sheet	
Eject Manual Cut Sheet	None This field is ignored.
Collate	
Collate	None This field is ignored.
PAPER POSITIONING	
Continuous Forms	
Dist. from Top Paper Edge	None This field is ignored.
Dist. from Left Paper Edge	None This field is ignored.
Location of First Print Column	None This field is ignored.
Manual Feed	
Dist. from Top Paper Edge	None This field is ignored.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
Dist. from Left Paper Edge	None
	This field is ignored.
Location of First Print Column	None
	This field is ignored.
Automatic Feed	
Dist. from Top Paper Edge	None
	This field is ignored.
Dist. from Left Paper Edge	None
	This field is ignored.
Location of First Print Column	None
	This field is ignored.
SET PAGE LENGTH (INCHES)	
Control Sequence	PAGE_LENGTH_TYPE?=INCH SET_PAGE_LENGTH=
	When the data is defined in the control sequence for SET PAGE LENGTH (INCHES) and if any of the data is not defined in the control sequence for SET PAGE LENGTH (LINES), this field is migrated.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
X/Y	None
	This field is ignored.
Top Margin Size	None
	This field is ignored.
Bottom Margin Size	None
	This field is ignored.
SET PAGE LENGTH (LINES)	
Control Sequence	PAGE_LENGTH_TYPE?=LINE SET_PAGE_LENGTH=
	When the length is defined in the control sequence for SET PAGE LENGTH (LINES), this field is migrated.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
X/Y	None

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	**PDF Field Name** **Description**
	This field is ignored.
Top Margin Size	None This field is ignored.
Bottom Margin Size	None This field is ignored.
SET LEFT MARGIN (INCHES)	
Control Sequence	None This field is ignored.
Maximum	None This field is ignored.
Offset	None This field is ignored.
X/Y	None This field is ignored.
SET LEFT MARGIN (COLUMNS)	
Control Sequence	SET_HORIZONTAL_MARGIN= When the data is defined in the control sequence field for SET LEFT MARGIN (COLUMNS), this field is migrated.
Maximum	None This field is ignored.
Offset	None This field is ignored.
CARRIER RETURN/LINE FEED	
Continuous Forms	
Carrier Return (Continuous Forms)	CARRIAGE_RETURN= This field is migrated for the following cases: • Case 1 ◦ The data is defined in the Carrier Return field for the Continuous Forms. ◦ Any of the data in the Carrier Return field is not defined for the Manual Feed and Automatic Feed. • Case 2

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	◦ The data is defined in the Carrier Return field for the Continuous Forms. ◦ The same data is defined in the Carrier Return field for the Manual Feed. ◦ Any of the data in the Carrier Return field is not defined for the Automatic Feed. • Case 3 ◦ The data is defined in the Carrier Return field for the Continuous Forms. ◦ The same data is defined in the Carrier Return field for the Automatic Feed. ◦ Any of the data in the Carrier Return field is not defined for the Manual Feed. • Case 4 ◦ The data is defined in the Carrier Return field for the Continuous Forms. ◦ The same data is defined in the Carrier Return field for the Automatic Feed and Manual Feed.
Line Feed (Continuous Forms)	LINE_FEED=
	This field is migrated for the following cases: • Case 1 ◦ The data is defined in the Line Feed field for the Continuous Forms. ◦ Any of the data in the Line Feed Line field is not defined for the Manual Feed and Automatic Feed. • Case 2 ◦ The data is defined in the Line Feed field for the Continuous Forms. ◦ The same data is defined in the Line Feed field for the Manual Feed. ◦ Any data is not defined in the Line Feed field for the Automatic Feed. • Case 3

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	**PDF Field Name** **Description**
	◦ The data is defined in the Line Feed field for the Continuous Forms. ◦ The same data is defined in the Line Feed field for the Automatic Feed. ◦ Any data is not defined in the Line Feed field for the Manual Feed. • Case 4 ◦ The data is defined in the Line Feed field for the Continuous Forms. ◦ The same data is defined in the Line Feed field for the Automatic Feed and Manual Feed.
Manual Feed	
Carrier Return (Manual Feed)	CARRIAGE_RETURN=
	This field is migrated for the following cases: • Case 1 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ Any of the data in the Line Feed Line field is not defined for the Continuous Forms and Automatic Feed. • Case 2 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Continuous Forms. ◦ Any data is not defined in the Line Feed field for the Automatic Feed. • Case 3 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Automatic Feed. ◦ Any data is not defined in the Line Feed field for the Continuous Forms. • Case 4 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Automatic Feed and Continuous Forms.
Line Feed (Manual Feed)	LINE_FEED=
	This field is migrated for the following cases:

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	• Case 1 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ Any of the data in the Line Feed Line field is not defined for the Continuous Forms and Automatic Feed. • Case 2 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Continuous Forms. ◦ Any data is not defined in the Line Feed field for the Automatic Feed. • Case 3 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Automatic Feed. ◦ Any data is not defined in the Line Feed field for the Continuous Forms. • Case 4 ◦ The data is defined in the Line Feed field for the Manual Feed. ◦ The same data is defined in the Line Feed field for the Automatic Feed and Continuous Forms.
Automatic Feed	
Carrier Return (Automatic Feed)	CARRIAGE_RETURN=
	This field is migrated for the following cases: • Case 1 ◦ The data is defined in the Carrier Return field for the Automatic Feed. ◦ Any of the data in the Carrier Return field is not defined for the Continuous Feed and Manual Feed. • Case 2 ◦ The data is defined in the Carrier Return field for the Automatic Feed. ◦ The same data is defined in the Carrier Return field for the Continuous Forms. ◦ Any of the data in the Carrier Return field is not defined for the Manual Feed. • Case 3

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name Description
	◦ The data is defined in the Carrier Return field for the Automatic Feed. ◦ The same data is defined in the Carrier Return field for the Manual Feed. ◦ Any of the data in the Carrier Return field is not defined for the Continuous Forms. • Case 4 ◦ The data is defined in the Carrier Return field for the Automatic Feed. ◦ The same data is defined in the Carrier Return field for the Manual Feed and Continuous Forms.
Line Feed (Automatic Feed)	LINE_FEED=
	This field is migrated for the following cases: • Case 1 ◦ The data is defined in the Line Feed field for the Automatic Feed. ◦ Any of the data in the Line Feed Line field is not defined for the Continuous Forms and Manual Feed. • Case 2 ◦ The data is defined in the Line Feed field for the Automatic Feed. ◦ The same data is defined in the Line Feed field for the Continuous Forms. ◦ Any data is not defined in the Line Feed field for the Manual Feed. • Case 3 ◦ The data is defined in the Line Feed field for the Automatic Feed. ◦ The same data is defined in the Line Feed field for the Manual Feed. ◦ Any data is not defined in the Line Feed field for the Continuous Forms. • Case 4 ◦ When the data is defined in the Line Feed field for the Automatic Feed. ◦ The same data is defined in the Line Feed field for the Manual Feed and Continuous Forms.
MULTIPLE COPIES	
Print without Clearing Page from	
Top Tray	None

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	This field is ignored.
Bottom Tray	None
	This field is ignored.
Envelope Tray	None
	This field is ignored.
Manual Tray	None
	This field is ignored.
Clear Page Buffer	None
	This field is ignored.
Multiple Copies Variable Control	
Control Sequence	None
	This field is ignored.
Maximum	None
	This field is ignored.
Offset	None
	This field is ignored.
LANDSCAPE PAPER HANDLING	
Bottom Tray Feed	
Bottom Tray Feed	SET_LANDSCAPELEFT_ORIENT=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control sequence for the landscape orientation. For this, the PFT Migration Utility divides the control sequence into two parts. The control sequence for the drawer select and paper feed is ignored because this control sequence is migrated when the data for PAPER HANDLING is processed. The control sequence for landscape orientation is migrated to SET_LANDSCAPE_ORIENT=. (See Definition of PAPER HANDLING Migration on page 481.)
Top Tray Feed	
Top Tray Feed	SET_LANDSCAPELEFT_ORIENT=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control sequence for the landscape orientation. For this, the PFT Migration Utility divides the control sequence into two parts. The control sequence for the drawer select and paper feed is ignored because this control sequence is migrated when the data for PAPER HANDLING is processed. The control sequence for landscape orien-

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	**PDF Field Name**
	Description
	tation is migrated to SET_LANDSCAPE_ORIENT=. (See Definition of PAPER HANDLING Migration on page 481.)
Envelope Feed	
Envelope Feed	SET_LANDSCAPELEFT_ORIENT=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control sequence for the landscape orientation. For this, the PFT Migration Utility divides the control sequence into two parts. The control sequence for the drawer select and paper feed is ignored because this control sequence is migrated when the data for PAPER HANDLING is processed. The control sequence for landscape orientation is migrated to SET_LANDSCAPE_ORIENT=. (See Definition of PAPER HANDLING Migration on page 481.)
Manual Feed	
Manual Feed	SET_LANDSCAPELEFT_ORIENT=
	If the data is defined in some fields for LANDSCAPE PAPER HANDLING, the PFT Migration Utility assumes that this control sequence includes the control sequence for the landscape orientation. For this, the PFT Migration Utility divides the control sequence into two parts. The control sequence for the drawer select and paper feed is ignored because this control sequence is migrated when the data for PAPER HANDLING is processed. The control sequence for landscape orientation is migrated to SET_LANDSCAPE_ORIENT=. (See Definition of PAPER HANDLING Migration on page 481.)
TYPESTYLE DEFINITION	
Default Typestyle Definition	
PC Character Set	None
	This field is ignored.
Initial Control Sequence	None
	This field is ignored.
Ending Control Sequence	None
	This field is ignored.
Individual Typestyle Definition	
Typestyle number	SET_GFID_
	This number is appended as a four-digit number after the string SET_GFID_. For example, when the typestyle number `9` is defined, `0009` is appended after the string SET_GFID_ and SET_GFID_0009= is migrated. And the numbers supported by PC400 are migrated. The numbers not supported by PC400 are ignored.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
PC Character Set	None
	This field is ignored.
Initial Control Sequence	SET_GFID_xxxx=
	This control sequence is appended after the string SET_GFID_xxxx=, where xxxx is the four-digit number defined in the typestyle number field.
Ending Control Sequence	None
	This field is ignored.
Characters	None
	This field is ignored.
Initial Control Sequence	None
	This field is ignored.
Ending Control Sequence	None
	This field is ignored.
Symbols	None
	This field is ignored.
Initial Control Sequence	None
	This field is ignored.
Ending Control Sequence	None
	This field is ignored.
Group Typestyle Definition	
Group Identifier	None
	This field is ignored.
Group Identifier Comment	None
	This field is ignored.
Typestyle number	SET_GFID_
	This number is appended as a four-digit number after the string SET_GFID_. For example, when the typestyle numbers 1, 2, 3, and 4 are defined, `0001`, `0002`, `0003`, and `0004` are appended after the string SET_GFID_ and SET_GFID_0001=, SET_-GFID_0002=, SET_GFID_0003=, and SET_GFID_0004= are migrated because the PC400 does not have the group typestyle definition. The typestyle numbers that are not supported by the PC400 are not migrated.
PC Character Set	None
	This field is ignored.
Initial Control Sequence	SET_GFID_xxxx=

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
	This control sequence is appended after the string SET_GFID_xxxx=, where xxxx is the four-digit number defined in the typestyle number field.
Ending Control Sequence	None
	This field is ignored.
Characters	None
	This field is ignored.
Initial Control Sequence	None
	This field is ignored.
Ending Control Sequence	None
	This field is ignored.
Symbols	None
	This field is ignored.
Initial Control Sequence	None
	This field is ignored.
Ending Control Sequence	None
	This field is ignored.
Character Set Number	
Character Set Number	None
	This field is ignored.
SLOT SELECTION	
Slot 1 Sequence	
Slot 1 Sequence	None
	This field is ignored.
Slot 2 Sequence	
Slot 2 Sequence	None
	This field is ignored.
Slot 3 Sequence	
Slot 3 Sequence	None
	This field is ignored.
Stop Sequence	
Stop Sequence	None
	This field is ignored.
USER DEFINED CONTROL	
Parameters of SET ENVELOP SIZE Command	
Control Number: 984	None

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name
	Description
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
ESC Sequence of SET ENVELOP SIZE Command	
Control Number: 985	None
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
Parameters of SET PAGE SIZE Command	
Control Number: 986	None
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
ESC Sequence of SET PAGE SIZE Command	
Control Number: 987	None
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
Printer Data Stream	
Control Number: 988	None
	This field is ignored.
Control Sequence	SET_FONT_GLOBAL=1B 5B 49 word-value(LH) word-value(HL) word-value(HL) byte-value word-value(HL)
	When `04` is defined in this field, SET_FONT_GLOBAL=1B 5B 49 … word-value(HL) is migrated. When the other value is defined, this field is ignored. `04` means IBM® Personal Printer Data Stream Level 2 or higher. When the migration is done for

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name Description
	PAPER HANDLING, use this information. (See Definition of PAPER HANDLING Migration on page 481.)
Control Sequence File Name	None This field is ignored.
Duplex long edge	
Control Number: 989	None This field is ignored.
Control Sequence	SET_DUPLEX= When the data is defined in the control sequence for the duplex long edge, the data is appended to the string SET_DUPLEX=.
Control Sequence File Name	None This field is ignored.
Duplex short edge	
Control Number: 990	None This field is ignored.
Control Sequence	SET_DUPLEX_TUMBLE= When the data is defined in the control sequence for the duplex short edge, the data is appended to the string SET_DUPLEX_TUMBLE=.
Control Sequence File Name	None This field is ignored.
Simplex	
Control Number: 991	None This field is ignored.
Control Sequence	RESET_DUPLEX= When the data is defined in the control sequence for the simplex, the data is appended to the string RESET_DUPLEX=.
Control Sequence File Name	None This field is ignored.
Jog the output tray	
Control Number: 992	None This field is ignored.
Control Sequence	None This field is ignored.
Control Sequence File Name	None This field is ignored.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name Description
Normal portrait orientation	
Control Number: 993	None
	This field is ignored.
Control Sequence	SET_PORTRAIT_ORIENT=
	When the data is defined in the control field for the normal portrait orientation, the data is appended to the string SET_PORTRAIT_ORIENT=.
Control Sequence File Name	None
	This field is ignored.
Landscape left	
Control Number: 994	None
	This field is ignored.
Control Sequence	SET_LANDSCAPELEFT_ORIENT=
	When the data is defined in the control field for the landscape left, the data is appended to the string SET_LANDSCAPELEFT_ORIENT=.
Control Sequence File Name	None
	This field is ignored.
Portrait upside down orientation	
Control Number: 995	None
	This field is ignored.
Control Sequence	SET_PORTRAITUPDWN_ORIENT=
	When the data is defined in the control field for the portrait upside down orientation, the data is appended to the string SET_PORTRAITUPDWN_ORIENT=.
Control Sequence File Name	None
	This field is ignored.
Landscape right	
Control Number: 996	None
	This field is ignored.
Control Sequence	SET_LANDSCAPERGHT_ORIENT=
	When the data is defined in the control field for the landscape right, the data is appended to the string SET_LANDSCAPERGHT_ORIENT=.
Control Sequence File Name	None
	This field is ignored.
COR in 10 pitch	
Control Number: 997	None
	This field is ignored.

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

	PDF Field Name
PFT Field Name	**Description**
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
COR in 12 pitch	
Control Number: 998	None
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
COR in 15 pitch	
Control Number: 999	None
	This field is ignored.
Control Sequence	None
	This field is ignored.
Control Sequence File Name	None
	This field is ignored.
FUNCTION SELECTION TEST RESPONSES	
Superscript /Subscript	None
	This data is ignored.
Underline	None
	This data is ignored.
Emphasis (Bold)	None
	This data is ignored.
Form Feed	None
	This data is ignored.
Back Space	BACKSPACE=
	This field is migrated when you type YES or NO in response to the prompt during the Backspace Function Selection Test.
Mid-line pitch change	None
	This data is ignored.
Horizontal Character spacing	None
	This data is ignored.
First character position	None

Table 64. Migration from the Printer Function Table to the Printer Definition File

(continued)

PFT Field Name	PDF Field Name
	Description
	This data is ignored.
PSM	None
	This data is ignored.
Cursor Draw	None
	This data is ignored.

Note: When you migrate IBM3812.PFT and IBM3812.MNL, the following fields are added to IBM3812.PDF:

- `FORWARD_VERTICAL_STEP_FEED=1B 5B 43 03 00 E3 word-value(LH)`
- `VERTICAL_PEL=240`

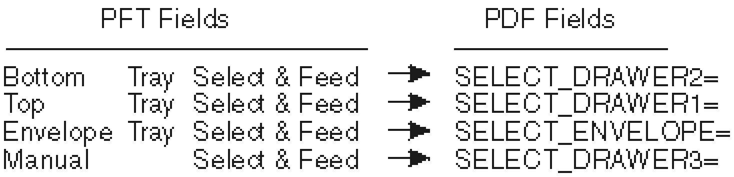
Definition of PAPER HANDLING Migration

The PFT Migration Utility migrates the data for PAPER HANDLING and LANDSCAPE PAPER HANDLING as follows:

1. The PFT Migration Utility checks whether the LANDSCAPE PAPER HANDLING controls are defined.
2. If the LANDSCAPE PAPER HANDLING controls are defined, the PFT Migration Utility assumes that the PAPER HANDLING control sequences include controls to set the page orientation to portrait. Each LANDSCAPE PAPER HANDLING control sequence has the controls to set the page orientation to landscape in addition to the tray select and feed controls.
3. If the LANDSCAPE PAPER HANDLING controls are not defined, the PAPER HANDLING controls do not have the controls to set the page orientation to portrait. If a personal printer supports normal portrait, landscape left, portrait upside down, and landscape right orientation, the page orientation controls are defined in the appropriate user-defined controls.

If the LANDSCAPE PAPER HANDLING controls are not defined

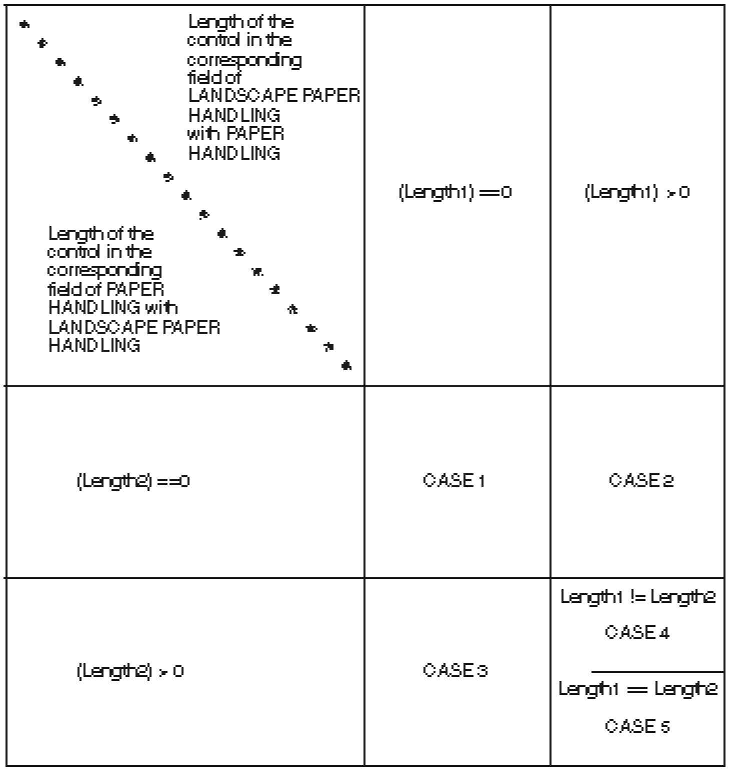
When the PAPER HANDLING controls are not defined, the controls are migrated as follows:



PFT Fields		PDF Fields
Bottom Tray Select & Feed	→	SELECT_DRAWER2=
Top Tray Select & Feed	→	SELECT_DRAWER1=
Envelope Tray Select & Feed	→	SELECT_ENVELOPE=
Manual Select & Feed	→	SELECT_DRAWER3=

If the LANDSCAPE PAPER HANDLING controls are defined

When the PAPER HANDLING controls are defined, these controls have the controls to set the page orientation to portrait. The LANDSCAPE PAPER HANDLING controls have the controls to set the page orientation to landscape. The PFT Migration Utility migrates the controls as follows:



Length of the control in the corresponding field of LANDSCAPE PAPER HANDLING with PAPER HANDLING Length of the control in the corresponding field of PAPER HANDLING with LANDSCAPE PAPER HANDLING	(Length1) ==0	(Length1) > 0
(Length2) ==0	CASE 1	CASE 2
(Length2) > 0	CASE 3	Length1 != Length2 CASE 4 Length1 == Length2 CASE 5

- **CASE 1**

  Since both of the fields are not defined, no data is migrated.

  Example: No data is migrated to SELECT_DRAWER2= under the following conditions. For this example, SELECT_DRAWER2= is not written in the output file.

  - No data is defined in the bottom tray select and feed for PAPER HANDLING.
  - No data is defined in the bottom tray select and feed for LANDSCAPE PAPER HANDLING.
- **CASE 2**

  Since no data is defined in the field for PAPER HANDLING, the PFT Migration Utility cannot compare the data in the corresponding field with the data for LANDSCAPE PAPER HANDLING. The data for LANDSCAPE PAPER HANDLING is migrated to the drawer selection field of PDF regardless, including control of the landscape orientation.

Example: The control of the top tray select and feed for LANDSCAPE PAPER HANDLING is migrated to SELECT_DRAWER1= under the following conditions.

  - No data is defined in the top tray select and feed for PAPER HANDLING.
  - The control is defined in the top tray select and feed for LANDSCAPE PAPER HANDLING.

- **CASE 3**

  Since no data is defined in the field for LANDSCAPE PAPER HANDLING, the PFT Migration Utility cannot compare the data in the corresponding field with the data for PAPER HANDLING. The data for PAPER HANDLING is migrated to the drawer selection field of PDF regardless, including the control of the portrait orientation.

  Example: The control of the manual select and feed for PAPER HANDLING is migrated to SELECT_DRAWER3= under the following conditions.

  - The control is defined in the manual select and feed for PAPER HANDLING.
  - No data is defined in the top tray select and feed for LANDSCAPE PAPER HANDLING.

- **CASE 4**

  Since the length is different, no data is migrated.

  Example: No data is migrated to SELECT_DRAWER2= under the following conditions. For this example, SELECT_DRAWER2= is not written in the output file.

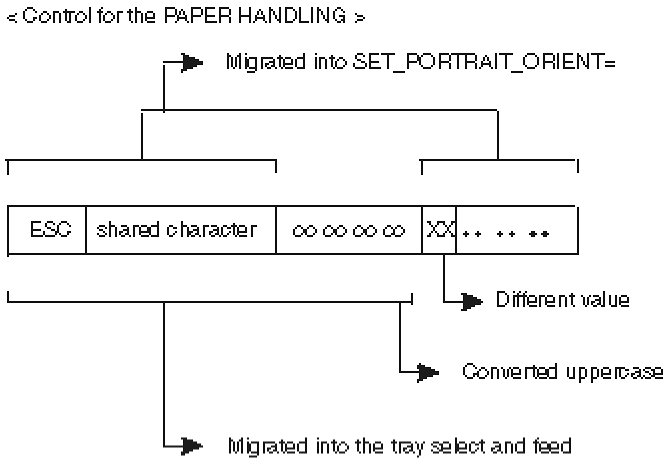
  - The data is defined in the bottom tray select and feed for PAPER HANDLING and the length is 8.
  - The data is defined in the bottom tray select and feed for LANDSCAPE PAPER HANDLING and the length is 10.

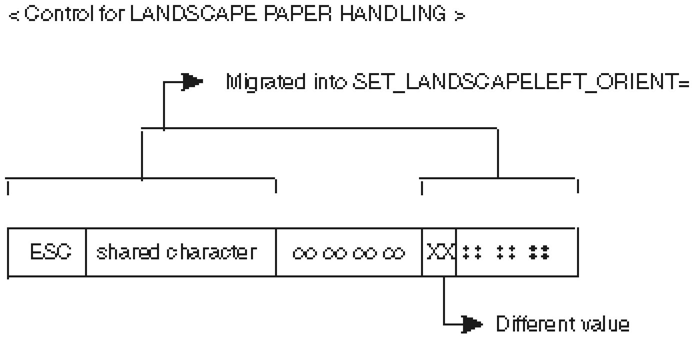
- **CASE 5**

  The PFT Migration Utility compares the data for PAPER HANDLING with the data for LANDSCAPE PAPER HANDLING as follows:

  - If a different value is found in the data, search the control backward for the escape character X'1B'.
  - If the escape character is found in the middle of the control, divide the control into two parts. The PFT Migration Utility assumes that the first part is the control for the tray select and feed, and the second part is the control for the page orientation. The PFT Migration Utility migrates the first part of the PAPER HANDLING control into the tray select and feed, the second part of the PAPER HANDLING control into the SET_PORTRAIT_ORIENT=, and the second part of the LANDSCAPE PAPER HANDLING control into SET_LANDSCAPELEFT_ORIENT=.
  - If the escape character is found at the top of the control and 03 (= HP PCL) is defined in the user-defined control 988, the PFT Migration Utility assumes that the controls are combined. For this, the control begins with the escape character and the two shared characters. For this, the PFT Migration Utility assumes that the last character of the first part is a lowercase letter and converts it to an uppercase letter to indicate that it is a terminating character.

    The PFT Migration Utility divides the controls into the two parts as follows:


< Control for the PAPER HANDLING >
Migrated into SET_PORTRAIT_ORIENT=
ESC
shared character
oo oo oo oo
XX
Different value
Converted uppercase
Migrated into the tray select and feed


< Control for LANDSCAPE PAPER HANDLING >
Migrated into SET_LANDSCAPELEFT_ORIENT=
ESC
shared character
oo oo oo oo
XX
Different value

- If the escape character is found, but the data stream is not HP PCL, the PFT Migration Utility displays the error message to indicate that the PFT Migration Utility cannot migrate the data.
- If the escape character is not found, the PFT Migration Utility displays the error message to indicate that the PFT Migration Utility cannot migrate the data.
- If a different value is not found, the same control is defined for PAPER HANDLING and LANDSCAPE PAPER HANDLING.

The PFT Migration Utility migrates the data in the following order:

1. Top Tray Select and Feed
2. Bottom Tray Select and Feed
3. Manual Select and Feed
4. Envelope Tray Select and Feed

The migration stops for the page orientation, SET_PORTRAIT_ORIENT= and SET_LANDSCAPELEFT_ORIENT=, when the controls for the page orientation are found. For example, the controls for the page orientation are found when the data is migrated for the Bottom Tray Select and Feed. The PFT Migration Utility does not care about the page orientation when migrating the data Manual Select and Feed, and Envelop Tray Select and Feed.

Troubleshooting

There are a number of self-help information resources and tools to help you troubleshoot problems. When you have any problem when using the product, you can perform the following tasks:I

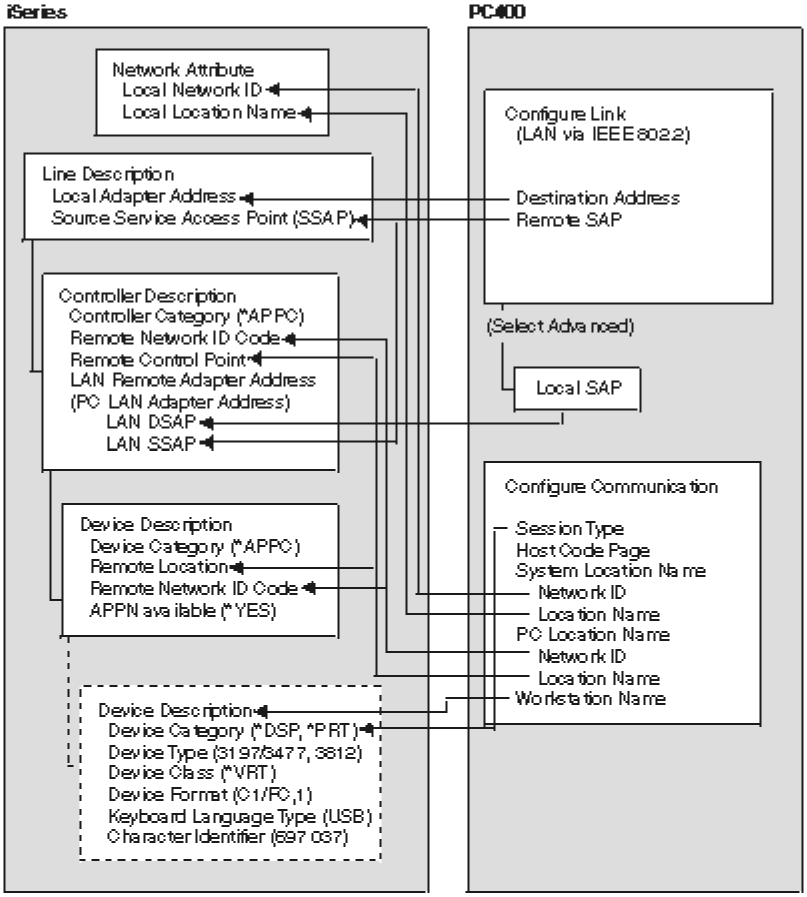
- Refer to the release information for your product for known issues, workarounds, and troubleshooting information.
- Check if a download or fix is available to resolve your problem.
- Search the available knowledge bases to see if the resolution to your problem is already documented.
- If you still need help, contact HCL Software Support and report your problem.

iSeries, eServer i5, or System i5 Configuration Examples

To connect to an iSeries™, eServer™ i5, or System i5™, you need to specify configuration information in the workstation profile that accurately corresponds to the information specified in the iSeries™, eServer™ i5, or System i5™ (referred to as the *device description*).

For example, the LAN attachment via IEEE 802.2 in the following figure shows how the configuration information specified in the workstation profile corresponds to the configuration information in the iSeries™, eServer™ i5, or System i5™.

Figure 10. LAN Attachment via IEEE 802.2



iSeries Device Description

To configure 5250 display or printer sessions, the following values must be set in the iSeries™ device description:

iSeries Device Description	Display Session		Printer Session
	24 x 80	27 x 132	
Device category	*DSP	*DSP	*PRT
Device class	*VRT	*VRT	*VRT
Device type	3197	3477	3812
Device model	C1	FC	1
Keyboard language type	USB+	USB+	-
Character identify code	697 037+	697 037+	-
+ For SBCS, depends on the host code page selection.			

5250 Sessions through One Link

If you want all your 5250 sessions to connect through one link to an iSeries™, eServer™ i5, or System i5™, use the same **PC Location Name** and the same **Link Parameters** for all the sessions.

Tip

Enter **DSPNETA** from a 5250 session to display iSeries™, eServer™ i5, or System i5™ network attributes.

System i5, iSeries, eServer i5, or System Mode Description

PC400 initially uses mode description **QPCSUPP** on the iSeries™, eServer™ i5, or System i5™. If the PC Support/400 program or iSeries™ Access is installed on the iSeries™, eServer™ i5, or System i5™, QPCSUPP need not be created. If mode description QPCSUPP does not exist on the iSeries™, eServer™ i5, or System i5™, create the mode description:

1. Enter the following command on the command line of the main menu of the iSeries™, eServer™ i5, or System i5™:

   ```
   CRTMODD
   ```

   The Creating Mode Description panel appears.

   ```
                      Creating Mode Description (CRTMODD)

   Type the selected items, and push the Enter key.

   Mode Description.................            Name
   Maximum Session................. 8            1-512
   Maximum number of interaction.... 8            1-512
   Number of Local Control Sessions. 4            0-512
   Number of Pre-joined Sessions.... 0            0-512
   Inbound Pacing Value............ 7            0-63
   Outbound Pacing Value........... 7            0-63
   Maximum Length of Request Unit... *CALC            241-16384, *CALC
   Text Description ................ *BLANK





                                                                      End
    F3=Exit    F4=Prompt        F5=Reshow    F10=Add parameter   F12= Cancel
    F13=How to use this panel                F24=More key
   ```

2. Type the necessary values in each field, according to the following table.

Field Name	Input Value
Mode description	**QPCSUPP**
Maximum session	**64**
Maximum number of interactions	**64**
Number of local control sessions	**0**
Number of pre-joined sessions	**0**
Inbound pacing value	**7**
Outbound pacing value	**7**
Maximum length of request unit	***CALC**

Field Name	Input Value
Text description	This field is optional

3. After you type all the values, press the Enter key.

   This completes the creation of the mode description QPCSUPP.

iSeries, eServer i5, or System i5 Device Description for Asynchronous Attachment Example

If you want to use an asynchronous dial attachment, the iSeries™, eServer™ i5, or System i5™ requires that you specify configuration parameters for the controller/line/devices to be used.

The following sample is a typical configuration on the iSeries™, eServer™ i5, or System i5™ for an asynchronous dialed connection through an ASCII Workstation Controller.

1. Enter the following command on the command line of the iSeries™, eServer™ i5, or System i5™ main menu:

   ```
   WRKCFGSTS *CTL CTL03
   ```

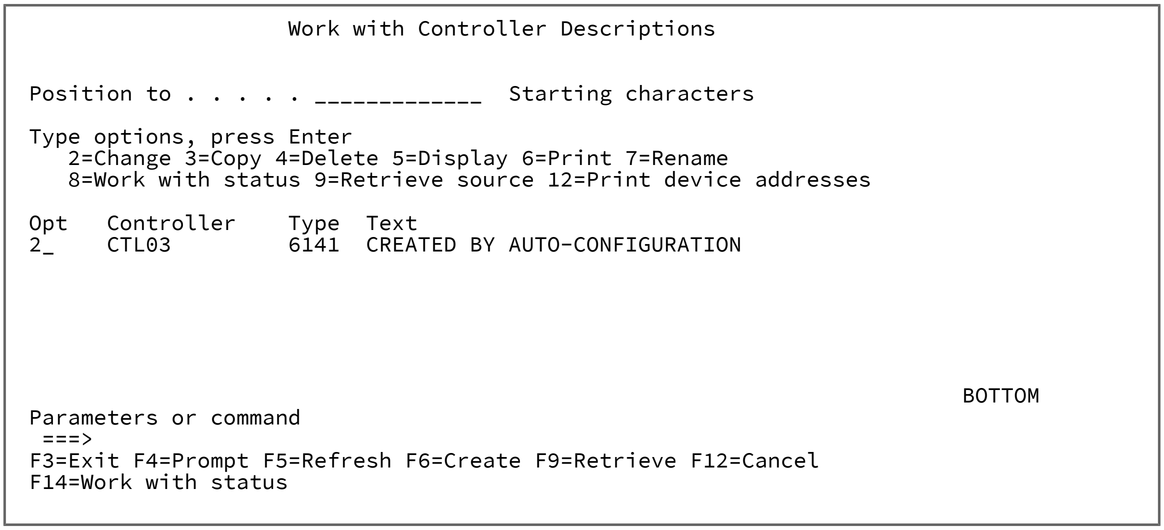
   where *CTL03* is the name of your controller.

   The Work with Configuration Status panel appears.

   ```
                         Work with Configuration Status


    Position to . . . . . _____________  Starting characters

    Type options, press Enter
      1=Vary on 2=Vary off 5=Work with job 8=Work with description
      9-Display mode status ...

    Opt   Description        Status         -------------Job--------------
    8_    CTL03              ACTIVE
    __      ADLCTST          VARY ON PENDING
    __      ASYNC            VARY ON PENDING
    __      ASYNCD           VARY ON PENDING
    __      EZASYNC          VARY ON PENDING
    __      ASYNCP0          VARY ON PENDING
    __      ASYNRTR          ACTIVE
                                                                          BOTTOM
    Parameters or command
     ===>

    F3=Exit F4=Prompt F12=Cancel F23=More options F24=More keys
   ```

2. Enter **8** in the **Opt** field to work with the controller description for CTL03.

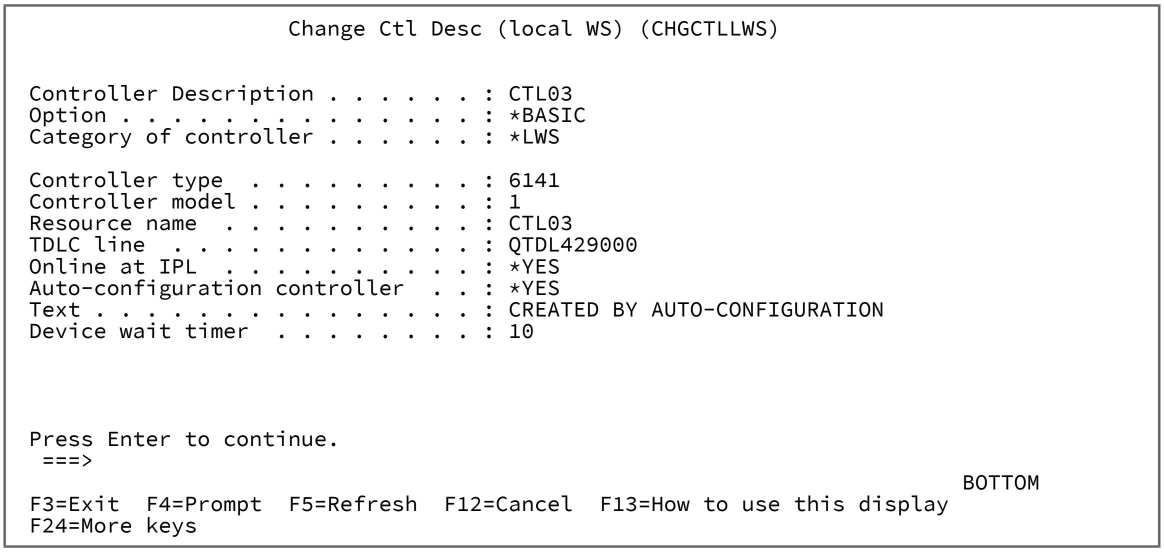
   The Work with Controller Descriptions panel appears.



```
                    Work with Controller Descriptions


Position to . . . . . _____________  Starting characters

Type options, press Enter
   2=Change 3=Copy 4=Delete 5=Display 6=Print 7=Rename
   8=Work with status 9=Retrieve source 12=Print device addresses

Opt   Controller    Type  Text
2_    CTL03         6141  CREATED BY AUTO-CONFIGURATION






                                                                        BOTTOM
Parameters or command
 ===>
F3=Exit F4=Prompt F5=Refresh F6=Create F9=Retrieve F12=Cancel
F14=Work with status
```

3. Enter **2** in the **Opt** field to change the controller description for CTL03.

   The Change Controller Description panel appears.

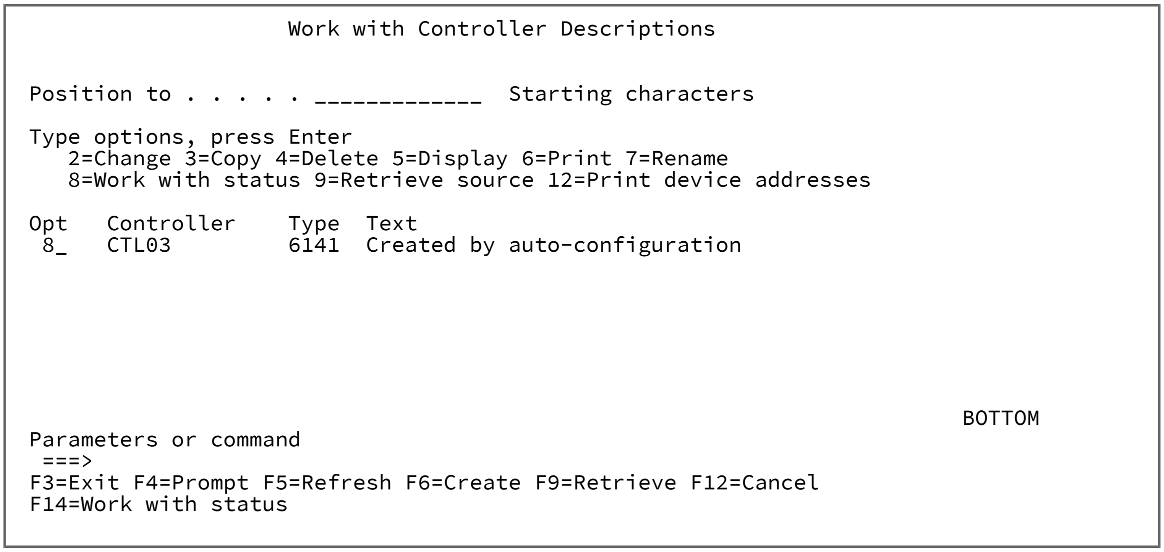


```
                    Change Ctl Desc (local WS) (CHGCTLLWS)


Controller Description . . . . . . : CTL03
Option . . . . . . . . . . . . . . : *BASIC
Category of controller . . . . . . : *LWS

Controller type  . . . . . . . . . : 6141
Controller model . . . . . . . . . : 1
Resource name  . . . . . . . . . . : CTL03
TDLC line  . . . . . . . . . . . . : QTDL429000
Online at IPL  . . . . . . . . . . : *YES
Auto-configuration controller  . . : *YES
Text . . . . . . . . . . . . . . . : CREATED BY AUTO-CONFIGURATION
Device wait timer  . . . . . . . . : 10




Press Enter to continue.
 ===>
                                                                        BOTTOM
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

4. Type the values in each field, according to the following table.

Field Name	Input Value
Controller description	CTL03
Resource name	CTL03
Online at IPL	*YES
Device wait timer	10
Auto-configuration controller	*YES
Text description	This field is optional

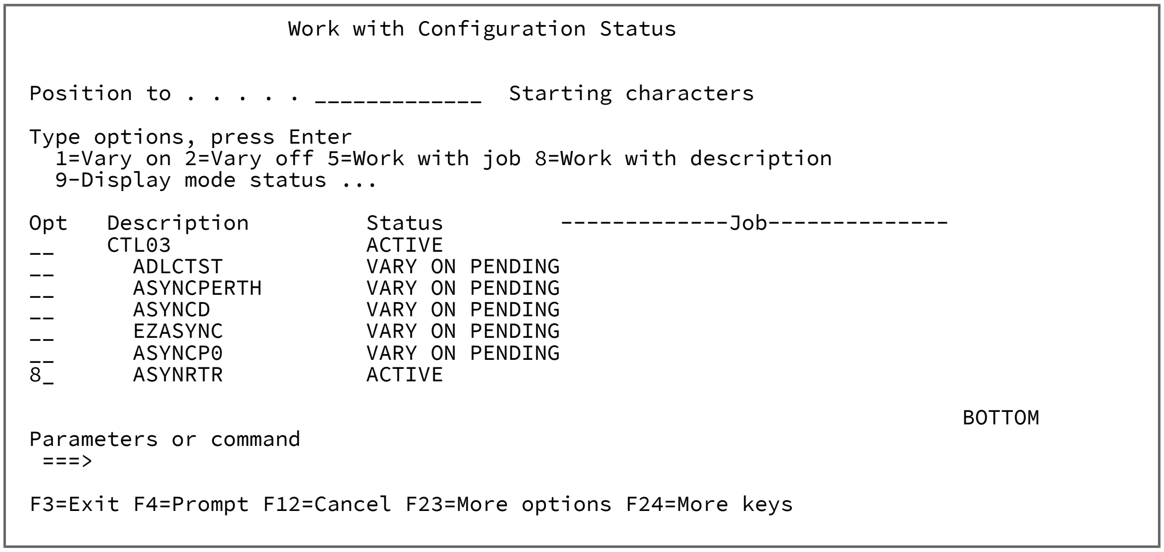
   The Work with Controller Descriptions panel appears.



```
                    Work with Controller Descriptions


Position to . . . . . _____________  Starting characters

Type options, press Enter
   2=Change 3=Copy 4=Delete 5=Display 6=Print 7=Rename
   8=Work with status 9=Retrieve source 12=Print device addresses

Opt   Controller    Type  Text
 8_   CTL03         6141  Created by auto-configuration








                                                                        BOTTOM
Parameters or command
 ===>
F3=Exit F4=Prompt F5=Refresh F6=Create F9=Retrieve F12=Cancel
F14=Work with status
```

5. Enter **8** in the **Opt** field to work with the configuration status.

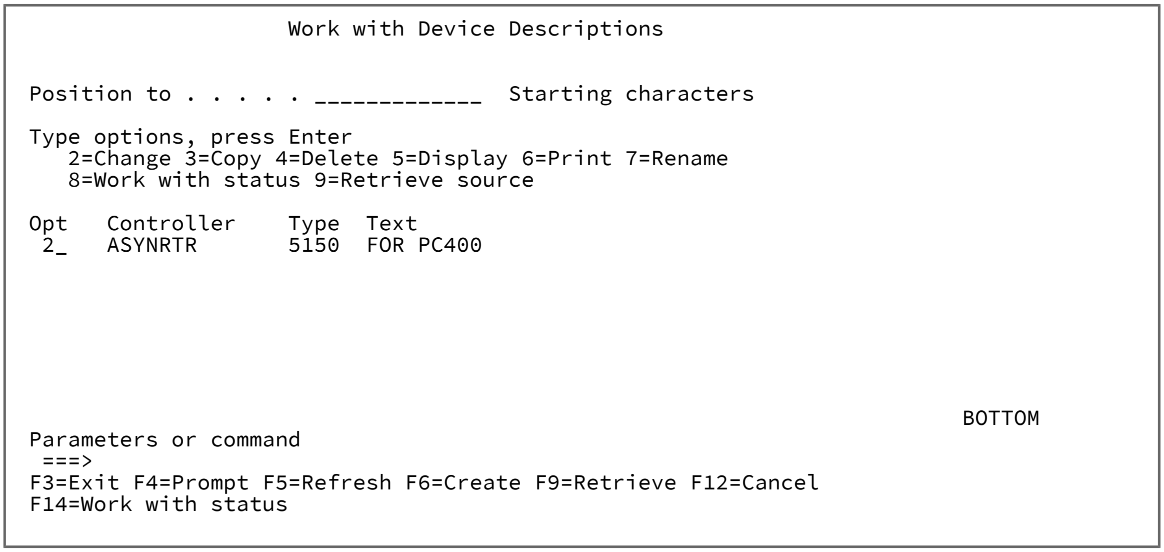
   The Work with Configuration Status panel appears.



```
                    Work with Configuration Status


Position to . . . . . _____________  Starting characters

Type options, press Enter
  1=Vary on 2=Vary off 5=Work with job 8=Work with description
  9-Display mode status ...

Opt   Description         Status         -------------Job--------------
__    CTL03               ACTIVE
__      ADLCTST           VARY ON PENDING
__      ASYNCPERTH        VARY ON PENDING
__      ASYNCD            VARY ON PENDING
__      EZASYNC           VARY ON PENDING
__      ASYNCP0           VARY ON PENDING
8_      ASYNRTR           ACTIVE

                                                                        BOTTOM
Parameters or command
 ===>

F3=Exit F4=Prompt F12=Cancel F23=More options F24=More keys
```

6. Enter **8** in the **Opt** field next to `ASYNRTR` to work with the display device description.

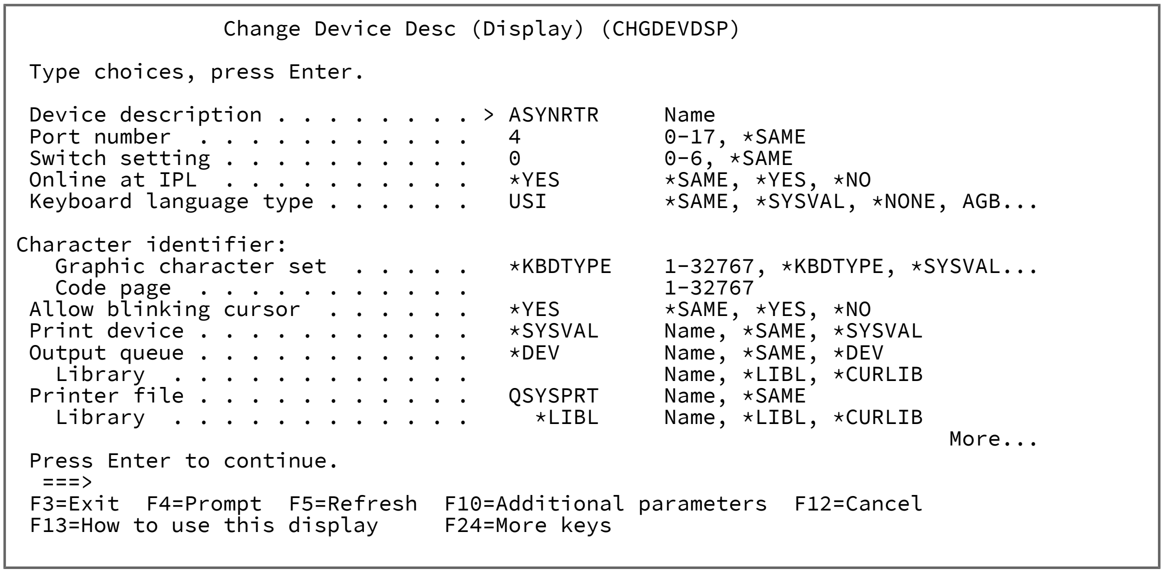
   The Work with Device Descriptions panel appears.



```
                     Work with Device Descriptions


 Position to . . . . . _____________  Starting characters

 Type options, press Enter
    2=Change 3=Copy 4=Delete 5=Display 6=Print 7=Rename
    8=Work with status 9=Retrieve source

 Opt   Controller    Type  Text
  2_   ASYNRTR       5150  FOR PC400








                                                                        BOTTOM
 Parameters or command
  ===>
 F3=Exit F4=Prompt F5=Refresh F6=Create F9=Retrieve F12=Cancel
 F14=Work with status
```

7. Enter **2** in the **Opt** field to change the device description.

   The Change Device Description panel appears.



```
                Change Device Desc (Display) (CHGDEVDSP)

 Type choices, press Enter.

 Device description . . . . . . . . > ASYNRTR     Name
 Port number  . . . . . . . . . . .   4           0-17, *SAME
 Switch setting . . . . . . . . . .   0           0-6, *SAME
 Online at IPL  . . . . . . . . . .   *YES        *SAME, *YES, *NO
 Keyboard language type . . . . . .   USI         *SAME, *SYSVAL, *NONE, AGB...

Character identifier:
   Graphic character set  . . . . .   *KBDTYPE    1-32767, *KBDTYPE, *SYSVAL...
   Code page  . . . . . . . . . . .               1-32767
 Allow blinking cursor  . . . . . .   *YES        *SAME, *YES, *NO
 Print device . . . . . . . . . . .   *SYSVAL     Name, *SAME, *SYSVAL
 Output queue . . . . . . . . . . .   *DEV        Name, *SAME, *DEV
   Library  . . . . . . . . . . . .               Name, *LIBL, *CURLIB
 Printer file . . . . . . . . . . .   QSYSPRT     Name, *SAME
   Library  . . . . . . . . . . . .     *LIBL     Name, *LIBL, *CURLIB
                                                                       More...
 Press Enter to continue.
  ===>
 F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
 F13=How to use this display     F24=More keys
```

This completes the creation of the controller and display device descriptions for an asynchronous dial attachment.

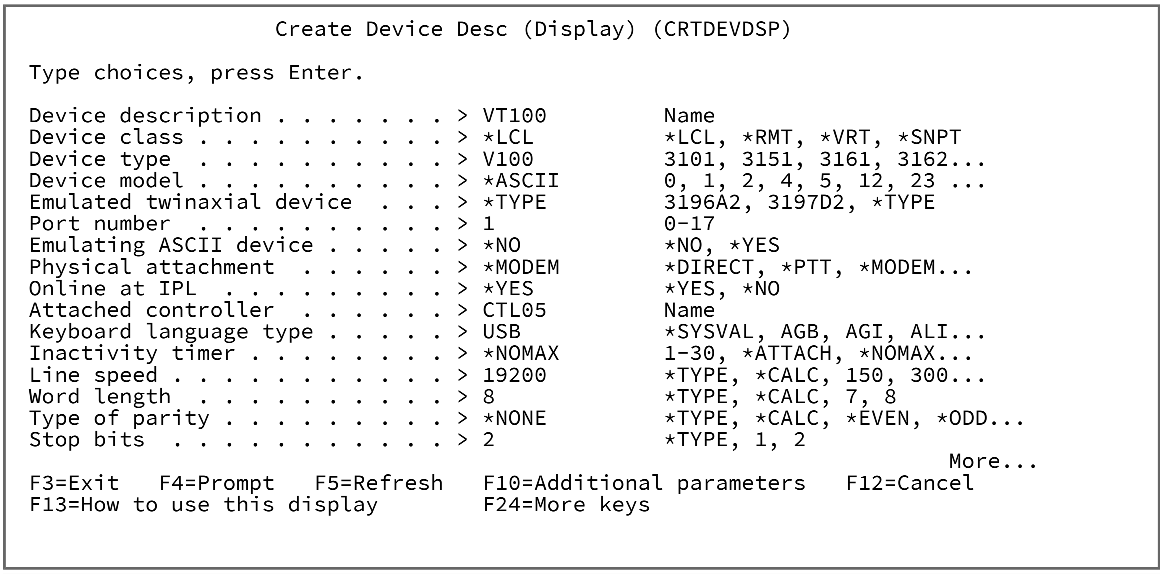
iSeries, eServer i5, or System i5 VT Asynchronous Attachment Example

The following sample is a typical configuration on the iSeries™, eServer™ i5, or System i5™ for a VT asynchronous dialed connection through an ASCII Workstation Controller. For more details on iSeries™, eServer™ i5, or System i5™ for VT asynchronous attachments, refer to *AS/400 ASCII Workstation Reference*.

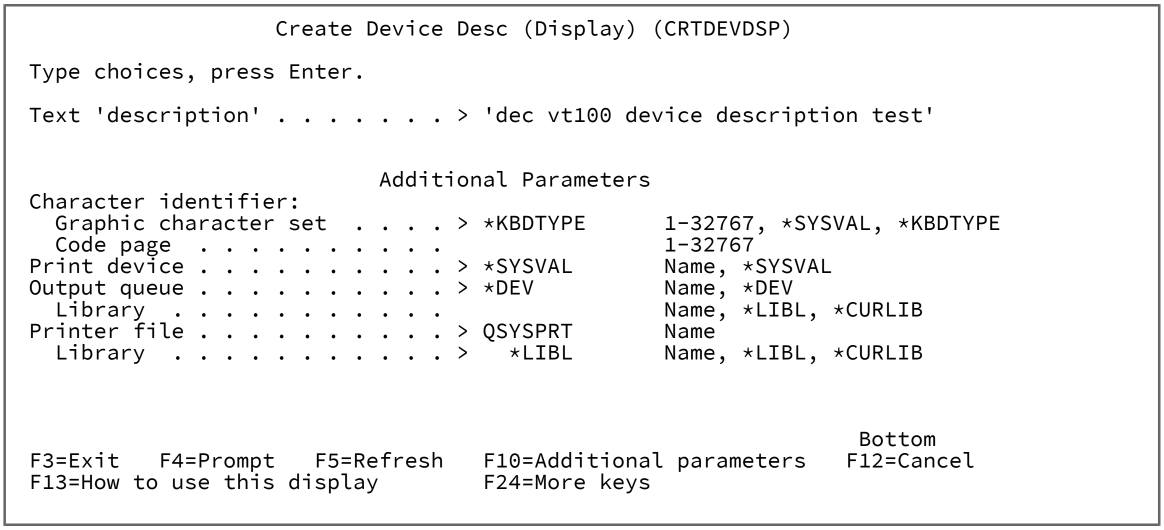
1. Enter the following command on the command line of the iSeries™, eServer™ i5, or System i5™ main menu:

```
CRTDEVDSP
```

The Create Device Description panel appears, in a first and second screen, as shown for the configuration of a VT100 device in the following screens. When filling in the panel, make sure that the name of the attached controller (arbitrarily shown as CTL05 here) matches the name of the twinaxial controller configured on your iSeries™, eServer™ i5, or System i5™.

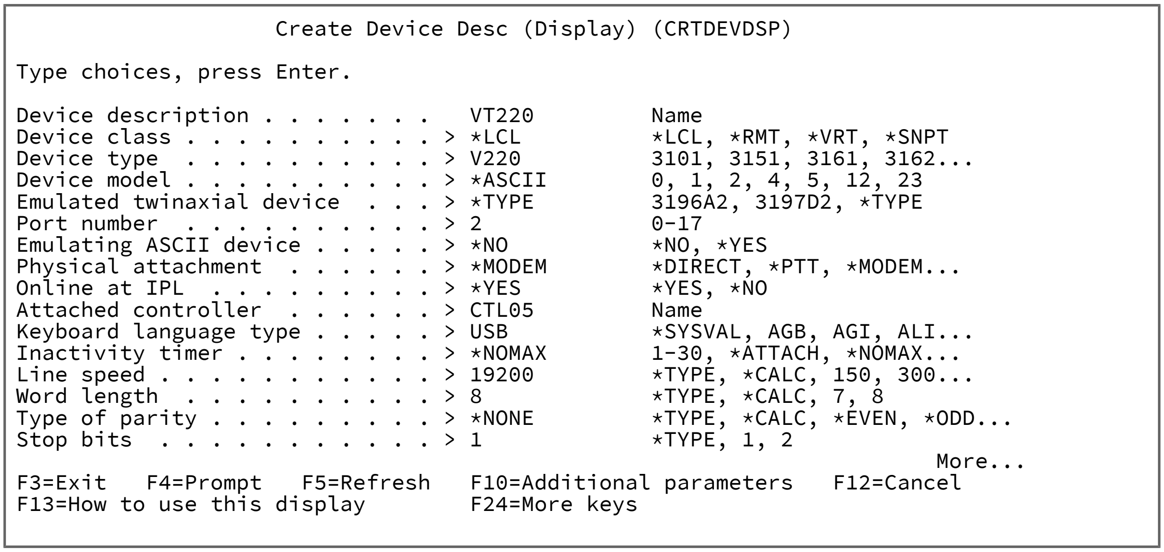


```
                   Create Device Desc (Display) (CRTDEVDSP)

Type choices, press Enter.

Device description . . . . . . . > VT100         Name
Device class . . . . . . . . . . > *LCL          *LCL, *RMT, *VRT, *SNPT
Device type  . . . . . . . . . . > V100          3101, 3151, 3161, 3162...
Device model . . . . . . . . . . > *ASCII        0, 1, 2, 4, 5, 12, 23 ...
Emulated twinaxial device  . . . > *TYPE         3196A2, 3197D2, *TYPE
Port number  . . . . . . . . . . > 1             0-17
Emulating ASCII device . . . . . > *NO           *NO, *YES
Physical attachment  . . . . . . > *MODEM        *DIRECT, *PTT, *MODEM...
Online at IPL  . . . . . . . . . > *YES          *YES, *NO
Attached controller  . . . . . . > CTL05         Name
Keyboard language type . . . . . > USB           *SYSVAL, AGB, AGI, ALI...
Inactivity timer . . . . . . . . > *NOMAX        1-30, *ATTACH, *NOMAX...
Line speed . . . . . . . . . . . > 19200         *TYPE, *CALC, 150, 300...
Word length  . . . . . . . . . . > 8             *TYPE, *CALC, 7, 8
Type of parity . . . . . . . . . > *NONE         *TYPE, *CALC, *EVEN, *ODD...
Stop bits  . . . . . . . . . . . > 2             *TYPE, 1, 2
                                                                   More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

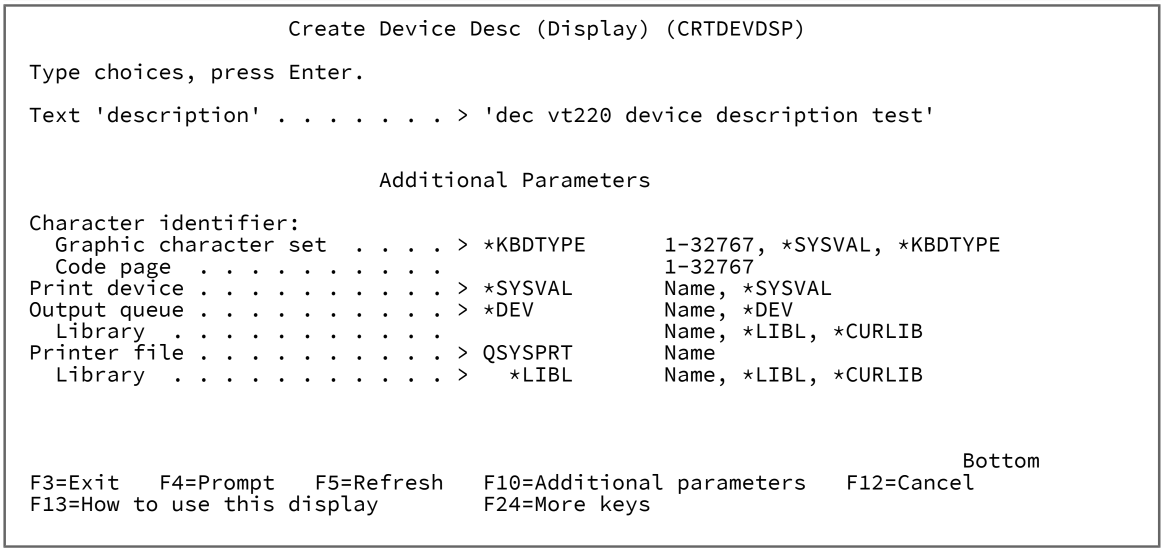


```
                   Create Device Desc (Display) (CRTDEVDSP)

Type choices, press Enter.

Text 'description' . . . . . . . > 'dec vt100 device description test'


                           Additional Parameters
Character identifier:
  Graphic character set  . . . . > *KBDTYPE      1-32767, *SYSVAL, *KBDTYPE
  Code page  . . . . . . . . . .                 1-32767
Print device . . . . . . . . . . > *SYSVAL       Name, *SYSVAL
Output queue . . . . . . . . . . > *DEV          Name, *DEV
  Library  . . . . . . . . . . .                 Name, *LIBL, *CURLIB
Printer file . . . . . . . . . . > QSYSPRT       Name
  Library  . . . . . . . . . . . >   *LIBL       Name, *LIBL, *CURLIB



                                                                Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

2. For a similar configuration for a VT220:
Use the same command, CRTDEVDSP, on the command line of the iSeries™, eServer™ i5, or System i5™ main menu. Again, the same kind of panels appear. You must make sure that the name of the attached controller matches the name of the twinaxial controller configured on your iSeries™, eServer™ i5, or System i5™.



```
                    Create Device Desc (Display) (CRTDEVDSP)

Type choices, press Enter.

Device description . . . . . . .   VT220         Name
Device class . . . . . . . . . . > *LCL          *LCL, *RMT, *VRT, *SNPT
Device type  . . . . . . . . . . > V220          3101, 3151, 3161, 3162...
Device model . . . . . . . . . . > *ASCII        0, 1, 2, 4, 5, 12, 23
Emulated twinaxial device  . . . > *TYPE         3196A2, 3197D2, *TYPE
Port number  . . . . . . . . . . > 2             0-17
Emulating ASCII device . . . . . > *NO           *NO, *YES
Physical attachment  . . . . . . > *MODEM        *DIRECT, *PTT, *MODEM...
Online at IPL  . . . . . . . . . > *YES          *YES, *NO
Attached controller  . . . . . . > CTL05         Name
Keyboard language type . . . . . > USB           *SYSVAL, AGB, AGI, ALI...
Inactivity timer . . . . . . . . > *NOMAX        1-30, *ATTACH, *NOMAX...
Line speed . . . . . . . . . . . > 19200         *TYPE, *CALC, 150, 300...
Word length  . . . . . . . . . . > 8             *TYPE, *CALC, 7, 8
Type of parity . . . . . . . . . > *NONE         *TYPE, *CALC, *EVEN, *ODD...
Stop bits  . . . . . . . . . . . > 1             *TYPE, 1, 2
                                                                      More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```



```
                    Create Device Desc (Display) (CRTDEVDSP)

Type choices, press Enter.

Text 'description' . . . . . . . > 'dec vt220 device description test'


                           Additional Parameters

Character identifier:
  Graphic character set  . . . . > *KBDTYPE      1-32767, *SYSVAL, *KBDTYPE
  Code page  . . . . . . . . . .                 1-32767
Print device . . . . . . . . . . > *SYSVAL       Name, *SYSVAL
Output queue . . . . . . . . . . > *DEV          Name, *DEV
  Library  . . . . . . . . . . .                 Name, *LIBL, *CURLIB
Printer file . . . . . . . . . . > QSYSPRT       Name
  Library  . . . . . . . . . . . >   *LIBL       Name, *LIBL, *CURLIB



                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Alerts

Alerts are generated by components of Z and I Emulator for Windows; corresponding messages are logged in the message log and can be viewed with the Z and I Emulator for Windows log viewer utility. Refer to the information about log viewer functions in the User's Reference for the specific emulator type for more information.

Note that some alerts can be caused by different situations. Each situation may generate a different message. Other alerts are informational and do not generate specific messages in the log, although messages about problems relating to the situation that generated the alert may be logged.

Alert ID number	Messages logged
APPN	
X'034A6F0B'	PCS4066E PCS4068E

```
                    PCS4093E
X'0DF28A14'         PCS4065E
                    PCS4066E
                    PCS4068E
                    PCS4069E
                    PCS4070E
X'170F7710'         No specific message
X'21745F28'         No specific message
X'2313A399'         PCS4364A
                    PCS4365A
                    PCS4379A
X'32CDF4E2'         PCS4073E
X'47302521'         No specific message
X'6D27D125'         PCS4066E
                    PCS4068E
X'7599A7D8'         No specific message
X'769022F0'         PCS4504E
X'9DCD7CCA'         PCS4275E
                    PCS4280E
                    PCS4282E
                    PCS4283A
                    PCS4284E
                    PCS4304E
                    PCS4305E
                    PCS4310A
                    PCS4311E
                    PCS4312E
X'9E452D9C'         PCS4593A
X'A89646AA'         PCS4275E
                    PCS4280E
                    PCS4282E
                    PCS4283A
                    PCS4284E
                    PCS4304E
                    PCS4305E
                    PCS4310A
                    PCS4311E
                    PCS4312E
X'B558D310'         PCS4324E
                    PCS4342E
                    PCS4347E
X'C781E91E'         No specific message
X'EBAA3C4F'         PCS4593A
X'EBEE390E'         PCS4063E
                    PCS4064E
                    PCS4066E
                    PCS4067E
                    PCS4068E
                    PCS4071A
                    PCS4091E
                    PCS4092E
                    PCS4094E
                    PCS4123E
                    PCS4124E
                    PCS4125E
X'F52A0C01'         PCS4061E
                    PCS4062E
X'FE1C42EB'         No specific message
```

```
LLC2 SAP

  X'016E5F4E'         PCS1066A
                      PCS1054A
  X'3BA03B6D'         PCS1066A
                      PCS1005E
  X'55BF3E1C'         PCS1066A
                      PCS1054A
  X'A676B230'         PCS1066A
                      PCS1005E
  X'CAF3C58A'         PCS1066A
                      PCS1054A
  X'D2E24978'         PCS1066A
                      PCS1005E
  X'D615A61E'         PCS1066A
                      PCS1054A
  X'EB1D6ABB'         PCS1066A
                      PCS1005E
  X'EB61E14F'         PCS1066A
                      PCS1005E

LLC2 Link Station

  X'216D1033'         PCS1065A
                      PCS1003E
  X'25AC0D84'         PCS1065A
                      PCS1004E
  X'28EF2B5D'         PCS1065A
                      PCS1001E
                      PCS1004E
                      PCS1006E
  X'5B8F5BA7'         PCS1065A
                      PCS1050A
  X'83D91642'         PCS1065A
                      PCS1000E
  X'87180BF5'         PCS1065A
                      PCS1000E
  X'8A5B2D2C'         PCS1065A
                      PCS1000E
  X'8E9A309B'         PCS1065A
                      PCS1000E
  X'E65B0B7F'         PCS1065A

pDLC

  Alert ID number     Messages logged      Alert type   Alert description

  X'0E499026'         PCS8607              01           3300
  X'0F935B3E'         PCS8603              01           3300
  X'21C346F0'         PCS8619              01           3300
  X'25025B47'         PCS8620              01           3300
  X'28417D9E'         PCS8617              01           3300
  X'2C806029'         PCS8618              01           3300
  X'4227687B'         PCS8610              01           3300
  X'6C6E2505'         PCS8604              01           8000
                      PCS8612
  X'7EA9C871'         PCS8608              01           3300
  X'8CEC6B74'         PCS8609              01           3300
  X'AB218ADF'         PCS8700              01          3300
  X'BB5C288E'         PCS8600              01          3300
  X'C16E9922'         PCS8615              01          3300
  X'C5AF8495'         PCS8616              01          3300
  X'C8ECA24C'         PCS8613              01          3300
  X'CC2DBFFB'         PCS8614              01          3300
  X'D3F9C6D8'         PCS8611              01          3300
  X'EBB67B65'         PCS8606              01          3300
```

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

> HCL
> 330 Potrero Ave.
> Sunnyvale, CA 94085
> USA
> Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Index

1012
1018
1012
1014
1017
1015
1018
1017
1016
1013
1015
1014
1013
1015
1016
1014
1019
1011
1019
1017
1013

1126
1135
1135
1136
1136
1127
1134
1132
1133
1132
1128
1130
1128
1129
1131
1129
1131
1130
1131
1134
1133

1137
1146
1153
1152
1147
1148
1149
1150
1151
1138
1145
1144
1145
1143
1143
1143
1139
1139
1140
1140
1142
1141
1142
1146
1144

1029

1067
1066
1065
1067
1065
1064
1066
1065
1063
1066
1063

1044
1043
1046
1034
1045
1036
1036
1036
1041
1041
1039
1038
1040
1044
1050
1048
1049
1038
1037
1037

1033
1030
1032
1033
1032
1031
1031
1032
1031
1030
1030
1029
1034
1033

1063
1050
1058
1051
1061
1053
1055
1059
1052
1062
1054
1056

1069

1069
1070
1071
1072
1073
1074
1075
1076
1069

1077

1082
1081
1080
1081
1080

1077
1078
1077
1079
1079
1080

1083

1095
1095
1094
1095
1094

1089
1090
1089
1092
1091
1093
1093
1091
1090

1086
1087
1088
1088
1087
1088
1087
1085
1086
1084
1084
1084
1083
1086
1085

1096

1111
1111
1110
1111
1110

1107, 1107
1105
1105
1107, 1107
1106, 1106
1108, 1108
1109
1109
1105

1100
1104
1103
1103
1102
1102

962
967
966
961
963
968
959
964
965
962
967
966
957
957

969
969
970
970
973
981
975
977
979
974
983
971
983
976
978
980
981
969
969

840
840

868
867
864
865
862
843
844
848
850
850
871
845
845
844
856
858
846
848
847
840
863
869
883
854
852
851
860
884
870
884
840

885
886

888
889
889
886
887
889
888
887
886

890
890

892
891
892
891

893
893

895
894
895
894

896
896

897
897
899
897
898
896

899
899

901
902
903
904
905
905
906
908
900
901
900
908

914
914

915
916
917
920
921
916
919
918
914
922
922

922
923

926
925
926

925

927

927
927

944
928
929
938
936
940
930
932
934
947
946
949
943
928
945
939
948
946
950
943
937
941
931
933
935

950
950

951
952
951
955
953

583
203

81
77
81

78
78
81
583

536
537
513

513
512
513
513
512
515
744
742
742
744
745
744
743

Emulator Programming

About This Book

This book provides necessary programming information for you to use the HCL Z and I Emulator for Windows Emulator High-Level Language Application Program Interface (EHLLAPI), and Z and I Emulator for Windows Session API (PCSAPI), and. The Host Access Class Library is described in *Host Access Class Library*.

EHLLAPI/PCSAPI is used with Z and I Emulator for Windows to provide a way for users and programmers to access the host presentation space with a set of functions that can be called from an application program running in a workstation session.

In this book, *Windows* refers to Windows® 7, Windows® 8/8.1, Windows® 10, Windows® Server 2008, and Windows® Server 2012. When information is relevant only to a specific operating system, this will be indicated in the text.

Who Should Read This Book

This book is intended for programmers who write application programs that use the APIs documented in this book.

A working knowledge of Windows® is assumed. For information about Windows®, refer to the list of publications under Where To Find More Information on page 508.

The programmer must also be familiar with connecting to a host system from a terminal or from a workstation with terminal emulation software.

This book assumes you are familiar with the language and the compiler that you are using. For information on how to write, compile, or link-edit programs, refer to Where To Find More Information on page 508 for the appropriate references for the specific language you are using.

Where To Find More Information

The Z and I Emulator for Windows library includes the following publications:

- *Installation Guide*
- *Quick Beginnings*
- *Emulator User's Reference*
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Client/Server Communications Programming*
- *System Management Programming*
- *Host Access Class Library*
- *Configuration File Reference*

In addition to the printed books, there are Hypertext Markup Language (HTML) documents provided with Z and I Emulator for Windows:

Host Access Class Library

The HACL Java HTML files describe how to write an ActiveX/OLE 2.0-compliant application to use Z and I Emulator for Windows as an embedded object. These files can be accessed from the Docs_Admin_Aids zipped folder delivered along with Z and I Emulator for Windows product documentation in the following path : *ZIEWin_3.0_Docs_Admin_Aids.zip\publications\en_US\doc\hacl*

Following is a list of related publications:

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

Notation

A table at the beginning of each section explains API functions in EHLLAPI Functions on page 536, PCSAPI Functions on page 690, and WinHLLAPI Extension Functions on page 678. It shows whether a function is supported for the products that provide the function described in the section. Yes means it is supported for a host type, and No means not supported. For example, the following table indicates that a function is available for 3270 and VT sessions but not for 5250 sessions.

3270	*5250*	*VT*
Yes	No	Yes

Introduction to Emulator APIs

The IBM® Z and I Emulator for Windows product supplies several application programming interfaces (APIs). Each interface has a specific set of functions and may be used for different purposes. Choose the programming interface that best matches the functional requirements of your application. Some applications may use more than one interface to achieve the desired results. The programming interfaces are:

- **Emulator High Level Language API (EHLLAPI)**: This interface provides functions to access emulator "presentation space" data such as characters on the host screen. It also provides functions for sending keystrokes to the host, intercepting user-entered keystrokes, querying the status of the host session, uploading and downloading files, and other functions. This interface is often used for *automated operator* applications which read host screens and enter keystrokes without direct user intervention. See EHLLAPI Functions on page 536.
  - **IBM® Standard HLLAPI Support**: This is a standard programming interface which allows programmatic access to a host emulator session. See Introduction to IBM Standard EHLLAPI, IBM Enhanced EHLLAPI and WinHLLAPI Programming on page 512.
  - **IBM® Enhanced HLLAPI Support**: This interface is based on the IBM® Standard HLLAPI interface. It provides all of the existing functionality but uses modified data structures. See Introduction to IBM Standard EHLLAPI, IBM Enhanced EHLLAPI and WinHLLAPI Programming on page 512.

    - **Windows® High Level Language API (WinHLLAPI)**: This interface provides much of the same functionality of IBM® Standard EHLLAPI and adds some extensions that take advantage of the Windows® environment. See Introduction to IBM Standard EHLLAPI, IBM Enhanced EHLLAPI and WinHLLAPI Programming on page 512.
- Any 32-bit APIs which accept\return Window Handles and pointers might not work correctly with HCL ZIEWin due to difference in pointer\handle sizes between x86 and x64 platforms.

  For Example:

  "Data String" parameter returned in byte numbers (9-12) in API Start Communication Notification (80) might be truncated on x64 platform.
- **Z and I Emulator for Windows Session API (PCSAPI)**: This interface is used to start, stop, and control emulator sessions and settings. See PCSAPI Functions on page 690.

  For Z and I Emulator for Windows Version 3.0, functions have been added to allow control and retrieval of page and printer settings. See Page Setup Functions on page 699 and Printer Setup Functions on page 708.
- **HCL Z and I Emulator for Windows Host Access Class Library (ECL)**: ECL is a set of objects that allow application programmers and scripting language writers to access host applications easily and quickly. Z and I Emulator for Windows supports three different ECL layers (C++ objects, ActiveAutomation (OLE), and LotusScript Extension (LSX)). Refer to *Host Access Class Library (HACL)* for more details.

Using API Header Files

The application program should include operating system header files before including API header files. For example:

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

Critical Sections

Use critical sections (**EnterCriticalSection** function) carefully when your program calls emulator APIs. Do not make emulator API calls within a critical section. If one thread of an application establishes a critical section and another thread is within an emulator API call, the call is suspended until you exit from the critical section.

During processing of an API call, all signals (except numeric coprocessor signals) are delayed until the call completes or until the call needs to wait for incoming data. Also, **TerminateProcess** issued from another process is held until the application completes an API call it might be processing.

Stack Size

Emulator APIs use the calling program's stack when they are executed. The operating system, the application, and the API all require stack space for dynamic variables and function parameters. At least 8196 bytes (8K) of stack space should be available at the time of an API call. It is the responsibility of the application program to ensure sufficient stack space is available for the API.

Windows x64 Platform Support

The x64-based versions of Microsoft® Windows® Server 2008 and Microsoft® Windows® 8/8.1/10 x64 Edition are optimized to run native 64-bit programs, but do not support 32-bit drivers or 16-bit applications.

For these platforms, Z and I Emulator for Windows does not install the following libraries.

- 16-bit API support:
  - Standard EHLLAPI 16-bit interface
  - WinHLLAPI 16-bit interface
  - PCSAPI 16-bit interface

Sample Programs

Several sample programs are provided, each of which illustrates the use of one of the Z and I Emulator for Windows APIs. If you choose to install the sample programs, they will be installed in the \SAMPLES directory.

Note: International Business Machines Corporation provides these files as is, without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

The sample program files include source and supporting files for the following Z and I Emulator for Windows APIs:

- Emulator High-Level Language Programming Interface (EHLLAPI)
- PCSAPI Functions

The following files are installed in the \SAMPLES directory.

Table 65. Sample Program Subdirectories

File Name	Description
DDE_C.H	DDE include file
EHLAPI32.H	IBM® standard 32-bit EHLLAPI include file
WHLLAPI.H	WinHLLAPI 16-bit include file
HAPI_C.H	EHLLAPI include file
PCSAPI.H	PCSAPI include file
PCSCALLS.LIB	Import library for standard interface
PCSCAL32.LIB	Import library for enhanced interface
EHLAPI32.LIB	Import library for IBM® Standard 32-bit EHLLAPI interface
WHLLAPI.LIB	Import library for WinHLLAPI 16-bit interface

Table 65. Sample Program Subdirectories (continued)

File Name	Description
WHLAPI32.LIB	Import library for WinHLLAPI 32-bit interface

The following subdirectories are created in the \SAMPLES directory.

Table 66. Sample Program Subdirectories

File Name	Description
ECL	The sample\ecl\cpp folder contains all files related HACL CPP sample. The sample\ecl\vb folder contains all files related to HACL VB.Net sample.
HLLSMP	Shows how to use EHLLAPI to request a keystroke and log on to a VM system.(X86). It supports the logon, pastetext and sendkey functionalities. Refer to the hllsmp\Readme.txt for the details on using the above-mentioned functionalities with hllsmp.exe.(X64).

Introduction to IBM Standard EHLLAPI, IBM Enhanced EHLLAPI and WinHLLAPI Programming

This chapter provides information needed to incorporate IBM® Standard EHLLAPI (16- and 32-bit), WinHLLAPI (16- and 32-bit), and IBM® Enhanced 32-bit EHLLAPI (EHLAPI32) functions into applications written in a high level language. It provides details on call format, memory allocation considerations, initializing the interfaces, and compiling and linking applications. Also included is a short sample EHLLAPI program and the compile/link instructions used to build it. Finally, a set of possible uses for the EHLLAPI interface (scenarios) is described.

An EHLLAPI application is any application program which uses the EHLLAPI interface to access the host 3270/5250/VT presentation space. The presentation space includes the visible emulator character data, fields and attribute data, keystroke data, and other information.

EHLLAPI Overviews

Following are overviews for HLLAPI programming interfaces.

IBM Standard EHLLAPI

EHLLAPI is a standard programming interface which allows programmatic access to a host emulator session. Functions are provided for reading host screen data (such as the characters and attributes), for sending keystrokes, and performing other emulator-related functions.

The EHLLAPI interface is a single call-point interface. There is a single callable API through which all EHLLAPI functions are requested. On each call to the interface the application provides a function number which identifies the function requested, a pointer to a data buffer, a pointer to the length of the data buffer, and a pointer to a return code (see EHLLAPI Call Format on page 513).

WinHLLAPI

WinHLLAPI is based on the familiar EHLLAPI.API. It encompasses all of the existing functionality and adds extensions that take advantage of the Windows® message driven environment. Users of the HCL Z and I Emulator for Windows EHLLAPI interface will notice no functional difference unless they incorporate the WinHLLAPI extensions.

The WinHLLAPI extension functions and any functions that deviate from the EHLLAPI form are described in WinHLLAPI Extension Functions on page 678. For information on common functions, refer to EHLLAPI Functions on page 536.

WinHLLAPI and IBM® Standard EHLLAPI

The entry symbol for WinHLLAPI, is appropriately, **WinHLLAPI**. EHLLAPI users wishing to switch to the WinHLLAPI implementation must change from the **hllapi** standard entry. New users should follow all of the directions in EHLLAPI Functions on page 536, and use the **WinHLLAPI** entry in place of the standard **hllapi** entry.

IBM Enhanced EHLLAPI and IBM Standard EHLLAPI

IBM Enhanced EHLLAPI is based on the familiar EHLLAPI API. It encompasses all of the existing functionality but takes advantage of the 32-bit environment and uses modified data structures. Standard interface users wishing to switch to IBM® Enhanced 32-bit EHLLAPI need to change only the entry symbol from LPWORD to LPINT in the first, third, and fourth parameters. New users should use the procedures in the following sections.

Languages

Any programming language which can invoke an entry point in a DLL with the "Pascal" calling convention can be used to execute EHLLAPI functions. However, the Z and I Emulator for Windows EHLLAPI toolkit provides header files and function prototypes only for the C++ languages. A clear understanding of data structure layout and calling conventions is required to use any other language. The EHLLAPI toolkit supports the following C/C++ compilers:

- Microsoft® Visual C/C++ Version 4.0 and higher

Most other C/C++ compilers will also work with the toolkit.
EHLLAPI C/C++ applications must include the Z and I Emulator for Windows EHLLAPI header file (HAPI_C.H). This file defines the layout of data structures and provides a prototype for the EHLLAPI entry point.

Note: The data structure layout for 16- and 32-bit applications are not the same (see Standard and Enhanced Interface Considerations on page 530).

EHLLAPI Call Format

The EHLLAPI entry point (**hllapi**) is always called with the following four parameters:

1. EHLLAPI Function Number (input)
2. Data Buffer (input/output)
3. Buffer Length (input/output)
4. Position (input); Return Code (output)

The prototype for IBM® Standard EHLLAPI is:

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

The prototype for IBM® Enhanced EHLLAPI is:

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

Each parameter is passed by *reference* not by value. Thus each parameter to the function call must be a *pointer* to the value, not the value itself. For example, the following is a correct example of calling the EHLLAPI Query Session Status function:

```
    #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
    }
```

All the parameters in the **hllapi** call are pointers and the return code of the EHLLAPI function is returned in the value of the 4th parameter, not as the value of the function. For example, the following is **not** correct:

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

Although the **hllapi** function is defined to return a **long** data type for IBM® Standard and Enhanced EHLLAPI, and **void** data type for WinHLLAPI, its value is undefined and should not be used.

The second through fourth parameters of the **hllapi** call can return information to the application. The description of each EHLLAPI function describes what, if any, information is returned in these parameters.

Data Structures

Many EHLLAPI functions use a formatted data structure to pass information to or from the application program. The description of each function shows the layout of the data structure. The data passed to or from the EHLLAPI function must exist in storage exactly as documented, byte for byte. Note that the structure layout is the same for all IBM® Standard and WinHLLAPI 16- and 32-bit applications. Data structures for the IBM® Enhanced 32-bit applications are packed to a 4-byte alignment.

It is *highly recommended* that the supplied header file and data structure definitions be used to ensure proper data alignment and layout. Although it is technically possible, the following is *not* recommended:

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

The recommended way to write this function would be:

```
#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
}
```

Memory Allocation

EHLLAPI functions do not allocate or free memory. The application program must preallocate buffer space for EHLLAPI functions which require it before calling the **hllapi** entry point. The buffer space may be pre-allocated as a dynamic variable such as:

```
struct HLDQuerySessionStatus QueryBuff;
```

or it may be allocated by a call to a C library or operating system function such as:

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

In any case, the application is responsible for allocating sufficient buffer space before calling EHLLAPI functions and for freeing buffers when they are not needed.

EHLLAPI Return Codes

EHLLAPI functions return a completion code or return codein the 4th parameter of the **hllapi** function call (except for the **Convert Position** or **RowCol** (99) function). The return code indicates the success or failure of the requested function.

Unless indicated otherwise in the description of each function, the following table shows the meaning of each return code value. Some functions may have a slightly different interpretation of these return codes; refer to the individual function descriptions for details.

Table 67. EHLLAPI Return Codes

Return Code	Explanation
0	The function successfully executed, or no update since the last call was issued.
1	An incorrect host presentation space ID was specified. The specified session either was not connected, does not exist, or is a logical printer session.
2	A parameter error was encountered, or an incorrect function number was specified. (Refer to the individual function for details.)
4	The execution of the function was inhibited because the target presentation space was busy, in X CLOCK state (X []), or in X SYSTEM state.
5	The execution of the function was inhibited for some reason other than those stated in return code 4.
6	A data error was encountered due to specification of an incorrect parameter (for example, a length error causing truncation).
7	The specified presentation space position was not valid.
8	A functional procedure error was encountered (for example, use of conflicting functions or missing prerequisite functions).
9	A system error was encountered.
10	This function is not available for EHLLAPI.
11	This resource is not available.
12	This session stopped.
24	The string was not found, or the presentation space is unformatted.
25	Keystrokes were not available on input queue.
26	A host event occurred. See **Query Host Update** (24) for details.
27	File transfer was ended by a Ctrl+Break command.
28	Field length was 0.
31	Keystroke queue overflow. Keystrokes were lost.
32	An application has already connected to this session for communications.
33	Reserved.
34	The message sent to the host was canceled.
35	The message sent from the host was canceled.
36	Contact with the host was lost.
37	Inbound communication has been disabled.
38	The requested function has not completed its execution.
39	Another DDM session is already connected.
40	The disconnection attempt was successful, but there were asynchronous requests that had not been completed at the time of the disconnection.
41	The buffer you requested is being used by another application.
42	There are no outstanding requests that match.
43	The API was already locked by another EHLLAPI application (on LOCK) or API not locked (on UNLOCK).

Compiling and Linking

The application program can be linked using dynamic linking method. This means that we can link the entry point by performing the dynamic linking. In this case, the application uses operating system calls to load the correct DLL and obtain the entry point address at run time.

The following table shows which .DLL should be used for dynamic loading.

Interface	Entry Point	DLL
IBM® Standard (64-bit)	hllapi	EHLAPI32.DLL
IBM® Enhanced (64-bit)	hllapi	PCSHLL32.DLL
WinHLLAPI (64-bit)	winhllapi	WHLAPI32.DLL

Dynamic Link Method

Using the dynamic link method the application makes calls to the operating system at run time to load the Z and I Emulator for Windows EHLLAPI module and to locate the **hllapi** entry point within it. This method requires more code in the application but gives the application greater control over error conditions. For example, the application can display a specific error message to the user if the Z and I Emulator for Windows EHLLAPI module cannot be found.

To use dynamic linking, the application needs to load the appropriate Z and I Emulator for Windows module and locate the entry point. It is recommended that the entry point be located by its ordinal number and not by name. The ordinal number is defined in the header file. The following 32-bit Windows® code loads the IBM® Standard 32-bit EHLLAPI module, locates the **hllapi** entry point, and makes an EHLLAPI function call.

```
#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
}
```

Multithreading

HCL Enhanced EHLLAPI (32-bit) and HCL® Standard EHLLAPI 16-bit connect on a per process basis. All threads access the same connected host session. The thread that performs the connections must also perform the disconnection.

HCL® Standard EHLLAPI (32-bit) and WinHLLAPI connect on a per thread basis. Each thread must maintain its own connections. This allows a multithreaded process to maintain connections to more than one connected host session at a time. This eliminates the need for multi-process schemes when using a WinHLLAPI program to coordinate data between different hosts. It also puts the burden of connecting and disconnecting as necessary on the individual thread.

Presentation Spaces

Many EHLLAPI functions require a *presentation space ID (PSID)* to indicate which host emulator session is to be used for the function. (This is also referred to as the *short session ID*). A presentation space ID is a single character in the range A to Z. There are a maximum of 26 sessions.

IBM® Enhanced 32-Bit Interface Presentation Space IDs

For IBM® Enhanced EHLLAPI applications, the session ID is extended with three additional bytes. These extended session bytes must be set to zero for future compatibility. This is most easily accomplished by setting the contents of EHLLAPI buffers to all binary zero before filling them in with the required information. For example, the following might be used to query the status of 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 of Presentation Spaces

An emulator session can be configured as a display session or a printer session. EHLLAPI applications cannot connect to printer or router sessions of PC400. The **Query Sessions (10)** function can be used to determine the type of a particular session.

Size of Presentation Spaces

An emulator display session can be configured for a range of screen sizes from 1920 bytes (24x80 screen size) to 9920 bytes (62x160 screen size). Some EHLLAPI functions such as **Copy PS to String (8)** require the application to allocate enough storage to hold (possibly) the entire presentation space. The size of the presentation space for a given session can be obtained using the **Query Session Status (22)** function.

Presentation Space IDs

EHLLAPI functions interact with only one presentation space at a time. The presentation space ID (PSID) is used to identify the particular presentation space in which a function is to operate.

For some functions, the PSID is contained in a preceding call to the **Connect Presentation Space** (1) function. For other functions, the PSID is contained in the calling data string parameter.

Host-Connected Presentation Space

Connection to the host presentation space (or session) is controlled by using the **Connect Presentation Space** (1) and **Disconnect Presentation Space** (2) functions. The status of the connection determines whether some functions can be executed. It also affects how the PSID is defined. The following text explains how to control the status of the connection to the host presentation space:

- At any given time, there can be either no host-connected presentation space, or there can be one and only one host-connected presentation space.
- There is no default host-connected presentation space.
- Following a connect, there is one and only one host-connected presentation space. The host presentation space that is connected is identified in the calling data string parameter of the connect function.
- A subsequent call to connect can be executed with no intervening disconnect. In this case, there is still one and only one host-connected presentation space. Again, the host presentation space that is connected is identified in the calling data string parameter of the connect function.
- Following a disconnect, there is no host-connected presentation space. This rule applies following multiple consecutive calls to connect or following a single call to connect.
- You cannot connect to a logical printer session.

Presentation Space ID Handling

The PSID is used to specify the host presentation space (or session) in which you desire a function to operate. The way the PSID is handled is affected by two factors:

1. The method used to specify the PSID:
    a. As the calling data string parameter of a preceding call to the **Connect Presentation Space** (1) function
    b. As a character in the calling data string of the function being executed. Handling varies depending on whether the character is:
        - A letter *A* through *Z*
        - A blank or a null
2. The status of the connection to the host presentation space.

The following paragraphs describe how the PSID is handled for the various combinations of these two factors.

PSID Handling for Functions Requiring Connect

Some functions interact only with the host-connected presentation space. These functions require the **Connect Presentation Space** (1) function as a prerequisite call. The PSID for these functions is determined by the **Connect Presentation Space** (1) and the **Disconnect Presentation Space** (2) functions as follows:

- When there is no host-connected presentation space, these functions do not interact with any presentation space. A return code of 1 is generated.
- When there is one host-connected presentation space, these functions interact with the presentation space specified in the calling data string parameter of the most recent call to the **Connect Presentation Space** (1) function.

PSID Handling for Functions Not Requiring Connect

Some functions can interact with a host presentation space whether it is connected or not. These functions allow you to specify the PSID in the calling data string parameter. They are as follows:

- **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)

All except the first two of these functions allow you to specify the PSID using either:

- A letter *A* through *Z*
- A blank or a null

The first two functions require that a letter be used to specify the PSID.

When there is no host-connected presentation space, the following rules apply:

- The function can interact with any host presentation space if a letter, not a blank or a null, is used to specify the PSID.
- If a blank or a null is used to specify the PSID, a return code of 1 is generated. The function does not execute.
- Using a letter to specify the PSID does not establish a host-connected presentation space, except on a connect PS request.

When there is one host-connected presentation space, the following rules apply:

- The function can interact with any host presentation space if a letter is used to specify the PSID.
- If a blank or a null is used to specify the PSID, the function operates in the presentation space identified in the most recent call to the **Connect Presentation Space** (1) function.
- Using a letter to specify the PSID does not change the established PSID of the host-connected presentation space, except on a connect PS request.

The following functions are available for printer sessions:

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

Sharing EHLLAPI Presentation Space between Processes

More than one EHLLAPI application can share a presentation space if the applications support sharing (that is, if they were developed to work together or if they exhibit predictable behavior[1]). To determine which applications support sharing, EHLLAPI applications are specified as one of following types:

[1] This means that two EHLLAPI programs will not be vying for the same Presentation Space at the same time; or that there is logic in those programs which will allow the program to wait until the PS is available; or that the applications never use the Session in a way which would lock out other applications.

- Supervisory
- Exclusive write with read privilege allowed
- Exclusive write without read privilege allowed
- Super write
- Read

The type of shared access can be defined by setting the following read and write sharing options for each function in the **Set Session Parameters** (9) function call:

SUPER_WRITE

The application allows other applications that allow sharing and have write access permissions to concurrently connect to the same presentation space. The originating application performs supervisory-type functions but does not create errors for other applications that share the presentation space.

WRITE_SUPER

The application requires write access and allows only supervisory applications to concurrently connect to its presentation space. This is the default value.

WRITE_WRITE

The application requires write access and allows partner or other applications with predictable behavior to share the presentation space.

WRITE_READ

The application requires write access and allows other applications that perform read-only functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.

WRITE_NONE

The application has exclusive use of the presentation space. No other applications are allowed to share the presentation space, including supervisory applications. The application is allowed to copy the presentation space and perform read-only operations as usual.

READ_WRITE

The application requires only read access to monitor the presentation space and allows other applications that perform read or write, or both, functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.

Note: Sharing presentation space is not available between threads in a process.

Table 68. EHLLAPI Read and Write Sharing Option Combinations

Calling Application	Super_Write	Write_Super	Write_Write	Write_Read	Write_None	Read_Write
Super_Write	Yes	Yes	Yes	No	No	Yes
Write_Super (default)	Yes	No	No	No	No	No
Write_Write	Yes	No	Yes	No	No	Yes
Write_Read	No	No	No	No	No	Yes
Write_None	No	No	No	No	No	No
Read_Write	Yes	No	Yes	Yes	No	Yes

In addition to specifying compatible read and write access options, applications that are designed to work together but cannot allow others to work in the same presentation space can optionally define a keyword, KEY$nnnnnnnn, in the **Set Session Parameters** (9) function call. This keyword allows only those applications that use the same keyword to share the presentation space.

Note:

1. The **Start Keystroke Intercept** (50) function is non-shareable. Only one application at a time can trap keystrokes.
2. The **Connect To Presentation Space** (1) and **Start Keystroke Intercept** (50) functions share common subsystem functions. Successful requests by an application to share either of these functions can affect the requests of these two functions by other applications. For example, if application A successfully requests a **Connect To Presentation Space** (1) with Write_Read access and KEY $abcdefgh as the keyword, a request by application B to **Connect To Presentation Space** (1) or **Start Keystroke Intercept** (50) is successful only if both applications have set compatible read and write options.

Table 69. Prerequisite Functions and Associated Dependent Functions

Prerequisite Call	Functions	Access
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

Table 69. Prerequisite Functions and Associated Dependent Functions (continued)

Prerequisite Call	Functions	Access
Connect Presentation Space (1)	Copy Field to String (34)	Read
	Copy OIA (13)	Read
	Copy Presentation Space (5)	Read
	Copy Presentation Space to String (8)	Read
	Copy Presentation Space to Clipboard (35)	Read
	Copy String to Field (33)	Write
	Copy String to Presentation Space (15)	Write
	Disconnect Presentation Space (2)	Write
	Find Field Length (32)	Read
	Find Field Position (31)	Read
	Query Cursor Location (7)	Read
	Query Field Attribute (14)	Read
	Paste Clipboard to Presentation Space (36)	Write
	Release (12)	Write
	Reserve (11)	Write
	Search Field (30)	Read
	Search Presentation Space (6)	Read
	Send key (3)	Read
	Set Cursor (40)	Write
	Start Playing Macro (110)	Write
	Wait (4)	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

Locking Presentation Space

An application, even if specified with shared presentation space, can obtain exclusive control of a presentation space by using the **Lock Presentation Space API** (60) or the **Lock Windows® Services API** (61) functions. Requests by the other applications to use a presentation space locked by these functions are queued and processed in first-in-first-out (FIFO) order when the originating application unlocks the presentation space.

If the application that locked the presentation space does not unlock it by using the same call with an **Unlock** option or **Reset System** (21) call, the lock is removed when the application terminates or the session stops.

Using mouse actions to select, copy, and paste text in the Presentation Space

The following mouse actions can be used in the Presentation Space.

- Select a word by double-clicking the left mouse button.
- Copy a selected word by clicking the right mouse button.
- Paste a copied word by double-clicking the mouse right button.

ASCII Mnemonics

Keystrokes originating at a host keyboard might have a corresponding ASCII value. The response of the **Get Key** (51) function to a keystroke depends on whether the key is defined and also on whether the key is defined as an ASCII value or an ASCII mnemonic.

The keyboard for one session might not be capable of producing some codes needed by the another session. ASCII mnemonics that represent these codes can be included in the data string parameter of the **Send Key** (3) function.

The capabilities of the **Send Key** (3) function and the **Get Key** (51) function allow sessions to exchange keystrokes that might not be represented by ASCII values or by an available key. A set of mnemonics that can be generated from a keyboard is provided. These mnemonics let you use ASCII characters to represent the special function keys of the workstation keyboard.

Mnemonics for unshifted keys consist of the escape character followed by an abbreviation. This is also true for the shift keys themselves, Upper shift, Alt, and Ctrl. Mnemonics for shifted keys consist of the mnemonic for the shift key followed by the mnemonic for the unshifted key. Hence the mnemonic for a shifted key is a 4-character sequence of escape character, abbreviation, escape character, abbreviation.

The default escape character is `@`. You can change the value of the escape character to any other character with the `ESC=c` option of the **Set Session Parameters** (9) function. The following text uses the default escape character, however.

Shift indicators that are not part of the ASCII character set are represented to the host application by 2-byte ASCII mnemonics as follows:

Upper shift	@S
Alt	@A
Ctrl	@r

Mnemonics for these shift indicators are never received separately by an application. Likewise, they are never sent separately by an application. Shift indicator mnemonics are always accompanied by a non-shift-indicator character or mnemonic.

The abbreviations used make the mnemonics for special keys easy to remember. An alphabetic key code has been used for the most common keys. For example, the Clear key is *C*; the Tab key is *T*, and so on. Please note that the uppercase and lowercase alphabetic characters are mnemonic abbreviations for different keys.

The following text describes the use of these functions.

General

All defined keys are represented by either:

- A 1-byte ASCII value that is part of the 256-element ASCII character set, or
- A 2-, 4-, or 6-byte ASCII mnemonic

To represent a key defined as an ASCII character, a 1-byte ASCII value that corresponds to that character is used.

To represent a key defined as a function, a 2-, 4-, or 6-byte ASCII mnemonic that corresponds to that function is used. For example, to represent the backtab key, `@B` is used. To represent PF1, `@1` is used. To represent Erase Input, `@A@F` is used. See the following lists:

`@B`	Left Tab	`@0`	Home	`@h`	PF17
`@C`	Clear	`@1`	PF1/F1	`@i`	PF18
`@D`	Delete	`@2`	PF2/F2	`@j`	PF19
`@E`	Enter	`@3`	PF3/F3	`@k`	PF20
`@F`	Erase EOF	`@4`	PF4/F4	`@l`	PF21
`@H`	Help (PC400)	`@5`	PF5/F5	`@m`	PF22
`@I`	Insert	`@6`	PF6/F6	`@n`	PF23
`@J`	Jump	`@7`	PF7/F7	`@o`	PF24
`@L`	Cursor Left	`@8`	PF8/F8	`@q`	End
`@N`	New Line	`@9`	PF9/F9	`@u`	Page UP (PC400)
`@O`	Space	`@a`	PF10/F10	`@v`	Page Down (PC400)
`@P`	Print	`@b`	PF11/F11	`@x`	PA1
`@R`	Reset	`@c`	PF12/F12	`@y`	PA2
`@T`	Right Tab	`@d`	PF13	`@z`	PA3
`@U`	Cursor Up	`@e`	PF14	`@@`	@ (at) symbol
`@V`	Cursor Down	`@f`	PF15	`@$`	Alternate Cursor
`@Z`	Cursor Right				

`@A@C`	Test (PC400)	`@A@e`	Pink (PC/3270)
`@A@D`	Word Delete	`@A@f`	Green (PC/3270)
`@A@E`	Field Exit	`@A@g`	Yellow (PC/3270)
`@A@F`	Erase Input	`@A@h`	Blue (PC/3270)
`@A@H`	System Request	`@A@i`	Turquoise (PC/3270)
`@A@I`	Insert Toggle	`@A@j`	White (PC/3270)
`@A@J`	Cursor Select	`@A@l`	Reset Host Color (PC/3270)
`@A@L`	Cursor Left Fast	`@A@t`	Print (Personal Computer)
`@A@Q`	Attention	`@A@u`	Rollup (PC400)
`@A@R`	Device Cancel	`@A@v`	Rolldown (PC400)
`@A@T`	Print Presentation Space	`@A@y`	Forward Word Tab
`@A@U`	Cursor Up Fast	`@A@z`	Backward Word Tab
`@A@V`	Cursor Down Fast	`@A@-`	Field - (PC400)
`@A@Z`	Cursor Right Fast	`@A@+`	Field + (PC400)
`@A@9`	Reverse Video	`@A@<`	Record Backspace (PC400)
`@A@b`	Underscore (PC/3270)	`@S@E`	Print Presentation Space on Host (PC400)
`@A@c`	Reset Reverse Video (PC/3270)	`@S@x`	Dup
`@A@d`	Red (PC/3270)	`@S@y`	Field Mark

Note:

1. The first `@` symbol in the first table represents the escape character. The first and second `@` symbol in the second table is the escape character. The `@` symbol is the default escape character. You can change the value of the escape character using the `ESC=c` option of the **Set Session Parameters** (9) function.

   If you change the escape character to `#`, the literal sequences used to represent the Backtab, Home, and Erase Input keys become `#B`, `#0`, and `#A#F`, respectively.

   Also, the literal sequence used to represent the `@` symbol becomes `#@`.
2. If you send the mnemonic for print screen (that is, either `@P` or `@A@T`), place it at the end of the calling data string.
3. If you send the mnemonic for device cancel (that is, `@A@R`), it is passed through with no error message; however, local copy is not stopped.

Get Key (51) Function

If the terminal operator types a key defined as an ASCII character, the host application receives a 1-byte ASCII value that corresponds to that character.

If the operator types a key defined as a function, the host application receives a 2-, 4-, or 6-byte ASCII mnemonic that corresponds to that function. For example, if the **Backtab** key is typed, `@B` is received. If **PF1** is pressed, `@1` is received. If **Erase Input** is pressed, `@A@F` is received.

If the operator types a defined shift key combination, the host application receives the ASCII character, or the 2-, 4-, or 6-byte ASCII mnemonic that corresponds to the defined character or function.

If the operator types an individual key that is not defined, the **Get Key** (51) function returns a return code of 20 and nothing is sent to the host application.

The **Get Key** (51) function prefixes all characters and mnemonics sent to the host application with two ASCII characters. The first ASCII character is the PSID of the host presentation space to which the keystrokes are sent. The other character is an *A*, *S*, or *M* for ASCII, special shift, or mnemonic, respectively. See Return Parameters on page 590.

Send Key (3) Function

To send an ASCII character to another session, include that character in the data string parameter of the **Send Key** (3) function.

To send a function key to another session, include the ASCII mnemonic for that function in the data string parameter of the **Send Key** (3) function.

If the **Send Key** (3) function sends an unrecognized mnemonic to the host session a return code rejecting the key might result.

Debugging

As an aid in debugging EHLLAPI applications, the Trace Facility of Z and I Emulator for Windows may be used. This facility will produce a log of all EHLLAPI calls, parameters, return values, and return codes. For more information on using the Trace Facility, refer to *Administrator's Guide and Reference*.

A Simple EHLLAPI Sample Program

The following sample Windows® application will enter the character string "Hello World!" in the first input field of host session '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;
}
```

The application could be built with the following command:

```
nmake /a all
```

Standard and Enhanced Interface Considerations

There is no functional difference between the standard and enhanced EHLLAPI interfaces on a given platform. However there are other important differences:

- The enhanced EHLLAPI interface extends the presentation space ID (PSID) from 1 byte to 4 bytes. Currently the additional bytes are not used, but your application should set them to binary zeros to ensure compatibility with future versions of enhanced EHLLAPI.
- The position (offset) of data elements in memory buffers passed to and from EHLLAPI functions are different. Data elements in enhanced EHLLAPI are aligned to double-word boundaries. Data elements in standard EHLLAPI are not aligned in any particular way. EHLLAPI applications should not be coded to set or retrieve data in the buffers by offset (byte) values. Instead, the supplied data structures in the HAPI_C.H file should be used to set and retrieve data elements. This will ensure that data is set and retrieved from the correct position for both 16- and 32-bit programs.

By prefilling EHLLAPI data buffers with binary zeros, and using the data structures supplied in HAPI_C.H, an application can be compiled for standard or enhanced operation without any source code changes. For example, the following section of code would work for standard EHLLAPI but would fail for enhanced EHLLAPI:

```
#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
...
```

The above example would fail if compiled as a enhanced EHLLAPI application because:

- The application does not set the extended session ID bytes to zero.
- The buffer length for this function is 20, not 18.
- The session type indicator is not at offset 9 in the data buffer, it is at offset 12.

The following is the same function written to work correctly if compiled for standard or enhanced operation. Changed lines are indicated with a >:

```
   #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
   ...
```

Host Automation Scenarios

The sample scenarios presented here provide conceptual information about activities that can be facilitated by using EHLLAPI. The scenarios deal with the duties your EHLLAPI programmed operator can perform in these areas:

- Host system operation, including:
  - Search function
  - Sending keystrokes
- Distributed processing, including:
  - Data extraction
  - File transfer
- Integrating interfaces

Scenario 1. A Search Function

There are four phases in a typical host system transaction:

1. Starting the transaction
2. Waiting for the host system to respond
3. Analyzing the response to see if it is the expected response
4. Extracting and using the data from the response

Your programmed operator can use a series of EHLLAPI functions to mimic these actions. After determining the correct starting point for the host system transaction, the programmed operator can call the **Search Presentation Space** (6) function to determine which keyword messages or prompting messages are on the display screen.

Next, the programmed operator can use the **Send Key** (3) function to type data into a host system session and enter a host system transaction. Then the programmed operator can:

- Use the **Wait** (4) function that waits for the X CLOCK, X [], or X SYSTEM condition to end (or returns a keyboard-locked condition if the terminal has locked up).

  If the keyboard is inhibited, your EHLLAPI program can call the **Copy OIA** (13) function to get more information about the error condition.
- Use the **Search Presentation Space** (6) function to look for an expected keyword to validate that the proper response had been received.
- Use the **Copy Presentation Space to String** (8) function (or any of several data access functions) to extract the desired data.

The **Search Presentation Space** (6) function is critical to simulate another task of the terminal operator. Some host systems do not stay locked in X CLOCK, X [], or X SYSTEM mode until they respond; instead, they quickly unlock the keyboard and allow the operator to stack other requests. In this environment, the terminal operator depends on some other visual prompt to know that the data has returned (perhaps a screen title or label). The **Search Presentation Space** (6) function allows your EHLLAPI program to search the presentation space while waiting. Also, while waiting for a response, calling the **Pause** (18) function allows other DOS sessions to share the central processing unit resource. The **Pause** (18) function has an option that allows your EHLLAPI program to wait for a host system update event to occur.

If no host system event occurs after a reasonable time-out period, your EHLLAPI program could call a customized error message such as:

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

In this environment, program revisions become very important considerations, because the programmed operator must be reprogrammed for even minor changes in the display messages.

For example, if a terminal operator expects the message:

```
Enter Part Number:
```

as a prompt, he or she will probably be able to respond properly to an application change that produces the message:

```
Enter Component Number:
```

However, because the programmed operator is looking for a literal keyword string, subtle changes in message syntax, even as trivial as uppercase versus lowercase, can make the program take a preprogrammed error action.

Scenario 2. Sending Keystrokes

There are several considerations that demand attention in designing programs that send keystrokes to the host system. In some application environments, issuing a command is as simple as typing a string and pressing Enter. Other applications involve more complex formatted screens in which data can be entered into any one of several fields. In this environment you must understand the keystrokes required to fill in the display screen.

The Tab key mnemonic (@T; see General on page 526 for a full list of mnemonics) can be used to skip between fields. When sending keystrokes to a field using the **Send Key** (3) function, you should be aware of the field lengths and contents. If you fill the fields completely and the next attribute byte is autoskip, your cursor will then be moved to the next field. If you then issued a tab, you would skip to yet another field.

Likewise, if your keystrokes do not completely fill the field, there might be data left from prior input. You should use the Erase End of Field (EOF) command to clear this residual data.

Scenario 3. Distributed Processing

Some applications fall into the category called *collaborative*. These applications provide a single end-user interface, but their processing is performed at two or more different physical locations.

An EHLLAPI application can interact with host system applications by intercepting the communication between the host system and the terminal user. The host system presentation space is the vehicle used to intercept this data. The local application can request to be notified each time the presentation space is updated or whenever an AID key is pressed by the operator.

This workstation application can then cooperate with a host system application in any of the following ways:

- On a field or presentation space basis using either the copy functions that address fields (**Copy String to Field** (33) function or **Copy Field to String** (34) function) or the functions that let you copy from and into presentation spaces (for example, **Copy String to Presentation Space** (15) function or **Copy Presentation Space to String** (8) function).
- On a keystroke basis, using the **Send Key** (3) function.
- On a file basis, for large blocks of data. You can have your application use the EHLLAPI file transfer capability (using **Send File** (90) function or **Receive File** (91) function) to transfer data or functions (such as load modules) and have it processed locally or remotely.

Scenario 4. File Transfer

In this scenario, assume that you want to automate a file transfer:

- You could begin by using the procedure discussed in the search scenario earlier to log on to a host system session.
- Instead of using one of the copy functions (which are inefficient for copying many screens of data), your EHLLAPI program could call file transfer functions **Send File** (90) and **Receive File** (91) to transfer data.
- Upon successful completion:
    - If the **Send File** (90) function finished executing, your EHLLAPI program could submit a batch job using either a copy function or the **Send Key** (3) function before logging off.
    - If the **Receive File** (91) function finished executing, your EHLLAPI program could start up a local application.

Scenario 5. Automation

An application can provide all the keystrokes for another application or can intersperse keystrokes to the target destination with those from the keyboard. Sometimes, to do this, the application must lock out other sources of keystroke input that might be destined for a target application or presentation space (using the **Reserve** (11) function) and the later unlock it (using the **Release** (12) function).

The origin of keystrokes presented to any application is determined by the design of the application. Keystrokes can originate from:

- The keyboard
- Data integrated into the source application
- Secondary storage retrieved through the DOS interface
- The Z and I Emulator for Windows interface

In all cases the keystrokes that are provided to the target application are indistinguishable from the ordinary operator input.

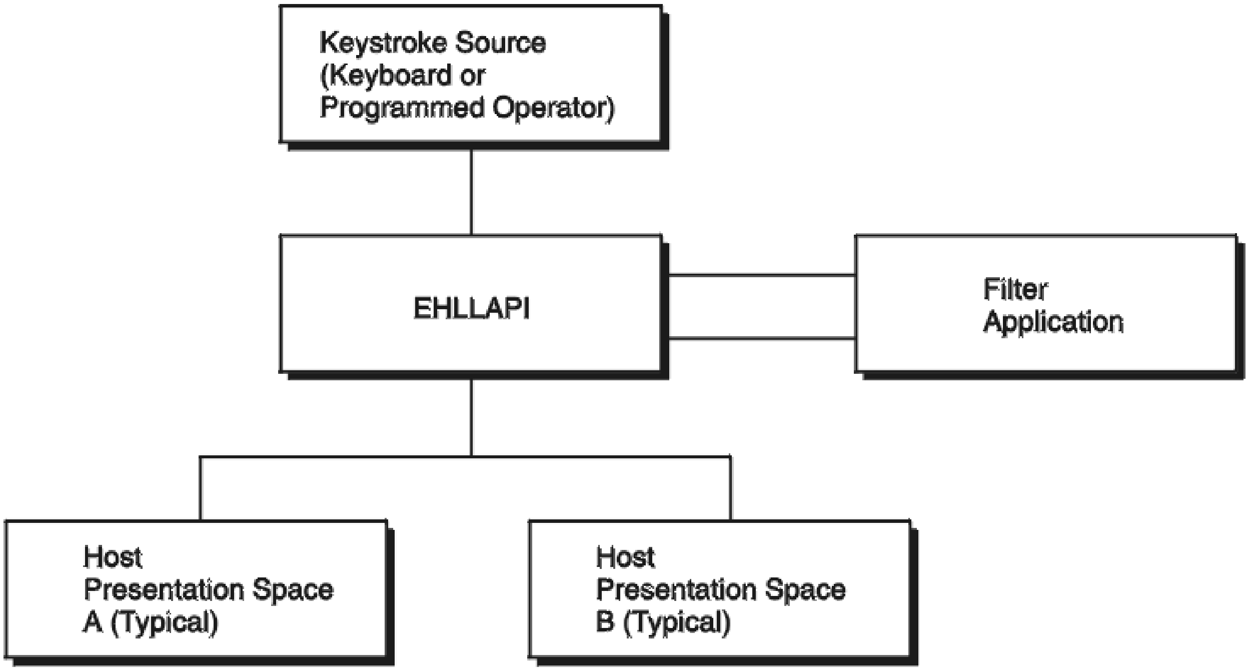
Scenario 6. Keystroke Filtering

An application that acts as a filter can intercept a keystroke coming from EHLLAPI (either from the keyboard or a source application) that is targeted for another destination. The keystroke can then be:

- Ignored (that is, deleted)
- Redirected to another application
- Validated
- Converted (for example, uppercase to lowercase)
- Enhanced (through keyboard macros)

Figure 11: Keystroke Flow on page 535 provides a simplified representation of the keystroke flow and the objects within a keyboard enhancement environment.

Figure 11. Keystroke Flow



Scenario 7. Keyboard Enhancement

This scenario makes use of filtering to create an **enhancer application program**. An enhancer application program is one that monitors the data coming in from the keyboard and changes it in some specified way. Typically, these application programs use instructions called **keyboard macros**, which tell them what keystrokes to look for and what changes to make. The change might involve suppressing a keystroke (so it appears to the target application as though it was never sent), replacing a keystroke with another, or replacing single keystroke with a series of keystrokes.

To do this using EHLLAPI, you might construct this scenario:

1. Your EHLLAPI application program calls the **Connect Presentation Space** (1) function to connect to the presentation space whose keystrokes are to be filtered.
2. Your EHLLAPI program next calls the **Start Keystroke Intercept** (50) function specifying the L option. This causes all keystrokes to be routed to the filtering application program.
3. The filtering application program can now define a loop in which:
    a. The **Get Key** (51) function intercepts all keystrokes being sent to the target presentation space.
    b. The filtering application examines each keystroke and performs a keyboard macro task, such as:
        - Abbreviating program commands so that three- or four-keystroke command can be condensed into a single keystroke
        - Customizing commands so that they are easier to remember or consistent with other software packages

- Creating **boiler plates** for contracts or frequently used letters
- Rearranging the keyboard for concurrent applications that use the same keys for differing functions

For example, the filtering application might convert a key combination such as Alt+Y into a command to move the cursor to column 35 of the second line in presentation space and write the string “XYZ Tool Corporation, Dallas, Texas”.

c. If a keystroke is rejected, your EHLLAPI program can cause a beep to be sounded, using the **Post Intercept Status** (52) function.

4. After your EHLLAPI program exits the filtering loop, **Stop Keystroke Intercept** (53) function to end the filtering process.

EHLLAPI Functions

This chapter describes each individual Z and I Emulator for Windows EHLLAPI function in detail and explains how to use the EHLLAPI program sampler. The functions are arranged alphabetically by name. The functions are explained for both the standard and enhanced interfaces.

Note: Throughout this chapter WinHLLAPI, IBM® Standard 32-bit HLLAPI and 16-bit EHLLAPI are referred to as Standard Interface, and IBM® Enhanced 32-bit EHLLAPI is referred to as Enhanced Interface.

Page Layout Conventions

All EHLLAPI function calls are presented in the same format so that you can quickly retrieve the information you need. The format is:

- Function Name (Function Number)
  - Prerequisite Calls
  - Call Parameters
  - Return Parameters
  - Notes on Using This Function

Prerequisite Calls

“Prerequisite Calls” lists any calls that must be made prior to calling the function being discussed.

Call Parameters

“Call Parameters” lists the parameters that must be defined in your program to call the discussed EHLLAPI function and explains how those parameters are to be defined. If a parameter is never used by a function, then *NA* (not applicable) is listed. If a parameter can be overridden by certain values of session parameters defined with calls to the **Set Session Parameters** (9) function, such session parameters are named.

Return Parameters

“Return Parameters” lists the parameters that must be received by your program after a call to the discussed EHLLAPI function and explains how to interpret those parameters.

Notes on Using This Function

“Notes on Using This Function” lists any session options that affect the function under discussion. It also provides technical information about using the function and application development tips.

Summary of EHLLAPI Functions

Table 70: EHLLAPI Functions Summary on page 537 is the summary of the EHLLAPI functions:

Table 70. EHLLAPI Functions Summary

Function	3270	5250	VT
Connect Presentation Space (1) on page 546	Yes	Yes	Yes
Disconnect Presentation Space (2) on page 581	Yes	Yes	Yes
Send Key (3) on page 634	Yes	Yes	Yes
Wait (4) on page 667	Yes	Yes	Yes
Copy Presentation Space (5) on page 564	Yes	Yes	Yes
Search Presentation Space (6) on page 631	Yes	Yes	Yes
Query Cursor Location (7) on page 607	Yes	Yes	Yes
Copy Presentation Space to String (8) on page 569	Yes	Yes	Yes
Set Session Parameters (9) on page 643	Yes	Yes	Yes
Query Sessions (10) on page 614	Yes	Yes	Yes
Reserve (11) on page 627	Yes	Yes	Yes
Release (12) on page 626	Yes	Yes	Yes
Copy OIA (13) on page 555	Yes	Yes	Yes
Query Field Attribute (14) on page 608	Yes	Yes	Yes
Copy String to Presentation Space (15) on page 575	Yes	Yes	Yes
Pause (18) on page 600	Yes	Yes	Yes
Query System (20) on page 616	Yes	Yes	Yes

Table 70. EHLLAPI Functions Summary (continued)

Function	3270	5250	VT
Reset System (21) on page 628	Yes	Yes	Yes
Query Session Status (22) on page 612	Yes	Yes	Yes
Start Host Notification (23) on page 657	Yes	Yes	Yes
Query Host Update (24) on page 611	Yes	Yes	Yes
Stop Host Notification (25) on page 665	Yes	Yes	Yes
Search Field (30) on page 629	Yes	Yes	Yes
Find Field Position (31) on page 587	Yes	Yes	Yes
Find Field Length (32) on page 585	Yes	Yes	Yes
Copy String to Field (33) on page 573	Yes	Yes	Yes
Copy Field to String (34) on page 551	Yes	Yes	Yes
Copy Presentation Space to Clipboard (35) on page 576	Yes	Yes	Yes
Paste Clipboard to Presentation Space (36) on page 578	Yes	Yes	Yes
Set Cursor (40) on page 642	Yes	Yes	Yes
Start Close Intercept (41) on page 653	Yes	Yes	Yes
Query Close Intercept (42) on page 604	Yes	Yes	Yes
Stop Close Intercept (43) on page 664	Yes	Yes	Yes
Query Additional Field Attribute DRB on page 603	No	Yes	No
Start Keystroke Intercept (50) on page 660	Yes	Yes	Yes
Get Key (51) on page 589	Yes	Yes	Yes
Post Intercept Status (52) on page 601	Yes	Yes	Yes
Stop Keystroke Intercept (53) on page 666	Yes	Yes	Yes
Lock Presentation Space API (60) on page 596	Yes	No	No
Lock Window Services API (61) on page 598	Yes	No	No
Start Communication Notification (80) on page 655	Yes	Yes	Yes
Query Communication Event (81) on page 606	Yes	Yes	Yes
Stop Communication Notification (82) on page 665	Yes	Yes	Yes
Send File (90) on page 683	Yes	Yes	No
Receive File (91) on page 624	Yes	Yes	No
Cancel File Transfer (92) on page 540	Yes	Yes	Yes
Convert Position or Convert RowCol (99) on page 548	Yes	Yes	Yes
Connect Window Services (101) on page 547	Yes	Yes	Yes
Disconnect Window Service (102) on page 582	Yes	Yes	Yes
Query Window Coordinates (103) on page 618	Yes	Yes	Yes
Window Status (104) on page 668	Yes	Yes	Yes
Change Switch List LT Name (105) on page 542	Yes	Yes	Yes
Change PS Window Name (106) on page 541	Yes	Yes	Yes
Start Playing Macro (110) on page 663	Yes	Yes	Yes

Table 70. EHLLAPI Functions Summary (continued)

Function	3270	5250	VT
Connect for Structured Fields (120) on page 544	Yes	No	No
Disconnect from Structured Fields (121) on page 580	Yes	No	No
Query Communications Buffer Size (122) on page 605	Yes	No	No
Allocate Communications Buffer (123) on page 539	Yes	No	No
Free Communications Buffer (124) on page 588	Yes	No	No
Get Request Completion (125) on page 593	Yes	No	No
Read Structured Fields (126) on page 619	Yes	No	No
Write Structured Fields (127) on page 673	Yes	No	No

Allocate Communications Buffer (123)

3270	*5250*	*VT*
Yes	No	No

The **Allocate Communications Buffer** function obtains a buffer from the operating system. A buffer address must be passed on both the **Read Structured Fields** (126) and **Write Structured Fields** (127) functions.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 123	
Data String	See the following table	
Length	Must be 6	Must be 8
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1–2	1–4	32-bit or 16-bit buffer length. (0 < size ≤ (64 KB-256 bytes)=X'FF00')
3–6	5–8	32-bit allocated buffer address (returned)

Return Parameters

Return Code	Explanation
0	The **Allocate Communications Buffer** function was successful.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).

Notes on Using This Function

1. The EHLLAPI obtains a buffer from the operating system memory management and places the buffer address into the return parameter string. The requested buffer size (length) is also passed in the parameter string. The buffer size can be from 1 byte to 64 KB minus 256 bytes (X'FF00' bytes) in length.

   See *“**Query Communications Buffer Size** (122)”* for information regarding buffer size.
2. Buffers obtained using this function must not be shared among different processes. If this is attempted, the applications will experience unpredictable results.
3. An EHLLAPI application must issue a **Free Communications Buffer** (124) function to free the allocated memory.
4. A maximum of 10 buffers can be allocated to an application. If this limit is reached, a return code for resource unavailable (RC=11) will be returned.
5. The **Reset System** (21) function frees buffers allocated by this function.

Cancel File Transfer (92)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Cancel File Transfer** function causes any current EHLLAPI initiated **Send File** or **Receive File** for the specified session to immediately return.

Prerequisite Calls

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

Call Parameters

	Enhanced Interface
Function Number	Must be 92
Data String	1-character short name of the host presentation space. A blank or null indicates request for updates to the host-connected presentation space
Length	4 is implied

	Enhanced Interface
PS Position	NA

The calling data structure contains these elements

Byte	Definition
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered

Notes on Using This Function

Since both **Send File** (90) and **Receive File** (91) are blocking calls, this function must always be issued on a different thread.

Change PS Window Name (106)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Change PS Window Name** function allows the application to specify a new name for the presentation space window or reset the presentation space window to the default name.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 106	
Data String	See the following table	
Length	Must be specified (See note.)	Must be 68
PS Position	NA	

Note: The data string length must be specified (normally 3–63 for PC/3270, 4–63 for PC400, 68 for enhanced interface).

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2	5	A change request option value, select one of: • X'01' for changing the presentation space window name. • X'02' for resetting the presentation space window name.
3–63	6–66	An ASCII string of from 1 (for PC/3270) or 2 (for PC400) to 61 bytes including a terminator byte. The ASCII string must end with a NULL character. This string must contain at least one non-NULL character followed by a NULL character.
	67–68	Reserved

Return Parameters

Return Code	Explanation
0	The **Change PS Window Name** function was successful.
1	An incorrect host presentation space short session ID was specified, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

A string is ended at the first NULL character found. The NULL character overrides the specified string length. If the NULL character is not at the end of the specified length, the last byte at the specified length is replaced by a NULL character, and the remainder of the data string is lost. If the NULL character is found before the specified length, the string is truncated at that point, and the remainder of the data string is lost.

If the application fails to reset the presentation space name before exiting, the exit list processing resets the name.

Change Switch List LT Name (105)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Change Switch List LT Name** function allows the application to change or reset a switch list for a selected logical terminal (LT). The application must specify on the call the name to be inserted in the switch list.

Note: This is for compatibility with Communication Manager EHLLAPI, and has the same result as the **Change PS Window Name** (106) function.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 105	
Data String	See the following table	
Length	Normally 4–63	Must be 68
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2	5	A change request option; select: • X'01' for changing a switch list LT name • X'02' for resetting a switch list LT name
3–63	6–66	An ASCII string of 2 to 61 bytes including a terminator byte. The ASCII string must end with a NULL character. This string must contain at least one non-NULL character followed by a NULL character.
	67–68	Reserved

Return Parameters

Return Code	**Explanation**
0	The **Change Switch List LT Name** function was successful.

Return Code	Explanation
1	An incorrect host presentation space short session ID was specified, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

A string is ended at the first NULL character found. The NULL character overrides the specified string length. If the NULL character is not at the end of the specified length, the last byte at the specified length is replaced by a NULL character, and the remainder of the data string is lost. If the NULL character is found before the specified length, the string is truncated at that point, and the remainder of the data string is lost.

If the application fails to reset the switch list LT name before exiting, the exit list processing resets the name.

Connect for Structured Fields (120)

3270	***5250***	***VT***
Yes	No	No

The **Connect for Structured Fields** function allows an application to establish a connection to the emulation program to exchange structured field data with a host application. The workstation application must provide the Query Reply data field and must point to it with in the parameter string. The destination/origin ID returned by the emulator will be returned to the application.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 120	
Data String	See the following table	
Length	7 or 11	Must be 16
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)

Byte		Definition
	2–4	Reserved
2–5	5–8	Address of the Query Reply data buffer
6–7	9–10	Destination/origin unique ID. (16-bit word, returned)
	11–12	Reserved
8–11	13–16	The data in these position is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

Return Code	Explanation
0	The **Connect for Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid, or the host presentation space was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
10	The function is not supported by the emulation program.
32	An application has already connected to this session for communications (successful connect).
39	One DDM session is already connected to this session.

Notes on Using This Function

1. EHLLAPI scans the query reply buffers for the destination/origin ID (DOID) self-defining parameter (SDP) to determine the contents of the DOID field of the query reply. If this value is X'0000', the emulator will assign a DOID to the application and EHLLAPI will fill in the DOID field of the query reply with the assigned ID. If the value specified by the application in the DOID field of the query reply is a nonzero value, the emulator will assign the specified value as the application's DOID, assuming that the ID has not been previously assigned. If the specified DOID is already in use, a return code of 2 will be returned by EHLLAPI.
2. The application should build the Query Reply Data structures in the application's private memory. Refer to Query Reply Data Structures Supported by EHLLAPI on page 721, for the detailed formats and usages of the query reply data structures supported by EHLLAPI.
3. Only cursory checking is performed on the Query Reply Data. Only the ID and the length of the structure are checked for validity.
4. Only one DDM base type connect is allowed per host session. If the DDM connection supports the self-defining parameter (SDP) for the destination origin ID (DOID), then multiple connects are allowed.
5. If return code RC=32 or RC=39 is received, an application is already connected to the selected session and use of that presentation space should be approached with caution. Conflicts with file transfer, and other EHLLAPI applications might result.

Connect Presentation Space (1)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Connect Presentation Space** function establishes a connection between your EHLLAPI application program and the host presentation space.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 1	
Data String	1-character short name of the host presentation space	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

The **Connect Presentation Space** function sets the return code to indicate the status of the attempt and, if successful, the status of the host presentation space.

Return Code	**Explanation**
0	The **Connect Presentation Space** function was successful; the host presentation space is unlocked and ready for input.
1	An incorrect host presentation space ID was specified. The specified session either does not exist or is a logical printer session. This return code could also mean that the API Setting for EHLLAPI is not set on.
4	Successful connection was achieved, but the host presentation space is busy.
5	Successful connection was achieved, but the host presentation space is locked (input inhibited).
9	A system error was encountered.

Return Code	Explanation
11	This resource is unavailable. The host presentation space is already being used by another system function.

Notes on Using This Function

1. The **Connect Presentation Space** function is affected by the `CONLOG`/`CONPHYS` session option.
2. An EHLLAPI application cannot be connected to multiple presentation spaces concurrently. Calls requiring the **Connect Presentation Space** function as a prerequisite use the currently connected presentation space. For example, if an application is connected to presentation space A, B, and C in that order, the application must connect to B or A again to issue functions.
3. Each thread that requests a **Connect Presentation Space** must have a corresponding **Disconnect Presentation Space** (2), or one of the threads must issue a **Reset System** (21), which affects all threads and disconnects any remaining connections.
4. More than one EHLLAPI application can share a presentation space, if the applications support sharing (that is, if they were developed to work together and if they exhibit predictable behavior) and have compatible read/write access and keyword options as set in the **Set Sessions Parameters** (9) function. For more information, see Set Session Parameters (9) on page 643.
5. Because the **Connect Presentation Space** and **Start Keystroke Intercept** (50) functions share common subsystem functions, successful requests by an application to share either of these functions for the same session can affect the request of these two functions by other applications. For example, if application A successfully requests a **Connect Presentation Space** for a session with Write_Read access and KEY $abcdefgh as the keyword, a request by application B to **Connect Presentation Space** for a session and **Start Keystroke Intercept** is successful only if both applications have set compatible read/write options.
6. You cannot connect to a session that is defined as a logical printer session. Refer to *Administrator's Guide and Reference* for more information.

Connect Window Services (101)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Connect Window Services** function allows the application to manage the presentation space windows. Only one EHLLAPI application at a time can be connected to a presentation space for window services.

An EHLLAPI application can connect to more than one presentation space concurrently for window services.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 101	
Data String	1-character short session ID of the host presentation space	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

Return Code	Explanation
0	The **Connect Window Services** function was successful.
1	An incorrect host presentation space short session ID was specified, or the Sessions Window Services manager was not connected. This return code could also mean that the API Setting for EHLLAPI is not set on.
9	A system error occurred.
10	The function is not supported by the emulation program.
11	This resource is unavailable. The host presentation space is already being used by another system function.

Notes on Using This Function

1. An EHLLAPI application can be connected to multiple presentation space windows at the same time. The application can go back and forth between the connected presentation space windows without having to disconnect. For example, if an application is connected to presentation space windows A, B, and C, the application can access all of A, B, and C at the same time, and the other applications cannot access A, B, or C.
2. A **Connect Window Services** function is sufficient for the process. However, each thread that requests a **Connect Window Services** must have a corresponding **Disconnect Window Services** (102), or one of the threads must issue a **Reset System** (21), which affects all threads and disconnects any remaining connections.

Convert Position or Convert RowCol (99)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Convert Position** or **Convert RowCol** function converts the host presentation space positional value into the display row and column coordinates or converts the display row and column coordinates into the host presentation space positional value. This function does not change the cursor position.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 99	
Data String	Host presentation space short name *and* `P` for the **Convert Position** function (for example, `AP` converts the presentation space position of session A); *or* Host presentation space short name and `R` for the **Convert RowCol** function (for example, `AR` converts the row and column coordinates of session A).	
Length	Row, when `R` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input depends on how your host presentation space is configured. See Notes on Using This Function on page 551. NA when `P` is specified as the second character in the data string parameter.	
PS Position	Column, when `R` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input ranges from 24 to 43 depending on how your host presentation space is configured. See Notes on Using This Function on page 551. Host presentation space position, when `P` is specified as the second character in the data string parameter. The lower limit for valid input is 1. The upper limit for valid input ranges from 1920 to 3564 depending on how your host presentation space is configured. See Notes on Using This Function on page 551.	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)

Byte		Definition
	2–4	Reserved
2	5	**Convert** option `P` or `R`
	6–8	Reserved

Return Parameters

This function returns a length and a return code.

Length:

For the **Convert Position** function (`P` as the second character in the calling data string), a number between 1 and 43 (for PC/3270) or 27 (for PC400) is returned. This value is the number of the row that contains the PS position contained in the calling PS position parameter. The upper limit can be smaller than 43 (for PC/3270) or 27 (for PC400) depending on how the host presentation space is configured.

For the **Convert RowCol** function (`R` as the second character in the calling data string), a value of 0 indicates an error in the input value for row (calling length parameter).

Return Code:

The **Convert Position or RowCol** function is the exception to the rule that the fourth return parameter always contains a return code. For this function, the value returned in the fourth parameter is called a status code. This status code can contain data or a return code. Your application must provide for processing of this status code to prevent unpredictable results or an error.

- If the value of the fourth parameter is 0, 9998, or 9999, it is a return code.
- For the **Convert Position** function (`P` as the second character of the calling data string), a value in the range of 1–132 is the number of the column that contains the PS position passed in the calling PS Position parameter. The upper limit can be smaller than 132 depending on how the host presentation space is configured.
- For the **Convert RowCol** function (`R` as the second character of the calling data string), a value in the range of 1–3564 represents the host presentation space position that corresponds to the row and column values passed in the calling length and PS position parameters, respectively. The upper limit can be smaller than 3564 depending on how the host presentation space is configured.

The following status codes are defined:

Status Code	Explanation
0	This is an incorrect PS position or column.
>0	This is the PS position or column.
9998	An incorrect host presentation space ID was specified or a system error occurred.
9999	Character 2 in the data string is not P or R.

Notes on Using This Function

1. To configure your presentation space, refer to *Administrator's Guide and Reference*
2. To find out how many rows and columns are in your presentation space, examine the returned data string parameter for the **Query Session Status** (22) function. See Query Session Status (22) on page 612.

Copy Field to String (34)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy Field to String** function transfers characters from a field in the host-connected presentation space into a string.

The **Copy Field to String** function translates the characters in the host source presentation space into American National Standard Code for Information Interchange (ASCII). Attribute bytes and other characters not represented in ASCII normally are translated into blanks.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 34	
Data String	Preallocated target data string. When the **Set Session Parameters** (9) function with Extended Attribute Bytes (EAB) option is issued, the length of the data string must be at least twice the length of the field.	
Length	Number of bytes to copy (the length of the data string).	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

A string containing data from the identified field in the host presentation space. The first byte in the returned data string is the beginning byte of the identified field in the host presentation space. The number of bytes in the returned data string is determined by the smaller of:

- Number of bytes specified in the calling length parameter
- Number of bytes in the identified field in the host presentation space

Length:

The length of the data returned.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Copy Field to String** function was successful.
1	Your program is not connected to a host session.
2	An error was made in specifying parameters.
6	The data to be copied and the target field are not the same size. The data is truncated if the string length is smaller than the field copied.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

1. The field position and length information can be found by using the **Find Field Position** (31) and **Find Field Length** (32) functions. The **Copy Field to String** function can be used with either protected or unprotected fields, but only in a *field-formatted* host presentation space.
2. The copy is ended when one of the following conditions is encountered:
   - When the end of the field is reached
   - When the length of the target string is exceeded
3. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned preceding each character.
4. The **Copy Field to String** function is affected by the `ATTRB`/`NOATTRB`/`NULLATTRB`, the `EAB`/`NOEAB`, the `XLATE`/`NOXLATE`, the `DISPLAY`/`NODISPLAY`, the `DISPLAY`/`NODISPLAY` session options. Refer to items 5 on page 646; 13 on page 649 and 14 on page 650; 17 on page 650; and and for more information.

   As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

   Set Session Parameter

   Effect on the COPY Function

   NOEAB and NOEAD

   Attributes are not returned. Only text is copied from the presentation space to the user buffer.

   EAB and NOXLATE

   Attributes are returned as defined in the following tables.

EAB and XLATE

The colors used for the presentation space display are returned. Colors can be remapped; so the attribute colors are not the ones returned by the **COPY** functions when XLATE and EAB are on at the same time.

The returned character attributes are defined in the following tables. The attribute bit positions are in IBM® format with bit 0 the left most bit in the byte.

3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	**Meaning**
0–1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2–4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5–6	Character attributes 00 = Default value
7	Reserved

5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	**Meaning**
0	Reverse image 0 = Normal image 1 = Reverse image
1	Underline 0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink

Bit Position	Meaning
3	Separator of columns 0 = No separator 1 = Separator
4–7	Reserved

The following table shows Z and I Emulator for Windows character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0–3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4–7	Foreground character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

For a PS/2® monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

5. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation on page 515 for more information.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on the status bar. By **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Copy OIA (13)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Copy OIA** function returns the current operator information area (OIA) data from the host-connected presentation space.

The OIA is located under the bottom dividing line of the screen and is used to display session status information about the connection between the workstation and the host.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 13	
Data String	Preallocated target data string	
Length	103	104
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Data String:

A 103-byte string for 16-bit and 104-byte string for 32-bit. See Format of the Returned OIA Data String on page 556 for more information.

Return Code:

The following codes are defined:

Return Code	Explanation
0	OIA data is returned. The target presentation space is unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length. OIA data was not returned.
4	OIA data is returned. The target presentation space is busy.
5	OIA data is returned. The target presentation space is locked. (Input inhibited)
9	An internal system error was encountered. OIA data was not returned.

Notes on Using This Function

1. The OIA Group consists of the bits that show the status of the connected sessions. The group is categorized by the represented host function. (For example, Group 8 consists of the bits that show all conditions of the input inhibit in the session.) The states of each group are ordered so that the high-order bits represent the indicators of higher priority. That is, bit 7 has priority over bit 0. Therefore, if more than one state is active within a group, the state with the highest priority is the active state within that group.
2. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation on page 515 for more information.

Format of the Returned OIA Data String

The OIA data string contains the following information:

Byte		Definition
Standard	Enhanced	
1	1	The OIA format byte. The value is 1 (PC/3270), 9 (PC400), or 5 (VT).
2–81	2–81	The OIA image in the host code points.
82–103	82–103	OIA group indicator meanings.
	104	Reserved.

PC/3270 OIA Group Indicator Meanings and Its Image

The OIA image group consists of an 80-byte ASCII character string with no attribute bytes that contains the OIA image in host code points. Figure 12: Host Presentation Space Characters on page 558 shows the hexadecimal codes found in the host presentation space, and the characters they represent. The returned data can be translated into OIA graphics characters. Refer to *Quick Beginnings* for information on the OIA indicators.

To translate the returned data into OIA graphics characters, proceed as follows:

1. Print the data returned in bytes 2 through 81 to the screen or to a printer.
2. Using the code page chart applicable to the device on which the output appears, find the hexadecimal value corresponding to each character.
3. Using Figure 12: Host Presentation Space Characters on page 558, find the OIA graphics character corresponding to each hexadecimal value found in step 2.

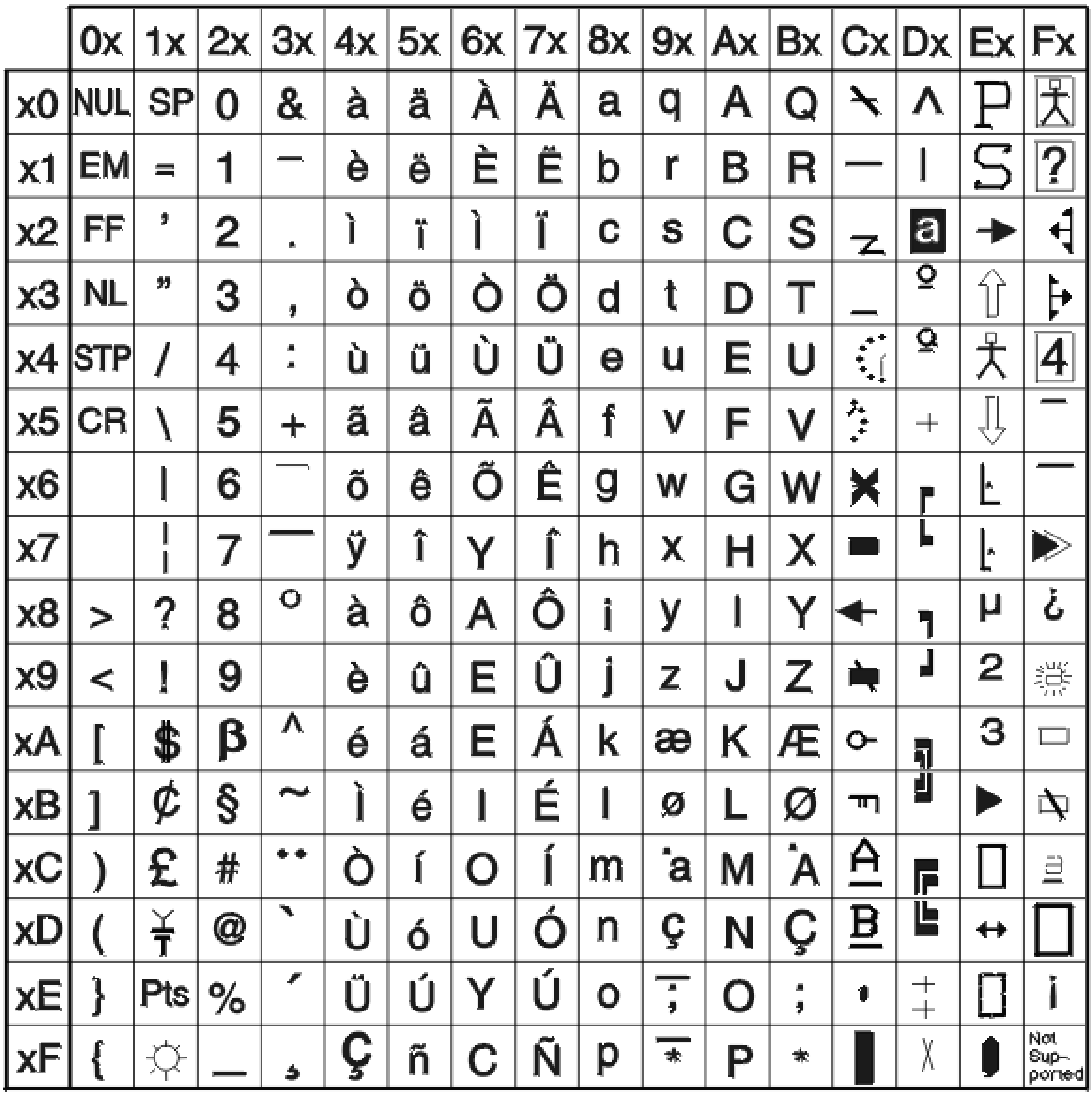
Note: Group 8 (byte 0) machine, communications, and program check images are followed by a three-digit number related to the type of check.

The short session ID followed by X'20' is in column 7.

All group images are represented by Main Frame Interactive (MFI) hex code points.

Note: The OIA image data string position minus 1 position equals the OIA column.

Figure 12. Host Presentation Space Characters



	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL	SP	0	&	à	ä	À	Ä	a	q	A	Q	↘	∧	P	웃
x1	EM	=	1	¯	è	ë	È	Ë	b	r	B	R	—	|	S	?
x2	FF	’	2	.	ì	ï	Ì	Ï	c	s	C	S	≥	a	→	⊣
x3	NL	”	3	,	ò	ö	Ò	Ö	d	t	D	T	_	♀	⇧	⊢
x4	STP	/	4	:	ù	ü	Ù	Ü	e	u	E	U	◌	♀	웃	4
x5	CR	\	5	+	ã	â	Ã	Â	f	v	F	V	⁘	+	⇩	¯
x6		|	6	¯	õ	ê	Õ	Ê	g	w	G	W	✖	┏	∟	¯
x7		¦	7	—	ÿ	î	Y	Î	h	x	H	X	▬	┗	∟	⊳
x8	>	?	8	°	à	ô	A	Ô	i	y	I	Y	←	┐	µ	¿
x9	<	!	9		è	û	E	Û	j	z	J	Z	■	┘	2	☼
xA	[	$	ß	^	é	á	E	Á	k	æ	K	Æ	⊸	⌉	3	▭
xB	]	¢	§	~	ì	é	I	É	l	ø	L	Ø	╥	⌋	▶	⊘
xC	)	£	#	¨	ò	í	O	Í	m	å	M	Å	A̲	╔	▯	≞
xD	(	¥	@	`	ù	ó	U	Ó	n	ç	N	Ç	B̲	╚	↔	▯
xE	}	Pts	%	´	ü	ú	Y	Ú	o	;	O	;	•	⁺₊	⌷	i
xF	{	☼	_	¸	Ç	ñ	C	Ñ	p	*	P	*	▌	╳	⬮	Not Sup- ported

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning
0–1	Reserved
2	SSCP-LU session owns screen
3	LU-LU session owns screen
4	Online and not owned

Bit	Meaning
5	Subsystem ready
6–7	Reserved

- Group 2 (Offset 83): Character Selection

Bit	Meaning
0	Reserved
1	APL
3	Alphanumeric
4–5	Reserved

- Group 3 (Offset 84): Shift State

Bit	Meaning
0	Upper shift
1	Numeric
2	CAPS
3–7	Reserved

- Group 4 (Offset 85): PSS Group 1

Bit	Meaning
0–7	Reserved

- Group 5 (Offset 86): Highlight Group 1

Bit	Meaning
0	Operator selectable
1	Field inherit
2–7	Reserved

- Group 6 (Offset 87): Color Group 1

Bit	Meaning
0	Operator selectable
1	Field inherit
2–7	Reserved

- Group 7 (Offset 88): Insert

Bit	Meaning
0	Insert mode
1–7	Reserved

- Group 8 (Offset 89–93): Input Inhibited (5 bytes)
    - Byte 1 (Offset 89)

Bit	Meaning
0	Non-resettable machine check

Bit	Meaning
1	Reserved
2	Machine check
3	Communications check
4	Program check
5–7	Reserved

- Byte 2 (Offset 90)

Bit	Meaning
0	Device busy
1	Terminal wait
2	Minus symbol
3	Minus function
4	Too much entered
5–7	Reserved

- Byte 3 (Offset 91)

Bit	Meaning
0–2	Reserved
3	Incorrect dead key combination, limited key.
4	Wrong place
5–7	Reserved

- Byte 4 (Offset 92)

Bit	Meaning
0–1	Reserved
2	System wait
3–7	Reserved

- Byte 5 (Offset 93)

Bit	Meaning
0–7	Reserved

- Group 9 (Offset 94): PSS Group 2

Bit	Meaning
0–7	Reserved

- Group 10 (Offset 95): Highlight Group 2

Bit	Meaning
0–7	Reserved

- Group 11 (Offset 96): Color Group 2

Bit	Meaning
0–7	Reserved

- Group 12 (Offset 97): Communication Error Reminder

Bit	Meaning
0-6	Communications error
1–7	Reserved

- Group 13 (Offset 98): Printer State

Bit	Meaning
0–7	Reserved

- Group 14 (Offset 99): Graphics

Bit	Meaning
0–7	Reserved

- Group 15 (Offset 100): Reserved
- Group 16 (Offset 101): Automatic Key Play/Record State

Bit	Meaning
0–7	Reserved

- Group 17 (Offset 102): Automatic Key Quit/Stop State

Bit	Meaning
0–7	Reserved

- Group 18 (Offset 103): Expanded State

Bit	Meaning
0–7	Reserved

PC400 OIA Group Indicator Meanings and Its Image

Details of the OIA group are listed in the following tables.

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning	Beginning Position of Data String
0–2	Reserved	
3	System available	1
4	Reserved	
5	Subsystem ready	
6–7	Reserved	

- Group 2 (Offset 83): Character Selection

Bit	Meaning	Beginning Position of Data String
0–1	Reserved	
3	Alphanumeric	
4–5	Reserved	

- Group 3 (Offset 84): Shift State

Bit	Meaning	Beginning Position of Data String
0	Reserved	
1	Keyboard shift	39
2	CAPS	
3–6	Reserved	

- Group 4 (Offset 85): PSS Group 1

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 5 (Offset 86): Highlight Group 1

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 6 (Offset 87): Color Group 1

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 7 (Offset 88): Insert

Bit	Meaning	Beginning Position of Data String
0	Insert mode	68
1–7	Reserved	

- Group 8 (Offset 89–93): Input Inhibited (5 bytes)
  - Byte 1 (Offset 89)

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

  - Byte 2 (Offset 90)

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

  - Byte 3 (Offset 91)

Bit	Meaning	Beginning Position of Data String
0–4	Reserved	
5	Operator input error	64
6–7	Reserved	

- Byte 4 (Offset 92)

Bit	Meaning	Beginning Position of Data String
0–1	Reserved	
2	System wait	64
3–7	Reserved	

- Byte 5 (Offset 93)

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 9 (Offset 94): PSS Group 2

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 10 (Offset 95): Highlight Group 2

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 11 (Offset 96): Color Group 2

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 12 (Offset 97): Communication Error Reminder

Bit	Meaning	Beginning Position of Data String
0	Communications Error	
1–5	Reserved	
7	Message wait	3

- Group 13 (Offset 98): Printer State

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 14 (Offset 99): Graphics

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 15 (Offset 100): Reserved
- Group 16 (Offset 101): Automatic Key Play/Record State

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 17 (Offset 102): Automatic Key Quit/Stop State

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

- Group 18 (Offset 103): Expanded State

Bit	Meaning	Beginning Position of Data String
0–7	Reserved	

VT Host OIA Group Indicator Meanings and Its Image

Details of the VT Host OIA group are listed in the following tables.

- Group 1 (Offset 82): Online and Screen Ownership

Bit	Meaning
5	Subsystem ready

- Group 2 (Offset 83): Character Selection

Bit	Meaning
0	Upper shift
2	CAPS

- Group 7 (Offset 88): Insert

Bit	Meaning
0	Insert mode

Some columns on the OIA line display different messages for VT than those messages displayed for 3270/5250. See the following table for specific details.

Column	Symbol
1–7	VT220 7
	VT220 8
	VT100
	VT52
	VTANSI
9 - 12	LOCK
61 - 64	HOLD

Copy Presentation Space (5)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Copy Presentation Space** function copies the contents of the host-connected presentation space into a data string that you define in your EHLLAPI application program.

The **Copy Presentation Space** function translates the characters in the host source presentation space into ASCII. Attribute bytes and other characters not represented in ASCII normally are translated into blanks. If you do not want the attribute bytes translated into blanks, you can override this translation with the ATTRB option under the **Set Session Parameters** (9) function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 5	
Data String	Preallocated target string the size of your host presentation space. This can vary depending on how your host presentation space is configured. When the **Set Session Parameters** (9) function with the EAB option is issued, the length of the data string must be at least twice the length of the presentation space.	
Length	NA (the length of the host presentation space is implied).	
PS Position	NA.	

Return Parameters

This function returns a data string, length, and a return code.

Data String:

Contents of the connected host presentation space.

Length:

Length of the data copied.

Return Code:

The following codes are defined:

Return Code	**Explanation**
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
4	The host presentation space contents were copied. The connected host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.

Return Code	Explanation
9	A system error was encountered.

Notes on Using This Function

1. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned preceding each character.
2. The **Copy Presentation Space** function is affected by the following session options:
   - ATTRB/NOATTRB/NULLATTRB
   - EAB/NOEAB
   - XLATE/NOXLATE
   - BLANK/NOBLANK
   - DISPLAY/NODISPLAY
   - EXTEND_PS/NOEXTEND_PS

Refer to items 5 on page 646; 13 on page 649, 14 on page 650, 15 on page 650 and 17 on page 650; and and for more information.

If the target data string provided is not long enough to hold the requested data, unpredictable results can occur.

As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

Set Session Parameter

Effect on the COPY Function

NOEAB and NOEAD

Attributes are not returned. Only text is copied from the presentation space to the user buffer.

EAB and NOXLATE

Attributes are returned as defined in the following tables.

EAB and XLATE

The colors used for the presentation space display are returned. Colors can be remapped; so the attribute colors are not the ones returned by the **Copy** functions when XLATE and EAB are on at the same time.

NOSO/SPACESO/SO

When NOSO is specified, it works as SPACESO. The size of the presentation space is not changed.

The returned character attributes are defined in the following tables. The attribute bit positions are in IBM® format with bit 0 the left most bit in the byte.

3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0–1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2–4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5–6	Character attribute 00 = Default value
7	Reserved

5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0	Reverse image 0 = Normal image 1 = Reverse image
1	Underline 0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink
3	Separator of columns 0 = No separator 1 = Separator
4–7	Reserved

The following table shows Z and I Emulator for Windows character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0–3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4–7	Foreground character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

For a PS/2® monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

If you want to copy only a portion of the host presentation space, use the **Copy Presentation Space to String** (8) function.

To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation on page 515 for more information.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Copy Presentation Space to String (8)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Copy Presentation Space to String** function is used to copy all or part of the host-connected presentation space into a data string that you define in your EHLLAPI application program.

The input PS position is the offset into the host presentation space. This offset is based on a layout in which the upper-left corner (row 1/column 1) is location 1 and the bottom-right corner is 3564, which is the maximum screen size for the host presentation space. The value of `PS Position + (Length – 1)` cannot exceed the configured size of your host presentation space.

The **Copy Presentation Space to String** function translates the characters in the host source presentation space into ASCII. Attribute bytes and other characters not represented in ASCII normally are translated into blanks. If you do not want the attribute bytes translated into blanks, you can override this translation with the ATTRB option under the **Set Session Parameters** (9) function.

Prerequisite Calls

Connect Presentation Space (1).

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 8	
Data String	Preallocated target string the size of your host presentation space. When the **Set Session Parameters** (9) function with the EAB option is issued, the length of the data string must be at least twice the length of the presentation space.	
Length	Length of the target data string.	
PS Position	Position within the host presentation space of the first byte in your target data string.	

Return Parameters

This function returns a data string and a return code.

Data String:

Contents of the host presentation space.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The host presentation space contents were copied to the application program. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length, or the sum of (Length - 1) + PS position is greater than the size of the connected host presentation space.
4	The host presentation space contents were copied. The host presentation space was waiting for host response.
5	The host presentation space was copied. The keyboard was locked.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

1. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned following each character.
2. The **Copy Presentation Space to String** function is affected by the following options:
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB
    - XLATE/NOXLATE
    - BLANK/NOBLANK
    - DISPLAY/NODISPLAY
    - EXTEND_PS/NOEXTEND_PS

    Refer to items 5 on page 646; 13 on page 649 and 14 on page 650; 15 on page 650; 17 on page 650; and and

    If the target data string provided is not large enough to hold the requested number of bytes, the copy ends successfully (RC=0, 4, or 5) when the end of the target data string is reached.

    As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters** (9) function. The involved set session parameters have the following effect:

    Set Session Parameter

    Effect on the Copy Function

NOEAB and NOEAD

Attributes are not returned. Only text is copied from the presentation space to the user buffer.

EAB and NOXLATE

Attributes are returned as defined in the following tables.

EAB and XLATE

The colors used for the presentation space display are returned. Colors can be remapped, so the attribute colors are not the ones returned by the **Copy** functions when XLATE and EAB are on at the same time.

The returned character attributes are defined in the following tables. The attribute bit positions are in IBM format with bit 0 the left most bit in the byte.

- 3270 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0–1	Character highlighting 00 = Normal 01 = Blink 10 = Reverse video 11 = Underline
2–4	Character color (Color remap can override this color definition.) 000 = Default 001 = Blue 010 = Red 011 = Pink 100 = Green 101 = Turquoise 110 = Yellow 111 = White
5–7	Reserved

- 5250 character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0	Reverse image 0 = Normal image 1 = Reverse image
1	Underline

Bit Position	Meaning
	0 = No underline 1 = Underline
2	Blink 0 = Not blink 1 = Blink
3	Separator of columns 0 = No separator 1 = Separator
4–7	Reserved

- VT character attributes are returned from the host to the emulator. The following table applies when EAB and NOXLATE are set.

Bit Position	Meaning
0-3	Reserved
4	Bold 1 = On 0 = Off
5	Underscore 1 = On 1 = Off
6	Blink 1 = On 0 = Off
7	Reverse 0 = On 1 = Off

- The following table shows Z and I Emulator for Windows character color attributes. The following table applies when EAB and XLATE are set.

Bit Position	Meaning
0–3	Background character colors 0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White
4–7	Foreground character colors

Bit Position	Meaning
	0000 = Black 0001 = Blue 0010 = Green 0011 = Cyan 0100 = Red 0101 = Magenta 0110 = Brown (3270), Yellow (5250) 0111 = White 1000 = Gray 1001 = Light blue 1010 = Light green 1011 = Light cyan 1100 = Light red 1101 = Light magenta 1110 = Yellow 1111 = White (high intensity)

For a PS/2 monochrome display, the characters in the application (workstation) session appear as various shades of gray. This is required to give users their remapped colors in the EHLLAPI application session so they can get what they see in their host application presentation spaces.

3. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation on page 515 for more information.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Copy String to Field (33)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Copy String to Field** function transfers a string of characters into a specified field in the host-connected presentation space. This function can be used only in a *field-formatted* host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 33	
Data String	String containing the data to be transferred to a target field in the host presentation space.	
Length	Length, in number of bytes, of the source data string. Overridden if in EOT mode.	
PS Position	Identifies the target field. This can be the PS position of any byte within the target field. Copy always starts at the beginning of the field.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target field was protected or inhibited, or incorrect data was sent to the target field (such as a field attribute).
6	Copy was completed, but data is truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Unformatted host presentation space.

Notes on Using This Function

1. The **Copy String to Field** function is affected by the following options:
   - STRLEN/STREOT
   - EOT
   - EAB/NOEAB
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB

   Refer to items 1 on page 645 and 2 on page 645; 13 on page 649 and 14 on page 650; 18 on page 650; and and for more information.
2. The string to be transferred is specified with the calling data string parameter. The string ends when one of these three conditions is encountered:

- When an end-of-text (EOT) delimiter is encountered in the string if EOT mode was selected using the **Set Session Parameters** (9) function. (See Set Session Parameters (9) on page 643).
- When the number specified in the length is reached if not in EOT mode.
- When an end-of-field is encountered in the field.

Note: If the field at the end of the host presentation space wraps, wrapping occurs when the end of the presentation space is reached.

3. The keyboard mnemonics (see **Send Key** (3) function) cannot be sent using the **Copy String to Field** function.
4. The first byte of the data to be transferred is always placed at the beginning of the field that contains the specified PS position.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Copy String to Presentation Space (15)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Copy String to Presentation Space** function copies an ASCII data string directly into the host presentation space at the location specified by the PS position calling parameter.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 15.	
Data String	String of ASCII data to be copied into the host presentation space.	
Length	Length, in number of bytes, of the source data string. Overridden if in EOT mode.	
PS Position	Position in the host presentation space to begin the copy, a value between 1 and the configured size of your host presentation space.	

Return Parameters

Return Code	Explanation
0	The **Copy String to Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy.
5	The target presentation space is protected or inhibited, or incorrect data was sent to the target presentation space (such as a field attribute byte).
6	The copy was completed, but the data was truncated.
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

1. The **Copy String to Presentation Space** function is affected by the following options:
   - STRLEN/STREOT
   - EOT
   - EAB/NOEAB
   - XLATE/NOXLATE
   - PUTEAB/NOPUTEAB
   - EXTEND_PS/NOEXTEND_PS

   Refer to items 1 on page 645 and 2 on page 645; 13 on page 649 and 14 on page 650; 18 on page 650; and and for more information.
2. The keyboard mnemonics (see **Send Key** (3) function) cannot be sent using the **Copy String to Presentation Space** function.
3. The string ends when an end-of-text (EOT) delimiter is encountered in the string if EOT mode was selected using the **Set Session Parameters** (9) function. (See Set Session Parameters (9) on page 643).
4. Although the **Send Key** (3) function accomplishes the same purpose, this function responds with the prompt and enters a command more quickly. Because the **Send Key** (3) function emulates the terminal operator typing the data from the keyboard, its process speed is slow for an application operating with a lot of data. This function provides a faster input path to the host.
5. The original data (the copied string) cannot exceed the size of the presentation space.
6. This function call may cause a cursor movement to an unexpected position with some host applications. A SendKey function may be a better choice for filling a field than this function.

   Note: This only occurs with VT sessions or connections to an ASCII host.

Copy Presentation Space to Clipboard (35)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Copy Presentation Space to Clipboard** function is used to copy all or part of the host-connected presentation space into clipboard. The input PS position is the offset into the host presentation space. This offset is based on a layout in which the upper-left corner (row 1/column 1) is location 1 and the bottom-right corner is 3564, which is the maximum screen size for the host presentation space. The value of PS Position + (Length – 1) cannot exceed the configured size of your host presentation space.

The **Copy Presentation Space to Clipboard** translates the characters in the host source presentation space into ASCII. Attribute bytes and other characters not represented in ASCII normally are translated into blanks. If you do not want the attribute bytes translated into blanks, you can override this translation with the ATTRB option under the **Set Sesssion Parameters (9)** function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 35.	
Data String	Preallocated target string the size of your host presentation space. When the **Set Session Parameters (9)** function with the EAB option is issued, the length of the data string must be at least twice the length of the presentation space.	
Length	Length of the target data string	
PS Position	Position within the host presentation space of the first byte in your target data string.	

Return Parameters

Return Code	**Explanation**
0	The host presentation space contents were copied to the clipboard. The target presentation space was active, and the keyboard was unlocked.
1	Your program is not connected to a host session.
2	An error was made in specifying string length, or the sum of (Length - 1) + PS position is greater than the size of the connected host presentation space.
4	The host presentation space contents were copied to clipboard. The host presentation space was waiting for host response.
5	The host presentation space was copied to clipboard. The keyboard was locked.

Return Code	Explanation
7	The host presentation space position is not valid.
9	A system error was encountered.

Notes on Using This Function

1. An EAB can be returned when the **Set Session Parameters** (9) function EAB option is used. EAB is related to each character in the presentation space and is returned following each character.
2. The **Copy Presentation Space to Clipboard** function is affected by the following options:
    - ATTRB/NOATTRB/NULLATTRB
    - EAB/NOEAB
    - XLATE/NOXLATE
    - BLANK/NOBLANK
    - DISPLAY/NODISPLAY
    - PUTEAB/NOPUTEAB
    - EXTEND_PS/NOEXTEND_PS
3. The data string buffer is used in processing the data retrieved from presentation space and copy to clipboard. If the data string buffer provided is not large enough to hold the requested number of bytes, the copy ends successfully (RC=0, 4, or 5) when the end of the data string buffer is reached. As previously stated, the return of attributes by the various **Copy** (5, 8, and 34) functions is affected by the **Set Session Parameters (9)** function. The involved set session parameters have the following effect:

    Set Session Parameter

    Effect on the Copy Function

    NOEAB and NOEAD

    Attributes are not returned. Only text is copied from the presentation space to the clipboard.

    EAB and NOXLATE

    Attributes are returned as defined in the following tables.

    EAB and XLATE

    The colors used for the presentation space display are returned. Colors can be remapped, so the attribute colors are not the ones returned by the **Copy** functions when XLATE and EAB are on at the same time.

Paste Clipboard to Presentation Space (36)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Paste Clipboard to Presentation Space** function pastes an ASCII data string directly into the host presentation space at the location specified by the PS position calling parameter.

Prerequisite Calls

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 36.	
Data String	String buffer that holds the data from clipboard to paste into the host presentation space.	
Length	Length, in number of bytes to be pasted	
PS Position	Position in the host presentation space to begin the copy, a value between 1 and the configured size of your host presentation space.	

Return Parameters

Return Code	**Explanation**
0	The **Paste Clipboard to Presentation Space** function was successful.
1	Your program is not connected to a host session.
2	Parameter error or zero length for copy
5	The target presentation space is protected or inhibited, or incorrect data was sent to the target presentation space (such as a field attribute byte).
6	The copy was completed, but the data was truncated.
7	The copy was completed, but the data was truncated.
9	A system error was encountered.

Notes on Using This Function

1. The **Paste Clipboard to Presentation Space** function is affected by the following options:
    - STRLEN/STREOT
    - EAB/NOEAB
    - EOT
    - XLATE/NOXLATE
    - PUTEAB/NOPUTEAB
    - EXTEND_PS/NOEXTEND_PS
2. The string ends when an end-of-text (EOT) delimiter is encountered in the string if EOT mode was selected using the **Set Session Parameters (9)** function. (See "Set Session Parameters (9)" on page 147).
3. The original data (the copied string) cannot exceed the size of the presentation space.

String	Meanings	Single-byte character field	
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 means single-byte characters.			

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows always displays the same information on the 24th row. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Disconnect from Structured Fields (121)

3270	*5250*	*VT*
Yes	No	No

The **Disconnect from Structured Fields** function drops the connection between the emulation program and the EHLLAPI application. The EHLLAPI application must disconnect from the emulation program before exiting from the system. The EHLLAPI application should issue this function request if a previous **Connect for Structured Fields** was issued.

The **Reset System (21)** function will also disconnect any outstanding SF connections.

Prerequisite Calls

Connect for Structured Fields (120)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 121	
Data String	See the following table	
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).

Byte		**Definition**
	2–4	Reserved.
2–3	5–6	Destination/origin unique ID returned by the Connect for structured field (120) functions.
	7–8	Reserved.

Return Parameters

Return Code	**Explanation**
0	The **Disconnect from Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
40	Disconnected with asynchronous requests pending.

Notes on Using This Function

1. When a **Disconnect from Structured Fields** function is called, any outstanding asynchronous **Read Structured Fields** (126) or **Write Structured Fields** (127) function requests are returned if the application issues the **Get Request Completion** (125) function call. Use the asynchronous form of this function when cleaning up after issuing a Disconnect call.
2. The **Reset System** (21) function will also free any outstanding asynchronous requests (requests that have not been retrieved by the application using the **Get Request Completion** (125) function).

Disconnect Presentation Space (2)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Disconnect Presentation Space** function drops the connection between your EHLLAPI application program and the host presentation space. Also, if a host presentation space is reserved using the **Reserve** (11) function, it is released upon execution of the **Disconnect Presentation Space** function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 2	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The **Disconnect Presentation Space** function was successful.
1	Your program was not currently connected to the host presentation space.
9	A system error was encountered.

Notes on Using This Function

1. After the **Disconnect Presentation Space** function is called, functions that interact with the host-connected presentation space are no longer valid (for example, the **Send Key** (3), **Wait** (4), **Reserve** (11) and **Release** (12) functions).
2. Your EHLLAPI application should disconnect from the host presentation space before exiting.
3. The **Disconnect Presentation Space** function does not reset the session parameters to the defaults. Your EHLLAPI application must call the **Reset System** (21) function to accomplish this.

Disconnect Window Service (102)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Disconnect Window Service** function disconnects the window services connection between the EHLLAPI program and the specified host presentation space window.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 102	

	Standard Interface	Enhanced Interface
Data String	See the following table	
Length	1	4
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

Return Code	Explanation
0	The **Disconnect Window Service** function was successful.
1	Your program is not connected for Window Services.
9	A system error occurred.

Notes on Using This Function

After the **Disconnect Window Service** function has been called, your application no longer manages the presentation space window.

Before exiting the application, you should request a **Disconnect Window Service** function for all presentation spaces that have been connected for Presentation Manager® services. If the application exits with an outstanding connection for window services, the subsystem cancels the outstanding connection.

EditKey Intercept

This feature enables you to intercept Edit keys in addition to the existing all keystrokes and send them to a session in a Windows 32-bit environment.

Prerequisites

1. Map the Edit functions in the Customize Keyboard window (for example Ctrl+C for edit copy function).
2. Call the Start Keystroke Intercept (50) EHLLAPI function with the call parameter data string value set. The values are as follows:

Byte Position	Contents
1	One of the following values: • A specific host presentation space short name (PSID) • A blank or null indicating a request for the host-connected host presentation space
2 to 4	Reserved
5	An option code character: • D for AID keystrokes only • L for all keystrokes • E for all keystrokes and Edit keys • M for requesting the asynchronous message mode of the notification (Windows only). If M is specified, a code character D or L, or E must be placed in position 13
6 to 8	Reserved
9 to 12	If M is specified in position 5, the window handle of the window that receives the message. The message is a non-zero return value of RegisterWindowMessage (PCSHLL).
13	If M is specified in position 5, one of the following values: • D for AID keystrokes only • L for all keystrokes • E for all keystrokes and Edit keys
14 to 16	Reserved

3. To get the intercepted Edit keys, use the Get Key (51) EHLLAPI function. The key mnemonic returned in the data string for the Edit keys will have M (keystroke type mnemonic) at the 5th byte position. The next 4 bytes will have one of the following Edit key mnemonics based on the Edit key intercepted:

Key mnemonic	Key intercepted
@W@C	Edit Copy
@W@D	Edit Clear
@W@E	Edit Copy Append
@W@L	Edit Copy Link
@W@N	Edit Paste Next
@W@V	Edit Paste
@W@X	Edit Cut
@W@Z	Edit Undo

4. To send Edit keys to the session, use the Send Key (3) EHLLAPI function. The data string passed as the call parameter can specify the following Edit key mnemonics:

Key mnemonic	Key sent
@W@C	Edit Copy

Key mnemonic	Key sent
@W@D	Edit Clear
@W@E	Edit Copy Append
@W@L	Edit Copy Link
@W@N	Edit Paste Next
@W@V	Edit Paste
@W@X	Edit Cut
@W@Z	Edit Undo

Note:

1. You do not have to call the Get Key (51) EHLLAPI function to use the Send Key (3) function. For both Get Key (51) and Send Key (3) functions to handle Edit keys, you must first call Start Keystroke Intercept (50) with the 5th byte position set to **E**. If the 5th byte contains **M**, then position 13 must contain **E**.
2. The expected return values for Start Keystroke Intercept (50), Get Key (51) and Send Key (3) functions have not changed.
3. Any prerequisites from the existing documentation should be followed as well as the prerequisites documented here.

Find Field Length (32)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Find Field Length** function returns the length of a target field in the connected presentation space. This function can be used to find either protected or unprotected fields, but only in a *field-formatted* host presentation space.

This function returns the number of characters contained in the field identified using the call PS position parameter. This includes all characters from the beginning of the target field up to the character preceding the next attribute byte.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 32	

	Standard Interface	Enhanced Interface
Data String	See the following table	
Length	NA	NA
PS Position	See note	

Note: PS Position: Identifies the field within the host presentation space at which to start the **Find**. It can be the PS position of any byte within the field in which you desire the **Find** to start.

The calling 2-character data string can contain:

Code	Explanation
ƀƀ or Tƀ	This field
Pƀ	The previous field, either protected or unprotected.
Nƀ	The next field, either protected or unprotected
NP	The next protected field
NU	The next unprotected field
PP	The previous protected field
PU	The previous unprotected field

Note: The ƀ symbol represents a required blank.

Return Parameters

This function returns a length and a return code.

Length:

The following lengths are valid:

Length	Explanation
= 0	When return code = 28, field length is 0. When return code = 24, host presentation space is not field formatted.
> 0	Required field length in the host presentation space.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Find Field Length** function was successful.
1	Your program is not connected to a host session.
2	A parameter error was encountered.
7	The host presentation space position is not valid.
9	A system error was encountered.

Return Code	Explanation
24	No such field was found.
28	Field length of 0 bytes.

Notes on Using This Function

Except when ƀƀ or Tƀ is used as the calling data string, if the field found is the same as the field from which the **Find** started, a return code of 24 is returned.

Find Field Position (31)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Find Field Position** function returns the beginning position of a target field in the host-connected presentation space. This function can be used to find either protected or unprotected fields but only in a *field-formatted* host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 31	
Data String	See the following table	
Length	NA	NA
PS Position	See note	

Note: PS Position: Identifies the field within the host presentation space at which to start the **Find**. It can be the PS position of any byte within the field in which you want the **Find** to start.

The calling 2-character data string can contain:

Code	Explanation
ƀƀ or Tƀ	This field
Pƀ	The previous field, either protected or unprotected
Nƀ	The next field, either protected or unprotected
NP	The next protected field
NU	The next unprotected field
PP	The previous protected field

Code	Explanation
PU	The previous unprotected field

Note: The ƀ symbol represents a required blank.

Return Parameters

This function returns a length and a return code.

Length:

The following lengths are valid:

Length	Explanation
= 0	When return code = 28, field length is 0. When return code = 24, host presentation space is not field-formatted.
> 0	Relative position of the requested field from the origin of the host presentation space. This position is defined to be the first position after the attribute byte.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Find Field Position** function was successful.
1	Your program is not connected to a host session.
2	A parameter error was encountered.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	No such field was found.
28	Field length of 0 bytes.

Notes on Using This Function

Except when ƀƀ or Tƀ is used as the calling data string, if the field found is the same as the field from which the **Find** started, a return code of 24 is returned.

Free Communications Buffer (124)

3270	*5250*	*VT*
Yes	No	No

The **Free Communications Buffer** function returns to management memory a buffer that is no longer required by the application. The application should free the buffer prior to exiting the system.

Prerequisite Calls

Allocate Communications Buffer (123)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 124	
Data String	See the following table	
Length	Must be 6	Must be 8
PS Position	NA	

Data String Contents

Byte		**Definition**
Standard	Enhanced	
1–2	1–4	Must be 0
3–6	5–8	The address of the buffer

Return Parameters

Return Code	**Explanation**
0	The **Free Communications Buffer** function was successful.
2	An error was made in specifying parameters.
9	A system error occurred.
41	The buffer is in use.

Notes on Using This Function

1. If the application attempts to free an in use buffer, the free request will be denied and a return code of 41 will be returned.
2. An application should request the **Free Communications Buffer** (124) function before exiting for all communication buffers that have been allocated using the **Allocate Communications Buffer** (123) function.
3. The **Reset System** (21) function will free buffers allocated by the **Allocate Communications Buffer** (123) function.

Get Key (51)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Get Key** function lets your EHLLAPI application program retrieve a keystroke from a session specified by the **Start Keystroke Intercept** (50) function and either process, accept, or reject that keystroke. By placing this function in a loop, you can use it to intercept a string.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 51	
Data String	See the following table	
Length	8	12
PS Position	NA	

Data String Contents

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
	2–4	Reserved
2–8	5–11	Blanks that hold space for the symbolic representation of the requested data
	12	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

See the following table:

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the host-connected presentation
	2–4	Reserved
2	5	An option code character, one of the following characters: • `A` for ASCII returned • `M` for keystroke mnemonic • `S` for special mnemonic
3–8	6–11	These 6 bytes of the preallocated buffer space are used internally to enqueue and dequeue keystrokes. Possible combinations include: • Byte 3 contains an ASCII character and byte 4 contains X'00' • Byte 3 contains the escape character (either `@` or another character specified using the `ESC=c` option of function 9) and byte 4 contains a 1-byte abbreviation for a function. (See ASCII Mnemonics on page 525) • Bytes 5 through 8 might be similar to bytes 3 and 4 if the returned ASCII mnemonic is longer than 2 bytes (for example, if the ASCII mnemonic represents Attn `@A@Q`, byte 5 contains `@` and byte 6 contains `Q`). If not used, bytes 5 through 8 are set to zero (X'00').

For clarification, some examples of returned data strings are provided below:

Note: The `@` symbol is the default escape character. The value of the escape character can be set to any keystroke represented in ASCII by using the `ESC=c` option of the **Set Session Parameters** (9) function. If the escape character has been changed to another character using this option, the `@` symbol in the following examples is replaced by the other character.

16-Bit Interface

`EAt`

`E` is the presentation space short name. The keystrokes are returned as ASCII (A), and the returned key is the lowercase letter t. (Bytes 4–8 = X'00').

EM@2

E is the presentation space short name. The keystrokes are returned as mnemonics, and the returned key is PF2 (Bytes 5–8 = X'00').

32-Bit Interface

Eƀƀƀ**At**

E is the presentation space short name. The keystrokes are returned as ASCII (A), and the returned key is the lowercase letter t. (Bytes 7–11 = X'00').

Eƀƀƀ**M@2**

E is the presentation space short name. The keystrokes are returned as mnemonics, and the returned key is PF2 (Bytes 8–11 = X'00').

Return Code:

The following codes are valid:

Return Code	Explanation
0	The **Get Key** function was successful.
1	An incorrect presentation space was specified.
5	You specified the AID only option under the **Start Keystroke Intercept** (50) function, and non-AID keys are inhibited by this session type when EHLLAPI tries to write incorrect keys to the presentation space.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.
20	An undefined key combination was typed.
25	The requested keystrokes are not available on the input queue.
31	Keystroke queue overflowed and keystrokes were lost.

Notes on Using This Function

1. If a return code of 31 occurs for the **Get Key** function, either:
    - Increase the value of the calling length parameter for the **Start Keystroke Intercept** (50) function, or
    - Execute the **Get Key** function more frequently.

    An intercepted keystroke occupies 3 bytes in the buffer. The next intercepted keystroke is placed in the adjacent three bytes. When the **Get Key** function retrieves a keystroke (first in first out, FIFO), the three bytes that it occupied are made available for another keystroke. By increasing the size of the buffer or the rate at which keystrokes are retrieved from the buffer, you can eliminate buffer overflow.

    For the PC/3270, another way to eliminate return code 31 is to operate the PC/3270 emulator in the resume mode.

2. You can use the **Send Key** (3) function to pass both original keystrokes and any others that your EHLLAPI application might need to the host-connected presentation space.
3. Keystrokes arrive asynchronously and are enqueued in the keystroke queue that you have provided in your EHLLAPI application program using the **Start Keystroke Intercept** (50) function.
4. The **Get Key** function behaves like a read. When keystrokes are available, they are read into the data area that you have provided in your application.
5. In the case of field support for a session, the application might be interested only in AID keys, for example the Enter key. If so, the **Start Keystroke Intercept** (50) function option code should be set to `D` (meaning for AID Keys only).
6. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. Refer to Memory Allocation on page 515 for more information.

Get Request Completion (125)

3270	*5250*	*VT*
Yes	No	No

The **Get Request Completion** function allows an application to determine the status of a previous asynchronous function request issued to the EHLLAPI and to obtain the function parameter list before using the data string again. This function is valid only if the user specified asynchronous (A) completion on a previous function call such as **Read Structured Fields** (126) or **Write Structured Fields** (127).

Each asynchronous request requiring the **Get Request Completion** function will return a unique ID from the asynchronous request. The application must save this ID. This ID is the identification used by the **Get Request Completion** function to identify the desired request. The user has three request options using this function:

1. The application can query or wait for a specific asynchronous function request by supplying the request ID of that function and a nonblank session short name.
2. The application can query or wait for the first completed asynchronous function request for a specified session by supplying a request ID of X'0000' and a nonblank session short name.

Prerequisite Calls

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

and

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

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 125	
Data String	See the following table	
Length	Must be 14	Must be 24
PS Position	NA	

Data String Contents

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2	5	N or W N=NOWAIT is required W=WAIT is required
	6–8	Reserved
3–4	9–10	Function request ID.
5–6	11–12	Reserved
7–10	13–16	Reserved
11–12	17–20	Reserved
13–14	21–24	Reserved

The **Get Request Completion** function behaves differently depending upon the second character of the parameter string, which is one of the following characters:

N

Nowait option: If a specific request ID was supplied and the function has completed, control will be returned to the application with a return code of zero and a completed data string as defined in Return Parameters on page 595. If a request ID of zero was supplied and any eligible asynchronous function has completed, control will be returned to the application with a return code of zero and a completed data string as defined in Return Parameters on page 595.

W

Wait option: If a specific request ID was supplied and the function has not completed, the call will wait until the function has completed before returning to the application. If the supplied request ID was zero and no eligible asynchronous function has completed, the call will wait until a function completes before returning to the calling application. On return, the return code value will be zero and the data string will be completed as defined in Return Parameters on page 595.

Return Parameters

Byte		**Definition**
Standard	Enhanced	
5–6	11–12	Function number of the completed asynchronous function (126 or 127). (returned)
7–10	13–16	Address of the data string of the completed asynchronous function call. (The application must not reuse the data string until the request has completed). (returned)
11–12	17–20	Length of the data string of the completed asynchronous function call. (returned)
13–14	21–24	Return code of the completed asynchronous function call. (returned)

Return Code	Explanation
0	The **Get Request Completion** function was successful.
2	An error was made in specifying parameters.
9	A system error was encountered.
38	Requested function was not complete.
42	No matching request was found.

There are some differences between return codes 38 and 42:

1. Return code 38
    a. If a specific request ID and session were requested, both the session and ID were found but the request is pending (not in a completed state).
    b. If a zero request ID and a specific session were requested, the specified session has pending requests, but they are not satisfied (complete).
    c. If a zero request ID and a blank session were requested, pending requests were found but none were satisfied (complete).
2. Return code 42
    a. If a specific request ID and session were requested, the specific request ID was not found in either a pending or a completed state.
    b. If a zero request ID and a specific session were requested, the specific session contains no pending or completed requests.
    c. If a zero request ID and a blank session were requested, no pending or completed requests were found.

Notes on Using This Function

1. This function is valid only if the user specified asynchronous completion (A for Asynchronous) on a previous function call such as **Read Structured Fields** or **Write Structured Fields**.
2. If the return code is a 0, the application should check the returned data string for information pertaining to the completion of the requested asynchronous function.

Lock Presentation Space API (60)

3270	*5250*	*VT*
Yes	No	No

The **Lock Presentation Space API** function allows the application to obtain or release exclusive control of the presentation space window over other Windows 32–bit applications. While locked, no other application can connect to the presentation space window.

Successful processing of this function with the Lock causes EHLLAPI presentation space window functions requested from other EHLLAPI applications to be queued until the requesting application unlocks the presentation space. Requests from the locking application are processed normally.

Prerequisite Calls

Connect to Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 60	
Data String	See the following table	
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–4	Reserved.
2	5	One of the following characters:

Byte		Definition
		• L to lock the API. • U to unlock the API.
3	6	One of the following characters: • R to return if the presentation space is already locked by an application. • Q to queue the Lock request if the presentation space is already locked by an application.
	7–8	Reserved.

Return Parameters

Return Code	Explanation
0	The **Lock Presentation Space API** function was successful.
1	An incorrect host presentation space short session ID was specified or was not connected.
2	An error was made in specifying parameters.
9	A system error was encountered.
43	The API was already locked by another EHLLAPI application (on LOCK) or API not locked (on UNLOCK).

Notes on Using This Function

The following EHLLAPI functions are queued when a lock is in effect:

- **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) with the CONPHYS parameter set in a previous **Set Sessions Parameter** (9) function call.

These queued requests are not serviced until the lock is removed. When the lock is removed, the queued requests are processed in first-in-first-out (FIFO) order. EHLLAPI functions not listed are run as if there was no lock. The requesting application unlocks the presentation space window by one of the following methods:

- Disconnecting from the presentation space while still owning the Lock.
- Issuing the **Reset System** (21) function while still owning the Lock.
- Stopping the application while still owning the Lock.
- Stopping the session.
- Successfully issuing the **Lock Presentation Space API** with the Unlock option.

Before exiting the application, you should unlock any presentation space windows that have been locked with the **Lock Presentation Space API** function. If the application exits with outstanding locks, or a **Reset System** (21), or **Disconnect Presentation Space** (2) function is issued, the locks are released.

It is recommended that applications lock the presentation space only for short periods of time and only when exclusive use of the presentation space is required.

Lock Window Services API (61)

3270	*5250*	*VT*
Yes	No	No

The **Lock Window Services API** function allows the application to obtain or release exclusive control of the presentation space window over other Windows 32-bit applications. While locked, no other application can connect to the presentation space window.

Successful processing of this function with the Lock causes EHLLAPI presentation space window functions requested from other EHLLAPI applications to be queued until the requesting application unlocks the presentation space. Requests from the locking application are processed normally.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 61	
Data String	See the following table.	

	Standard Interface	**Enhanced Interface**
Length	Must be 3	Must be 8
PS Position	NA	

Data String Contents

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–4	Reserved.
2	5	One of the following characters: • L to lock the API. • U to unlock the API.
3	6	One of the following characters: • R to return if the presentation space is already locked by an application. • Q to queue the Lock request if the presentation space is already locked by an application.
5–6	11–12	Function number of the completed asynchronous function (126 or 127). (returned)
	7–8	Reserved.

Return Parameters

Return Code	**Explanation**
0	The **Lock Window Services API** function was successful.
1	An incorrect host presentation space short session ID was specified or was not connected.
2	An error was made in specifying parameters.
9	A system error was encountered.
38	Requested function was not complete.
43	The API was already locked by another EHLLAPI application (on LOCK) or API not locked (on UNLOCK).

Notes on Using This Function

The following EHLLAPI functions are queued when a lock is in effect:

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

These queued requests are not serviced until the lock is removed. When the lock is removed, the queued requests are processed in first-in-first-out (FIFO) order.

The requesting application unlocks the presentation space window by one of the following methods:

- Successfully issuing the **Lock Window Services API** with the UNLOCK option.
- Disconnecting from the presentation space while still owning the Lock.
- Issuing the **Reset System** (21) function while still owning the Lock.
- Stopping the application while still owning the Lock.
- Stopping the session.

Before exiting the application, you should Unlock any presentation space windows that have been locked with the **Lock Window Services API** function. If the application exits with outstanding locks, the subsystem releases the locks.

It is recommended that applications lock the presentation space only for short periods of time and only when exclusive use of the presentation space is required.

Pause (18)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Pause** function waits for a specified amount of time. It should be used in place of *timing loops* to wait for an event to occur. A **Pause** function can be ended by a host event if a prior **Start Host Notification** (23) function has been called and the IPAUSE option is selected.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 18	
Data String	NA	
Length	Contains the pause duration in half-second increments	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The wait duration has expired.
9	An internal system error was encountered. The time results are unpredictable.
26	The host session presentation space or OIA has been updated. Use the **Query Host Update** (24) function to get more information.

Notes on Using This Function

1. Selecting the FPAUSE or IPAUSE option using the **Set Session Parameters** (9) function affects the length of the pause you get when you call this function. See item 6 on page 646 for more information.
2. The value entered in the calling length parameter is the maximum number of half-second intervals that the **Pause** function waits. For a pause of 20 seconds, a hex value of `0028` (decimal 40) must be passed in the calling length parameter.
3. If you use the IPAUSE option and the pause value is zero, then the function waits up to 2400 half-second intervals, unless interrupted sooner. If you use the FPAUSE option and the pause value is zero, then the function returns immediately.
4. If you use the IPAUSE option, once a pause has been satisfied by a host event, you should call the **Query Host Update** (24) function to clear the queue prior to the next **Pause** function. The **Pause** function will continue to be satisfied with the pending event until the **Query Host Update** (24) function is completed.
5. A practical maximum value for the **Pause** function is `2400`. You should not use the **Pause** function for these kinds of tasks:
   - Delay for very long durations (of several hours, for example).
   - Delay for more than a moderate length of time (20 minutes) before checking the system time-of-day clock and proceeding with your EHLLAPI program execution.
   - With applications requiring a high-resolution timer because the time interval created by a **Pause** function is approximate.
   - Set the time interval to zero in a loop.
6. IPAUSE set and the interruptible pause allow an EHLLAPI application to determine whether the specified host presentation space (PS) or operator information area (OIA) is updated. The following three functions are used:
   - **Start Host Notification** (23)
   - **Query Host Update** (24)
   - **Stop Host Notification** (25)

   By using IPAUSE when the **Start** function is called, you can make an application wait until the host presentation space or OIA (or both) receives an update. When the receive is completed and the application can issue the **Query** function to determine the changes, **Pause** terminates. Then the application issues the **Search Presentation Space** (6) to check whether the expected update occurred.

Post Intercept Status (52)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Post Intercept Status** function informs the Z and I Emulator for Windows emulator that a keystroke obtained through the **Get Key** (51) function was accepted or rejected. When the application rejects a keystroke, the **Post Intercept Status** function issues a beep.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 52	
Data String	See the following table	
Length	Must be 2	Must be 8
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • The 1-letter short name of the presentation space. • A blank or null indicating a function call for the host-connected presentation space.
	2–4	Reserved
2	5	One of the following characters: • A for accepted keystroke. • R for rejected keystroke.
	6–8	Reserved.

Return Parameters

Return Code	**Explanation**
0	The **Post Intercept Status** function was successful.
1	An incorrect presentation space was specified.
2	An incorrect session option was specified.

Return Code	Explanation
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space ID.
9	A system error was encountered.

Query Additional Field Attribute (45)

3270	*5250*	*VT*
No	Yes	No

The **Query Additional Field Attribute** function returns additional information about the 5250 field containing the input host presentation space position. This information is returned in the data string parameter in the form of a defined structure.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 45.	
Data String	8 bytes long character string.	
Length	8 is implied.	
PS Position	Identifies the target. This can be the PS position of any byte within the target field.	

The calling data string can contain:

Byte	Definition
1–8	Reserved

Return Parameters

This function returns a data string and a return code.

Data String:

The function returns the following data string.

Byte	Definition
1–6	Reserved
7–8	Two 8–bit unsigned characters that return:

Byte	**Definition**
	• R if field is RTL and L if field is LTR. • U if field is upper case and L if field is a normal case field.

Return Code:

The following return codes are defined:

Return Code	**Explanation**
0	The **Query Additional Field Attribute** was successful.
1	Your program is not currently connected to a host session.
7	The host presentation space position is not valid.
9	No field was found in this position.
24	Field is unformatted.

Query Close Intercept (42)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Query Close Intercept** function allows the application to determine if the close option was selected.

Prerequisite Calls

Start Close Intercept (41)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 42	
Data String	See the following table.	
Length	Must be 1	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	1-character short session ID of the host presentation space, or a blank or null indicating request for querying the host-connected session
	2–4	Reserved

Return Parameters

Return Code	Explanation
0	A close intercept event did not occur.
1	The presentation source was not valid.
2	An error was made in specifying parameters.
8	No prior **Start Close Intercept** (41) function was called for this host presentation space.
9	A system error occurred.
12	The session stopped.
26	A close intercept occurred since the last query close intercept call.

Query Communications Buffer Size (122)

3270	***5250***	***VT***
Yes	No	No

The **Query Communications Buffer Size** function allows an application to determine both the maximum and the optimum buffer sizes supported by the emulation program.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 122	
Data String	See the following table	
Length	Must be 9	Must be 20
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2–3	5–8	16- or 32-bit field for the optimum supported inbound buffer size (Returned value)

Byte		Definition
4–5	9–12	16- or 32-bit field for the maximum supported inbound buffer size (Returned value)
6–7	13–16	16- or 32-bit field for the optimum supported outbound buffer size (Returned value)
8–9	17–20	16- or 32-bit field for the maximum supported outbound buffer size (Returned value)

Return Parameters

Return Code	Explanation
0	The **Query Communications Buffer Size** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
10	The function was not supported by the emulation program.

Notes on Using This Function

1. There is no way to require the user to use this function. It is not a required function so that the application can be tailored to run on any system.
2. The buffer sizes returned represent the record sizes that are actually transmitted across the medium. For a DDM connection, the 8-byte header supplied in the **Read** and **Write Structured Fields** data buffer is stripped off and 1 byte containing the structured field AID value is prefixed. The application should compare the size of the actual data in the data buffer (which does not include the 8-byte header) with the buffer sizes returned by the **Query Communications Buffer Size** minus 1 byte. For destination/origin connections, the 8-byte header supplied in the **Read** and **Write Structured Fields** data buffer is stripped off and 9 bytes are then prefixed to the data. The application should compare the size of the actual data in the data buffer (which does not include the 8-byte header) with the buffer size returned by the **Query Communications Buffer Size** minus 9 bytes.
3. The maximum buffer sizes returned represent the maximum number of bytes supported by the workstation hardware and by the emulator. The maximum buffer size can be used only if the host is also configured to accept at least these maximum sizes.
4. The optimum buffer sizes returned represent the optimum number of bytes supported by the both the workstation hardware and the emulator. Some network configurations might set transmission limits smaller than these values. In these cases, the data transfer buffer size override value in the emulator configuration profile will be used for structured field support. The **Query Communications Buffer Size** will reflect any buffer size override values entered in the emulator configuration profile.

Query Communication Event (81)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Query Communication Event** function lets the EHLLAPI program determine whether any communication events have occurred.

Prerequisite Calls

Start Communication Notification (80)

Call Parameters

	Enhanced Interface
Function Number	Must be 81
Data String	1-character short name of the host presentation space or a blank or null indicating request for updates to the host-connected presentation space
Length	4 is implied
PS Position	NA

The calling data structure contains these elements:

Byte	**Definition**
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	**Definition**
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered
21	The indicated PSID was connected
22	The Indicated PSID was disconnected

Query Cursor Location (7)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Query Cursor Location** function indicates the position of the cursor in the host-connected presentation space by returning the cursor position.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 7	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

This function returns a length and a return code.

Length:

Host presentation space position of the cursor.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Cursor Location** function was successful.
1	Your program is not currently connected to a host session.
9	A system error was encountered.

Query Field Attribute (14)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Field Attribute** function returns the attribute byte of the field containing the input host presentation space position. This information is returned in the returned length parameter.

For the PC/3270, note also that:

- The returned length parameter is set to 0 if the screen is unformatted.
- Attribute bytes are equal to or greater than hex C0.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 14.	
Data String	NA.	
Length	NA.	
PS Position	Identifies the target. This can be the PS position of any byte within the target field.	

Return Parameters

This function returns a length and a return code.

Length:

The attribute value if the screen is formatted, or 0 if the screen is unformatted.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Field Attribute** was successful.
1	Your program is not currently connected to a host session.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	Attribute byte not found or unformatted host presentation space.

Notes on Using This Function

The returned field attributes are defined in the following tables. The bit positions are in IBM format with bit 0 as the left most bit in the byte.

- 3270 field attribute:

Bit Position	Meaning
0–1	Both = 1, field attribute byte
2	Unprotected/protected 0 = Unprotected data field 1 = Protected field

Bit Position	Meaning
3	A/N 0 = Alphanumeric data 1 = Numeric data only
4–5	I/SPD 00 = Normal intensity, pen not detectable 01 = Normal intensity, pen detectable 10 = High intensity, pen detectable 11 = Nondisplay, pen not detectable
6	Reserved
7	MDT 0 = Field has not been modified 1 = Field has been modified

- 5250 field attributes:

Bit Position	Meaning
0	Field attribute flag 0 = Nonfield attribute flag 1 = Field attribute flag
1	Visibility 0 = Nondisplay 1 = Display
2	Unprotected/protected 0 = Unprotected data field 1 = Protected field
3	Intensity 0 = Normal intensity 1 = High intensity
4–6	Field type 000 = Alphanumeric data: All characters are available 001 = Alphabet only: Uppercase and lowercase, comma, period, hyphen, blank, or Dup key are available 010 = Numeric shift: Automatic shift for number

Bit Position	Meaning
	011 = Numeric data only: 0–9, comma, period, plus, minus, blank, or Dup key are available 101 = Numeric data only: 0–9, or Dup key are available 110 = Magnetic stripe reading device data only 111 = Signed-numeric data: 0–9, plus, minus, or Dup key are available
7	MDT 0 = Field has not been modified 1 = Field has been modified

Query Host Update (24)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Host Update** function lets the programmed operator determine if the host has updated the host presentation space or OIA because:

- The **Start Host Notification** (23) function was called (on first call to the **Query Host Update** function only)
- The previous call to the **Query Host Update** function (for all calls to the **Query Host Update** function except the first).

Prerequisite Calls

Start Host Notification (23)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 24	
Data String	1-character short name of the host presentation space, or a blank or null indicating request for updates to host-connected presentation space	
Length	1 is implied	4 is implied
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

Return Code	Definition
0	No updates have been made since the last call.
1	An incorrect host presentation space was specified.
8	No prior **Start Host Notification** (23) function was called for the host presentation space ID.
9	A system error was encountered.
21	The OIA was updated.
22	The presentation space was updated.
23	Both the OIA and the host presentation space were updated.
44	Printing has completed in the printer session.

Notes on Using This Function

The target presentation space must be specified in the data string, even though a connection to the host presentation space is not necessary to check for updates.

Query Session Status (22)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Session Status** function is used to obtain session-specific information.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	16-bit	32-bit
Function Number	Must be 22.	
Data String	An 18/20-byte string consisting of a 1-byte short name of the target presentation space plus 17 bytes for returned data. Position 1 can be filled with: 1. A blank or a null to indicate a request for the host_connected presentation space. 2. An * (asterisk) to indicate a request for the keyboard-owner presentation space.	
Length	Must be 18	Must be 20

	16-bit	**32-bit**
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2–9	5–12	Session long name (same as profile name; or, if profile not set, same as short name)
10	13	**Session** **Type** **D** 3270 display **E** 3270 printer **F** 5250 display **G** 5250 printer **H** ASCII VT
11	14	Session characteristics expressed by a binary number including the following session-characteristics bits **Bit 0** EAB 0: Session has the basic attribute. 1: Session has the extended attribute **Bit 1** PSS 0: Session does not support the programmed symbols 1: Session supports the programmed symbols **Bits 2–7** Reserved

Byte		Definition
12–13	15–16	Number of rows in the host presentation space, expressed as a binary number
14–15	17–18	Number of columns in the host presentation space, expressed as a binary number
16–17	19–20	Host code page expressed as a binary number
18		Reserved

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Session Status** function was successful.
1	An incorrect host presentation space was specified.
2	An incorrect string length was made.
9	A system error was encountered.

Notes on Using This Function

1. To use this function, preallocate memory to receive the returned data string parameter. The statements required to preallocate this memory vary depending on the language in which your application is written. See Memory Allocation on page 515 for more information.

Query Sessions (10)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Sessions** function returns a 16-byte (12-byte for standard interface) data string describing each host session.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

Function	Description	
	Standard Interface	Enhanced Interface
Function Number	Must be 10	

Function	Description	
Data String	Preallocated string of 16*n* bytes long (12*n* for 16-bit) (*n* =number of sessions) or more	
Length	12*n* bytes	16*n* bytes
PS Position	NA	

Note: When the length is not matched to the number of sessions, the return code is 2.

Return Parameters

This function returns a data string, a length, and a return code.

Data String:

The returned data string is 16*n* bytes long (12*n* for standard interface), where *n* is the number of host sessions. The descriptors are concatenated into the data string and each session type, and presentation space size of a host session.

The format of each 16-byte (12-byte for standard interface) session descriptor is as follows:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2–9	5–12	Session long name (same as profile name; or, if profile not set, same as short name)
10	13	Connection type H=host
	14	Reserved
11–12	15–16	Host presentation space size (this is a binary number and is not in display format). If the session type is a print session, the value is 0.

Length:

The number of host sessions started.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Sessions** function was successful.
2	An incorrect string length was made.
9	A system error was encountered.

Notes on Using This Function

1. If an application program receives RC=2 or RC=0, the number of the active sessions is returned in the length field. The application program can recognize the minimum string length by this number.
2. The **Query Sessions** function is affected by the CFGSIZE/NOCFGZISE session option (see item 16 on page 650 for more information) and by the EXTEND_PS/NOEXTEND_PS option (see item 20 on page 651 for more information).

Note:

1. When NOCFGSIZE is set in **Set Session Parameters** (9) for a 5250 session, the value of presentation space size returned in byte position 11 and 12 from **Query Sessions**(10) will be changed in accordance with the selection of EXTEND_PS or NOEXTEND_PS.
2. When EXTEND_PS is set in **Set Session Parameters** (9), presentation space size returned from **Query Sessions** (10) will include the size of the message line, if it exists.
3. When NOEXTEND_PS is set, the value will not change regardless of the existence of a message line. In the case of 25 row, 80 column presentation space, the value can be 1920 or 2000.

Query System (20)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query System** function can be used by an EHLLAPI application program to determine the level of Z and I Emulator for Windows support and other system-related values. This function returns a string that contains the appropriate system data. Most of this information is for use by a service coordinator when you call the IBM Support Center after receiving a return code 9   (a system error was encountered).

The bytes in this returned string are defined in Return Parameters on page 617.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 20	
Data String	Preallocated string of 35 bytes	36 bytes
Length	Must be 35	Must be 36

	Standard Interface	Enhanced Interface
PS Position	NA	

Return Parameters

This function returns a data string and a return code.

Data String:

A data string of 35 bytes (for 16–bit) or 36 bytes (for 32–bit) is returned. The bytes are defined as follows:

Byte		Definition
Standard	Enhanced	
1	1	EHLLAPI version number
2–3	2–3	EHLLAPI level number
4–9	4–9	Reserved
10–12	10–12	Reserved
13	13	Hardware base, U=Unable to determine
14	14	Program type, where P=HCL Z and I Emulator for Windows
15–16	15–16	Reserved
17–18	17–18	Z and I Emulator for Windows version/level as a 2-byte ASCII value
19	19	Reserved
20–23	20–23	Reserved
24–27	24–27	Reserved
28–29	28–29	Reserved
	30	Reserved
30–31	31–32	NLS type expressed as a 2-byte binary number
33–35	34–36	Reserved

Return Code

The following codes are defined:

Return Code	Explanation
0	The **Query System** function was successful; data string has been returned.
1	EHLLAPI is not loaded. (PC/3270 only)
2	An incorrect string length was specified. (PC/3270 only)
9	A system error was encountered.

Notes on Using This Function

To use this function, preallocate memory to receive the returned data string parameter. See Memory Allocation on page 515 for more information.

Query Window Coordinates (103)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Query Window Coordinates** function requests the coordinates for the window of a presentation space. The window coordinates are returned in pels.

Note: (0,0) indicates the top-left of the window.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 103	
Data String	1-character short session ID of the host presentation space	
Length	17 is implied	20 is implied
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a function call for the current connection presentation space
	2–4	Reserved
2-17	5–20	Reserved

Return Parameters

This function returns a data string and a return code.

Byte		**Definition**
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short session ID • A blank or null indicating a function call for the current connection presentation space
	2–4	Reserved
2–17	5–20	Four 32-bit unsigned integers that return:
2–5	5–8	XLeft Long integer in pels of the left X coordinate of the rectangular window relative to the desktop window
6–9	9–12	YBottom Long integer in pels of the bottom Y coordinate of the rectangular window relative to the desktop window
10–13	13–15	XRight Long integer in pels of the right X coordinate of the rectangular window relative to the desktop window
14–17	16–20	YTop Long integer in pels of the top Y coordinate of the rectangular window relative to the desktop window

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Query Window Coordinates** function was successful.
1	Your program was not currently connected to the host session.
9	A system error occurred.
12	The session stopped.

Read Structured Fields (126)

3270	*5250*	*VT*
Yes	No	No

The **Read Structured Fields** function allows an application to read structured field data from the host application. If the call specifies S (for Synchronous), the application does not receive control until the **Read Structured Fields** is completed. If the call specifies A (for Asynchronous), the application receives control immediately after the call. If the call specifies M (for Asynchronous, message mode), the application receives control immediately after the call. The application can wait for the message. In any case (S, A, or M), the application provides the buffer address in which the data from the host is to be placed.

For a successful asynchronous completion of this function, the following statements apply:

The return code field in the parameter list might not contain the results of the requested I/O. If the return code is not 0, the request failed. The application must take the appropriate action based on the return code.

If the return code for this request is 0, the application must use the request ID returned with this function call to issue the **Get Request Completion** function call to determine the completion results of the function associated with the request ID. The **Get Request Completion** function call returns the following information:

1. Function request ID
2. Address of the data string from the asynchronous request
3. Length of the data string
4. Return code of the completed function

Prerequisite Calls

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

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 126	
Data String	See the following table	
Length	8, 10 or 14	20
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–4	Reserved.
2	5	S or A or M **S =** Synchronous. Control is not returned to the application until the read is satisfied. **A =** Asynchronous. Control is returned immediately to the application, can wait for the event object. **M =** Asynchronous. Control is returned immediately to the application, can wait for the message.
	6	Reserved.

Byte		Definition
3–4	7–8	2-byte destination/origin ID.
5–8	9–12	4-byte address of the buffer into which the data is to be read. The buffer must be obtained using the **Allocate Communications Buffer** (123) function.
9–10	13–16	Reserved.
11–12	17–20	When M is specified in position 2 the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage ("PCSHLL")(not equal 0).
13–14		The data in these positions is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous) is specified in position 5, (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
9–10	13–14	2-byte function request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15–16	Reserved.
	17–20	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. When the event object is cleared, the application must issue the **Get Request Completion** (125) function call (32-bit only).

Note: A event object address is returned for each successful asynchronous request. The event object should not be used again. A new event object is returned for each request and is valid for only the duration of that request.

Data String:

If "M" (asynchronous message mode) is specified in position 5 (2 for 16-bit applications) and the function is completed successfully, the following data string is returned:

Byte		Definition
9–10	13–14	A 2-byte function request ID. It is used by the **Get Request Completion (125)** function to determine the completion of this function call.
	15–16	Reserved.
11–12	17–18	Task ID of asynchronous message mode.
	19–20	Reserved.

Note: If the function is completed successfully, an application window receive a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains Task ID returned by the function call. The HIWORD of lParam parameter contains Return Code 0, which shows the function was successful, and LOWORD of lParam parameter contains function number 126.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Read Structured Fields** function was successful.
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).
35	Request rejected. An outbound transmission from the host was canceled.
36	Request rejected. Lost contact with the host.
37	The function was successful, but the host is inbound disabled.

Notes on Using This Function

1. Return code 35 will be returned when the first **Read Structured Fields** or **Write Structured Fields** is requested after an outbound transmission from the host is canceled. Corrective action is the responsibility of the application.
2. Return code 36 requires that the application disconnect from the emulation program and then reconnect to reestablish communication with the host. Corrective action is the responsibility of the application.
3. Return code 37 will be returned if the host is inbound disabled. The **Read Structured Fields** function was successfully requested.
4. The EHLLAPI allows for a maximum of 20 asynchronous requests per application to be outstanding. A return code for unavailable resources (RC=11) is returned if more than 20 asynchronous requests are attempted.

The structured field data contains the application structured fields received from the host. Structured field headers are removed by the EHLLAPI before the structured field data reaches the application.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'.
2	1 word	m (message length: The number of bytes of data in the message, the number does not include the buffer header prefix, which contains 8 bytes). This value is returned by EHLLAPI.
4	1 word	n (buffer size: the supplied length of the data buffer that does include the 8-byte message header). This value must be set by the application.
6	1 word	X'C000'.
8	8 bytes	Length of the first (or only) structured field message.
10	1 byte	First nonlength byte of the structured field message.
		:
m+7	1 byte	Last byte in the structured field message.

Bytes 0 through 7 are the buffer header. These first 8 bytes are used by the emulation program. The user section of the buffer begins with offset 8. Bytes 8 and 9 contain the number of bytes in the first structured field (a structured field message can contain multiple structured fields), including 2 bytes for bytes 8 and 9. Bytes 8 through *m*+7 are used for the structured field message received from the host (which could contain multiple structured fields).

The using application must furnish the complete buffer with the word at offset 0 set to zero. The buffer length must be in the word at offset 4. The word at offset 6 must be X'C000'. The emulation program will place the data message beginning at offset 8 and place the length of the message in the word at offset 2. The buffer length is not disturbed by EHLLAPI.

Synchronous Requests

When **Read Structured Fields** is requested synchronously (the S option in the data string), control is returned to the application only after the request is satisfied. The application can assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

Asynchronous Requests

When **Read Structured Fields** is requested asynchronously (the A option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

When requested asynchronously, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13–14 (9–10 for standard interface) of the data string
- The address of a event object in positions 17—20 of the data string

These are used to complete the asynchronous **Read Structured Fields** call.

The following steps must be completed to determine the outcome of an asynchronous **Read Structured Fields** function call:

- If the EHLLAPI return code is not zero, the request failed. No asynchronous request has been made. The application must take appropriate actions before attempting the call again.
- If the return code is zero, the application should wait until the event object is in the signaled state by using the **Get Request Completion** (125) function or **Wait For Single Object**. The event object should not be reused. The event object is valid only for the duration of the **Read Structured Fields** function call through the completion of the **Get Request Completion** (125) function call.
- Once the event object is in the signaled state, use the returned 16-bit Request ID as the Request ID parameter in a call to the **Get Request Completion** (125) function. The data string returned from the **Get Request Completion** (125) function call contains the final return code of the **Read Structured Fields** function call.

When **Read Structured Fields** is requested asynchronously (the M option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Read Structured Fields** request.

When requested asynchronously with the M option, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13–14 (9–10 for standard interface) of the data string
- Task ID of asynchronous message mode in positions 17–18 (11–12 for standard interface) of the data string.

These are used to complete the asynchronous **Read Structured Fields** call.

Receive File (91)

3270	***5250***	***VT***
Yes	Yes	No

The **Receive File** function is used to transfer a file from the host session to the workstation session. It is used the same way as the RECEIVE command is used in the PC/3270. The **Receive File** function can be called by an EHLLAPI application program.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 91.	
Data String	Refer to the examples.	
Length	Length, in number of bytes, of the data string. Overridden if in EOT mode.	

Following are examples of the data strings for a single-byte character set (SBSC):

3270 Session

- To receive the file from the VM/CMS host system:

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

- To receive the file from the MVS™/TSO host system:

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

- To receive the file from the CICS® host system:

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

5250 Session

- To receive the file from the iSeries™, eServer™ i5, or System i5™ host system:

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

Return Parameters

Return Code	Explanation
2	Parameter error or you have specified a length that is too long (more than 255 bytes) for the EHLLAPI buffer. The file transfer was unsuccessful.
3	File transfer complete.
4	File transfer complete with segmented records.
9	A system error was encountered.
27	File transfer terminated because of either a Cancel button or the timeout set by the **Set Session Parameter** (9) function.
101	File transfer was successful (transfer to/from CICS®).

If you receive return code 2 or 9, there is a problem with the system or with the way you specified your data string.

Other return codes can also be received, which relate to message numbers generated by the host transfer program. For transfers to a CICS® host transfer program, subtract 100 from the return code to give you the numeric portion of the message. For example, a return code of 101 would mean that the message number INW0001 was issued by the host. For other host transfer programs, just use the return code as the numerical part of the message. For example, a return of 34 would mean that message TRANS34 was issued by the host transfer program. The documentation for your host transfer program should give more information about the meanings of the specific messages.

Operating system error codes reported by EHLLAPI are greater than 300. To determine the error code, subtract 300 and refer to the operating system documentation for return codes.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the STRLEN/STREOT, EOT=c, QUIET/NOQUIET and the TIMEOUT=c/TIMEOUT=0 session options. See items 1 on page 645 and 2 on page 645 and items 7 on page 646 and 8 on page 647 for more information.
2. If no path is specified when the **Receive File** function is executed, the received file is stored in the current subdirectory, which is the directory in which your application is running.

Release (12)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Release** function unlocks the keyboard that is associated with the host presentation space reserved using the **Reserve** (11) function.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 12	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	**Explanation**
0	The **Release** function was successful.
1	Your program is not connected to a host session.
9	A system error was encountered.

Notes on Using This Function

If you do not **Release** a host presentation space reserved by using the **Reserve** (11) function, you are locked out of that session until you call the **Reset System** (21) function, you call the **Disconnect Presentation Space** (2) function, or you terminate the EHLLAPI application program.

Reserve (11)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Reserve** function locks the keyboard that is associated with the host-connected presentation space to block input from the terminal operator.

The reserved host presentation space remains locked until one of the following occurs:

- **Connect** (1) function is executed to a new session.
- **Disconnect Presentation Space** (2) function is executed.
- **Release** (12) function is executed.
- **Reset System** (21) function is executed.
- **Start Keystroke Intercept** (50) function is executed.
- EHLLAPI application program is terminated.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 11	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	**Explanation**
0	The **Reserve** function was successful.
1	Your program is not connected to a host session.
5	Presentation space cannot be used.

Return Code	Explanation
9	A system error was encountered.

Notes on Using This Function

1. If your EHLLAPI application program is sending a series of transactions to the host, you might need to prevent the user from gaining access to that session until your application processing is complete.
2. The keyboard input that a user makes while the keyboard is locked by this function is enqueued and processed after the session is terminated.
3. This function locks both the mouse and the keyboard input. The application program must unlock the presentation space to enable either the mouse or the keyboard input.

Reset System (21)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Reset System** function reinitializes EHLLAPI to its starting state. The session parameter options are reset to their defaults. Event notification is stopped. The reserved host session is released. The host presentation space is disconnected. Keystroke intercept is disabled.

You can use the **Reset System** function during initialization or at program termination to reset the system to a known initial condition.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 21	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The **Reset System** function was successful.
1	EHLLAPI is not loaded.
9	A system error was encountered.

Notes on Using this Function

For the PC/3270, this function can be used to check whether EHLLAPI is loaded. Place a call to this function at the start of your application and check for a return code of 1.

Search Field (30)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Search Field** function examines a field within the connected host presentation space for the occurrence of a specified string. If the target string is found, this function returns the decimal position of the string numbered from the beginning of the host presentation space. (For example, in a 24-row by 80-column presentation space, the row 1, column 1 position is numbered *1* and the row 5, column 1 position is numbered *321*.)

This function can be used to search either protected or unprotected fields, but only in a *field-formatted* host presentation space.

Note: If the field at the end of the host presentation space wraps, wrapping occurs when the end of the presentation space is reached.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 30.	
Data String	Target string for search.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Identifies the target field. For `SRCHALL`, this can be the PS position of any byte within the target field. For `SRCHFROM`, it is the beginning point of the search for `SRCHFRWD` or the ending point of the search for `SRCHBKWD`. See note 3 on page 630.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	Explanation
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Search Field** function was successful.
1	Your program is not connected to a host session.
2	Parameter error. Either the string length was zero, or EOT mode was specified but no EOT character was found in calling data string.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found, or the host presentation space was unformatted.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and the EOT=c session options. See items 1 on page 645 through 4 on page 646 for more information.
2. You can use the **Set Session Parameters** (9) function to determine whether your searches proceed forward (SRCHFRWD) or backward (SRCHBKWD) in a field.
3. The **Search Field** function normally checks the entire field (SRCHALL default mode). However, you can use the function 9 to specify SRCHFROM. In this mode, the calling PS position parameter does more than identify the target field. It also provides a beginning or ending point for the search.
    - If the SRCHFRWD option is in effect, the search for the designated string begins at the specified PS position and proceeds toward the end of the field.
    - If the SRCHBKWD option is in effect, the search for the designated string begins at the end of the field and proceeds backward toward the specified PS position. If the target string is not found, the search ends at the PS position specified in the calling PS position parameter.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Search Presentation Space (6)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Search Presentation Space** function lets your EHLLAPI program examine the host presentation space for the occurrence of a specified string.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 6.	
Data String	Target string for search.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Position within the host presentation space where the search is to begin (SRCHFRWD option) or to end (SRCHBKWD option). Overridden in SRCHALL (default) mode.	

Return Parameters

This function returns a length and a return code.

Length:

The following codes are defined:

Length	**Explanation**
= 0	The string was not found.
> 0	The string was found at the indicated host presentation space position.

Return Code:

The following codes are defined:

Return Code	**Explanation**
0	The **Search Presentation Space** function was successful.
1	Your program is not connected to a host session.

Return Code	Explanation
2	An error was made in specifying parameters.
7	The host presentation space position is not valid.
9	A system error was encountered.
24	The search string was not found.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the SRCHALL/SRCHFROM, STRLEN/STREOT, SRCHFRWD/SRCHBKWD, and the EOT=c session options. See items 1 on page 645 through 4 on page 646 through for more information.
2. You can use the **Set Session Parameters** (9) function to specify SRCHBKWD. When this option is in effect, the search operation locates the *last* occurrence of the string.
3. The **Search Presentation Space** function normally checks the entire host presentation space. However, you can use the **Set Session Parameters** (9) function to specify SRCHFROM. In this mode, the calling PS position parameter specifies a beginning or ending point for the search.
   - If the SRCHFRWD option is in effect, the search for the designated string begins at the specified PS position and proceeds toward the end of the host presentation space.
   - If the SRCHBKWD option is in effect, the search for the designated string begins at the end of the PS and proceeds backward toward the specified PS position. If the target string is not found, the search ends at the PS position specified in the calling PS position parameter.
4. The SRCHFROM option is also useful if you are looking for a keyword that might occur more than once in the host presentation space.
5. The **Search Presentation Space** function is useful in determining when the host presentation space is available. If your EHLLAPI application is expecting a specific prompt or message before sending data, the **Search Presentation Space** function allows you to check for a prompt message before continuing.

Note: 5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, or on the status bar. For information to be displayed on the status bar, the status bar must be configured. Refer to *Quick Beginnings* for information on configuring the status bar. By the **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.

Send File (90)

3270	*5250*	*VT*
Yes	Yes	No

The **Send File** function is used to transfer a file from the workstation session where EHLLAPI is running to a host session.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 90.	
Data String	Refer to the examples.	
Length	Length of the target data string. Overridden in EOT mode.	
PS Position	Must be 0.	

Following are examples of the data strings for SBCS

3270 Session

- To send the file to the VM/CMS host system:

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

- To send the file to the MVS/TSO host system:

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

- To send the file to the CICS host system:

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

5250 Session

- To send the file to the iSeries™, eServer™ i5, or System i5™ host system:

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

Return Parameters

Return Code	Explanation
2	Parameter error or you have specified a length that is too long (more than 255 bytes) for the EHLLAPI buffer. The file transfer was unsuccessful.
3	File transfer complete.
4	File transfer complete with segmented records.
5	Workstation file name is not valid or not found. File transfer was canceled.
9	A system error was encountered.
27	File transfer terminated because of either a Cancel button or the timeout set by the **Set Session Parameter** (9) function.
101	File transfer was successful (transfer to/from CICS).

If you receive return code 2 or 9, there is a problem with the system or with the way you specified your data string.

Other return codes can also be received which relate to message numbers generated by the host transfer program. For transfers to a CICS host transfer program, subtract 100 from the return code to give you the numeric portion of the message. For example, a return code of 101 would mean that the message number INW0001 was issued by the host. For other host transfer programs, just use the return code as the numerical part of the message. For example, a return of 34 would mean that message TRANS34 was issued by the host transfer program. The documentation for your host transfer program should give more information about the meanings of the specific messages.

Operating system error codes reported by EHLLAPI are greater than 300. To determine the error code, subtract 300 and refer to the operating system documentation for return codes.

Notes on Using This Function

1. Four sets of parameters under the **Set Session Parameters** (9) function are related to this function. They are the QUIET/NOQUIET, STRLEN/STREOT, TIMEOUT=c/TIMEOUT=0, and the EOT=c session options. See items 1 on page 645 and 2 on page 645 plus items 7 on page 646 and 8 on page 647 for more information.

Send Key (3)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Send Key** function is used to send either a keystroke or a string of keystrokes to the host presentation space.

You define the string of keystrokes to be sent with the calling data string parameter. The keystrokes appear to the target session as though they were entered by the terminal operator. You can also send all attention identifier (AID) keys such as Enter and so on. All host fields that are input protected or are numeric only must be treated accordingly.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 3.	
Data String	A string of keystrokes, maximum 255. Uppercase and lowercase ASCII characters are represented literally. Function keys and shifted function keys are represented by mnemonics. See Keyboard Mnemonics on page 636.	
Length	Length of the source data string. Overridden if in EOT mode.	
PS Position	NA	

Return Parameters

Return Code	Explanation
0	The keystrokes were sent; status is normal.
1	Your program is not connected to a host session.
2	An incorrect parameter was passed to EHLLAPI.
4	The host session was busy; all of the keystrokes could not be sent.
5	Input to the target session was inhibited or rejected; all of the keystrokes could not be sent.
9	A system error was encountered.

Notes on Using This Function

1. The parameters under the **Set Session Parameters** (9) function are related to this function. They are the AUTORESET/NORESET, STRLEN/STREOT, EOT=c, ESC=c, and RETRY/NORETRY session options. See items 1 on page 645 and 2 on page 645, 9 on page 648 and 10 on page 648, and 19 on page 651 for more information.
2. Keystrokes cannot be sent to the host session when the keyboard is locked or busy. You can check this condition with the **Wait** (4) function.
3. If the host is busy, input might be rejected.
4. The length of the data string must be explicitly defined by the default length parameter, but it can be defined implicitly by the EOT=c option of the **Set Session Parameters** (9) function.

   When explicitly defining length (see item 1), the value for the length parameter passed by the application must be calculated. For this calculation, allow 2 bytes for compound keystrokes such as `@E` and allow 4 bytes for compound keystrokes such as `@A@C`.
5. To send special control keys, a compound character coding scheme is used. In this coding scheme, one keystroke is represented by a sequence of two to four ASCII characters. The first and third character are always the escape character. The second and fourth character are always a keycode.

   To send the sequence `LOGON ABCDE` followed by the Enter key, you would code the string `LOGON ABCDE@E`. A complete list of these keycodes is represented in Keyboard Mnemonics on page 636.

   This compound coding technique allows an ASCII string representation of all necessary keystroke codes without requiring the use of complex hexadecimal key codes.

   The default escape character is `@`. The value of the escape character can be changed to any other character with the ESC=c option of the **Set Session Parameters** (9) function.
6. Users needing higher levels of performance should use the **Copy String to Field** (33) or **Copy String to Presentation Space** (15) function rather than send keystrokes with the **Send Key** (3) function. But remember, only the **Send Key** (3) function can send the special control keys.
7. Refer to **Set Session Parameters (9) on page 643** session option 10 on page 648 (NORESET option) to improve the performance of this function.

Unless NORESET is required, the reset mnemonic is added to the keystroke strings as a prefix. Therefore, all resettable status except input inhibit are reset.

The NORESET option is not the same as the **Reset System** (21) function.

8. The keystroke strings, including the AID key, are sent to the host via multiple paths. Each path sends the strings before the first AID key (or including the AID key). EHLLAPI adjusts the string length and the start position of each path. For a host application program, any keystroke might be lost by the AID key process. Therefore, you should not send a keystroke list that includes plural AID keys.
9. During the `@P` (Print) or `@A@T` (Print Presentation Space) process, all requests that update the presentation space are rejected. If the presentation space is busy or the interruption request occurs during the print request, the mnemonic @A@R (Device Reset – Cancel to print the Presentation Space) cancels the request and resets the status.

Keyboard Mnemonics

The keyboard mnemonics provide the ASCII characters representing the special function keys of the keyboard in the workstation. The abbreviation codes make the mnemonics for special keys easy to remember. An alphabetic key code is used for the most common keys. For example, the **Clear** key is *C*, and the **Tab** key is *T*.

Table 71: Mnemonics with Uppercase Alphabetic Characters on page 636 shows the mnemonics using uppercase alphabetic characters:

Table 71. Mnemonics with Uppercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@B	Left Tab	Yes	Yes	No
@C	Clear	Yes	Yes	No
@D	Delete	Yes	Yes	No
@E	Enter	Yes	Yes	No
@F	Erase EOF	Yes	Yes	No
@H	Help	No	Yes	No
@I	Insert	Yes	Yes	No
@J	Jump (Set Focus)	Yes	Yes	No
@L	Cursor Left	Yes	Yes	Yes
@N	New Line	Yes	Yes	Yes
@O	Space	Yes	Yes	Yes
@P	Print	Yes	Yes	Yes
@R	Reset	Yes	Yes	No
@T	Right Tab	Yes	Yes	Yes
@U	Cursor Up	Yes	Yes	Yes
@V	Cursor Down	Yes	Yes	Yes
@Z	Cursor Right	Yes	Yes	Yes

Table 72: Mnemonics with Numbers or Lowercase Characters on page 637 shows the mnemonics using a number or lowercase alphabetic characters.

Table 72. Mnemonics with Numbers or Lowercase Characters

Mnemonic	Meaning	3270	5250	VT
@0	Home	Yes	Yes	No
@1	PF1/F1	Yes	Yes	No
@2	PF2/F2	Yes	Yes	No
@3	PF3/F3	Yes	Yes	No
@4	PF4/F4	Yes	Yes	No
@5	PF5/F5	Yes	Yes	No
@6	PF6/F6	Yes	Yes	Yes
@7	PF7/F7	Yes	Yes	Yes
@8	PF8/F8	Yes	Yes	Yes
@9	PF9/F9	Yes	Yes	Yes
@a	PF10/F10	Yes	Yes	Yes
@b	PF11/F11	Yes	Yes	Yes
@c	PF12/F12	Yes	Yes	Yes
@d	PF13	Yes	Yes	Yes
@e	PF14	Yes	Yes	Yes
@f	PF15	Yes	Yes	Yes
@g	PF16	Yes	Yes	Yes
@h	PF17	Yes	Yes	Yes
@i	PF18	Yes	Yes	Yes
@j	PF19	Yes	Yes	Yes
@k	PF20	Yes	Yes	Yes
@l	PF21	Yes	Yes	No
@m	PF22	Yes	Yes	No
@n	PF23	Yes	Yes	No
@o	PF24	Yes	Yes	No
@q	End	Yes	Yes	No
@u	Page Up	No	Yes	No
@v	Page Down	No	Yes	No
@x	PA1	Yes	Yes	No
@y	PA2	Yes	Yes	No
@z	PA3	Yes	Yes	No

Table 73: Mnemonics with @A and @ Uppercase Alphabetic Characters on page 638 shows the mnemonics using the combination @A and @alphabetic uppercase (A–Z) key.

Table 73. Mnemonics with @A and @ Uppercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@A@C	Test	No	Yes	No
@A@D	Word Delete	Yes	Yes	No
@A@E	Field Exit	Yes	Yes	No
@A@F	Erase Input	Yes	Yes	No
@A@H	System Request	Yes	Yes	No
@A@I	Insert Toggle	Yes	Yes	No
@A@J	Cursor Select	Yes	Yes	No
@A@L	Cursor Left Fast	Yes	Yes	No
@A@Q	Attention	Yes	Yes	No
@A@R	Device Cancel (Cancels Print Presentation Space)	Yes	Yes	No
@A@T	Print Presentation Space	Yes	Yes	Yes
@A@U	Cursor Up Fast	Yes	Yes	No
@A@V	Cursor Down Fast	Yes	Yes	No
@A@Z	Cursor Right Fast	Yes	Yes	No

Table 74: Mnemonics with @A and @ Lowercase Alphabetic Characters on page 638 shows the mnemonics using the combination @A and @number or @A and @alphabetic lowercase (a–z) key.

Table 74. Mnemonics with @A and @ Lowercase Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@A@9	Reverse Video	Yes	Yes	No
@A@b	Underscore	Yes	No	No
@A@c	Reset Reverse Video	Yes	No	No
@A@d	Red	Yes	No	No
@A@e	Pink	Yes	No	No
@A@f	Green	Yes	No	No
@A@g	Yellow	Yes	No	No
@A@h	Blue	Yes	No	No
@A@i	Turquoise	Yes	No	No
@A@j	White	Yes	No	No
@A@l	Reset Host Colors	Yes	No	No
@A@t	Print (Personal Computer)	Yes	Yes	No
@A@y	Forward Word Tab	Yes	Yes	No
@A@z	Backward Word Tab	Yes	Yes	No

Table 75: Mnemonics with @A and @ Alphanumeric (Special) Characters on page 639 shows the mnemonics using the combination @A and @special character.

Table 75. Mnemonics with @A and @ Alphanumeric (Special) Characters

Mnemonic	Meaning	3270	5250	VT
@A@-	Field -	No	Yes	No
@A@+	Field +	No	Yes	No
@A@<	Record Backspace	No	Yes	No

Table 76: Mnemonics with @S (Shift), @W (Edit) and @ Alphabetic Characters on page 639 shows the mnemonics using the combination @S , @W, and @alphabetic lowercase.

Table 76. Mnemonics with @S (Shift), @W (Edit) and @ Alphabetic Characters

Mnemonic	Meaning	3270	5250	VT
@S@E	Print Presentation Space on Host	No	Yes	No
@S@x	Dup	Yes	Yes	No
@S@y	Field Mark	Yes	Yes	No
@W@C	Edit Copy	Yes	Yes	Yes
@W@D	Edit Clear	Yes	Yes	Yes
@W@E	Edit Copy Append	Yes	Yes	Yes
@W@L	Edit Copy Link	Yes	Yes	Yes
@W@N	Edit Paste Next	Yes	Yes	Yes
@W@V	Edit Paste	Yes	Yes	Yes
@W@X	Edit Cut	Yes	Yes	Yes
@W@Z	Edit Undo	Yes	Yes	Yes

Note: @W Edit mnemonics are supported only in EHLLAPI functions in Enhanced mode. See Start Keystroke Intercept function under Summary of EHLLAPI Functions on page 537.

VT Only: Table 77: Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only) on page 639 shows the mnemonics using the combination @M and @number or @alphabetic lowercase (a-z)

Table 77. Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only)

Mnemonic	Meaning	3270	5250	VT
@M@0	VT Numeric Pad 0	No	No	Yes
@M@1	VT Numeric Pad 1	No	No	Yes
@M@2	VT Numeric Pad 2	No	No	Yes
@M@3	VT Numeric Pad 3	No	No	Yes
@M@4	VT Numeric Pad 4	No	No	Yes
@M@5	VT Numeric Pad 5	No	No	Yes

Table 77. Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only) (continued)

Mnemonic	Meaning	3270	5250	VT
@M@6	VT Numeric Pad 6	No	No	Yes
@M@7	VT Numeric Pad 7	No	No	Yes
@M@8	VT Numeric Pad 8	No	No	Yes
@M@9	VT Numeric Pad 9	No	No	Yes
@M@-	VT Numeric Pad -	No	No	Yes
@M@,	VT Numeric Pad ,	No	No	Yes
@M@.	VT Numeric Pad .	No	No	Yes
@M@e	VT Numeric Pad Enter	No	No	Yes
@M@f	VT Edit Find	No	No	Yes
@M@i	VT Edit Insert	No	No	Yes
@M@r	VT Edit Remove	No	No	Yes
@M@s	VT Edit Select	No	No	Yes
@M@p	VT Edit Previous Screen	No	No	Yes
@M@n	VT Edit Next Screen	No	No	Yes
@M@a	VT PF1	No	No	Yes
@M@b	VT PF2	No	No	Yes
@M@c	VT PF3	No	No	Yes
@M@d	VT PF4	No	No	Yes
@M@h	VT HOld Screen	No	No	Yes
@M@(space)	Control Code NUL	No	No	Yes
@M@A	Control Code SOH	No	No	Yes
@M@B	Control Code STX	No	No	Yes
@M@C	Control Code ETX	No	No	Yes
@M@D	Control Code EOT	No	No	Yes
@M@E	Control Code ENQ	No	No	Yes
@M@F	Control Code ACK	No	No	Yes
@M@G	Control Code BEL	No	No	Yes
@M@H	Control Code BS	No	No	Yes
@M@I	Control Code HT	No	No	Yes
@M@J	Control Code LF	No	No	Yes
@M@K	Control Code VT	No	No	Yes
@M@L	Control Code FF	No	No	Yes
@M@M	Control Code CR	No	No	Yes
@M@N	Control Code SO	No	No	Yes
@M@O	Control Code SI	No	No	Yes
@M@P	Control Code DLE	No	No	Yes

Table 77. Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only) (continued)

Mnemonic	Meaning	3270	5250	VT
@M@Q	Control Code DC1	No	No	Yes
@M@R	Control Code DC2	No	No	Yes
@M@S	Control Code DC3	No	No	Yes
@M@T	Control Code DC4	No	No	Yes
@M@U	Control Code NAK	No	No	Yes
@M@V	Control Code SYN	No	No	Yes
@M@W	Control Code ETB	No	No	Yes
@M@X	Control Code CAN	No	No	Yes
@M@Y	Control Code EM	No	No	Yes
@M@Z	Control Code SUB	No	No	Yes
@M@u	Control Code ESC	No	No	Yes
@M@v	Control Code FS	No	No	Yes
@M@w	Control Code GS	No	No	Yes
@M@x	Control Code RS	No	No	Yes
@M@y	Control Code US	No	No	Yes
@M@z	Control Code DEL	No	No	Yes
@Q@A	VT User Defined Key 6	No	No	Yes
@Q@B	VT User Defined Key 7	No	No	Yes
@Q@C	VT User Defined Key 8	No	No	Yes
@Q@D	VT User Defined Key 9	No	No	Yes
@Q@E	VT User Defined Key 10	No	No	Yes
@Q@F	VT User Defined Key 11	No	No	Yes
@Q@G	VT User Defined Key 12	No	No	Yes
@Q@H	VT User Defined Key 13	No	No	Yes
@Q@I	VT User Defined Key 14	No	No	Yes
@Q@J	VT User Defined Key 15	No	No	Yes
@Q@K	VT User Defined Key 16	No	No	Yes
@Q@L	VT User Defined Key 17	No	No	Yes
@Q@M	VT User Defined Key 18	No	No	Yes

Table 77. Mnemonics Using @M, @Q and @Alphabetic Lowercase (For VT Only) (continued)

Mnemonic	Meaning	3270	5250	VT
@Q@N	VT User Defined Key 19	No	No	Yes
@Q@0	VT User Defined Key 20	No	No	Yes
@Q@a	VT Backtab	No	No	Yes
@Q@r	VT Clear Page	No	No	Yes
@Q@s	VT Edit	No	No	Yes

The following table shows the mnemonics using a special character.

Table 78. Mnemonics with Special Character Keys

Mnemonic	Meaning	3270	5250	VT
@@	@	Yes	Yes	Yes
@$	Alternate Cursor (The Presentation Manager® Interface only)	Yes	Yes	Yes
@<	Backspace	Yes	Yes	Yes

The following character keys are interpreted as they are.

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

Set Cursor (40)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Set Cursor** function is used to set the position of the cursor within the host presentation space. Before using the **Set Cursor** function, a workstation application must be connected to the host presentation space.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 40	
Data String	NA	
Length	NA	
PS Position	Desired cursor position in the connected host presentation space	

Return Parameters

Return Code	Explanation
0	Cursor was successfully located at the specified position.
1	Your program is not connected to a host session.
4	The session is busy.
7	A cursor location less than 1 or greater than the size of the connected host presentation space was specified.
9	A system error occurred.

Set Session Parameters (9)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Set Session Parameters** function lets you change certain default session options in EHLLAPI for all sessions. When EHLLAPI is loaded, the default settings for session options are as indicated by the underscored entries in the tables that appear in Session Options on page 645 . Any, some, or all of these settings can be changed by including the desired option in the calling data string as explained below. Specified settings remain in effect until:

- Changed by a subsequent **Set Session Parameters** (9) function that specifies a new value.
- The **Reset System** (21) function is executed.
- The EHLLAPI application program is terminated.

The following table lists those EHLLAPI functions that are affected by session options. Functions not listed in the table are not affected by any of the session options. Session options that affect each function are indicated by corresponding entries in the *"See Items"* column. These entries are indexed to the list that follows Call Parameters on page 645.

Function Number	Function Name	See Items
1	**Connect Presentation Space**	**11 on page 648**, **21 on page 651**, **22 on page 652**

Function Number	Function Name	See Items
3	Send Key	1 on page 645, 2 on page 645, 9 on page 648, 10 on page 648, 19 on page 651
4	Wait	12 on page 648
5	Copy Presentation Space	5 on page 646, 13 on page 649, 14 on page 650, 15 on page 650, 17 on page 650, 20 on page 651
6	Search Presentation Space	1 on page 645, 2 on page 645, 3 on page 646, 4 on page 646
8	Copy Presentation Space to String	5 on page 646, 13 on page 649, 14 on page 650, 15 on page 650, 17 on page 650, 20 on page 651
10	Query Sessions	16 on page 650, 20 on page 651
15	Copy String to Presentation Space	1 on page 645, 2 on page 645, 13 on page 649, 14 on page 650, 18 on page 650, 20 on page 651
18	Pause	6 on page 646
30	Search Field	1 on page 645, 2 on page 645, 3 on page 646, 4 on page 646, 20 on page 651
33	Copy String to Field	1 on page 645, 2 on page 645, 13 on page 649, 14 on page 650, 18 on page 650, 20 on page 651
34	Copy Field to String	5 on page 646, 13 on page 649, 14 on page 650, 17 on page 650, 20 on page 651
35	Copy Presentation Space to Clipboard	5 on page 646, 13 on page 649, 14 on page 650, 17 on page 650, 20 on page 651
36	Paste Clipboard to Presentation Space	1 on page 645, 2 on page 645, 13 on page 649, 14 on page 650, 18 on page 650, 20 on page 651
51	Get Key	9 on page 648, 12 on page 648
90	Send File	1 on page 645, 2 on page 645, 7 on page 646, 8 on page 647
91	Receive File	1 on page 645, 2 on page 645, 7 on page 646, 8 on page 647
101	Connect Window Services	21 on page 651, 22 on page 652

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 9.	
Data String	String containing the desired values of those session options that are to be changed. The data string can contain any of the values in the tables of Session Options on page 645. The values should be placed on the data string line, separated by commas or blanks. The sets of parameters are explained in terms of the functions they affect.	
Length	Explicit length of the source data string (the `STREOT` option is not allowed).	
PS Position	NA.	

Session Options

The following tables show the session options. The default is underlined.

1. The values in the following table determine how the data string length is defined for functions **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90), and **Receive File** (91).

Value	Explanation
STRLEN	An explicit length is passed for all strings.
STREOT	Lengths are not explicitly coded. Calling (source) data strings are terminated with an EOT character.

2. The statement in the following table is used to specify the character that is used as the end-of-text (EOT) delimiter in the calling (source) data string for EHLLAPI functions **Send Key** (3), **Search Presentation Space** (6), **Copy String to Presentation Space** (15), **Search Field** (30), **Copy String to Field** (33), **Send File** (90), and **Receive File** (91).

Value	Explanation
EOT=c	Allows you to specify the EOT character for string terminators (in STREOT mode). Binary zero is the default. Do not leave a blank after the equal sign.

To be valid, `c` must be entered as a 1-byte string literal character with no preceding blanks. The EOT character specified by this statement is used to determine the length of a calling data string only when the STREOT option (see item 1) is in effect.

3. The values in the following table affect the **Search Presentation Space** (6) and **Search Field** (30) search functions.

Value	Explanation
SRCHALL	The **Search Presentation Space** (6) function and **Search Field** (30) function scan the entire host presentation space or field.
SRCHFROM	The **Search Presentation Space** (6) function and **Search Field** (30) function start from a specified PS position (for `SRCHFRWD`) or end at a specified PS position (for `SRCHBKWD`).

4. The values in the following table affect the **Search Presentation Space** (6) and **Search Field** (30) search functions. They determine the direction for the search.

Value	Explanation
SRCHFRWD	The **Search Presentation Space** (6) function and **Search Field** (30) function perform in an ascending direction.
SRCHBKWD	The **Search Presentation Space** (6) function and **Search Field** (30) function perform in a descending direction. A search is satisfied if the first character of the requested string starts within the bounds specified for the search.

5. The values in the following table determine how attribute bytes are treated for functions **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), and **Copy Field to String** (34).

Value	Explanation
NOATTRB	Convert all unknown values to blanks.
ATTRB	Pass back all codes that do not have an ASCII equivalent as their original values.
NULLATTRB	Convert all field attributes to null characters.

6. The values in the following table affect the **Pause** (18) function.

Value	Explanation
FPAUSE	A full-duration pause lasts for however long you specified in the **Pause** (18) function.
IPAUSE	Interruptible pause. After the **Start Host Notification** (23) function is executed, a host event satisfies a pause.

7. The values in the following table determine whether messages generated by file transfer functions **Send File** (90) and **Receive File** (91) are displayed.

Value	Explanation
NOQUIET	SEND and RECEIVE messages are displayed.
QUIET	SEND and RECEIVE messages are not displayed.

8. The statements in the following table determine how long Z and I Emulator for Windows EHLLAPI waits before it automatically issues a Cancel during execution of file transfer functions **Send File** (90) and **Receive File** (91). To be valid, `c` must be a capital letter J–N and must not be preceded by a blank.

Value	Explanation
TIMEOUT=0	A Cancel is automatically issued following a 20-second (approximate) delay.
TIMEOUT=c	A Cancel is automatically issued following a specified delay. A 1-character indicator from the table below tells Z and I Emulator for Windows how many 30-second cycles it should accept before issuing a Cancel itself. **Character** **Value (in 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

Value	Explanation
	L 6.0 **M** 6.5 **N** 7.0

9. The statement in the following table is used to define the escape character for keystroke mnemonics. This session option affects functions **Send Key** (3) and **Get Key** (51). The value of `c` must be entered as a 1-byte literal character string with no preceding blanks.

Value	Explanation
ESC=c	Specifies the escape character for keystroke mnemonics (@ is the default). Do not leave a blank after the equal sign. A blank is not a valid escape character.

10. The values in the following table determine whether EHLLAPI automatically precedes strings sent using the **Send Key** (3) function with a reset.

Value	Explanation
AUTORESET	EHLLAPI attempts to reset all inhibited conditions by prefixing all strings of keys sent using the **Send Key** (3) function with a reset.
NORESET	Do not AUTORESET.

11. The values in the following table affect the manner in which the **Connect Presentation Space** (1) command function.

Value	Explanation
CONLOG	Establishes a logical connection between the workstation session and a host session. During Connect, does not jump to the requested presentation space.
CONPHYS	Establishes a physical connection between the workstation session and a host session. During Connect, jumps to the requested presentation space.

12. The values in the following table affect the **Wait** (4) function and **Get Key** (51) function. For each value, there are two different effects, one for each function.

Value	Explanation
TWAIT	For the **Wait** (4) function, waits up to a minute before timing out on XCLOCK (X []) or XSYSTEM.

Value	Explanation
	For the **Get Key** (51) function, does not return control to your EHLLAPI application program until it has intercepted a key (normal or AID key based on the option specified under the **Start Keystroke Intercept** (50) function).
LWAIT	For the **Wait** (4) function, waits until XCLOCK (X ▯)/XSYSTEM clears. This option is not recommended, because control does not return to your application until the host is available. For the **Get Key** (51) function, does not return control to your EHLLAPI application program until it has intercepted a key (normal or AID key based on the option specified under the **Start Keystroke Intercept** (50) function).
NWAIT	For the **Wait** (4) function, checks status and returns immediately (no wait). For the **Get Key** (51) function, returns return code 25 (keystrokes not available) in the fourth parameter if nothing is queued matching the option specified under the **Start Keystroke Intercept** (50) function.

Note: Use of NWAIT is recommended.

13. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34). Extended attribute bytes (EAB) include extended character attributes and extended field attributes.

Value	Explanation
NOEAB	Pass data only, no EABs.
EAB	Pass the presentation space data with extended attribute bytes. For each character that appears on the screen, 2 bytes of data are passed. Therefore, a buffer twice the size of the presentation space must be preallocated; for example 2 x 1920 = 3840 for a 24-row by 80-column presentation space. Extended attributes for a string of characters may be reported as attributes of the field byte, rather than as attributes of each individual character in the field. In this case, to tell if a particular character or set of characters on a screen is underscored, do a CopyPStoString specifying the position of the field attribute byte (the byte before the field that is displayed on the screen) to get the EAB information that applies to all of the characters in that field.

Note: When using **EHLLAPI Copy PS to String**, text is copied which should be invisible to the operator. Use the EHLLAPI Set Session Parameters function to set the NODISPLAY option to determine if there is hidden data. This causes EHLLAPI to return nondisplay fields as nulls. Another common procedure for hiding data is to set the foreground and background colors the same (BLACK, for instance) so the text is displayed, but not visible to the human operator. The only way for your application to detect this is to use the EAB and XLATE session parameters and then copying the PS. The foreground/background color of each position is returned and you can determine which characters are invisible.

14. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy String to Presentation Space** (15), **Copy String to Field** (33), and **Copy Field to String** (34).

Value	Explanation
NOXLATE	EABs are not translated.
XLATE	EABs are translated to the PC color graphics adapter (CGA) format.

15. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8) and **Copy Presentation Space to Clipboard** (35) if NOATTRB and NOEAB are specified.

Value	Explanation
BLANK	Convert all unknown values to X'20'.
NOBLANK	Convert all unknown values to X'00'.

The default value is BLANK. If you want to change the default value to NOBLANK, add the following statement in the PCSWIN.INI file located in the Z and I Emulator for Windows user-class application data directory:

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

16. The values in the following table affect the presentation space size that is returned by the **Query Sessions** (10).

Value	Explanation
CFGSIZE	Returns the configured size of the connected presentation space. This option ignores any override of the configured size by the host.
NOCFGSIZE	Returns the current size of the connected presentation space.

17. The values in the following table affect **Copy Presentation Space** (5), **Copy Presentation Space to String** (8), **Copy Field to String** (34) and **Copy Presentation Space to Clipboard** (35).

Value	Explanation
DISPLAY	Copy nondisplay fields in the presentation space to the target buffer area in the same manner as display fields. Current applications function normally.
NODISPLAY	Do not copy nondisplay fields in the presentation space to the target buffer area. Copy the nondisplay fields to the target buffer as a string of null characters. This allows applications to display the copied buffers in the presentation widow without displaying confidential information, such as passwords.

18. The values in the following table affect **Copy String to Presentation Space** (15), **Copy String to Field** (33) and **Paste Clipboard to Presentation Space** (36).

Value	Explanation
NOPUTEAB	EAB is not contained in the data string of **Copy String to Presentation Space** or **Copy String to Field**.
PUTEAB	EAB is contained with character data in the data string of **Copy String to Presentation Space** or **Copy String to Field**.

This option is used for the compatibility with Communication Manager/2. For Communication Manager/2, the data string, which is specified in **Copy String to Presentation Space** or **Copy String to Field**, must be contain EAB (or EAD) with character data when EAB (or EAD) is valid in **Set Session Parameters**. Whereas, for the previous Z and I Emulator for Windows, the data string specified in these functions must consist of character data only even if EAB (or EAD) is valid. But Z and I Emulator for Windows allows that the data string contains EAB (or EAD) by setting PUTEAB to provide the compatibility with Communication Manager/2.

19. The values in the following table affect the **Send Key** (3) function. Keystrokes are not processed if the keyboard is blocked or in use. The options determine whether the function tries to resend the keystrokes until a 4-minute timeout occurs or if the function returns immediately after determining the keyboard is blocked or in use.

Value	Explanation
RETRY	Continues to attempt to send keystrokes until they are sent or until a 4-minute timeout occurs.
NORETRY	Returns immediately after determining the keyboard is blocked or in use.

20. The values in the following table affect **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) and **Paste Clipboard to Presentation Space** (36).

Value	Explanation
EXTEND_PS	5250 emulation supports a presentation space of 24 rows by 80 columns. In some instances, Communication Manager 5250 emulation displays a 25th row. This occurs when either an error message from the host is displayed or when the operator selects the SysReq key. Z and I Emulator for Windows displays 25th row information on row 24, but EHLLAPI normally sees the *real* 24th row. By **EXTEND_PS** option, an EHLLAPI application can use the same interface with Communication Manager EHLLAPI and valid presentation space is extended when this condition occurs.
NOEXTEND_PS	The presentation space is not extended when the above condition occurs. This is the default value.

21. The values in the following table affect the **Connect Presentation Space** (1) and **Connect Window Services** (101) functions. The options specify whether an application can or will share the presentation space to which it is connected with another application. Only one of the following values can be specified with each **Set Session Parameter** call.

Value	**Explanation**
SUPER_WRITE	The application allows other applications that allow sharing and have write access permissions to concurrently connect to the same presentation space. The originating application performs supervisory-type functions but does not create errors for other applications that share the presentation space.
WRITE_SUPER	The application requires write access and allows only supervisory application to concurrently connect to its presentation space. This is the default value.
WRITE_WRITE	The application requires write access and allows partner or other applications with predictable behavior to share the presentation space.
WRITE_READ	The application requires write access and allows other applications that perform read-only functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.
WRITE_NONE	The application has exclusive use of the presentation space. No other applications are allowed to share the presentation space, including supervisory applications. The application is allowed to copy the presentation space and perform read-only operations as usual.
READ_WRITE	The application requires only read access to monitor the presentation space and allows other applications that perform read or write, or both, functions to share the presentation space. The application is also allowed to copy the presentation space and perform other read-only operations as usual.

22. The values in the following table allow applications that have presentation space sharing requirements to limit the sharing to a partner application (an application that was developed to work with it).

Value	**Explanation**
NOKEY	Allows the application to be compatible with existing applications that do not specify the **KEY** parameter.
KEY$*nnnnnnnn*	Uses a keyword to restrict sharing access to the presentation space that it supports. The keyword must be exactly 8 bytes in length.

Return Parameters

This function returns a length and a return code.

Length:

Number of valid session parameters that are set.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The session parameters have been set.
2	One or more parameters were not valid.
9	A system error was encountered.

Start Close Intercept (41)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Close Intercept** function allows the application to intercept close requests generated when a user selects the close option from the emulator session window. This function intercepts the close request and discards it until a **Stop Close Intercept** (43) function is requested.

After using this function, your application program can use the **Query Close Intercept** (42) function to determine when a close request has occurred.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

Byte	Definition	
	Standard Interface	Enhanced Interface
Function Number	Must be 41	
Data String	See the following table	
Length	5 or 6	Must be 12
PS Position	NA	

The data string contains the following items.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–4	Reserved.
4–5		The data in these positions is ignored by EHLLAPI. However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.
6	5	Specify M to request asynchronous message mode (Windows only).
	6–8	Reserved.

Byte		Definition
2–3	9–12	When M is specified in position 5 (6 for 16-bit), the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL) (not equal 0).

Return Parameters

This function returns a data string and a return code.

Data String:

If asynchronous message mode is not specified in position 5 (6 for standard interface) and the function is completed successfully, the following data string is returned.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–8	Reserved.
	9–12	4 byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. (32-bit only).

Data String:

If M (asynchronous message mode) is specified in position 5 (6 for standard interface) and the function is completed successfully, the following data string is returned.

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–8	Reserved
2–3	9–10	Task ID of asynchronous message mode

Note: If a user selects the close option, an application window receives a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter will contain the Task ID returned by this function call. The HIWORD of the lParam parameter will contain the Return Code 26, which shows a close intercept occurred, and the LOWORD of the lParam parameter will contain the function number 41.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Start Close Intercept** function was successful.
1	An incorrect host presentation space was specified.
2	A parameter error occurred.

Return Code	Explanation
9	A system error occurred.
10	The function is not supported by the emulation program.

Notes on Using This Function

1. The returned event object or semaphore is in a non-signaled state when the start request function returns. The event object is in the signaled state each time a close request occurs. To receive notification of multiple close request events, put the event object into the signaled state each time using **SetEvent** or the **Query Close Intercept** (42) function.
2. After using this function, your application program can use the **Query Close Intercept** (42) function to determine when a close request has occurred. The application can wait on the returned event object to determine when the event has occurred.
3. This is not an exclusive call. Multiple applications can request this function for the same short session ID.
4. If there are no applications intercepting close requests for a session, any subsequent close requests selected by the user from the emulator operations dialog result in a normal stop requested for that session.

Start Communication Notification (80)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Communication Notification** function begins the process by which your EHLLAPI application can determine whether the specified session is connected to a host.

After using this function, the application can use **Query Communication Event** (81) to determine whether the session is connected or disconnected.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Enhanced Interface
Function Number	Must be 80
Data String	Preallocated structure; see the following table
Length	16
PSPosition	NA

The calling data structure contains these elements

Byte	Definition

1	A 1-character presentation space short name (PSID).
2-4	Reserved
5	One of the following values: • The character C asks for notification when the session either disconnects or connects to the host. • The character A requests the asynchronous mode of notification. When A is specified, position 9-12 returns the address of an event object (Windows). The character C must be placed in position 13. • The character M requests the asynchronous message mode of the notification. When M is specified, the event selection character C must be placed in position 13.
6-8	Reserved
9-12	When M is specified in position 5, the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL)—(not zero).
13	This should contain the character C if position 5 is A or M.
14-16	Reserved

Data String

If A (asynchronous mode) is specified in position 5 of the calling data structure and the function is completed successfully, the following data string is returned:

Byte	**Definition**
1	A 1-character presentation space short-name (PSID)
2-8	Reserved
9-12	4-byte binary value in which the event object handle is returned by EHLLAPI. The application can wait for this event object.

If M (asynchronous message mode) is specified in position 5 of the calling data structure and the function is completed successfully, the following data string is returned:

Byte	**Definition**
1	A 1-character presentation space short-name (PSID)
2-8	Reserved
9-10	Task ID of asynchronous message mode

When the session connects or disconnects an application window receives a message. The message is the return value of RegisterWindow Message (PCSHLL). The wParam contains the Task ID returned by the function call. HIWORD of lParam contains a `21` if the session is connected to the host or a 22 if the session is disconnected. The LOWORD of lParam contains the function number `80`.

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
2	An error was made in designating parameters
9	A system error was encountered

Notes on using this Function

1. An application program can issue this function for multiple host sessions. The **Query Communication Event** (81) function can be used to determine the session communication status.
2. If the application chooses the asynchronous option, it can use the Windows SDK call **WaitForSingleObject** to wait until the sessions communication status has changed.
3. The event object is initially in a non-signaled state. It is signaled each time an event occurs. To receive notification for multiple events the application must put the event object into the non-signaled state each time it is signaled, by using the Windows SDK call **ResetEvent**, or by using function 81 **Query Communications Event**.
4. Multiple calls to this function with the same options from the same application will be ignored.
5. This is not exclusive to one application. Several applications can request this function for the same Session ID.

Start Host Notification (23)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Host Notification** function begins the process by which your EHLLAPI application program determines if the host presentation space or OIA have been updated.

After using this function, your application program can use the **Query Host Update** (24) function to determine when a host event has occurred.

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 23	

	Standard Interface	Enhanced Interface
Data String	Preallocated string; see the following table	
Length	6 or 7 implied	16
PS Position	NA	

The calling data string contains these elements:

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A 1-character presentation space short name (PSID) • A blank or null indicating a request for the host-connected host presentation space
	2–4	Reserved.
2	5	One of the following values: • The character B asking for notification of both host presentation space and OIA updates. • The character O asking for notification of only OIA updates. • The character P asking for notification of only host presentation space updates. • The character A requesting the asynchronous mode of the notification When A is specified, position 9–12 returns the address of an event object. The event selection character B, O, or P must be placed in position 13. • The character M requesting the asynchronous message mode of the notification. When M is specified, the event selection character B, O, or P must be placed in position 13 (7 for 16-bit). • E The character E asking for notification of completion during a printer session.
	6–8	Reserved.
3–4	9–12	When M is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL) (not equal 0).
7	13	One of the following values if position 5 (2 for 16-bit) is A or M:

Byte		Definition
		• The character B asking for notification of both host presentation space and OIA updates • The character O asking for notification of only OIA updates • The character P asking for notification of only host presentation update.
	14–16	Reserved.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous mode of notification) is specified in position 5 and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–8	Reserved.
	9–12	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object (32-bit only).

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–8	Reserved
3–4	9–10	Task ID of asynchronous message mode

Note: If OIA or presentation space is updated, an application window receives a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam contains Return Code 21 (shows the OIA is updated), 22 (shows the host presentation space is updated), or 23 (shows both the OIA and the host presentation space are updated), and LOWORD of lParam parameter contains function number 23.

Return Code:

The following codes are defined:

Return Code	Definition
0	The **Start Host Notification** function was successful.
1	An incorrect host presentation space was specified.
2	An error was made in designating parameters.
9	A system error was encountered.

Notes on Using This Function

1. An application program can issue this function for multiple host sessions. The **Pause** (18) function can notify the application when one or more host sessions (PS, OIA, or both of them) are updated. The **Query Host Update** (24) function can be used to determine whether a PS, OIA, or both of them have been updated.
2. If the application chooses the asynchronous option, it can wait for the returned event object or semaphore to determine when a host event has occurred.
3. The event object or semaphore is initially in a non-signaled state and is signaled each time an appropriate event occurs. To receive notification for multiple events, the application must put the event object into the non-signaled state each time it has been signaled using either the **ResetEvent** or the **Query Host Update** (24) function.
4. An application cannot request Start Host Notification more than once with the same options.
5. This is not an exclusive call. Multiple applications can request this function for the same short session ID.

Start Keystroke Intercept (50)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Start Keystroke Intercept** function allows a workstation application to filter any keystrokes sent to a session by a terminal operator. After a call to this function, keystrokes are intercepted and saved until the keystroke queue overflows or until the **Stop Keystroke Intercept** (53) function or **Reset System** (21) function is called. The intercepted keystrokes can be:

- Received through the **Get Key** (51) function and sent to the same or another session with the **Send Key** (3) function
- Accepted or rejected through the **Post Intercept Status** (52) function
- Replaced by other keystrokes with the **Send Key** (3) function
- Used to trigger other processes

Prerequisite Calls

There are no prerequisite calls for this function.

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 50	
Data String	See the following table	
Length	Keystroke buffer size EHLLAPI allocates 32 bytes minimum for this buffer.	
PS Position	NA	

The calling data string contains:

Byte		Definition
Standard	Enhanced	
1	1	One of the following values: • A specific host presentation space short name (PSID) • A blank or null indicating a request for the host-connected host presentation space
	2–4	Reserved.
2	5	An **option code** character: • D for AID keystrokes only. • L for all keystrokes. • E for edit keys and all keystrokes (Available in Enhanced mode only) • M for requesting the asynchronous message mode of the notification (Windows only). When M is specified, a code character D, or L, or E (Enhanced Monde) must be placed in position 13 (7 for 16-bit). Prerequisite: keyboard keys must be mapped to edit functions, e.g. Ctrl +C mapped to edit copy function. See Table 76: Mnemonics with @S (Shift), @W (Edit) and @ Alphabetic Characters on page 639 for edit functions supported.
	6–8	Reserved.
3–4	9–12	When M is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set. The message is a return value of RegisterWindowMessage (PCSHLL) (not equal 0).
7	13	One of the following values if position 5 (2 for 16-bit) is M:

Byte		Definition
		• D for AID keystrokes only. • L for all keystrokes. • E for edit keys and all keystrokes. (Available in Enhanced mode only.)
	14–16	Reserved.

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–8	Reserved
3–4	9–10	Task ID of asynchronous message mode

Note: If a user sends keystrokes to a session, an application window receives a message. The message is a return value of RegisterWindowMessge (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam parameter contains return code 0, which shows that the function was successful, and LOWORD of lParam parameter contains function number 50.

Return Parameters

Return Code	Explanation
0	The **Start Keystroke Intercept** function was successful.
1	An incorrect presentation space was specified.
2	An incorrect option was specified.
4	The execution of the function was inhibited because the target presentation space was busy.
9	A system error was encountered. Release is being used.

Notes on Using This Function

1. If a return code of 31 occurs for the **Get Key** (51) function, either:
    - Increase the value of the calling length parameter for this function, or
    - Execute the **Get Key** (51) function more frequently.

   An intercepted keystroke occupies 3 bytes in the buffer. The next intercepted keystroke is placed in the adjacent 3 bytes. When the **Get Key** (51) function retrieves a keystroke (first-in first-out, or FIFO), the 3 bytes that it occupied are made available for another keystroke. By increasing the size of the buffer or the rate at which keystrokes are retrieved from the buffer, you can eliminate buffer overflow.

In the PC/3270, another way to eliminate return code 31 is to operate the PC/3270 emulator in the resume mode.

2. If option code D is provided, EHLLAPI writes intercepted non-AID keys to the presentation space to which they were originally intended, and returns only AID keys to the application.
3. Call the **Stop Keystroke Intercept** (53) function before exiting your EHLLAPI application. Otherwise, keystroke interception remains enabled with unpredictable results.

Start Playing Macro (110)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Start Playing Macro** function invokes a macro. The macro will be executed in the connected session.

Note: This macro must exist in the Z and I Emulator for Windows user-class application data directory and no extension should be specified in the function call for the macro name.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface
Function Number	Must be 110
Data String	See the following table
Length	Length of macro name, plus 3
PS Position	NA

Byte		**Definition**
Standard	Enhanced	
1-2		Reserved
3-n		Null terminated macro name

Return Parameters

Return Code	**Explanation**
0	The **Start Playing Macro** function was successful.

Return Code	Explanation
1	The programs is not connected to a host session.
2	An error was made in specifying parameters.
9	A system error was encountered.

Stop Close Intercept (43)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Close Intercept** function allows the application to turn off the **Start Close Intercept** (41) function. After the application has issued the **Stop Close Intercept** function, subsequent close requests result in a normal stop sent to the logical terminal session.

Prerequisite Calls

Start Close Intercept (41)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 43	
Data String	1-character short session ID of the host presentation space	
Length	1	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

Return Code	Explanation
0	The **Stop Close Intercept** function was successful.
1	An incorrect host presentation space was specified.
2	An error was made in specifying parameters.
8	No previous **Start Close Intercept** (41) function was issued.
9	A system error occurred.

Return Code	Explanation
12	The session stopped.

Stop Communication Notification (82)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Communication Notification** function disables the capability of the **Query Communication Event** (81) function to determine whether any communication events have occurred in the specified Session.

Prerequisite Calls

Start Communication Notification (80)

Call Parameters

	Enhanced Interface
Function Number	Must be 82
Data String	1-character short name of the host presentation space, or a blank or null indicating request for updates to the host-connected presentation space
Length	4 is implied
PSPosition	NA

The calling data structure contains these elements:

Byte	Definition
1	A 1-character presentation space short name (PSID)
2-4	Reserved

Return Parameters

Return Code	Definition
0	The function was successful
1	An incorrect PSID was specified
8	No prior call to **Start Communication Notification** (80) function was called for the PSID
9	A system error was encountered

Stop Host Notification (25)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Stop Host Notification** function disables the capability of the **Query Host Update** (24) function to determine if the host presentation space or OIA has been updated. This function also stops host events from affecting the **Pause** (18) function.

Prerequisite Calls

Start Host Notification (23)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 121	
Data String	See the following note	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Note: 1-character short name of the target presentation space ID, or a blank or a null to indicate a request for the host-connected presentation space.

Return Parameters

Return Code	Definition
0	The **Stop Host Notification** function was successful.
1	An incorrect host presentation space was specified.
8	No previous **Start Host Notification** (23) function was issued.
9	A system error was encountered.

Stop Keystroke Intercept (53)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Stop Keystroke Intercept** function ends your application program's ability to intercept keystrokes.

Prerequisite Calls

Start Keystroke Intercept (50)

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 53	
Data String	Short name of the target presentation space (PSID)	
Length	1 is implied	Must be 4
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved

Return Parameters

Return Code	**Explanation**
0	The **Stop Keystroke Intercept** function was successful.
1	An incorrect presentation space was specified.
8	No prior **Start Keystroke Intercept** (50) function was called for this presentation space.
9	A system error was encountered.

Wait (4)

3270	***5250***	***VT***
Yes	Yes	Yes

The **Wait** function checks the status of the host-connected presentation space. If the session is waiting for a host response (indicated by XCLOCK (X ▯) or XSYSTEM), the **Wait** function causes EHLLAPI to wait up to 1 minute to see if the condition clears.

Prerequisite Calls

Connect Presentation Space (1)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 4	
Data String	NA	
Length	NA	
PS Position	NA	

Return Parameters

Return Code	Definition
0	The keyboard is unlocked and ready for input.
1	Your application program is not connected to a valid session.
4	Timeout while still in XCLOCK (X ▯) or XSYSTEM.
5	The keyboard is locked.
9	A system error was encountered.

Notes on Using This Function

1. The **Wait** function is used to give host requests like those made by the **Send Key** (3) function the time required to be completed. Using the **Set Session Parameters** (9) function, you can request the `TWAIT`, `LWAIT`, or the `NWAIT` option. See item 12 on page 648.
2. You can use this function to see if the host OIA is inhibited.
3. The **Wait** function is satisfied by the host unlocking the keyboard. Therefore, a return code of 0 does not necessarily mean that the transaction has been completed. To verify completion of the transaction, you should use the **Search Field** (30) function or **Search Presentation Space** (6) function combined with the **Wait** function to look for expected keyword prompts.

Window Status (104)

3270	*5250*	*VT*
Yes	Yes	Yes

The **Window Status** function allows the application to query or change a window's presentation space size, location, or visible state.

Prerequisite Calls

Connect Window Services (101)

Call Parameters

	Standard Interface	Enhanced Interface
Function Number	Must be 104	
Data String	See the following table	
Length	16 or 20	24 or 28
PS Position	NA	

The calling data string can contain:

Byte		Definition
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID)
	2–4	Reserved
2	5	A request option value, select one of the following values: • X'01' for set status **Note:** When the session is embedded In-Place in a compound OLE document, the set form of this function (byte 5 = X'01') always returns 0 but has no effect. • X'02' for query for status • X'03' for query for extended status
	6	Reserved

If the request option value is X'01' (set status):

Byte		Definition
Standard	Enhanced	
3–4	7–8	A 16- or 32-bit word containing the status set bits if the request option is 1 (set status). The following codes are valid return values if the request option is set status: **X'0001'** Change the window size. (Not valid with minimize, maximize, restore, or move.)

Byte		Definition
		X'0002' Move the window. (Not valid with minimize, maximize, size, or restore.) **X'0004'** ZORDER window replacement. **X'0008'** Set the window to visible. **X'0010'** Set the window to invisible. **X'0080'** Activate the window. (Sets focus to window and places it in the foreground unless ZORDER is specified. In this case, the ZORDER placement is used.) **X'0100'** Deactivate the window. (Deactivates the window and makes the window the bottom window unless ZORDER is also specified. In this case, the ZORDER placement is used.) **X'0400'** Set the window to minimized. (Not valid with maximize, restore, size, or move.) **X'0800'** Set the window to maximized. (Not valid with minimize, restore, size, or move.) **X'1000'** Restore the window. (Not valid with minimize, maximize, size, or move.)
5–6	9–12	A 16- or 32-bit word containing the X window position coordinate. (Ignored if the move option is not set.)
7–8	13–16	A 16- or 32-bit word containing the Y window position coordinate. (Ignored if the move option is not set.)
9–10	17–20	A 16- or 32-bit word containing the X window size in device units. (Ignored if the size option is not set.)
11–12	21–24	A 16- or 32-bit word containing the Y window size in device units. (Ignored if the size option is not set.)

Byte		Definition
13–16	25–28	A 16- or 32-bit word containing a window handle for relative window placement. These two words are only for the set option. (Ignored if the ZORDER option is not set.) Valid values are as follows: X'00000003' Place in front of all sibling windows. X'00000004' Place behind all sibling windows.

If the request option value is X'02' (query for status):

Byte		Definition
Standard	Enhanced	
3–4	7–8	A 16- or 32-bit word containing X'0000' if the request option is 2 (query for status). The following codes are possible return values if the request option is query for status. More than one state is possible. **X'0008'** The window is visible. **X'0010'** The window is invisible. **X'0080'** The window is activated. **X'0100'** The window is deactivated. **X'0400'** The window is minimized. **X'0800'** The window is maximized.
5–6	9–12	A 16- or 32-bit word containing the X window position coordinate. (Ignored if the move option is not set.)
7–8	13–16	A 16- or 32-bit word containing the Y window position coordinate. (Ignored if the move option is not set.)
9–10	17–20	A 16- or 32-bit word containing the X window size in device units. (Ignored if the size option is not set.)
11–12	21–24	A 16- or 32-bit word containing the Y window size in device units. (Ignored if the size option is not set.)
13–16	25–28	A 16- or 32-bit word containing a window handle for relative window placement. These two words are only for the set option. (Ignored if the ZORDER option is not set.) Valid values are as follows:

Byte		Definition
		X'00000003' Place in front of all sibling windows. X'00000004' Place behind all sibling windows.

If the request option value is X'03' (query for extended status):

Byte		Definition
Standard	Enhanced	
3–4	7–8	A 16- or 32-bit word containing X'0000' if the request option is 3 (query for extended status). The following codes are possible return values if the request option is query for extended status. More than one state is possible. **X'0008'** The window is visible. **X'0010'** The window is invisible. **X'0080'** The window is activated. **X'0100'** The window is deactivated. **X'0400'** The window is minimized. **X'0800'** The window is maximized.
5–6	9–10	A 16- or 32-bit word containing the current font size in the X-dimension. The value is in screen pels.
7–8	11–12	A 16- or 32-bit word containing the current font size in the Y-dimension. The value is in screen pels.
9–12	13–16	Reserved. This value is always zero.
13–14	17–18	A 16- or 32-bit word containing the row number of the first visible character of the presentation space. This value is usually one, unless the Fixed Size font option is in effect, and the window has been resized such that some of the presentation space is hidden.
15–16	19–20	A 16- or 32-bit word containing the column number of the first visible character of the presentation space.
17–20	21–24	A 16- or 32-bit word containing the presentation space window handle of the session.

Return Parameters

Return Code	Explanation
0	The **Window Status** function was successful.
1	The presentation space was not valid or not connected.
2	An incorrect option was specified.
9	A system error occurred.
12	The session stopped.

Notes on Using This Function

The logical terminal (LT) windows use character cells. When resizing the LT windows, the LT rounds the number to prevent character cell truncation. The requested size and position might be slightly different from what was requested. Follow the set option with a query option to determine the final Presentation Manager® window position and size. All x and y coordinate positions and sizes are in pels.

Write Structured Fields (127)

3270	*5250*	*VT*
Yes	No	No

The **Write Structured Fields** function allows an application to write structured field data to the host application. If the call specifies S (for Synchronous), the application does not receive control until the **Write Structured Fields** function is completed. If the call specifies A (for Asynchronous), the application receives control immediately after the call. If the call specifies M, the application receives control immediately after the call. The application may wait for the message. In any case (S, A or M), the application provides the buffer address in which data to the host is to be placed.

For a successful asynchronous completion of this function, the following statements apply:

The return code field in the parameter list might not contain the results of the requested I/O. If the return code is not 0, then the request failed. The application must take the appropriate action based on the return code.

If the return code for this request is 0, the application must use the request ID returned with this function call to issue the **Get Request Completion** function call to determine the completion results of the function associated with the request ID. The **Get Request Completion** function call returns the following information:

1. Function request ID
2. Address of the data string from the asynchronous request
3. Length of the data string
4. Return code of the completed function

Prerequisite Calls

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

Call Parameters

	Standard Interface	**Enhanced Interface**
Function Number	Must be 127	
Data String	See the following table	
Length	8, 10, or 14	Must be 20
PS Position	NA	

The calling data string can contain:

Byte		**Definition**
Standard	Enhanced	
1	1	A 1-character presentation space short name (PSID).
	2–4	Reserved.
2	5	S or A or M **S =** Synchronous. Control is not returned to the application until the read is satisfied. **A =** Asynchronous. Control is returned immediately to the application, can wait for the event object. **M =** Asynchronous. Control is returned immediately to the application, can wait for the message.
	6	Reserved.
3–4	7–8	2-byte destination/origin ID.
5–8	9–12	4-byte address of the buffer from which the data is to be written. The buffer must be obtained using the **Allocate Communications Buffer** (123) function.
9–10	13–16	Reserved.
11–12	17–20	When “M” is specified in position 5 (2 for 16-bit), the window handle of the window that receives the message should be set, The message is a return value of RegisterWindowMessage (“PCSHLL”) (not equal 0).
13–14		The data in these positions is ignored by EHLLAPI However, no error is caused if the migrating program has data in these positions. This data is accepted to provide compatibility with migrating applications.

Return Parameters

This function returns a data string and a return code.

Data String:

If A (asynchronous) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
9–10	13–14	2-byte Function Request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15–16	Reserved.
	17–20	4-byte value in which the event object address is returned by EHLLAPI. The application can wait for this event object. When the event object is cleared, the application must issue the **Get Request Completion** (125) function call to get results of the **Write Structured Fields** request. (32-bit only).

Note: An event object is returned for each successful asynchronous request. The event object should not be used again. A new event object is returned for each request and is valid for only the duration of that request.

Data String:

If M (asynchronous message mode) is specified in position 5 (2 for standard interface) and the function is completed successfully, the following data string is returned:

Byte		Definition
9–10	13–14	2-byte Function Request ID. It is used by the **Get Request Completion** (125) function to determine the completion of this function call.
	15–16	Reserved.
11–12	17–18	Task ID of asynchronous message mode.
	19–20	Reserved.

Note: If the function is completed successfully, an application window receive a message. The message is a return value of RegisterWindowMessage (PCSHLL). The wParam parameter contains the Task ID returned by the function call. HIWORD of lParam parameter contains return code 0, which shows the function was successful, and LOWORD of lParam parameter contains function number 127.

Return Code:

The following codes are defined:

Return Code	Explanation
0	The **Write Structured Fields** function was successful.

Return Code	Explanation
1	A specified host presentation space short session ID was not valid or was not connected.
2	An error was made in specifying parameters.
9	A system error occurred.
11	Resource unavailable (memory unavailable).
34	The message sent inbound to the host was canceled.
35	An outbound transmission from the host was canceled.
36	Request rejected. Lost contact with the host.
37	Failed. The host is inbound disabled.

Notes on Using This Function

1. Return code 35 will be returned when the first **Read Structured Fields** or **Write Structured Fields** is requested after an outbound transmission from the host is canceled. Corrective action is the responsibility of the application.
2. Return code 36 requires that the application disconnect from the emulation program and then reconnect to reestablish communications with the host. Corrective action is the responsibility of the application.
3. Return code 37 will be returned if the host is inbound disabled.
4. The EHLLAPI allows for a maximum of 20 asynchronous requests per application to be outstanding. A return code for unavailable resources (RC=11) is returned if more than 20 asynchronous requests are attempted.

The structured field data format is as follows:

Offset	Length	Contents
0	1 word	X'0000'
2	1 word	m (message length: the number of bytes of data in the message, the number does not include the buffer header prefix, which contains 8 bytes) This value must be set by the application.
4	1 word	X'0000'
6	1 word	X'0000'
8	8 bytes	Length of the first (or only) structured field message.
10	1 byte	First nonlength byte of the structured field message.
		⋮
m+7	1 byte	Last byte in the structured field message.

Bytes 0 through 7 are the buffer header. These first 8 bytes are used by the emulation program. The user section of the buffer begins with offset 8. Bytes 8 and 9 contain the number of bytes in the first structured field (a structured field message can contain multiple structured fields) including 2 bytes for bytes 8 and 9. Bytes 8 through *m*+7 are used for the structured field message sent to the host.

Synchronous Requests

When **Write Structured Fields** is requested synchronously (the S option in the data string), control is returned to the application only after the request is satisfied. The application can assume:

- The return code is correct.
- The data in the communications buffer (read buffer) is correct.
- The host is no longer processing the **Write Structured Fields** request.

Asynchronous Requests

When **Write Structured Fields** is requested asynchronously (the A option in the data string), the application *cannot* assume:

- The return code is correct.
- The data in the communications buffer (write buffer) is correct.
- The host is no longer processing the **Write Structured Fields** request.

When requested asynchronously, EHLLAPI returns the following values:

- A 16-bit Request ID in positions 13–14 (9–10 for standard interface) of the data string
- The address of a event object in positions 17–20 of the data string.

These are used to complete the asynchronous **Write Structured Fields** call.

The following steps must be completed to determine the outcome of an asynchronous **Write Structured Fields** function call:

- If the EHLLAPI return code is not zero, the request failed. No asynchronous request has been made. The application must take appropriate actions before attempting the call again.
- If the return code is zero, the application should wait until the event object is in the signaled state by using the **Get Request Completion** (125) function. The event object **Get Request Completion** (125) function) and should not be reused. The event object is valid only for the duration of the **Write Structured Fields** function call through the completion of the **Get Request Completion** (125) function call.
- Once the event object is in the signaled state use the returned 16-bit Request ID as the Request ID parameter in a call to the **Get Request Completion** (125) function. The data string returned from the **Get Request Completion** (125) function call contains the final return code of the **Write Structured Fields** function call.

Asynchronous Requests

When **Write Structured Fields** is requested asynchronously (the M option in the data string), the application cannot assume:

- The return code is correct
- The data in the communications buffer (write buffer) is correct
- The host is no longer processing the **Write Structured Fields** request

When requested asynchronously with the M option, EHLLAPI returns the following values:

- A 16-bit request ID in positions 13–14 (9–10 for standard interface) of the data string
- Task ID of asynchronous message mode in position 17–18 (11–12 for standard interface)

These are used to complete the asynchronous **Write Structured Fields** call.

WinHLLAPI Extension Functions

This chapter describes the extension functions provided when using WinHLLAPI programming support.

Summary of WinHLLAPI Functions

The following WinHLLAPI functions are available for 3270, 5250, and VT:

- Wait (4) on page 679
- Start Host Notification (23) on page 680
- Start Close Intercept (41) on page 681
- Start Keystroke Intercept (50) on page 682
- Send File (90) on page 683
- Receive File (91) on page 685

WinHLLAPI Asynchronous Functions

The following sections describe the WinHLLAPI asynchronous functions.

WinHLLAPIAsync

This entry point is used for six WinHLLAPI functions that often take a long time to complete. With WinHLLAPIAsync, the function will be launched asynchronously and will not interfere with the continued progression of the calling application. These functions are: **Wait** (04), **Start Host Notify** (23), **Start Close Intercept** (41), **Start Keystroke Intercept** (50), **Send File** (90), and **Receive File** (91), and are described in WinHLLAPI Extension Functions on page 678.

HANDLE WinHLLAPIAsync (HWIND hWnd, LPWORD *lpnFunction*, LPBYTE *lpData*, LPWORD *lpnLength*, LPWORD *lpnRetC*)*

The parameter list is the same as WinHLLAPI except a window handle is required before the function number. Since the function operates asynchronously, its completion is signaled by a registered message. The window handle is required as the target of the message.

There are two messages that must be registered by the WinHLLAPI application through calls to **RegisterWindowsMessage()** with the strings **WinHLLAPIAsync**(for all functions except 90 and 91) and **WinHLLAPIAsyncFileTransfer** (for functions 90 and 91). The standard format is as follows:

WPARAM

contains the Task Handle returned by the original function call.

LPARAM

the high word contains the error code and the low word contains the original function number.

Wait (4)

This function determines whether the Host session is in an inhibited state. If, for some reason, the session is in an inhibited state, this function will signal your application with a message when either the inhibited state expires or your wait period has expired. The amount of time to wait is set with the **Set Session Parameters (9)** function.

Prerequisite Functions

Connect Presentation Space (1)

WinHLLAPIAsync(*hWnd*, *lpwFunction*, *lpbyString*, *lpwLength*, *lpwReturnCode***)**

Call Parameters

Parameter	**Description**
Data String	NA
Data Length	NA
PS Position	NA

Return Codes

Code	**Description**
WHLLOK	The PS is uninhibited and ready for input.
WHLLNOTCONNECTED	Your WinHLLAPI application is not connected to a valid host session.
WHLLPSBUSY	Function timed out while still inhibited.
WHLLNHIBITED	The PS is inhibited.
SHLLSYSERROR	The function failed due to a system error.
WHLLCANCEL	The asynchronous function was cancelled.

Remarks

Asynchronous Wait is used to notify the calling application when the inhibited state of the PS is expired. When inhibited state has expired, this version of **Wait** will post a **WinHLLAPIAsync** message to the window specified by the *hWnd*. The session options **TWAIT**, **LWAIT**, and **NWAIT**affect the length of time that this function will wait. See Set Session Parameters (9) on page 643 for details on these session options.

Note: If **NWAIT** is specified in the session parameters and the application registers using revision 1.1 of the WinHLLAPI implementation, the **WINHLLAPIAsync** call will work the same as the **WinHLLAPI** call and not send a message. If revision 1.0 is being used then **Wait** will return a message immediately with the inhibited status of the PS.

Start Host Notification (23)

This function enables you to notify your WinHLLAPI application of changes in the Host Session Presentation Space (PS) or Operation Information Area (OIA).

Prerequisite Functions

There are no prerequisite functions for this function.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

Call Parameters

Parameter	Description
Data String	A 7-byte string in the following format: **Byte 1** Short name session ID of the desired Host session, or space or null for the current Host session. **Byte 2** Notification mode. "P" for presentation space update only, "O" for OIA update only, "B" for both presentation space and OIA updates. When calling WinHLLAPIAsync, this position can be "A". **Byte 3-6** Not used. Provided for compatibility with older applications. **Byte 7** Reserved or replaced with one of the following if using WinHLLAPIAsync and A in byte 2: P for presentation space update only, O for OIA update only; and B for both presentation space and OIA updates.
Data Length	Length of Host event buffer (256 recommended).

Parameter	Description
PS Position	NA

Return Parameters

Parameter	Description
Data String	Same as *Data String* on the call.

Return Codes

Code	Description
WHLLOK	Host notification enabled.
WHLLNOTCONNECTED	The specified Host session is invalid.
WHLLPARAMETERERROR	One of more parameters are invalid.
WHLLSYSERROR	The function failed due to a system error.
WHLLCANCEL	The asynchronous function was cancelled.

Remarks

Once enabled, Host notification is enabled until you call **Stop Host Notification (25)** or **WinHLLAPICancelAsyncRequest()**. The function initiates host notification and immediately returns control to your Windows HLLAPI application. This frees your application to perform other tasks while waiting for host updates. When an update occurs, the function will notify the window specified by *hWnd* with the registered message **WinHLLAPIAsync**.

Start Close Intercept (41)

This function intercepts user requests to close Z and I Emulator for Windows.

Prerequisite Functions

There are no prerequisite functions for this function.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

Call Parameters

Parameter	Description
Data String	A 5-byte string for returned semaphore address. The first byte is the session short name of the session to query, or space or null for the current session.
Data Length	Must be specified.
PS Position	NA

Return Parameters

Parameter	Description
Data String	A 5-byte string with the following format: **Byte 1** Session short name, or space or null for the current session **Bytes 2-5** Semaphore address.

Return Code

Code	Description
WHLLOK	The function was successful.
WHLLNOTCONNECTED	An invalid presentation space was specified.
WHLLPARAMETERERROR	An invalid option was specified.
WHLLSYSERROR	The function failed due to a system error.
WHLLCANCEL	The asynchronous function was cancelled.

Remarks

Once enabled, Host notification remains enabled until you call **Stop Close Intercept (43)** or **WinHLLAPICancelAsyncRequest ()**. Initially, the semaphore is set. After using this function, close requests from the user are discarded and the semaphore is cleared.

The function initiates close intercept and immediately returns control to your Windows HLLAPI application. This frees your application to perform other tasks while waiting for close requests. When a close request occurs, the function will notify the window specified by *hWnd* with the registered message **WinHLLAPIAsync**.

Start Keystroke Intercept (50)

This function intercepts keystrokes sent to a session by the user.

Prerequisite Functions

There are no prerequisite functions for this function.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

Call Parameters

Parameter	Description
Data String	A 6-byte string in the following format:

Parameter	Description
	Byte 1 Session short name, or space or null for the current Host session. **Byte 2** Keystroke intercept code. "D" causes only AID keystrokes to be intercepted; "L" causes all keystrokes to be intercepted. **Bytes 3-6** Reserved
Data Length	Variable (256 is recommended)
PS Position	NA

Return Code

Code	Description
WHLLOK	Keystroke intercept has been initiated.
WHLLNOTCONNECTED	The Host session presentation space is invalid.
WHLLPARAMETERERROR	One or more parameters are invalid.
WHLLPSBUSY	Session is busy.
WHLLSYSERROR	Function failed due to a system error.
WHLLCANCEL	Asynchronous function was cancelled.

Remarks

The function initiates keystroke intercept and immediately returns control to your Windows HLLAPI application. This frees your application to perform other tasks while waiting for keystrokes. Once initiated, the function will post a **WinHLLAPIAsync** message to the window specified by *hWnd* whenever the user sends a key to the PS. After notification, the intercepted keystrokes can be handled in any way that is allowed by a normal EHLLAPI application. Take note that the keystroke buffer is of limited size so each keystroke should be handled and removed from the buffer.

Send File (90)

This function transfers a file from the PC to the Host.

Prerequisite Functions

There are no prerequisite functions for this function.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

Call Parameters

Parameter	Description
Data String	SEND command parameters.
Data Length	Length of *Data String*. NA if session option EOT is specified.
PS Position	NA

Return Codes

Code	Description
WHLLOK	File transfer started successfully.
WHLLPARAMETERERROR	Parameter error or *Data Length* is zero or greater than 255.
WHLLFTXCOMPLETE	File transfer complete.
WHLLFTXSEGMENTED	Transfer is complete with segmented records.
WHLLSYSERROR	The function failed due to a system error.
WHLLTRANSABORTED	File transfer aborted, either due to the user clicking the cancel button or because the timeout period has elapsed.
WHLLFILENOTFOUND	PC file not found.
WHLLFTXCOMPLETECICS	File transfer was successful (transfer to CICS).
WHLLACCESSDENIED	Access denied to PC file.
WHLLMEMORY	Insufficient memory.
WHLLINVALIDENVIRONMENT	Invalid environment.

Remarks

Only one file transfer operation is supported per connected Host session.

The function initiates the file transfer and immediately returns control to your Windows HLLAPI application. This frees your application to perform other tasks while the file transfer is occurring. Once initiated the function will regularly post **WinHLLAPIAsyncFileTransfer** messages to the window specified by *hWnd*. These messages will notify the WinHLLAPI application of the status of the transfer and send a final message when the transfer is complete.

wParm

Is the status indicator: the high byte contains the Session ID, the low byte contains the status. If the low byte is zero, the file transfer is still in progress. If the low byte is one, the file transfer has completed.

lParm

If the low byte of *wParm* is zero (in progress), *lParm* is the number of bytes transferred. If the low byte *wParm* is one (completed), *lParm* is the completion code.

Receive File (91)

This function transfers a file from the PC to the Host.

Prerequisite Functions

There are no prerequisite functions for this function.

WinHLLAPIAsync (*hWnd, lpwFunction, lpbyString, lpwLength, lpwReturnCode***)**

Call Parameters

Parameter	**Description**
Data String	RECEIVE command parameters.
Data Length	Length of *Data String*. NA if session option EOT is specified.
PS Position	NA

Return Codes

Code	**Description**
WHLLOK	File transfer started successfully.
WHLLPARAMETERERROR	Parameter error or *Data Length* is zero or greater than 255.
WHLLFTXCOMPLETE	File transfer complete.
WHLLFTXSEGMENTED	Transfer is complete with segmented records.
WHLLSYSERROR	The function failed due to a system error.
WHLLTRANSABORTED	File transfer aborted, either due to the user clicking the cancel button or because the timeout period has elapsed.
WHLLFILENOTFOUND	PC file not found.
WHLLFTXCOMPLETECICS	File transfer was successful (transfer to CICS).
WHLLACCESSDENIED	Access denied to PC file.
WHLLMEMORY	Insufficient memory.
WHLLINVALIDENVIRONMENT	Invalid environment.

Remarks

Only one file transfer operation is supported per connected Host session.

The function initiates the file transfer and immediately returns control to your Windows HLLAPI application. This frees your application to perform other tasks while the file transfer is occurring. Once initiated the function will regularly post **WinHLLAPIAsyncFileTransfer** messages to the window specified by *hWnd*. These messages will notify the WinHLLAPI application of the status of the transfer and send a final message when the transfer is complete.

wParm

Is the status indicator: the high byte contains the Session ID, the low byte contains the status. If the low byte is zero, the file transfer is still in progress. If the low byte is one, the file transfer has completed.

lParm

If the low byte of *wParm* is zero (in progress), *lParm* is the number of bytes transferred. If the low byte *wParm* is one (completed), *lParm* is the completion code.

WinHLLAPICancelAsyncRequest

This function cancels an outstanding asynchronous function launched by a call to **WinHLLAPIAsync()**.

Syntax

int WinHLLAPICancelAsyncRequest (HANDLE *hAsyncTask*, WORD *wFunction***)**

Parameters

hAsyncTask

The handle returned by WinHLLAPIAsync() when the function was initiated.

wFunction

The function number of the asynchronous task to cancel. Because this parameter is required for revision 1.1 but not in 1.0, it is optional.

With this function, any asynchronous task previously initiated by a call to WinHLLAPIAsync() may be canceled while still outstanding.

Returns

The return value indicates if the specified function was, in fact, canceled. If the function was canceled then the return value is WHLLOK (0). If the outstanding asynchronous function was not cancelled, one of the following codes will be returned.

WHLLINVALID

hAsyncTask is not a valid task handle.

WHLLALREADY

The asynchronous task specified by hAsyncTask has already completed.

Initialization and Termination Functions

The following section describes the initialization and termination functions of WinHLLAPI programming support.

WinHLLAPI Startup

This function is used to register the application with the WinHLLAPI implementation and should be called before any other call to the WinHLLAPI implementation. This implementation supports Versions 1.0 and 1.1 of the WinHLLAPI specification. The WinHLLAPI application should negotiate version compatibility with this function.

Syntax

int WinHLLAPIStartup(WORD *wVersionRequired*, LPWHLLAPIDATA *lpData***)**

Parameters

wVersionRequired

This is the version required by the WinHLLAPI application. The low byte contains the major version number and the high byte contains the minor version (or revision) number.

lpData

This is a pointer to a WHLLAPIDATA structure which will receive the implementations version number and a string describing the WinHLLAPI implementation provider. The WHLLAPIDATA structure is defined as:

```
#define  WHLLDESCRIPTION_LEN   127
typedef  struct  tagWHLLAPIDATA
{
     WORD  wVersion;
     Char  szDescription[WHLLDESCRIPTION_LEN + 1];
}WHLLAPIDATA,  *  PWHLLAPIDATA,  FAR  *LPWHLLAPIDATA;
```

Returns

The return value indicates success or failure of registering the WinHLLAPI application with the implementation. If registration was successful, the return value is WHLLOK (zero). Otherwise, it is one of the following:

WHLLSYSNOTREADY

Indicates that the underlying network subsystem is unavailable.

WHLLVERNOTSUPPORTED

Indicates that the version requested is not provided by this implementation. This implementation supports Versions 1.0 and 1.1 only.

WinHLLAPI Cleanup

The WinHLLAPI specification recommends that this function be used by the WinHLLAPI application to de-register from the WinHLLAPI implementation.

Syntax

BOOL WinHLLAPICleanup()

Returns

Returns TRUE if the unregistration was successful. Otherwise, it returns FALSE.

Blocking Routines

The following sections describe the blocking routines supported by WinHLLAPI programming.

Note: Although blocking routines are supported for WinHLLAPI compliance, use of them is not recommended. Use of the WinHLLAPIAsync functions are the recommended method for asynchronous processing.

WinHLLAPIIsBlocking

This function tells the calling WinHLLAPI application thread whether it is in the process of executing a blocking call. A blocking call is any synchronous function that takes a long time to execute and does not return until complete. There are five blocking calls in this implementation of WinHLLAPI. The blocking calls are: **Get Key (51)**, **Wait (4)**, **Pause (18)**, **Send File (90)**, and **Receive File (91)**.

Syntax

BOOL WinHLLAPIIsBlocking()

Returns

If the WinHLLAPI application thread is in the middle of a blocking call, the function returns TRUE, otherwise, it returns FALSE.

Remarks

Because the default blocking-hook allows messages to be processed during blocking calls, it is possible to call the blocking call again.

WinHLLAPISetBlockingHook

This function sets an application-defined procedure to be executed while waiting for the completion of a blocking call. A blocking call is any synchronous function that takes a long time to execute and does not return until complete. There are five blocking calls in this implementation of WinHLLAPI. The blocking calls are: **Get Key (51)**, **Wait (4)**, **Pause (18)**, **Send File (90)**, and **Receive File (91)**.

Syntax

FARPROC WinHLLAPISetBlockingHook(FARPROC *lpfnBlockingHook***)**

Parameters

lpfnBlockingHook

This is a pointer to the new blocking procedure.

Description

The WinHLLAPI implementation has a default blocking procedure that consists of nothing more than a message handler. This default mechanism is shown in the following example:

```
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;
}
```

The blocking hook is implemented on a per-thread basis. A blocking hook set by this function will stay in effect for the thread until it is replaced by another call to **WinHLLAPISetBlockingHook()** or until the default is restored by a call to **WinHLLAPIUnhookBlockingHook()**.

The Blocking function must return **FALSE** if it receives a **WM_QUIT** message so WinHLLAPI can return control to the application to process the message and terminate gracefully. Otherwise, the function should return **TRUE**.

Returns

This function returns a pointer to the blocking function being replaced.

WinHLLAPIUnhookBlockingHook

This function restores the default blocking-hook for the calling thread.

Syntax

BOOL WinHLLAPIUnhookBlockingHook()

Returns

This function returns TRUE if the default blocking mechanism was successfully restored, otherwise it returns FALSE.

WinHLLAPICancelBlockingCall

This function cancels an executing blocking call in the *current thread*. A blocking call is any synchronous function that takes a long time to execute and does not return until complete. There are five blocking calls in this implementation of WinHLLAPI. The blocking calls are **Get Key** (51), **Wait** (4), **Pause** (18), **Send File** (90), and **Receive File** (91). If one of these is blocking calls are cancelled, the cancelled function will return WHLLCANCEL.

Syntax

int WinHLLAPICancelBlockingCall()

Returns

The return value indicates if the specified function was, in fact, canceled. If the function was canceled, then the return value is WHLLOK (0). If there are no outstanding blocking functions, then the following return code will be returned:

WHLLINVALID

Indicates that there is no blocking call currently executing.

PCSAPI Functions

Z and I Emulator for Windows provides an API set, which is defined here and called *PCSAPI*. Whereas EHLLAPI is used to manage the interaction between a workstation application program and host systems after the session is established, the PCSAPI can be used to control the Z and I Emulator for Windows session itself.

How to Use PCSAPI

You can write application programs using the PCSAPI in C or C++. To develop a PCSAPI application, do the following:

1. Prepare source code and add the appropriate PCSAPI calls.
2. Include the header file PCSAPI.H in the application program.
3. Compile the source code.
4. Link the resultant .OBJ files with the appropriate object file or libraries.

   You must also link it with the PCSAPI import library, PCSCALLS.LIB for 16-bit and PCSCAL32.LIB for 32-bit.

Page Layout Conventions

All PCSAPI function calls are presented in the same format so that you can quickly retrieve the information you need. The format is:

- Function Name
    - Function Type
    - Parameter Type and Description
    - Return Code

Function Type

“Function Type” shows the type of the function in the following format:

TYPE FunctionName(*TYPE Parameter1, ...***)**

Parameter Type and Description

“Parameter Type and Description” lists the type and describes each of the parameters to be specified in the PCSAPI function call.

Return Code

“Return Code” lists the codes that must be received by your program after a call to the PCSAPI function.

pcsConnectSession

3270	***5250***	***VT***
Yes	Yes	Yes

The **pcsConnectSession** function starts the communications with a host session specified by the short session ID. The session must already be started. This call is equivalent to the **Communications → Connect** menu item on the emulator session panel.

Function Type

BOOL WINAPI pcsConnectSession(*char cShortSessionID***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

Return Code

Return Code	**Meaning**
TRUE	Function ended successfully.
FALSE	It means one of the following things:

Return Code	Meaning
	• The session has not started. • An incorrect session ID was specified. • Call failed.

pcsDisconnectSession

3270	*5250*	*VT*
Yes	Yes	Yes

The **pcsDisconnectSession** function stops the communications link with a host session specified by the short session ID. This only disconnects the link; it does not stop the session. This call is equivalent to the **Communications → Disconnect** menu item on the emulator session panel.

Function Type

BOOL WINAPI pcsDisconnectSession(*char cShortSessionID***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

Return Code

Return Code	Meaning
TRUE	Function ended successfully.
FALSE	It means one of the following things: • The session has not started. • An incorrect session ID was specified. • Call failed.

pcsQueryConnectionInfo

3270	*5250*	*VT*
Yes	No	No

The **pcsQueryConnectionInfo** function returns information about the Telnet connection of the specified host session. The resulting information is returned into the buffer supplied by the application.

Function Type

BOOL WINAPI pcsQueryConnectionInfo(*char cShortSessionID, CONNECTIONINFO *ConnectionInfo***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

CONNECTIONINFO *ConnectionInfo

Pointer to a CONNECTIONINFO structure where the connection info data will be returned.

Return Code

Return Code	Meaning
TRUE	Function ended successfully.
FALSE	It means one of the following things: • The session has not started. • An incorrect session ID was specified. • The session specified was not a supported connection type for this API (not Telnet).

ConnectionInfo

The CONNECTIONINFO structure will be filled with the information about the host connection, consisting of the following information:

Structure	Information
Host name	States the name of the currently connected Telnet host.
LU name	States the LU name currently assigned.
Port number	States the host port number being used for the connection.
SSL indicator	Indicates a Secure Connection (1 = secure; Ø = not secure).

Note: This API is valid only with the 32-bit version of PCSAPI, and only works for Telnet connections.

Example

```
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*
Yes	Yes	Yes

The **pcsQueryEmulatorStatus** function returns the status of the host session specified by the short session ID.

Function Type

ULONG WINAPI pcsQueryEmulatorStatus(*char cShortSessionID***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

Return Code

The return code value should be processed bit-significantly, that is, by either one of the following values or an ORed value out of the following values:

Return Code	Value	Meaning
PCS_SESSION_STARTED	0x00000001	Specified session has started. When this bit is off, the specified session has not started or an incorrect session ID was specified.
PCS_SESSION_ONLINE	0x00000002	Specified session is online (connected). When this bit is off, the specified session is offline (disconnected).
PCS_SESSION_API_ENABLED	0x00000004	API (EHLLAPI) is enabled on the specified session. If this bit is off, API is disabled on this session.

pcsQuerySessionList

3270	***5250***	***VT***
Yes	Yes	Yes

The **pcsQuerySessionList** function returns a list of all the current host sessions. The application must supply an array of SESSINFO structures as defined in the PCSAPI.H file, and a count of the number of elements in the array. This function fills in the structures with information about each session and returns the number of sessions found.

If the array has fewer elements than there are host sessions, then only the supplied elements of the array are filled in. The function always returns the actual number of sessions, even if the array is too small.

An application can call this function with zero array elements to determine how many sessions exist. A second call can then be made to obtain the session information.

Function Type

ULONG WINAPI pcsQuerySessionList(*ULONG Count, SESSINFO *SessionList***)**

Parameter Type and Description

ULONG Count

Number of elements in the SessionList array.

SESSINFO *SessionList

Pointer to an array of SESSINFO structures as defined in PCSAPI.H.

Return Parameters

Return Code

Total number of Z and I Emulator for Windows sessions. This may be greater than or less than the Count parameter.

SessionList

The array of SESSINFO structures is filled with information about the host sessions. Sessions may be placed in the list in any order. Each SESSINFO structure contains the following fields (defined in PCSAPI32.H)

Name

A union of char and ULONG which contains the session ID (A–Z). In the current implementation of Z and I Emulator for Windows, only the lower byte (char) is used, the other bytes are returned as zero.

Status

A combination of bit flags which indicate the current status of the session. The flags (PCS_SESSION_*) are defined in the following table.

The status value should be processed bit-significantly, that is, by either one of the following values or an ORed value out of the following values:

Return Code	Meaning
PCS_SESSION_STARTED	The session is running. If this flag is not set, all others are undefined.
PCS_SESSION_ONLINE	The session has established a communications link to the host (this is, the session is connected).
PCS_SESSION_API_ENABLED	The session is enabled for programming APIs. If this flag is not set, the EHLLAPI and Host Access Class Library APIs cannot be used on this session.

Example

```
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*
Yes	Yes	Yes

The **pcsQueryWorkstationProfile** function returns the workstation profile name that has been used to invoke the host session. To specify the host session, the short session ID must be used. The workstation profile name is copied to the work buffer supplied by the application.

Function Type

BOOL WINAPI pcsQueryWorkstationProfile(*char cShortSessionID, PSZ lpBuffer***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

PSZ lpBuffer

Work buffer to copy a null-terminated workstation profile name. The buffer must be large enough to contain a fully qualified file name.

Return Code

Return Code	Meaning
TRUE	Function ended successfully.
FALSE	It means one of the following things: • The session has not started. • An incorrect session ID was specified.

pcsSetLinkTimeout

3270	*5250*	*VT*
Yes	Yes	Yes

The **pcsSetLinkTimeout** function sets the idle timeout of a Telnet link which is SSCP owned. This function has no effect on non-TN connections or connections which are not in SSCP owned state. If the timeout value is set to zero the link will not time out. Otherwise the link will time out (disconnect) after being idle in SSCP-owned state for the number of minutes specified.

Function Prototype

ULONG WINAPI pcsSetLinkTimeout(*char cShortSessionID, USHORT Timeout***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

USHORT Timeout

Timeout value in minutes. A value of zero disables timeout.

Return Code

Return Code	Meaning
PCS_SUCCESSFUL	The function ended successfully.
PCS_SYSTEM_ERROR	A system error occurred.

pcsStartSession

3270	*5250*	*VT*
Yes	Yes	Yes

The **pcsStartSession** function starts a host session by using a specified workstation profile. A short session ID can also be specified.

Function Type

ULONG WINAPI pcsStartSession(*PSZ lpProfile, char cShortSessionID, USHORT fuCmdShow***)**

Parameter Type and Description

PSZ lpProfile

Path and complete filename of the profile to load. Path is optional but complete filename must be specified (.ws extension is not assumed).

char cShortSessionID

Presentation space short session ID. Space or NULL indicates the next available session ID.

USHORT fuCmdShow

Specifies how the window is to be displayed. One of the following values from PCSAPI.H:

- PCS_HIDE
- PCS_SHOW
- PCS_MINIMIZE
- PCS_MAXIMIZE

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	The function ended successfully.
PCS_INVALID_ID	1	An incorrect session ID was specified.
PCS_USED_ID	2	The specified short session ID is already used.

Return Code	Value	Meaning
PCS_INVALID_PROFILE	3	An error was made in specifying the workstation profile, or the window parameter was not valid.
PCS_SYSTEM_ERROR	9	A system error occurred.

pcsStopSession

3270	*5250*	*VT*
Yes	Yes	Yes

The **pcsStopSession** function stops a host session specified by the short session ID.

Function Type

BOOL WINAPI pcsStopSession(*char cShortSessionID, USHORT fuSaveProfile***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

USHORT fuSaveProfile

This parameter can be one of the following values:

fuSaveProfile	Value	Meaning
PCS_SAVE_AS_PROFILE	0	Save the profile as specified in the current profile.
PCS_SAVE_ON_EXIT	1	Save the profile on exit.
PCS_NOSAVE_ON_EXIT	2	Do not save the profile on exit.

Return Code

Return Code	Meaning
TRUE	The function ended successfully.
FALSE	It means one of the following things: • The session has not started. • An incorrect session ID was specified.

Page Setup Functions

The PCSAPI functions listed in this section enable you to control and retrieve the Z and I Emulator for Windows emulator session **Page Setup** settings.

Restrictions

If the following restrictions are not satisfied, the API will fail. The return code indicates the reason for the failure.

- The host session specified in the argument `cShortSessionID` should not be in PDT mode.
- The host session should not be printing when the API is invoked.
- The **File → Page Setup** dialog should not be in use.

Some members in the PAGEINFO structure might be valid or supported only for specific session types. If a restriction is not specified, then that member is valid or supported for the following session types:

- 3270 display
- 3270 printer
- 5250 display
- ASCII VT

5250 printer sessions are not supported.

pcsGetPageSettings

3270	***5250***	***VT***
Yes	Yes	Yes

The **pcsGetPageSettings** function retrieves the host session page settings values (similar to the **File → Page Setup** dialog settings). Only the settings in the **Text** tab of the dialog are supported.

Function Type

ULONG WINAPI pcsGetPageSettings(*char cShortSessionID, PAGEINFO * const pPageInfo, ULONG * const pErrorInfo***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

PAGEINFO * const pPageInfo

Pointer to PAGEINFO structure, where the page settings are returned.

nFlags

Combination of bit flags that indicates which members in the structure are valid. These flags can be used independently or by ORing them together to restore the property page (defined in PCSAPI32.H). The flags, along with the corresponding valid members in the structure, are as follows:

Flag	Valid members in the structure
PCS_PAGE_CPI	nCPI
PCS_PAGE_LPI	nLPI
PCS_PAGE_FACE_NAME	szFaceName
PCS_PAGE_MPL	nMPL
PCS_PAGE_MPP	nMPP

nCPI

The number of characters printed per inch.

LOWORD is the actual CPI value.

If Font CPI is configured in the session, HIWORD is 1. If Font CPI is not configured, HIWORD is 0.

nLPI

The number of lines printed per inch.

LOWORD is the actual LPI value.

If Font LPI is configured in the session, HIWORD is 1. If Font LPI is not configured, HIWORD is 0.

szFaceName

Face name of the printer font. This must be a null-terminated string.

nMPL

Maximum number of lines that can be printed per page.

This is also called MPL (Maximum Print Lines). Supported range is 1 to 255.

nMPP

Maximum number of characters that can be printed per line.

This is also called MPP (Maximum Print Position). Supported range is 1 to 255.

ULONG * const pErrorInfo

Not used. This must be set to NULL by the caller.

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	Function ended successfully.
PCS_INVALID_ID	1	Incorrect session ID was specified.
PCS_INVALID_SESS_TYPE	2	Not supported for the host session type.
PCS_DIALOG_IN_USE	3	Failed because the host session Page Setup or Printer Setup dialog was in use.
PCS_PRINTING	4	Page settings cannot be obtained because host session was printing.
PCS_PDT_MODE	5	Page settings cannot be obtained because host session is in PDT mode.
PCS_SYSTEM_ERROR	9	A system error occurred.

Example

```
{
   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*
Yes	Yes	Yes

The **pcsRestorePageDefaults** function restores the system default values of the Page Setup property pages defined in the nFlags field. This is equivalent to clicking **Default** in the property pages of the **File → Page Setup** dialog. Only the settings in the **Text** tab are supported.

Function Type

ULONG WINAPI pcsRestorePageDefaults(*char cShortSessionID, ULONG nFlags***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

ULONG nFlags

The following flag describes the name of the specified **Page Setup** dialog property page. This flag can be bitwise ORed to restore the property page (defined in PCSAPI32.H).

PCS_PAGE_TEXT

This flag describes the Text property page. This is the only property page currently supported.

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	Function ended successfully.
PCS_INVALID_ID	1	Incorrect session ID was specified.
PCS_INVALID_SESS_TYPE	2	The nFlags parameter has one or more options that are not valid for the host session type. No settings were restored.
PCS_DIALOG_IN_USE	3	Failed because the host session Page Setup or Printer Setup dialog was in use.
PCS_PRINTING	4	Page settings cannot be changed because host session was printing.
PCS_PDT_MODE	5	Page settings cannot be changed because host session is in PDT mode.

Return Code	Value	Meaning
PCS_SYSTEM_ERROR	9	A system error occurred.

Example

```
{
   ULONG Rc = 0;

   Rc = pcsRestorePageDefaults('A', PCS_PAGE_TEXT);

   if (Rc != PCS_SUCCESSFUL)
      printf("Failure. Return code = %d\n", Rc);
}
```

pcsSetPageSettings

3270	*5250*	*VT*
Yes	Yes	Yes

The **pcsSetPageSettings** function sets the host session page settings. This is similar to configuring the **File → Page Setup** dialog settings. Only the settings in the **Text** tab are supported.

Note:

1. CPI, LPI, and FontSize are dependent on the FaceName configured in the host session. If this API is used to set CPI, LPI, FontSize, and FaceName together, FaceName is set first, then the dependent properties.
2. If this API is used to set FaceName and the dependent properties in separate invocations, set FaceName first, then set CPI, LPI and FontSize. Otherwise, each time FaceName is set, query CPI, LPI and FontSize and ensure that they have the desired values.
3. If CPI, LPI, or FontSize are set before FaceName, then different values for CPI, LPI, or FontSize might be configured in the host session. This might occur if the current CPI, LPI, or FontSize values are not valid for the new FaceName set.

Function Type

ULONG WINAPI pcsSetPageSettings(*char cShortSessionID, const PAGEINFO * const pPageInfo, ULONG * const pErrorInfo***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

const PAGEINFO * const pPageInfo

Pointer to PAGEINFO structure, where the page settings are mentioned.

nFlags

Combination of bit flags that indicates which members in the structure are valid. These flags can be used independently or by ORing them together to restore the property page (defined in PCSAPI32.H). The flags, along with the corresponding valid members in the structure, are as follows:

Flag	Valid members in the structure
PCS_PAGE_CPI	nCPI
PCS_PAGE_LPI	nLPI
PCS_PAGE_FACE_NAME	szFaceName
PCS_PAGE_MPL	nMPL
PCS_PAGE_MPP	nMPP

nCPI

The number of characters printed per inch.

To select Font CPI, set the HIWORD of nCPI to 1. LOWORD of nCPI will be ignored.

To select a particular CPI value, do the following:

1. Set the HIWORD of nCPI to 0.
2. Set the LOWORD of nCPI to the actual CPI value.

nLPI

The number of lines printed per inch.

To select Font LPI, set the HIWORD of nLPI to 1. LOWORD of nLPI will be ignored

To select a particular LPI value, do the following:

1. Set the HIWORD of nLPI to 0.
2. Set the LOWORD of nLPI to the actual LPI value.

szFaceName

Face name of the printer font. This must be a null-terminated string.

nMPL

Maximum number of lines that can be printed per page.

This is also called MPL (Maximum Print Lines). Supported range is 1 to 255.

nMPP

Maximum number of characters that can be printed per line.

This is also called MPP (Maximum Print Position). Supported range is 1 to 255.

ULONG * const pErrorInfo

Contains the extended error info when the API fails with the return code of PCS_FAILURE. If the detailed error information is not needed, this flag must be set to NULL by the caller.

This is a combination of bit flags that describe which members of the PAGEINFO structure could not be set successfully. The flags that are defined in PCSAPI32.H are as follows:

Flag

Valid members in the structure

PCS_PAGE_CPI

Only nCPI is not valid.

PCS_PAGE_LPI

Only nLPI is not valid.

PCS_PAGE_FACE_NAME

Only szFaceName is not valid.

PCS_PAGE_MPL

Only nMPL is not valid.

PCS_PAGE_MPP

Only nMPP is not valid.

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	Function ended successfully.
PCS_INVALID_ID	1	Incorrect session ID was specified.
PCS_INVALID_SESS_TYPE	2	Not supported for the host session type.
PCS_DIALOG_IN_USE	3	Failed because the host session Page Setup or Printer Setup dialog was in use.

Return Code	Value	Meaning
PCS_PRINTING	4	Page settings cannot be changed because host session was printing.
PCS_PDT_MODE	5	Page settings cannot be changed because host session is in PDT mode.
PCS_FAILURE	6	Host session page settings are not fully applied. This could be because invalid data was given for some or all fields in the PAGEINFO structure. Examine pErrorInfo for details about settings that are not applied.
PCS_SYSTEM_ERROR	9	A system error occurred.

Example

```
{
   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);
}
```

Printer Setup Functions

The PCSAPI functions listed in this section enable you to control and retrieve the Z and I Emulator for Windows emulator session **Printer Setup** settings.

Restrictions

If the following restrictions are not met, the API will fail. The return code indicates the reason for the failure.

- The host session should not be printing when the API is invoked.
- The **File → Printer Setup** dialog should not be in use.

pcsGetPrinterSettings

3270	***5250***	***VT***
Yes	Yes	Yes

The **pcsGetPrinterSettings** function retrieves the host session printer settings (similar to the **File → Printer Setup** dialog settings).

Function Type

ULONG WINAPI pcsGetPrinterSettings(*char cShortSessionID, PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

PRINTINFO * const pPrintInfo

Pointer to PRINTINFO structure, where the printer settings are specified.

nFlags

Must be set to 0. This is ignored.

nBufSize

Size of the buffer allocated for the following fields:

- lpPDTFile
- lpPrtToDskAppFile
- lpPrtToDskSepFile
- lpPrinterName

If more than one of these members is retrieved in a single API call, then the caller must allocate the same size for all the buffers and pass that size in this member.

If this member is set to 0, the fields are ignored. The maximum size required for the buffers of the fields is returned in nSizeNeeded.

nSizeNeeded

The value of this member is determined by conditions related to the following fields:

- lpPDTFile
- lpPrtToDskAppFile
- lpPrtToDskSepFile
- lpPrinterName

The conditions are as follows:

- The value is the number of bytes needed, if the size of the buffer allocated by the caller is not big enough to return the fields listed above.
- The value is the maximum size of the required buffer, if more than one of the fields listed above are obtained by the caller.
- If nBufSize is set to 0 by the caller, this member contains the maximum size required for the buffers of the fields listed above.

bPromptDialog

Possible values are as follows:

- If TRUE, the Printer Setup dialog is shown before printing.
- If FALSE, the Printer Setup dialog is not shown before printing.

bPDTMode

Possible values are as follows:

- If TRUE, the host session is in PDT mode.
- If FALSE, the host session is in non-PDT mode (GDI mode).

lpPDTFile

Must be set to NULL if the caller is not interested in getting this member. The PDT file is returned if this is not a null pointer. This must point to the buffer of size nBufSize allocated by the caller.
When the API returns, this member contains one of the following:

- The fully qualified path name of the session PDT file.
- An empty string ("") if no PDT file is configured in the session.
- A truncated file name if the buffer size is not sufficient. The member nSizeNeeded contains the size of the buffer needed.

nPrtMode

This is an enumerated value that indicates the PrintMode of the connection. The enum data type PRINTMODE is defined in PCSAPI32.H. The nPrtMode setting must be one of the following:

- **PrtToDskAppend** (**Print to Disk-Append** mode)

  This is equivalent to selecting the **Append** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **PrtToDskSeparate** (**Print to Disk-Separate** mode)

  This is equivalent to selecting the **Separate** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **WinDefaultPrinter** (**Windows Default Printer** mode)

  This is equivalent to selecting the **Use Windows Default Printer** option in the host session **Printer Setup** dialog.
- **SpecificPrinter** (**Specific Printer** mode)

  This is equivalent to selecting a printer in the host session **Printer Setup** dialog, while leaving **Use Windows Default Printer** unchecked.

lpPrtToDskAppFile

Must be set to NULL if the caller is not interested in getting this member. The **Print to Disk-Append** file is returned if this is not a null pointer. This must point to the buffer of size nBufSize allocated by the caller.
When the API returns, this member contains one of the following:

- The fully qualified path name of the session **Print to Disk-Append** file.
- An empty string ("") if no **Print to Disk-Append** file is configured for the session.
- A truncated file name if the buffer size is not sufficient. The nSizeNeeded member contains the size of the buffer needed.

lpPrtToDskSepFile

Must be set to NULL if the caller is not interested in getting this member. The **Print to Disk-Separate** file is returned if this is not a null pointer. This must point to the buffer of size nBufSize allocated by the caller.
When the API returns, this member contains one of the following:

- The fully qualified path name of the session **Print to Disk-Separate** file.
- An empty string ("") if no **Print to Disk-Separate** file is configured for the session.
- A truncated file name if the buffer size is not sufficient. The nSizeNeeded member contains the size of the buffer needed.

lpPrinterName

Must be set to NULL if the caller is not interested in getting this member. The name of the printer is returned if this is not a null pointer. This must point to the buffer of size nBufSize allocated by the caller.
When the API returns, this member has one of the following:

- The name of the specific printer configured in the session, if the host session nPrtMode is SpecificPrinter.
- The name of the Windows default printer configured in the session, if the host session nPrtMode is WinDefaultPrinter.
- An empty string (""), if the host session nPrtMode is PrtToDskAppend or PrtToDskSeparate.
- A truncated printer name, if the buffer size is not sufficient. nSizeNeeded has the size of the buffer needed.

PrinterName must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `IBM InfoPrint 40 PS on Network Port`
- `HP LaserJet 4050 Series PCL 6 on LPT1`

ULONG * const pErrorInfo

This is filled with the extended error info when the API fails with the return code of PCS_FAILURE. pErrorInfo must be set to NULL by the caller, if the details of errors are not needed.

The following section describes the flags that are defined in PCSAPI32.H.

Flags for the pErrorInfo member of the PRINTINFO structure

PCS_PRINT_PRINTMODE_ERROR

PrintMode is not configured in the host session.

PCS_PRINT_PDTFILE_SIZEERR

The buffer size is not sufficient for lpPDTFile, so the file name is truncated. The nSizeNeeded member contains the actual size of the buffer required to return the PDT file.

PCS_PRINT_DSKAPPFILE_SIZEERR

The buffer size is not sufficient for lpPrtToDskAppFile, so the file name is truncated. The nSizeNeeded member contains the actual size of the buffer required to return the **Print to Disk-Append** file.

PCS_PRINT_DSKSEPFILE_SIZEERR

The buffer size is not sufficient for lpPrtToDskSepFile, so the file name is truncated. The nSizeNeeded member contains the actual size of the buffer required to return the **Print to Disk-Separate** file.

PCS_PRINT_PRINTERNAME_SIZEERR

The buffer size is not sufficient for lpPrinterName, so the printer name is truncated. The nSizeNeeded member contains the actual size of the buffer required to return the printer name.

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	The function ended successfully.
PCS_INVALID_ID	1	An incorrect session ID was specified.
PCS_DIALOG_IN_USE	3	Failed because the host session Page Setup or Printer Setup dialog was in use.
PCS_PRINTING	4	The printer settings could not be changed because the host session was printing. The application must retry later
PCS_FAILURE	6	Some printer settings could not be retrieved successfully. pErrorInfo contains detailed error information on which settings could not be retrieved.
PCS_SYSTEM_ERROR	9	A system error occurred.

Example

```
{
   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*
Yes	Yes	Yes

The **pcsSetPrinterSettings** function controls the host session printer settings (similar to the **File → Printer Setup** dialog settings).

Function Type

ULONG WINAPI pcsSetPrinterSettings(*char cShortSessionID, const PRINTINFO * const pPrintInfo, ULONG * const pErrorInfo***)**

Parameter Type and Description

char cShortSessionID

Presentation space short session ID.

const PRINTINFO * const pPrintInfo

Pointer to PRINTINFO structure, where the printer settings are mentioned.

nFlags

Combination of bit flags that indicates which members in the structure are valid. These flags can be used independently or by ORing them together to restore the property page (defined in PCSAPI32.H). The flags, along with the corresponding valid members in the structure, are as follows:

Flag

Valid members in the structure

PCS_PRINT_PDT

bPDTMode, lpPDTFile

PCS_PRINT_PRINTMODE

nPrtMode, lpPrtToDskAppFile, lpPrtToDskSepFile, lpPrinterName

PCS_PRINT_PROMPT_DIALOG

bPromptDialog

nBufSize

Must be set to 0. This is ignored.

nSizeNeeded

Must be set to 0. This is ignored.

bPromptDialog

Possible values are as follows:

- If TRUE, the Printer Setup dialog is shown before printing.
- If FALSE, the Printer Setup dialog is not shown before printing.

bPDTMode

Possible values are as follows:

- If TRUE, the connection is set to PDT mode.
- If FALSE, the connection is set to non-PDT mode (GDI mode).

lpPDTFile

Used only if bPDTMode is set to TRUE. This is ignored if bPDTMode is set to FALSE.
This is a null-terminated string containing the name of the PDT file and must be one of the following:

- NULL

  The PDT file that is currently configured in the connection is used. If there is no PDT file already configured in the connection, the API fails with an exception.
- File name, without the path

  lpPDTFile in the PDFPDT subfolder in the Z and I Emulator for Windows installation path is used.
- Fully qualified path name of the file

  If lpPDTFile does not exist, the API fails.

nPrtMode

This is an enumerated value that indicates the PrintMode of the connection. The enum data type PRINTMODE is defined in PCSAPI32.H. The nPrtMode setting must be one of the following:

- **PrtToDskAppend** (**Print to Disk-Append** mode)

  This is equivalent to selecting the **Append** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **PrtToDskSeparate** (**Print to Disk-Separate** mode)

  This is equivalent to selecting the **Separate** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **WinDefaultPrinter** (**Windows Default Printer** mode)

  This is equivalent to selecting the **Use Windows Default Printer** option in the host session **Printer Setup** dialog.
- **SpecificPrinter** (**Specific Printer** mode)

  This is equivalent to selecting a printer in the host session **Printer Setup** dialog, while leaving the **Use Windows Default Printer** option unchecked.

lpPrtToDskAppFile

This is used only if nPrtMode is set to PrtToDskAppend.
This is a null-terminated string containing the name of the **Print to Disk-Append** file and must be one of the following:

- NULL

  The file that is currently configured for the PrtToDskAppend mode in the connection is used. If there is no PDT file already configured in the connection, the API will fail.
- File name, without the path

  The user-class application data directory path is used to locate the file. If the file exists, it is used. Otherwise, it will be created when printing is complete.
- Fully qualified path name of the file

  The directory must exist in the path, or the API will fail. It is not necessary that the file exist in the path.

lpPrtToDskSepFile

The possible values are as follows:

- Fully qualified path name of the **Print to Disk-Separate** file for the session.
- An empty string ("") if no **Print to Disk-Separate** file is configured for the session.
- A truncated file name if the buffer size is not sufficient. The nSizeNeeded member contains the size of the buffer needed.

lpPrinterName

This is used only if nPrtMode is set to SpecificPrinter. It is ignored otherwise. This is a null-terminated string containing the printer name. If the printer does not exist, this member fails.
PrinterName must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `IBM InfoPrint 40 PS on Network Port`
- `HP LaserJet 4050 Series PCL 6 on LPT1`

ULONG * const pErrorInfo

This is filled with the extended error info when the API fails with the return code of PCS_FAILURE. pErrorInfo must be set to NULL by the caller, if the details of errors are not needed.

The following section describes the flags that are defined in PCSAPI32.H.

Flags for the pErrorInfo member of the PRINTINFO structure

PCS_PRINT_PDTMODE_ERROR

This can occur for one of one of the following reasons:

- bPDTMode is set to TRUE, lpPDTFile is set to NULL, and there is no PDT file already configured for the host session.
- nPrtMode is set to PrtToDskAppend or PrtToDskSeparate, PCS_PRINT_PDT is not set in nFlags, and the host session is not already in PDT mode.
- nPrtMode is set to PrtToDskAppend or PrtToDskSeparate and bPDTMode is set to FALSE.

PCS_PRINT_PDTFILE_ERROR

The file or the path specified in lpPDTFile was not found.

PCS_PRINT_PRTTODSK_FILE_ERROR

This can occur for one of one of the following reasons:

- The folder specified in the field lpPrtToDskAppFile or lpPrtToDskSepFile does not exist or does not have write access.
- An extension is specified in the field lpPrtToDskSepFile.

PCS_PRINT_PRINTMODE_ERROR

nPrtMode cannot be set successfully. This can occur for one of the following reasons:

- The value of nPrtMode is not one of the enumerated constants of the PRINTMODE enum data type.
- nPrtMode is set to PrtToDskAppend, lpPrtToDskAppFile is set to NULL, and there is no **Print to Disk-Append** file already configured in the host session.
- nPrtMode is set to PrtToDskSeparate, lpPrtToDskSepFile is set to NULL, and there is no **Print to Disk-Separate** file already configured in the host session.
- nPrtMode is set to SpecificPrinter and the printer given in the lpPrinterName field was not found.
- nPrtMode is set to WinDefaultPrinter and there is no default Windows® printer configured in the system.
- bPDTMode is set to FALSE and PCS_PRINT_PRINTMODE is not set in nFlags, but the host session PrintMode is PrtToDskAppend or PrtToDskSeparate.

Return Code

Return Code	Value	Meaning
PCS_SUCCESSFUL	0	The function ended successfully.
PCS_INVALID_ID	1	An incorrect session ID was specified.

Return Code	Value	Meaning
PCS_DIALOG_IN_USE	3	Failed because the host session Page Setup or Printer Setup dialog was in use.
PCS_PRINTING	4	The printer settings could not be changed because the host session was printing. The application must retry later.
PCS_FAILURE	6	No host session printer settings were applied. This might occur because invalid data was given for some or all of the fields in the PRINTINFO structure. pErrorInfo contains details about the errors.
PCS_SYSTEM_ERROR	9	A system error occurred.

Example

```
{
   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);
}
```

Troubleshooting for Emulator programming

You can use the following self-help information resources and tools to help you troubleshoot problems:

- Refer to the release information for your product for known issues, workaround, and troubleshooting information.
- Check if a download or fix is available to resolve your problem.

- Search the available knowledge bases to see if the resolution to your problem is already documented.
- If you still need help, contact HCL Software Support and report your problem.

Partial EHLLAPI input on Z and I Emulator for Windows host screen

Problem

Truncated command text was sent to a host when using HCL Z and I Emulator for Windows.

Cause

If an EHLLAPI application sends a SYSREQ key to the host and then tries to input a command onto the host screen, sometimes only a truncated part of the command is sent to the host. This problem occurs due to lack of synchronization between the SYSREQ processing at the Z and I Emulator for Windows host side and the input of commands from the EHLLAPI application.

When the application sends a SYSREQ command to the host, the following situations occur:

- The OIA is updated to indicate that you are in a SSCP-LU session.
- The Z and I Emulator for Windows session sends the AO command (the SYSREQ) to the 3270 host.

As soon as the host receives the SYSREQ, it responds to Z and I Emulator for Windows with the 0x15 or NL (NewLine) code. When Z and I Emulator for Windows processes this NL command by filling the rest of the line with NULLs, and moving the cursor to the beginning of the next line.

A problem occurs when the EHLLAPI application continues to input various commands in the host screen (through the SendKeys function), even before the Z and I Emulator for Windows session has received the NL command from the host and processed it. As a result, a part of the input command is first entered onto the screen, while the NL command is processed and the cursor is moved over to the next line. Then the remaining part of the command is input on the next line. Thus, only the truncated second part of the command is sent to the host, causing erroneous results.

Resolution

The solution for this problem is to force the EHLLAPI application to wait until the NL command is received and processed, before continuing to input the commands to the host screen. Once the session has notified the EHLLAPI application that the host response for SYSREQ has been processed, the EHLLAPI application can then continue with its input (because the session is now in the right state to accept new input). To accomplish this, use the following EHLLAPI function calls:

```
Start_Host_Notification (23)
Pause (18)
Set_Session_Parameters (9)
Query_Host_Update (24).
```

Possible code in the EHLLAPI application is as follows:

- Call Sendkeys(@A@H). This sends the SYSREQ command to the session.
- Call StartHostNotify with input B, where B indicates notification of both OIA and PS. This tells the session to notify the EHLLAPI application when the session's OIA and/or PS is updated by the host.
- Call Pause, specifying a sufficient timeout period. This causes the EHLLAPI application to wait until the session notifies it of a host update to the session's OIA and/or PS. This occurs when the session receives the most-awaited host response for the SYSREQ command. Note that if the timeout value has been exceeded, and no host notification has been received, the Pause function call still returns.

Also, for this Pause call to work, you must use the Set_Session_Parameters (9) function call to enable the IPAUSE option. This is required because it tells the Pause API call to return when the host notifies the session of an OIA and/or PS update.

If Pause has returned due to an OIA/PS update (host notification), it has a return value of 26. If this is the case, you are ready to send the host command. Otherwise, you must wait again for the host response.

The EHLLAPI application can continue with the command once it knows that either the OIA or the Presentation Space (or both) has been updated by the host. The QueryHostUpdate is used to check what was updated: that is, whether the OIA alone was updated (return code 21), or the PS alone was updated (return code 22) or whether both the OIA and the PS were updated (return code 23).

For example, the EHLLAPI code might resemble the following part:

```
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 */
```

This is the most appropriate solution for this problem, because the EHLLAPI application waits for the exact minimum time required to allow the session to receive and process the SYSREQ host response, before sending its command input.

Another solution is to add a delay [for example, Sleep(1000)] in the EHLLAPI application between the SYSREQ command and the subsequent command, so that the session has enough time to receive and process the host response. However, this solution is not the best, because the delay might be too little or might be excessive.

Refer to RFC 2355 (TN3270 Enhancements) for more information about the 3270 SYSREQ functionality.

HCL Z and I Emulator for Windows VBHLLAPI sample does not run in FDCC Windows Vista

Problem

The HCL Z and I Emulator for Windows VBHLLAPI sample uses controls provided by comdlg32.ocx, which is not installed in the Federal Desktop Core Configuration (FDCC) of Microsoft Windows Vista.

Cause

VBHLLAPI uses ActiveX and Common Dialog controls that are provided by the Microsoft comdlg32.ocx module. For security purposes, the FDCC of Windows Vista does not contain this particular module.

Resolution

The FDCC version of Windows Vista is customized, and changes are not recommended.

If HLLAPI samples containing VBHLLAPI need to be run, then the comdlg32.ocx module must be copied from a standard Windows Vista machine into the `\Windows\System32\` directory of the FDCC Windows Vista installation.

Then reboot the system for the change to take effect.

Query Reply Data Structures Supported by EHLLAPI

This appendix lists and defines the query reply structures supported by the EHLLAPI structured field interface for PC/3270. Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* or, in the case of an IBM licensed program, the documentation for the specific licensed program.

Note:

1. EHLLAPI must scan the query reply buffers to locate the destination/origin ID (DOID) self-defining parameter (SDP) for the structured field support to work and be reliable. The DOID field is then filled in with the assigned ID.
2. The application should build the query reply data structures in the application's private memory.
3. Only cursory checking is performed on the query reply data. Only the ID and the length of the structure are checked for validity.
4. The 2-byte length field at the beginning of each query reply **is not byte reversed**.

5. Only one distributed data management (DDM) base-type connection is allowed per host session. If the DDM connection supports the SDP for the DOID, multiple connections are allowed.
6. If a nonzero return code is received indicating that an application is already connected to the selected session (RC 32 or 39), use that presentation space with caution. Conflicts with File Transfer, and other EHLLAPI applications might result.

The DDM Query Reply

Several DDM query reply formats are supported. Here are some of them:

Table 79. DDM Query Reply Base Format

Offset	Length	Content	Meaning
0	1 word	Length	Length of structure
2	1 byte	X'81'	Query reply ID
3	1 byte	X'95'	Query reply type
4–5	2 bytes	FLAGS	Reserved
6–7	2 bytes	LIMIN	Maximum DDM bytes allowed in inbound transmission
8–9	2 bytes	LIMOUT	Maximum DDM bytes allowed in outbound transmission
10	1 byte	NSS	Number of subsets identifier
11	1 byte	DDMSS	DDM subset identifier

DDM Application Name Self-Defining Parameter

The DDM application name self-defining parameter provides the host application with the name of the application containing control of the DDM auxiliary device. The controlling application is identified by the DOID in the Direct Access self-defining parameter.

This self-defining parameter is optional, but it is necessary if a host application is to identify a distinct DDM auxiliary device when more than one application is in existence at a remote workstation.

Table 80. DDM Application Name Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	Length	Parameter length
1	1 byte	X'02'	DDM application name
2–n	n-2 bytes	NAME	Name of the remote application program

NAME

The name consists of 8 characters or less and is the means by which a host application can relate to an application in a remote workstation. It is the responsibility of the host and remote application users to ensure that the name is understood by the application at each end.

PCLK Protocol Controls Self-Defining Parameter

The PCLK Protocol Controls self-defining parameter indicates that the PCLK Protocol Controls structured field, ID = X'1013', can be used for both inbound and outbound in data streams destined to or from the DDM auxiliary device processor.

Table 81. DDM PCLK Auxiliary Device Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	X'04'	Parameter length
1	1 byte	X'03'	PCLK protocol controls
2–3	2 bytes	VERS	Protocol version

VERS

The value given in VERS is used to indicate the versions of PCLK installed in the terminal at the time the query reply is returned. For example, X'0001' indicates PCLK Version 1.1.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for the field definitions for this query reply.

Base DDM Query Reply Formats

The following query reply formats are *examples* of some of the Base + SDP (self-defining parameter) combinations possible. Not all of the combinations are shown.

Table 82. Base DDM Query Reply Format with Name and Direct Access Self-Defining Parameters

Offset	Length	Content	Meaning
0	1 word	Length	Length of structure (includes self-defining parameters)
2	1 byte	X'81'	Query reply ID
3	1 byte	X'95'	Query Reply type
4–5	2 bytes	FLAGS	Reserved
6–7	2 bytes	LIMIN	Maximum DDM bytes allowed in inbound transmission
8–9	2 bytes	LIMOUT	Maximum DDM bytes allowed in outbound transmission

Table 82. Base DDM Query Reply Format with Name and Direct Access Self-Defining Parameters (continued)

Offset	Length	Content	Meaning
10	1 byte	NSS	Number of subsets supported
11	1 byte	DDMSS	DDM subset identifier
12	1 byte	Length (n+2)	Parameter length
13	1 byte	X'02'	DDM application name
14– (13+n)	n bytes	Name	Name of the remote application program
14+n	1 byte	X'04'	Parameter length
15+n	1 byte	X'01'	Direct access ID
16+n – 17+n	2 bytes	DOID	Destination/origin ID assigned by the subsystem

The self-defining parameters begin at offsets 12 and (14 + *n*) where *n* is the length of the application name supplied at offset 14.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for the field definitions for this query reply.

Table 83. Base DDM Query Reply Format with Direct Access and Name Self-Defining Parameters

Offset	Length	Content	Meaning
0	1 word	Length	Length of structure (includes self-defining parameters)
2	1 byte	X'81'	Query reply ID
3	1 byte	X'95'	Query reply type
4–5	2 bytes	FLAGS	Reserved
6–7	2 bytes	LIMIN	Maximum DDM bytes allowed in inbound transmission
8–9	2 bytes	LIMOUT	Maximum DDM bytes allowed in outbound transmission
10	1 byte	NSS	Number of subsets supported
11	1 byte	DDMSS	DDM subset identifier
12	1 byte	X'04'	Parameter length
13	1 byte	X'01'	Direct access ID
14–15	2 bytes	DOID	Destination/origin ID assigned by the subsystem

Table 83. Base DDM Query Reply Format with Direct Access and Name Self-Defining Parameters (continued)

Offset	Length	Content	Meaning
16	1 byte	Length (n+2)	Parameter length
17	1 byte	X'02'	DDM application name
16+*n* – 17+*n*	*n* bytes	Name	Name of the remote application program

The self-defining parameters begin at offsets 12 and 16.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for the field definitions for this query reply.

The IBM Auxiliary Device Query Reply

The Auxiliary Device Query Reply is used to indicate to the host application the support of an IBM auxiliary device that uses a data stream defined by IBM, refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for more details.

When the function is supported, the query reply is transmitted inbound in reply to a Read Partition structured field specifying Query or Query List (QCODE List = X'9E', Equivalent, or All).

When a workstation supports multiple auxiliary devices, the IBM auxiliary devices query reply must be sent for each device.

Optional Parameters

All parameters shown in the base part of the query reply must be present. Parameters not used are set to X'00'.

At least one self-defining parameter must be present.

Table 84. IBM Auxiliary Device Base Format with Direct Access Self-Defining Parameter

Offset	Length	Content	Meaning
0–1	1 word	Length	Length of structure (includes self-defining parameters)
2	1 byte	X'81'	Query reply ID
3	1 byte	X'9E'	IBM auxiliary device reply
4	1 byte BIT 0 1–7	FLAGS QUERY B'1' RES	Reserved Read Part (Query, Query List) Auxiliary device supports Query Reserved, must be B'0's
5	1 byte	FLAGS	Reserved
6–7	2 bytes	LIMIN	Maximum DDM bytes allowed in inbound transmission
8–9	2 bytes	LIMOUT	Maximum DDM bytes allowed in outbound transmission

Table 84. IBM Auxiliary Device Base Format with Direct Access Self-Defining Parameter (continued)

Offset	Length	Content	Meaning
10	1 byte	TYPE X'01' X'02' Others	Type of auxiliary device supported IBM auxiliary device display IBM auxiliary device printer Reserved
11	1 byte	X'04'	Parameter length
12	1 byte	X'01'	Direct access
13–14	1 word	DOID	Destination/origin ID assigned by the subsystem

QUERY	This bit must be set to B'1' for all IBM auxiliary devices to indicate that it supports receiving a Read Partition (Query, Query List). The host applications can then use a Read Partition directed to the auxiliary device to determine its characteristics. The destination/origin structured field is used to direct the Read Partition structured field to the auxiliary device. The minimum support level for the IBM auxiliary device is to return the Null query reply in response to the Read Partition.
LIMIN	States the maximum number of bytes that can be sent in an inbound transmission. A LIMIN value of X'0000' indicates no implementation limit on the number of bytes transmitted inbound.
LIMOUT	States the maximum number of bytes that can be sent to an IBM auxiliary device in an outbound transmission. A LIMOUT value of X'0000' indicates no implementation limit on the number of bytes transmitted outbound.
TYPE	Identifies the auxiliary device being supported. Two values are valid. One identifies an auxiliary display and the other identifies an auxiliary printer. All other values are reserved.

The IBM auxiliary device processor supports two self-defining parameters, 01 and 03. These are defined in Table 85: IBM Auxiliary Device Direct Access Self-Defining Parameter on page 726.

Direct Access Self-Defining Parameter

The direct access self-defining parameter provides the ID for use in the destination/origin structured field in the direct access of the IBM auxiliary device.

This SDP is always required to accompany the base query reply.

Table 85. IBM Auxiliary Device Direct Access Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	X'04'	Parameter length
1	1 byte	X'01'	Direct access ID
2–3	2 bytes	DOID	Destination/origin ID

DOID

The value in these bytes is used in the ID field of the destination/origin structured field to identify the auxiliary device as the destination or origin of the data that follows.

PCLK Protocol Controls Self-Defining Parameter

The presence of the PCLK protocol controls self-defining parameter indicates that the PCLK protocol controls structured field, ID = X'1013', can be used for both inbound and outbound in data streams destined to or from the IBM auxiliary device processor.

Table 86. IBM Auxiliary Device PCLK Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	X'04'	Parameter length
1	1 byte	X'03'	PCLK protocol controls
2–3	2 bytes	VERS	Protocol version

VERS

The value given in VERS is used to indicate the versions of PCLK installed in the terminal at the time the query reply is returned. For example, X'0001' indicates PCLK version 1.1.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for the field definitions for this query reply.

The Product-Defined Query Reply

This query reply is used by IBM products using registered subidentifiers within the X'9C' data structure. The Product-Defined Data Stream query reply indicates support of a 3270DS workstation auxiliary device that uses an IBM product-defined data stream. The data stream is *not* defined by a format architecture document having an identifiable control point such as an architecture review board.

When an auxiliary device supports an IBM product-defined data stream, this query reply is transmitted inbound in reply to a Query List (QCODE List = X'9C' or All).

Optional Parameters

All parameters shown in the base part of the query reply and the direct access self-defining parameter must be present.

The format of the Product-Defined query reply is as follows:

Table 87. IBM Product-Defined Query Reply Base Format

Offset	Length	Content	Meaning
0–1	1 word	Length	Length of structure (includes self-defining parameters)

Table 87. IBM Product-Defined Query Reply Base Format (continued)

Offset	Length	Content	Meaning
2	1 byte	X'81'	Query reply ID
3	1 byte	X'9C'	IBM product-defined data stream
4–5	2 bytes	FLAGS	Reserved
6	1 byte	REFID	Reference identifier
7	1 byte	SSID	Subset identifier
8	1 byte	X'04'	Parameter length
9	1 byte	X'01'	Direct access
10–11	1 word	DOID	Destination/origin ID assigned by the subsystem

Valid values for REFID (offset 6) and SSID (offset 7) of the Product-Defined query reply are as follows:

Table 88. Valid REFID and SSID Values for the IBM Product-Defined Query Reply

REFID	SSID	Product and Data Stream Documentation
X'01'		5080 Graphics System: This reference ID indicates the 5080 Graphics System data stream is supported by the auxiliary device. Descriptions of the 5080 Graphics Architecture, structured field, subset ID, DOID, and associated function sets are defined in *IBM 5080 Graphics System Principles of Operation*
	X'01' X'02'	5080 HGFD Graphics Subset 5080 RS232 Ports Subset
X'02'		WHIP API (replaced by SRL name when written) This reference ID indicates that the WHIP API data stream is supported by the auxiliary device. A description of the WHIP API architecture is defined in *IBM RT PC Workstation Host Interface Program Version 1.1 User's Guide and Reference Manual*
	X'01'	WHIP Subset 1
X'03' to X'FF'		All other values are reserved.

The IBM product-defined processor supports only the direct access self-defining parameter. It is defined in Table 89: IBM Product-Defined Direct Access Self-Defining Parameter on page 729.

Direct Access Self-Defining Parameter

The presence of the Direct Access ID self-defining parameter indicates that the auxiliary device can be accessed directly by using the destination/origin structured field. When multiple auxiliary devices are supported that use a product-defined data stream, separate Product-Defined Data Stream query replies must be provided, each of which has a unique DOID.

Table 89. IBM Product-Defined Direct Access Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	X'04'	Parameter length
1	1 byte	X'01'	Direct access ID
2–3	2 bytes	DOID	Destination/origin ID

DOID

The value in these bytes is used in the ID field of the destination/origin structured field to identify the auxiliary device as the destination or origin of the data that follows.

The Document Interchange Architecture Query Reply

This query reply indicates the Document Interchange Architecture (DIA) function set supported. The format of the DIA Query Reply is as follows:

Table 90. IBM DIA Base Format

Offset	Length	Content	Meaning
0	1 word	Length	Length of structure (includes self-defining parameters)
2	1 byte	X'81'	Query reply ID
3	1 byte	X'97'	IBM DIA
4–5	2 bytes	FLAGS	Reserved
6–7	2 bytes	LIMIN	Maximum DDM bytes allowed in inbound transmission
8–9	2 bytes	LIMOUT	Maximum DDM bytes allowed in outbound transmission
10	1 byte	NFS	Number of 3-byte function set IDs that follow
11–13	3 bytes	DIAFS	DIA function set identifier
14– (13+(N*3))	N*3 bytes	DIAFSs	Additional DIA function set IDs
14+(N*3)	1 byte	X'04'	Parameter length
15+(N*3)	1 byte	X'01'	Direct access
16+(N*3)	1 word	DOID	Destination/origin ID assigned by the subsystem

The DIA auxiliary device processor supports only the direct access self-defining parameter. It is defined in Table 91: IBM Product-Defined Direct Access Self-Defining Parameter on page 730.

The presence of the direct access ID self-defining parameter indicates that the auxiliary device can be accessed directly by using the destination/origin structured field.

Table 91. IBM Product-Defined Direct Access Self-Defining Parameter

Offset	Length	Content	Meaning
0	1 byte	X'04'	Parameter length
1	1 byte	X'01'	Direct access ID
2–3	2 bytes	DOID	Destination/origin ID

DOID

The value in these bytes is used in the ID field of the destination/origin structured field to identify the auxiliary device as the destination or origin of the data that follows.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for the field definitions for this query reply.

Differences from Communication Manager/2 EHLLAPI

This appendix describes the differences between EHLLAPI of Z and I Emulator for Windows and EHLLAPI for Communication Manager/2.

The following EHLLAPI functions are different from those with the same names in Communication Manager/2. You need to understand the differences when you use these functions:

- **Set Session Parameter** (9)
- **Copy OIA** (13)
- **Copy String to PS** (15)
- **Storage Manager** (17)
- **Copy String to Field** (33)
- **Get Key** (51)
- **Window Status** (104)
- **Query Sessions** (10)
- **Connect for Structured Field** (120)
- **Allocate Communications Buffer** (123)
- ASCII mnemonics

Set Session Parameter (9)

Set Options

Z and I Emulator for Windows does not provide the following set options provided by Communication Manager:

OLDOIA, NEWOIA
COMPCASE, COMPICASE
OLD5250OIA, NEW5250OIA

Return Parameters

When the **Set Session Parameter** (9) function is terminated, Communication Manager returns a length of the valid data string as the third parameter, the data string length. However, Z and I Emulator for Windows returns a number of the valid set options as the data string length.

EAB Option

In Communication Manager/2, a color remap affects the value of the character color in the EAB attribute copied by **Copy PS** (5) or **Copy PS to String** (8) function when the EAB option is specified in the **Set Session Parameter** (9) function.

In Z and I Emulator for Windows, however, the value of the character color in the EAB attribute depends on the contents of the presentation space regardless of a color remap, and it is not affected by a color remap.

Copy OIA (13)

The **Copy OIA** (13) function has the following differences between Communication Manager/2 and Z and I Emulator for Windows. For more information of the group and the column positions, refer to Copy OIA (13) on page 555.

- Byte Position 21
  - Z and I Emulator for Windows returns X'F6'.
  - Communication Manager/2 returns X'20'.
- Byte Positions 61–63
  - Z and I Emulator for Windows does not return the printer information.
  - Communication Manager/2 returns the printer information.
- Group 3: Shift State

  Communication Manager/2 does not return the value of bit 2. Bit 2 is reserved, and bit 0 contains both the Upper Shift and the Caps Lock.
- Group 8 Byte 1: Input Inhibited
  - Z and I Emulator for Windows does not return bit 6 (Device not working).
  - Communication Manager/2 can return bit 6.
- Group 8 Byte 3: Input Inhibited
  - Z and I Emulator for Windows does not return bit 1 (Operator unauthorized) and bit 2 (Operator unauthorized -f).
  - Communication Manager/2 can return bits 1 and 2.
- Group 8 Byte 4: Input Inhibited
  - Z and I Emulator for Windows does not return bit 2 (System wait).
  - Communication Manager/2 can return bit 2.

- Group 10: Highlight Group 2
    - Z and I Emulator for Windows does not return bit 0 (Selected).
    - Communication Manager/2 can return bit 0.
- Group 11: Color Group 2
    - Z and I Emulator for Windows does not return bit 0 (Selected).
    - Communication Manager/2 can return bit 0.
- Group 13: Printer Status
    - In Z and I Emulator for Windows, this group is reserved.
    - Communication Manager/2 can return this group.
- Group 14: Graphics

  Communication Manager/2 does not return bit 0 (Graphic cursor).

Copy String to PS (15)

In Communication Manager/2, the EAB option of the **Set Session Parameter** (9) function affects the **Copy String to PS** function. When you specify the EAB option, pass the attribute data that has the same size as the text data to the function with the text data.

In Z and I Emulator for Windows, however, the data to be passed is only text data regardless of EAB option. If you want to use the same interface with Communication Manager/2, use the `PUTEAB` option of **Set Session Parameter** (9).

Storage Manager (17)

Storage Manager (17) function provided by Communication Manager/2 is not supported by Z and I Emulator for Windows. Use the APIs provided by Windows® to allocate the memory for the applications.

Copy String to Field (33)

In Communication Manager/2, when the EAB option of the **Set Session Parameter** (9) function is specified, the attribute data is passed to the function as a part of the data. Therefore, when you specify the EAB option, pass the attribute data that has the same size as the text data to the function with the text data.

In Z and I Emulator for Windows, however, the EAB option does not affect the data contents of the **Copy String to Field** (33) function. The data to be passed is not the attribute data, but only the text data. If you want to use the same interface with Communication Manager/2, use the `PUTEAB` option of **Set Session Parameter** (9).

Get Key (51)

Communication Manager/2 returns shift state using @A, @S, or @r, if the shift state of a passed key is not a key or function recognized by the emulator session. Z and I Emulator for Windows does not support these ASCII mnemonics.

Window Status (104)

EHLLAPI function 104 (PM_WINDOW_STATUS) ‘query extended status’ command (0x03) will return the handle of the emulator presentation space window. This is consistent with the definition of the function and the Communication Manager/2 implementation. However, Z and I Emulator for Windows EHLLAPI returns the handle of the frame window. EHLLAPI applications written for Z and I Emulator for Windows using this function need to use the parent of the window handle returned.

Query Sessions (10)

In Communication Manager/2, the descriptor for personal computer is returned. However, the descriptor is not returned in Z and I Emulator for Windows.

Connect for Structured Fields (120)

The event object for communication connection status provided by Communication Manager/2 is not in Z and I Emulator for Windows.

Allocate Communications Buffer (123)

In Communication Manager/2, the maximum value of the requested buffer size is 64 KB minus 8 bytes (X'FFF8').

In Z and I Emulator for Windows, however, it is 64 KB minus 256 bytes (X'FF00').

ASCII Mnemonics

The following ASCII mnemonics are not supported in Z and I Emulator for Windows:

Mnemonics	Meaning
@A@N	Get Cursor
@A@O	Locate Cursor
@A@X	Hexadecimal
@A@Y	Cmd (Function) Key
@A@a	Destructive Backspace
@S@A	Erase EOL
@S@B	Field Advance
@S@C	Field Backspace
@S@D	Valid Character Backspace
@S@P	POR (For sending only)
@S@T	Jump to Task Manager
@/	Overrun of queue (Only in the **Get Key** function)

Get Request Completion (125)

Z and I Emulator for Windows does not support a blank or null session ID.

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

> HCL
> 330 Potrero Ave.
> Sunnyvale, CA 94085
> USA
> Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Host Access Class Library

About This Book

This book provides necessary programming information for you to use the HCL Z and I Emulator for Windows, Version 3.0 Host Access Class Library (HACL). In this book, *Windows*® refers to Windows® 7, Windows® 8, Windows® 8.1, Windows® 10, Windows® Server 2008, and Windows® Server 2012. Throughout this book, *workstation* refers to all supported personal computers. When only one model or architecture of the personal computer is referred to, only that type is specified.

Who Should Read This Book

This book is intended for programmers and developers who write application programs that use the Host Access Class Library (HACL) functions.

A working knowledge of Windows® is assumed. For information about Windows®, see the list of publications under Where to Find More Information on page 736.

This book assumes you are familiar with the language and compiler that you are using. For information on how to write, compile, or link-edit programs, refer to Where to Find More Information on page 736 for the appropriate references for the specific language you are using.

How to Use This Book

This book is organized as follows:

- Introduction on page 736, gives an overview of the Host Access Class Library.
- Host Access Class Library C++ on page 745, describes the Host Access Class Library C++ methods and properties.
- Host Access Class Library Automation Objects on page 984, describes the methods and properties of the Host Access Class Library Automation Objects.
- Host Access Class Library for Java on page 1155, explains where you can find detailed information about the Host Access Class Library (HACL) Java™ classes.
- Sendkeys Mnemonic Keywords on page 1156, contains the mnemonic keywords for the Sendkeys method.
- ECL Planes — Format and Content on page 1159, describes the format and contents of the different data planes in the HACL presentation space model.

Where to Find More Information

The Z and I Emulator for Windows library includes the following publications:

- *Installation Guide*
- *Quick Beginnings*
- *Emulator User's Reference*
- *Administrator's Guide and Reference*
- *Emulator Programming*
- *Host Access Class Library*

In addition to the printed books, there are HTML documents provided with Z and I Emulator for Windows:

Host Access Class Library for Java

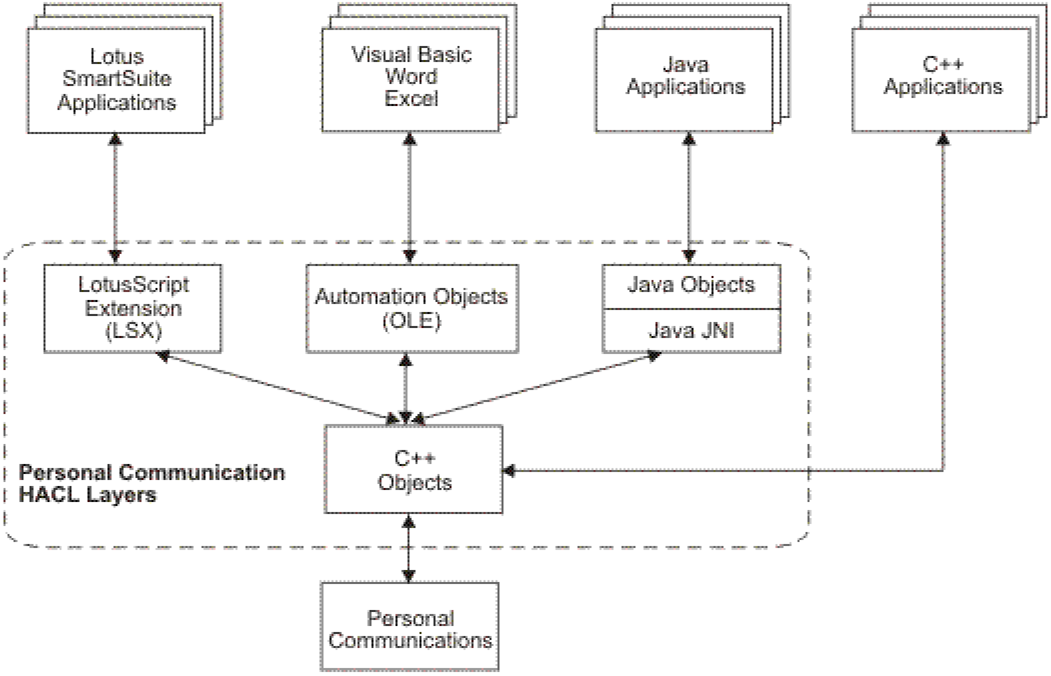
The HACL Java HTML files describe how to write an ActiveX/OLE 2.0-compliant application to use Z and I Emulator for Windows as an embedded object. These files can be accessed from the Docs_Admin_Aids zipped folder delivered along with Z and I Emulator for Windows product documentation in the following path : *ZIEWin_3.0_Docs_Admin_Aids.zip\publications\en_US\doc\hacl*

Introduction

The Host Access Class Library (HACL) is a set of objects that allows application programmers to access host applications easily and quickly. HCL Z and I Emulator for Windows provides support for a wide variety of programming languages and environments by supporting several different HACL layers: C++ objects, Java™ objects, Microsoft® COM-based automation technology (OLE). Each layer provides the same basic functionality, but each layer has some differences due to the different syntax and capabilities of each environment. The most functional and flexible layer is the C++ layer, which provides the basis for all others.

This layering concept allows the basic HACL functions to be used with a wide variety of programming environments including Java™, Microsoft® Visual Basic®, Visual Basic® for Applications, Lotus® Notes™, Lotus® WordPro and Visual C++®. The following figure shows the HACL layers.

Figure 13. HACL Layers



C++ Objects

This C++ class library presents a complete object-oriented abstraction of a host connection that includes: reading and writing the host presentation space (screen), enumerating the fields on the screen, reading the Operator Indicator Area (OIA) for status information, accessing and updating information about the visual emulator window, transferring files, and performing asynchronous notification of significant events.

See Host Access Class Library C++ on page 745 for details on C++ objects.

Java Objects

Java™ objects provides Java™ wrapping for all HACL functions similar to Host-on-Demand Version 3. See Host Access Class Library for Java on page 1155 for details on HACL Java™ classes.

Automation Objects

The Host Access Class Library Automation Objects allow Z and I Emulator for Windows to support the Microsoft® COM-based automation technology (formerly known as OLE automation). The HACL Automation Objects are a series of automation servers that allow automation controllers, for example, Microsoft® Visual Basic®, to programmatically access Z and I Emulator for Windows' data and functionality. In other words, applications that are enabled for controlling the automation protocol (automation controller) can control some of Z and I Emulator for Windows' operations (automation server).

Note: The Automation Objects provided by HCL Z and I Emulator for Windows are 32-bit in nature. These can be used only with 32-bit Microsoft Office programs.

See Host Access Class Library Automation Objects on page 984 for details on the Automation Objects layer.

ECL Concepts

The following sections describe several essential concepts of the *Emulator Class Library* (ECL). Understanding these concepts will aid you in making effective use of the library.

Connections, Handles and Names

In the context of the ECL, a connection is a single, unique Z and I Emulator for Windows emulator window. The emulator window may or may not be actually connected to a host and may or may not be visible on the screen. For instance, a Z and I Emulator for Windows window can be in a disconnected state. Connections are distinguished by their connection handle or by their connection name. Most HACL objects are associated with a specific connection. Typically, the object takes a connection handle or connection name as a parameter on the constructor of the object. For languages like Visual Basic® that do not support parameters on constructors, a member function is supplied for making the association. Once constructed, the object cannot be associated with any other connection. For example, to create an ECLPS (Presentation Space) object associated with connection 'B', the following code would be used:

C++

```
ECLPS *PSObject;
PSObject = new ECLPS('B');
```

Visual Basic®

```
Dim PSObject as Object
Set PSObject = CreateObject("ZIEWin.autECLPS")
PSObject.SetConnectionByName("B")
```

An HACL connection name is a single character from A–Z or a-z. There are a maximum of 52 connection names, and Z and I Emulator for Windows is currently limited to 52 concurrent connections. A connection's name is the same as its EHLLAPI short session ID, and the session ID shown on the Z and I Emulator for Windows window title and OIA.

An HACL handle is a unique 32-bit number that represents a single connection. Unlike a connection name, a connection handle is not limited to 52 values, and the value itself has no significance to the application. You can use a connection handle across threads and processes to refer to the same connection.

For future expansion, applications should use the connection handle whenever possible. Most HACL objects accept a handle or a name when a connection needs to be identified. There are functions available in the base HACL class to convert a handle to a name, and a name to a handle. These functions are available from any HACL object.

Note: Connection properties are dynamic. For example, the connection type returned by GetConnType may change if you reconfigure the connection to a different host. In general, the application should not assume that connection properties remain fixed.

Sessions

In the context of the ECL, a session object (ECLSession) is only a container for all the other connection-specific objects. It provides a shortcut for an application to create a complete set of HACL objects for a particular connection. The term *session* should not be confused with the Z and I Emulator for Windows session concept. A Z and I Emulator for Windows session refers to a physical emulation window on the screen.

Creating or destroying ECLSession objects does not affect Z and I Emulator for Windows sessions (windows). An application can create any number of ECLSession objects that refer to the same or different connections.

ECL Container Objects

Several of the HACL classes act as containers of other objects. For example, the ECLSession object contains an instance of the ECLPS, ECLOIA, ECLWinMetrics, and ECLXfer objects. Containers provide methods to return a pointer to the contained object. For example, the ECLSession object has a GetOIA method, which returns a pointer to an OIA object. Contained objects are not implemented as public members of the container's class, but rather are accessed only through methods.

For performance or other reasons, the contained objects may or may not be created when the container object is created. The class implementation may choose to defer construction of the contained objects until the first time the application requests a pointer to them. The application should not assume that contained objects are created at the same time as the container. For example, an instance of the ECLPS object may not be constructed when an ECLSession object is constructed. Instead, the ECLSession class may delay the construction of the ECLPS object until the first time the GetPS method is called.

When a container class is destroyed, all the contained instances are also destroyed. Any pointers that have been returned to the application become invalid and must not be used.

Note: Some HACL layers (such as the Automation Objects) may hide the containment scheme or recast it into a naming scheme that does not use explicit pointers

ECL List Objects

Several HACL classes provide list iteration capabilities. For example, the ECLConnList class manages the list of connections. ECL list classes are not asynchronously updated to reflect changes in the list content. The application must explicitly call the Refresh method to update the contents of a list. This allows an application to iterate a list without concern that the list may change during the iteration.

Events

The HACL provides the capability of asynchronous notification of certain events. An application can choose to be notified when specific events occur. For example, the application can be notified when a new Z and I Emulator for Windows connection starts. Currently the HACL supports notification for the following events:

- Connection start/stop
- Communications connect/disconnect
- Operator keystrokes
- Presentation space or OIA updates

Notification of events is implemented by the ECLNotify abstract base classes. A separate class exists for each event type. To be notified of an event, the application must define and create an object derived from one of the ECLNotify abstract base classes. That object must then be registered by calling the appropriate HACL registration function. Once an application object is registered, its NotifyEvent method is called whenever the event of interest occurs.

Note:

1. The application’s NotifyEvent method is called asynchronously on a separate thread of execution. Therefore, the NotifyEvent method should be reentrant, and if it accesses application resources, appropriate locking or synchronization should be used.
2. Some HACL layers (such as the Automation Objects) may not fully support or implement HACL events.

Error Handling

At the C++ layer, HACL uses C++ structured exception handling. In general, errors are indicated to the application by the throwing of a C++ exception with an ECLErr object. To catch errors, the application should enclose calls to the HACL objects in a try/catch block such as:

```
try {
   PSObj = new ECLPS('A');
   x = PSObj->GetSize();

   //...more references to HACL objects...

} catch (ECLErr ErrObj) {
   ErrNumber = ErrObj.GetMsgNumber();
   MessageBox(NULL, ErrObj.GetMsgText(), "ECL Error");
}
```

When a HACL error is caught, the application can call methods of the ECLErr object to determine the exact cause of the error. The ECLErr object can also be called to construct a complete language-sensitive error message.

In the Automation Objects layer , runtime errors cause an appropriate scripting error to be created. An application can use an On Error handler to capture the error, query additional information about the error and take appropriate action.

Addressing (Rows, Columns, Positions)

The HACL provides two ways of addressing points (character positions) in the host presentation space. The application can address characters by row/column numbers, or by a single linear position value. Presentation space addressing is always 1-based (not zero-based) no matter what addressing scheme is used.

The row/column addressing scheme is useful for applications that relate directly to the physical screen presentation of the host data. The rectangular coordinate system (with row 1 column 1 in the upper left corner) is a natural way to address points on the screen. The linear positional addressing method (with position 1 in the upper left corner, progressing from left to right, top to bottom) is useful for applications that view the entire presentation space as a single array of data elements, or for applications ported from the EHLLAPI interface which uses this addressing scheme.

At the C++ layer, the different addressing schemes are chosen by calling different signatures for the same methods. For example, to move the host cursor to a given screen coordinate, the application can call the ECLPS::SetCursorPos method in one of two signatures:

```
PSObj->SetCusorPos(81);
PSObj->SetCursorPos(2, 1);
```

These statements have the same effect if the host screen is configured for 80 columns per row. This example also points out a subtle difference in the addressing schemes — the linear position method can yield unexpected results if the application makes assumptions about the number of characters per row of the presentation space. For example, the first line of code in the example would put the cursor at column 81 of row 1 in a presentation space configured for 132 columns. The second line of code would put the cursor at row 2 column 1 no matter what the configuration of the presentation space.

Note: Some HACL layers may expose only a single addressing scheme.

Migrating from EHLLAPI

Applications currently written to the Emulator High Level Language API (EHLLAPI) can be modified to use the Host Access Class Library. In general it requires significant source code changes or application restructuring to migrate from EHLLAPI to HACL. HACL presents a different programming model than EHLLAPI and in general requires a different application structure to be effective.

The following sections will help a programmer familiar with EHLLAPI understand how HACL is similar and how HACL is different than EHLLAPI. Using this information you can understand how a particular application can be modified to use the HACL.

Note: EHLLAPI uses the term *session* to mean the same thing as an HACL *connection*. The terms are used interchangeably in this section.

Execution/Language Interface

At the most fundamental level, EHLLAPI and HACL differ in the mechanics of how the API is called by an application program.

EHLLAPI is implemented as a single call-point interface with multiple-use parameters. A single entry point (hllapi) in a DLL provides all the functions based on a fixed set of four parameters. Three of the parameters take on different meanings depending on the value of the forth command parameter. This simple interface makes is easier to call the API from a variety of programming environments and languages. The disadvantage is a lot of complexity packed into one function and four parameters.

HACL is an object-oriented interface that provides a set of programming objects instead of explicit entry points or functions. The objects have properties and methods that can be used to manipulate a host connection. You do not have to be concerned with details of structure packing and parameter command codes, but can focus on the application functions. HACL objects can only be used from one of the supported HACL layer environments (C++ or Automation Objects). These three layers are accessible to most modern programming environments such as Microsoft® Visual C++®, Visual Basic® and Lotus® SmartSuite® applications.

Features

At a high level, HACL provides a number of features not available at the EHLLAPI level. There are also a few features of EHLLAPI not currently implemented in any HACL class.

HACL unique features include:

- Connection (session) start/stop functions
- Event notification for host communications link connect/disconnect
- Event notification for connection (session) start/stop
- Comprehensive error trapping
- Generation of language-specific error message text
- No architectural limit to the number of connections (sessions); currently, Z and I Emulator for Windows is limited to 52
- Support for multiple concurrent connections (sessions) and multithreaded applications
- Row/column addressing for host presentation space
- Simplified model for presentation space
- Automatic generation of list of fields and attributes
- Keyword-based function key strings

EHLLAPI features not currently implemented in the HACL include:

- Structured field support
- OIA character images
- Lock/unlock presentation space

Session IDs

The HACL architecture is not limited to 52 sessions. Therefore, a single character session ID such as that used in EHLLAPI is not appropriate. The HACL uses the concept of a connection handle, which is a simple 32-bit value that has no particular meaning to the application. A connection handle uniquely identifies a specific connection (session). You can use a connection handle across threads and processes to refer to the same connection.

All HACL objects and methods that need to reference a particular connection accept a connection handle. In addition, for backward compatibility and to allow a reference from the emulator user interface (which does not display the handle), some objects and methods also accept the traditional session ID. The application can obtain a connection handle by enumerating the connections with the ECLConnList object. Each connection is represented by an ECLConnection object. The ECLConnection::GetHandle method can be used to retrieve the handle associated with that specific connection.

It is highly recommended that applications use connection handles instead of connection names (EHLLAPI short session ID). Future implementations of the HACL may prevent applications that use connection names from accessing more than 52 sessions. In some cases it may be necessary to use the name, such as when the user is required to input the name of a specific session the application is to utilize. In the following C+ + example, you supply the name of a session. The application then finds the connection in the connection list and creates PS and OIA objects for that session:

```
ECLConnList        ConnList;  // Connection list
ECLConnection       *ConnFound;  // Ptr to found connection
ECLPS           *PS;        // Ptr to PS object
ECLOIA             *OIA;      // Ptr to OIA object
char              UserRequestedID;

//... user inputs a session name (A-Z or a-z) and it is put
//... into the UserRequesteID variable.  Then...

ConnList.Refresh();  // Update list of connections
ConnFound = ConnList.FindConnection(UserRequestedID);
if (ConnFound == NULL) {
  // Session name given by user does not exist...
}
else {
  // Create PS and OIA objects using handle of the
  // connection just found:
  PS = new ECLPS(ConnFound.GetHandle());
  OIA= new  ECLOIA(ConnFound.GetHandle());

  // The following would also work, but is not the
  // preferred method:
  PS = new ECLPS(UserRequestedID);
  OIA= new ECLOIA(UserRequestedID);
}
```

The second way of creating the PS and OIA objects shown in the example is not preferred because is uses the session name instead of the handle. This creates an implicit 52-session limit in this section of the code. Using the first example shown allows that section of code to work for any number of sessions.

Presentation Space Models

The HACL presentation space model is easier to use than that of EHLLAPI. The HACL presentation space consists of a number of planes, each of which contains one type of data. The planes are:

- Text
- Field attributes
- Color
- Extended attributes

The planes are all the same size and contain one byte for each character position in the host presentation space. An application can obtain any plane of interest using the ECLPS::GetScreen method.

This model is different from the EHLLAPI, in which text and non-text presentation space data is often interleaved in a buffer. An application must set the EHLLAPI session parameter to specify what type of data to retrieve, then make another call to copy the data to a buffer. The HACL model allows the application to get the data of interest in a single call and different data types are never mixed in a single buffer.

SendKey Interface

The HACL method for sending keystrokes to the host (ECLPS::Sendkeys) is similar to the EHLLAPI SendKey function. However, EHLLAPI uses cryptic escape codes to represent non-text keys such as Enter, PF1 and Backtab. The ECLPS object uses bracketed keywords to represent these keystrokes. For example, the following C++ sample would type the characters ABC at the current cursor position, followed by an Enter key:

```
ECLPS  *PS;

PS = new ECLPS('A');   // Get PS object for "A"
PS->SendKeys("ABC[enter]");  // Send keystrokes
```

Events

EHLLAPI provides some means for an application to receive asynchronous notification of certain events. However, the event models are not consistent (some events use semaphores, others use window system messages), and the application is responsible for setting up and managing the event threads. The HACL simplifies all the event handling and makes it consistent for all event types. The application does not have to explicitly create multiple threads of execution, the HACL takes care of the threading internally.

However, you must be aware that the event procedures are called on a separate thread of execution. Access to dynamic application data must be synchronized when accessed from an event procedure. The event thread is spawned when the application registers for the event, and is terminated when the event is unregistered.

PS Connect/Disconnect and Multithreading

An EHLLAPI application must manage a connection to different sessions by calling ConnectPS and DisconnectPS EHLLAPI functions. The application must be carefully coded to avoid being connected to a session indefinitely because sessions have to be shared by all EHLLAPI applications. You must also ensure that an application is connected to a session before using certain other EHLLAPI functions.

The HACL does not require any explicit session connect or disconnect by the application. Each HACL object is associated with a particular connection (session) when it is constructed. To access different connections, the application only needs to create objects for each one. For example, the following example sends the keystrokes ABC to session A, then DEF to session B, and then the Enter key to session A. In an EHLLAPI program, the application would have to connect/disconnect each of the sessions since it can interact with only one at a time. An HACL application can just use the objects in any order needed:

```
ECLPS  *PSA, *PSB;

PSA = new ECLPS('A');
PSB = new ECLPS('B');

PSA->Sendkeys("ABC");
PSB->Sendkeys("DEF");
PSA->Sendkeys("[enter]");
```

For applications that interact with multiple connections (sessions), this can greatly simplify the code needed to manage the multiple connections.

In addition to the single working session, EHLLAPI also places constraints on the multithreaded nature of the application. Connecting to the presentation space and disconnecting from the presentation space has to be managed carefully when the application has more than one thread calling the EHLLAPI interface, and even with multiple threads the application can interact with only one session at a time.

The ECLPS does not impose any particular multithreading restrictions on applications. An application can interact with any number of sessions on any number of threads concurrently.

Host Access Class Library C++

This C++ class library presents a complete object-oriented abstraction of a host connection that includes: reading and writing the host presentation space (screen), enumerating the fields on the screen, reading the Operator Indicator Area (OIA) for status information, accessing and updating information about the visual emulator window, transferring files, and performing asynchronous notification of significant events. The class libraries support Microsoft® Visual C ++® compilers.

The Host Access Class Library C++ layer consists of a number of C++ classes arranged in a class hierarchy. Figure 14: Host Access Class Objects on page 746 illustrates the C++ inheritance hierarchy of the Host Access Class Library C++ layer. Each object inherits from the class immediately above it in the diagram.

Figure 14. Host Access Class Objects

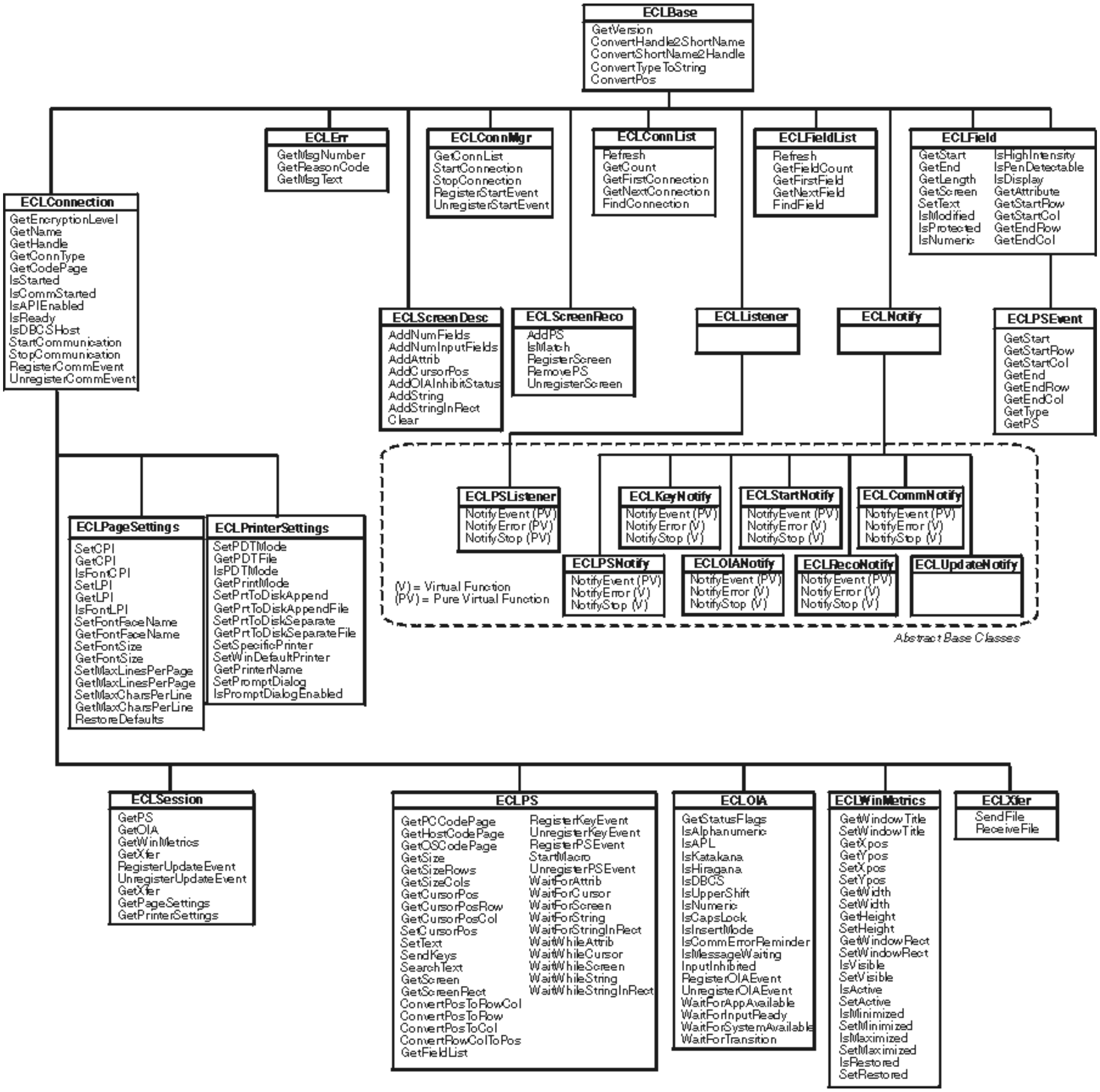


Figure 14: Host Access Class Objects on page 746 also shows all the member functions of each class. Note that in addition to the functions shown for each class, classes inherit all the functions of the parent class. For example, the function IsReady() is available on ECLSession, ECLPS, ECLOIA, ECLWinMetrics, and ECLXfer classes.

Each class is described briefly in the following sections. See the individual class descriptions in this chapter for more details.

All the examples shown in this chapter are supplied in the ECLSAMPS.CPP file. This file can be used to compile and execute any of the examples using any supported compiler.

The following is a brief overview of the Host Access Class Library C++ classes. Each class name begins with ECL, which is the common prefix for the Host Access Class Library.

- ECLBase, on page ECLBase Class on page 749, is the base class for all ECL objects. It provides some basic utility methods such as the conversion of connection names and handles. Because all ECL objects inherit from this class, these methods can be used on any ECL object.
- ECLConnection, on page ECLConnection Class on page 754, represents a single Z and I Emulator for Windows connection and contains connection information such as the connection status, the type of connection (for example, 3270 or 5250), and the name and handle of the connection. This class is also the base class for all the connection-specific ECL objects such as ECLPS and ECLOIA.
- ECLConnList, on page ECLConnList Class on page 769, contains a list of all the Z and I Emulator for Windows connections that were in existence at the time the object was created or the last time the Refresh method was called. Each connection is represented by an ECLConnection object.
- ECLConnMgr, on page ECLConnMgr Class on page 777, enumerates all the currently running Z and I Emulator for Windows connections (windows) using the ECLConnList object. Is also provides methods for starting new connections and stopping connections.
- ECLCommNotify, on page ECLCommNotify Class on page 784, is a notification class that an application can use to be notified whenever a connection is disconnected from or connected to a host. It can be used to monitor the status of a connection and take action when a connection is disconnected unexpectedly.
- ECLErr, on page ECLErr Class on page 789, provides a method for returning run-time error information from Host Access Class Library classes.
- ECLField, on page ECLField Class on page 793, contains information about a single field on the screen, such as the field attributes, field color, position on the screen or length. A method is also supplied to update input fields.
- ECLFieldList, on page ECLFieldList Class on page 810, contains a collection of ECLField objects. When the Refresh method is called, the current host screen is examined, and the list of fields is extracted and used to build the list of ECLField objects. An application can use this collection to manage fields without having to build the list itself.
- ECLKeyNotify, on page ECLKeyNotify Class on page 818, is a notification class that an application can use to be notified of keystroke events. The application can filter (remove) keystrokes, replace them with other keystrokes or discard them.
- ECLListener, on page ECLListener Class on page 823, is the base class for all new HACL event listener objects. It provides common functions for all listener objects.
- ECLOIA, on page ECLOIA Class on page 823, provides access to operator status information such as shift indicators, input inhibited conditions and communications errors.
- ECLOIANotify, on page ECLOIANotify Class on page 837, is an abstract base class. Applications create objects derived from this class to receive notification of OIA changes.
- ECLPS, on page ECLPS Class on page 840, represents the presentation space (screen) of a single connection. It contains methods for obtaining a copy of the screen contents in the form of data planes. Each plane represents a specific aspect of the presentation space, such as the text, field attributes and color attributes. Methods are provided for searching for strings in the presentation space, sending keystrokes to the

host, getting and setting the host cursor position, and many other functions. Also provided is an ECLFieldList object that can be used to enumerate the list of fields on the screen.

- ECLPSEvent, on page ECLPSEvent Class on page 885, is an event object which is passed to PS event listeners when the presentation space has been updated. It contains information about the event including what caused the update and the portion of the screen which has been updated.
- ECLPSListener, on page ECLPSListener Class on page 890, is an abstract base class. Applications create objects derived from this class to receive presentation space update events with all the information provided by the ECLPSEvent object.
- ECLPSNotify, on page ECLPSNotify Class on page 893, is an abstract base class. Applications create objects derived from this class to receive notification of presentation space updates with minimal information.
- ECLRecoNotify, on page ECLRecoNotify Class on page 896, is an abstract base class. Applications create objects derived from this class to receive notifications of screen recognitions.
- ECLScreenDesc, on page ECLScreenDesc Class on page 899, is a class used to describe a single host screen. Screen description class objects are then used to trigger events when the described host screen appears, or to synchronously wait for a particular host screen.
- ECLScreenReco, on page ECLScreenReco Class on page 908, is a class used to collect a set of screen description objects and generate asynchronous events when any of the screens in the collection appear in the presentation space.
- ECLSession, on page ECLSession Class on page 914, contains a collection of all the connection-specific objects. ECLSession can be used to easily create a complete set of objects for a particular connection.
- ECLStartNotify, on page ECLStartNotify Class on page 922,is a notification class that an application can use to be notified whenever a connection is started or stopped. It can be used to monitor the status of the system and take action when a connection is closed unexpectedly.
- ECLUpdateNotify, on page ECLUpdateNotify Class on page 927, is a notification class that an application can use to be notified whenever the host screen or OIA is updated.
- ECLWinMetrics, on page ECLWinMetrics Class on page 927, represents the physical window in which the emulation is running. Methods are provided for getting and setting the window state (min, max, restored), window size and visibility.
- ECLXfer, on page ECLXfer Class on page 950, initiates file transfers to or from the host over the connection.
- ECLPageSettings, on page ECLPageSettings Class on page 956, control and retrieve the settings of the emulator session **File > Page Setup** dialog.
- ECLPrinterSettings, on page ECLPrinterSettings Class on page 969, control and retrieve the settings of the emulator session **File > Printer Setup** dialogs.

Building C++ ECL Programs

This section describes the mechanics of how to build a C++ program which uses the ECL. The source code preparation, compiling and linking requirements are described.

Microsoft Visual C++

The following sections describe how to prepare, compile, and link Microsoft® Visual C++ applications that use the ECL. Z and I Emulator for Windows currently supports Microsoft® Visual C++ compiler Version 4.2 and later.

Source Code Preparation

Programs that use ECL classes must include the ECL header files to obtain the class definitions and other compile-time information. Although it is possible to include only the subset of header files the application requires, for simplicity it is recommended that applications include all ECL header files using the ECLALL.HPP file.

Any C++ source file which contains references to ECL objects or definitions should have the following statement before the first reference:

```
#include "eclall.hpp"
```

Compiling

The compiler must be instructed to search the ZIEWin subdirectory containing the ECL header files. This is done using the /I compiler option, or the Developer Studio Project Setting dialog.

The application must be compiled for multithreaded execution by using the /MT (for executable files), or /MD (for DLLs) compiler options.

Linking

The linker must be instructed to include the ECL linkable library file (PCSECLVC.LIB). This is done by specifying the fully qualified name of the library file on the linker command line, or by using the Developer Studio Project Settings dialog.

Executing

When an application that uses the ECL is executed, the ZIEWin libraries must be found in the system path. By default, the ZIEWin directory is added to the system path during ZIEWin installation.

ECLBase Class

ECLBase is the base class for all ECL objects. It provides some basic utility methods such as the conversion of connection names and handles. Because all ECL objects inherit from this class, these methods can be used on any ECL object.

An application should not create objects of this class directly.

Derivation

None

ECLBase Methods

The following shows the methods that are valid for ECLBase classes.

int GetVersion(void)
char ConvertHandle2ShortName(long ConnHandle)
long ConvertShortName2Handle(char Name)
void ConvertTypeToString(int ConnType,char *Buff)
inline void ConvertPos(ULONG Pos, ULONG *Row, ULONG *Col, ULONG PSCols)

GetVersion

This method returns the version of the Host Access Class Library. The value returned is the decimal version number multiplied by 100. For example, version 1.02 would be returned as 102.

Prototype

int GetVersion(void)

Parameters

None

Return Value

int

The ECL version number multiplied by 100.

Example

```
//-----------------------------------------------------------------
// ECLBase::GetVersion
//
// Display major version number of ECL library.
//-----------------------------------------------------------------
void Sample2() {

if (ECLBase::GetVersion() >= 200) {
  printf("Running version 2.0 or later.\n");
}
else {
  printf("Running version 1.XX\n");
}

} // end sample
```

ConvertHandle2ShortName

This method returns the name (A–Z or a-z) of the ECL connection handle specified. Note that this function may return a name even if the specified connection does not exist.

Prototype

char ConvertHandle2ShortName(long ConnHandle)

Parameters

long ConnHandle

The handle of an ECL connection.

Return Value

char

The name of the ECL connection in the range A–Z or a-z.

Example

```
//-------------------------------------------------------------------
// ECLBase::ConvertHandle2ShortName
//
// Display name of first connection in the connection list.
//-------------------------------------------------------------------
void Sample3() {

ECLConnList  ConnList;
long Handle;
char Name;

if (ConnList.GetCount() > 0) {
  // Print connection name of first connection in the
  // connection list.
  Handle = ConnList.GetFirstConnection()->GetHandle();
  Name = ConnList.ConvertHandle2ShortName(Handle);
  printf("Name of first connection is: %c \n", Name);
}
else printf("There are no connections.\n");

} // end sample
```

ConvertShortName2Handle

This method returns the connection handle of the ECL connection with the specified name. The name must be in the range A–Z or a-z. Note that this function may return a handle even if the specified connection does not exist.

Prototype

char ConvertShortName2Handle(char Name)

Parameters

char Name

The name of an ECL connection in the range A–Z or a-z.

Return Value

char

The handle of the ECL connection.

Example

```
//-------------------------------------------------------------------
// ECLBase::ConvertShortName2Handle
//
// Display handle of connection 'A'.
//-------------------------------------------------------------------
void Sample4() {

ECLConnList  ConnList;
long Handle;
char Name;

Name = 'A';
Handle = ConnList.ConvertShortName2Handle(Name);
printf("Handle of connection A is: 0x%lx \n", Handle);

} // end sample
```

ConvertTypeToString

This method converts a connection type returned by ECLConnection::GetConnType() into a null terminated string. The string returned is not language sensitive.

ConnType	Returned String
HOSTTYPE_3270DISPLAY	"3270 DISPLAY"
HOSTTYPE_3270PRINTER	"3270 PRINTER"
HOSTTYPE_5250 DISPLAY	"5250 PRINTER"
HOSTTYPE_5250PRINTER	"5250 PRINTER"
HOSTTYPE_VT	"ASCII TERMINAL"
HOSTTYPE_PC	"PC SESSION"
Any other value	"UNKNOWN"

Prototype

void ConvertTypeToString(int ConnType,char *Buff)

Parameters

int ConnType

The connection type and must be one of the HOSTTYPE_* constants defined in ECLBASE.HPP.

char *Buff

A buffer of size TYPE_MAXSTRLEN as defined in ECLBase.hpp in which the string will be returned.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLBase::ConvertTypeToString
//
// Display type of connection 'A'.
//-------------------------------------------------------------------
void Sample5() {

ECLConnection  *pConn;
char           TypeString[21];

pConn = new ECLConnection('A');

pConn->ConvertTypeToString(pConn->GetConnType(), TypeString);
// Could also use:
// ECLBase::ConvertTypeToString(pConn->GetConnType(), TypeString);

printf("Session A is a %s \n", TypeString);

delete pConn;

} // end sample
```

ConvertPos

This method is an inline function (macro) to convert an ECL position coordinate into a row/column coordinate given a position and the width of the presentation space. This function is faster than using ECLPS::ConvertPosToRowCol() for applications that already know (or assume) the width of the presentation space.

Prototype

inline void ConvertPos(ULONG Pos,ULONG *Row,ULONG *Col,ULONG PSCols).

Parameters

ULONG Pos

The linear positional coordinate to be converted (input).

ULONG *Row

The pointer to the returned row number of the given position (output).

ULONG *Col

The pointer to the returned column number of the given position (output).

ULONG *PSCols

The number of columns in the host presentation space (input).

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLBase::ConvertPos
//
// Display row/column coordinate of a given point.
//-------------------------------------------------------------------
void Sample6() {

ECLPS     *pPS;
ULONG     NumRows, NumCols, Row, Col;

try {
  pPS = new ECLPS('A');

  pPS->GetSize(&NumRows, &NumCols);  // Get height and width of PS

  // Get row/column coordinate of position 81
  ECLBase::ConvertPos(81, &Row, &Col, NumCols);
  printf("Position 81 is row %lu, column %lu \n", Row, Col);

  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLConnection Class

ECLConnection contains connection-related information for a given connection. This object can be created directly by an application, and is also created indirectly by the ECLConnList object or when creating any object that inherits from ECLConnection (for example, ECLSession).

The information returned by the methods of this object are current as of the time the method is called.

ECLConnection is inherited by ECLSession, ECLPS, ECLOIA, ECLWinMetrics, and ECLXfer.

Derivation

ECLBase > ECLConnection

ECLConnection Methods

The following shows the methods that are valid for ECLConnection classes.

```
ECLConnection(char ConnName)
ECLConnection(long ConnHandle)
~ECLConnection()
long GetHandle()
int GetConnType()
int GetEncryptionLevel()
char GetName()
BOOL IsStarted()
BOOL IsCommStarted()
BOOL IsAPIEnabled()
BOOL IsReady()

unsigned int GetCodePage()
void StartCommunication()
void StopCommunication()
void RegisterCommEvent(ECLCommNotify *NotifyObject, BOOL InitEvent = TRUE)
void UnregisterCommEvent(ECLCommNotify *NotifyObject)
```

ECLConnection Constructor

This method constructs an ECLConnection object from either a connection name or a handle.

Prototype

ECLConnection(long ConnHandle)

ECLConnection(char ConnName)

Parameters

long ConnHandle

Handle of connection to create a connection object.

char ConnName

Name (A–Z or a-z) of connection to create a connection object.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnection::ECLConnection    (Constructor)
//
// Create two connection objects for connection 'A', one created
// by name, the other by handle.
//-------------------------------------------------------------------
void Sample7() {

ECLConnection   *pConn1, *pConn2;
long              Hand;

try {
  pConn1  = new ECLConnection('A');
  Hand    = pConn1->GetHandle();
  pConn2  = new ECLConnection(Hand);  // Another ECLConnection for 'A'

  printf("Conn1 is for connection %c, Conn2 is for connection %c.\n",
         pConn1->GetName(), pConn2->GetName());

  delete pConn1;  // Call destructors
  delete pConn2;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLConnection Destructor

This method destroys an ECLConnection object.

Prototype

~ECLConnection()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnection::~ECLConnection    (Destructor)
//
// Create two connection objects, then delete both of them.
//-------------------------------------------------------------------
void Sample8() {

ECLConnection   *pConn1, *pConn2;
long              Hand;

try {
  pConn1  = new ECLConnection('A');
  Hand    = pConn1->GetHandle();
  pConn2  = new ECLConnection(Hand);  // Another ECLConnection for 'A'

  printf("Conn1 is for connection %c, Conn2 is for connection %c.\n",
         pConn1->GetName(), pConn2->GetName());

  delete pConn1;  // Call destructors
  delete pConn2;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetCodePage

This method returns the host code page for which the connection is configured.

Prototype

unsigned int GetCodePage()

Parameters

None

Return Value

unsigned int

Host code page of the connection.

Example

```
//-------------------------------------------------------------------
// ECLConnection::GetCodePage
//
// Display host code page for each ready connection.
//-------------------------------------------------------------------
void Sample16() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (Info->IsReady())
    printf("Connection %c is configured for host code page %u.\n",
            Info->GetName(), Info->GetCodePage());
}

} // end sample
```

GetHandle

This method returns the handle of the connection. This handle uniquely identifies the connection and may be used in other ECL functions that require a connection handle.

Prototype

long GetHandle()

Parameters

None

Return Value

long

Connection handle of the ECLConnection object.

Example

The following example shows how to return the handle of the first connection in the connection list.

```
//-------------------------------------------------------------------
// ECLConnection::GetHandle
//
// Get the handle of connection 'A' and use it to create another
// connection object.
//-------------------------------------------------------------------
void Sample9() {

ECLConnection   *pConn1, *pConn2;
long              Hand;

try {
  pConn1  = new ECLConnection('A');
  Hand    = pConn1->GetHandle();
  pConn2  = new ECLConnection(Hand);  // Another ECLConnection for 'A'

  printf("Conn1 is for connection %c, Conn2 is for connection %c.\n",
         pConn1->GetName(), pConn2->GetName());

  delete pConn1;  // Call destructors
  delete pConn2;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetConnType

This method returns the connection type. This connection type may change over time (for example, you may reconfigure the connection for a different host). The application should not assume the connection type is fixed. See below for connection types returned.

Note: The ECLBase::ConvertTypeToString function converts the connection type to a null terminated string.

Prototype

int GetConn Type()

Parameters

None

Return Value

int

Connection type constant (HOSTTYPE_* from HOSTBASE.HPP). The following table shows the value returned and its meaning.

Value Returned	Meaning
HOSTTYPE_3270DISPLAY	3270 display
HOSTTYPE_3270PRINTER	3270 printer
HOSTTYPE_5250DISPLAY	5250 display
HOSTTYPE_5250PRINTER	5250 printer
HOSTTYPE_VT	ASCII VT display
HOSTTYPE_UNKNOWN	Unknown connection type

Example

The following example shows how use the GetConnType method to return the connection type.

```
//-------------------------------------------------------------------
// ECLConnection::GetConnType
//
// Find the first 3270 display connection in the current list of
// all connections.
//-------------------------------------------------------------------
void Sample10() {

ULONG      i;             // Connection counter
ECLConnList ConnList;     // Connection list object
ECLConnection *Info=NULL; // Pointer to connection object

for (i=0; i<ConnList.GetCount(); i++) {

  Info = ConnList.GetNextConnection(Info);
  if (Info->GetConnType() == HOSTTYPE_3270DISPLAY) {
    // Found the first 3270 display connection
    printf("First 3270 display connection is '%c'.\n",
            Info->GetName());
    return;
  }

} // for
printf("Found no 3270 display connections.\n");

} // end sample
```

GetName

This method returns the connection name (a single, alphabetic character from A–Z or a-z) of the connection. This name also corresponds to the EHLLAPI session ID.

Prototype

char GetName()

Parameters

None

Return Value

char

Connection short name.

Example

The following example shows how to use the GetName method to return the connection name.

```
//-------------------------------------------------------------------
// ECLConnection::GetName
//
// Find the first 3270 display connection in the current list of
// all connections and display its name (session ID).
//-------------------------------------------------------------------
void Sample11() {

ULONG      i;             // Connection counter
ECLConnList ConnList;     // Connection list object
ECLConnection *Info=NULL; // Pointer to connection object

for (i=0; i<ConnList.GetCount(); i++) {

  Info = ConnList.GetNextConnection(Info);
  if (Info->GetConnType() == HOSTTYPE_3270DISPLAY) {
    // Found the first 3270 display connection, display the name
    printf("First 3270 display connection is '%c'.\n",
            Info->GetName());
    return;
  }

} // for
printf("Found no 3270 display connections.\n");

} // end sample
```

GetEncryptionLevel

This method returns the encryption level of the current connection.

Prototype

int GetEncryptionLevel()

Parameters

None

Return Value

int

Encryption level constant. The following table shows the value returned and its meaning.

Value Returned	Meaning
ENCRYPTION_NONE	No Encryption
ENCRYPTION_40BIT	40 bit encryption
ENCRYPTION_56BIT	56 bit encryption
ENCRYPTION_128BIT	128 bit encryption
ENCRYPTION_168BIT	168 bit encryption
ENCRYPTION_NOKEY	Encrypted without a key

Example

The following example shows how use the GetEncryptionLevel method to return the encryption level.

```
//-------------------------------------------------------
// ECLConnection::GetEncryptionLevel
//
// Display the encryption level of session A
//
//-------------------------------------------------------
void SampleEL()
{
int EncryptionLevel = 0;   //Encryption Level
ECLConnection * Info = NULL;  //Pointer to connection object

Info = new ECLConnection('A');
If (Info != NULL)
{
 EncryptionLevel = Info->GetEncryptionLevel();
 switch (EncryptionLevel)
 {
 case ENCRYPTION_NONE:
  printf("Encryption Level = None");
  break;
 case ENCRYPTION_40BIT:
  printf("Encryption Level = 40 BIT");
  break;
 case ENCRYPTION_56BIT:
  printf("Encryption Level = 56 BIT");
  break;
 case ENCRYPTION_128BIT:
  printf("Encryption Level = 128 BIT");
  break;
 case ENCRYPTION_168BIT:
  printf("Encryption Level = 168 BIT");
  break;

     default:
 }
}
}
```

IsStarted

This method indicates if the connection is started. A started connection may or may not be connected to a host. Use the IsCommStarted function to determine if the connection is currently connected to a host.

Prototype

BOOL IsStarted()

Parameters

None

Return Value

BOOL

TRUE value if the connection is started; FALSE value if the connection is not started.

Example

```
//-------------------------------------------------------------------
// ECLConnection::IsStarted
//
// Display list of all started connections.  Note they may or may
// not be communications-connected to a host, and may or may not
// be visible on the screen.
//-------------------------------------------------------------------
void Sample12() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

// Print list of started connections

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (Info->IsStarted())
    printf("Connection %c is started.\n", Info->GetName());
}

} // end sample
```

IsCommStarted

This method indicates if the connection is currently connected to the host (for example, it indicates if host communications is active for the connection). This function returns a FALSE value if the connection is not started (see IsStarted on page 763).

Prototype

BOOL IsCommStarted()

Parameters

None

Return Value

BOOL

TRUE value if the connection is connected to the host; FALSE value if the connection is not connected to the host.

Example

```
//-------------------------------------------------------------------
// ECLConnection::IsCommStarted
//
// Display list of all started connections which are currently
// in communications with a host.
//-------------------------------------------------------------------
void Sample13() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (Info->IsCommStarted())
    printf("Connection %c is connected to a host.\n", Info->GetName());
}
} // end sample
```

IsAPIEnabled

This method indicates if the connection is API-enabled. A connection that does not have API enabled cannot be used with the Host Access Class Library. This function returns a FALSE value if the connection is not started.

Prototype

BOOL IsAPIEnabled()

Parameters

None

Return Value

BOOL

TRUE value if API is enabled; FALSE value if API is not enabled.

Example

```
//-------------------------------------------------------------------
// ECLConnection::IsAPIEnabled
//
// Display list of all started connections which have APIs enabled.
//-------------------------------------------------------------------
void Sample14() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (Info->IsAPIEnabled())
    printf("Connection %c has APIs enabled.\n", Info->GetName());
}

} // end sample
```

IsReady

This method indicates that the connection is ready, meaning the connection is started, connected, and API-enabled. This function is faster and easier than calling IsStarted, IsCommStarted, and IsAPIEnabled.

Prototype

BOOL IsReady()

Parameters

None

Return Value

BOOL

TRUE if the connection is started, CommStarted, and API-enabled; FALSE if otherwise.

Example

```
//-------------------------------------------------------------------
// ECLConnection::IsReady
//
// Display list of all connections which are started, comm-connected
// to a host, and have APIs enabled.
//-------------------------------------------------------------------
void Sample15() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

   if (Info->IsReady())
    printf("Connection %c is ready (started, comm-connected, API
          enabled).\n", Info->GetName());
}

} // end sample
```

StartCommunication

This method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator communication menu and choosing Connect.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnection::StartCommunication
//
// Start communications link for any connection which is currently
// not comm-connected to a host.
//-------------------------------------------------------------------
void Sample17() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (!(Info->IsCommStarted())) {
    printf("Starting comm-link for connection %c...\n", Info->GetName());
    Info->StartCommunication();
  }
}

} // end sample
```

StopCommunication

This methods disconnects the ZIEWin emulator from the host data stream. This has the same effect as going to the ZIEWin emulator communication menu and choosing Disconnect.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnection::StopCommunication
```

```
//
// Stop comm-link for any connection which is currently connected
// to a host.
//-------------------------------------------------------------------
void Sample18() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object

for (Info = ConnList.GetFirstConnection();
     Info != NULL;
     Info = ConnList.GetNextConnection(Info)) {

  if (Info->IsCommStarted()) {
    printf("Stopping comm-link for connection %c...\n", Info->GetName());
    Info->StopCommunication();
  }
}

} // end sample
```

RegisterCommEvent

This member function registers an application object to receive notification of all communication link connect/disconnect events. To use this function, the application must create an object derived from the ECLCommNotify class. A pointer to that object is then passed to this registration function. *Implementation Restriction:* An application can register only one object for communication event notification.

After a notify object has been registered with this function, it will be called whenever the connections communication link with the host connects or disconnects. The object will receive notification for all communication events whether they are caused by the StartCommunication() function or explicitly by the user. This event should not be confused with the connection start/stop event which is triggered when a new ZIEWin connection starts or stops.

The optional InitEvent parameter causes an initial event to be generated when the object is registered. This can be useful to synchronize an event object with the current state of the communications link. If InitEvent is specified as FALSE, no initial event is generated when the object is registered. The default for this parameter is TRUE.

The application must call UnregisterCommEvent() before destroying the notification object. The object is automatically unregistered if the ECLConnection object where it is registered is destroyed.

See the description of ECLCommNotify Class on page 784 for more information.

Prototype

void RegisterCommEvent(ECLCommNotify *NotifyObject, BOOL InitEvent = TRUE)

Parameters

ECLCommNotify *NotifyObject

Pointer to an object derived from ECLCommNotify class.

BOOL InitEvent

Generate an initial event with the current state.

Return Value

None

Example

See ECLCommNotify Class on page 784 for an example of ECLConnection::RegisterCommEvent.

UnregisterCommEvent

This member function unregisters an application object previously registered for communication events with the RegisterCommEvent() function. A registered application notify object should not be destroyed without first calling this function to unregister it. If there is no notify object currently registered, or the registered object is not the NotifyObject passed in, this function does nothing (no error is thrown).

When a notify object is unregistered, its NotifyStop() member function will be called.

See the description of ECLCommNotify Class on page 784 for more information.

Prototype

void UnregisterCommEvent(ECLCommNotify *NotifyObject)

Parameters

ECLCommNotify *NotifyObject

This is a currently registered application notification object.

Return Value

None

Example

See ECLCommNotify Class on page 784 for an example of ECLConnection::UnregisterCommEvent.

ECLConnList Class

ECLConnList obtains information about all host connections on a given machine. An ECLConnList object contains a collection of all the connections that are currently known in the system.

The ECLConnList object contains a collection of ECLConnection objects. Each element of the collection contains information about a single connection. A connection in this list may be in any state (for example, stopped or disconnected). All started connections appear in this list. The ECLConnection object contains the state of the connection.

The list is a snapshot of the set of connections at the time this object is created, or the last time the Refresh method was called. The list is not dynamically updated as connections are started and stopped. An application can use the RegisterStartEvent member of the ECLConnMgr object to be notified of connection start and stop events.

An ELCConnList object may be created directly by the application or indirectly by the creation of an ECLConnMgr object.

Derivation

ECLBase > ECLConnList

Usage Notes

An ECLConnList object provides a static snapshot of current connections. The Refresh method is automatically called upon construction of the ECLConnList object. If you use the ECLConnList object right after construction it contains an accurate representation of the list of connections at that moment. However, you should call the Refresh method in the ECLConnList object before you start accessing it if some time has passed since its construction.

The application can iterate over the collection by using the GetFirstConnection and GetNextConnection methods. The object pointers returned by GetFirstConnection and GetNextConnection are valid only until the Refresh member is called, or the ECLConnList object is destroyed. The application can locate a specific connection of interest in the list using the FindConnection function. Like GetNextConnection, the returned pointer is valid only until the next Refresh or the ECLConnList object is destroyed.

The order of connections in the connection list is undefined. An application should not make any assumptions about the list order. The order of connections in the list does not change until the Refresh function is called.

An ECLConnList object is automatically created when an ECLConnMgr object is created. However, the ECLConnList object can be created without an ECLConnMgr object.

ECLConnList Methods

The following section describes the methods that are valid for the ECLConnList class.

ECLConnection * GetFirstConnection()
ECLConnection * GetNextConnection(ECLConnection *Prev)
ECLConnection * FindConnection(Long ConnHandle)
ECLConnection * FindConnection(char ConnName)
ULONG GetCount()
void Refresh()

ECLConnList Constructor

This method creates an ECLConnList object and initializes it with the current list of connections.

Prototype

ECLConnList();

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnList::ECLConnList        (Constructor)
//
// Dynamically construct a connection list object, display number
// of connections in the list, then delete the list.
//-------------------------------------------------------------------
void Sample19() {

ECLConnList *pConnList;  // Pointer to connection list object

try {
  pConnList = new ECLConnList();
  printf("There are %lu connections in the connection list.\n",
         pConnList->GetCount());

  delete pConnList;  // Call destructor
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLConnList Destructor

This method destroys an ECLConnList object.

Prototype

~ECLConnList()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnList::~ECLConnList        (Destructor)
//
// Dynamically construct a connection list object, display number
// of connections in the list, then delete the list.
//-------------------------------------------------------------------
void Sample20() {

ECLConnList *pConnList;  // Pointer to connection list object

try {
  pConnList = new ECLConnList();
  printf("There are %lu connections in the connection list.\n",
         pConnList->GetCount());

  delete pConnList;  // Call destructor
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetFirstConnection

The GetFirstConnection method returns a pointer to the first connection information object in the ECLConnList collection. See ECLConnection Class on page 754 for details on its contents. The returned pointer becomes invalid when the ECLConnList Refresh member is called or the ECLConnList object is destroyed. The application should not delete the returned object. If there are no connections in the list, NULL is returned.

Prototype

ECLConnection *GetFirstConnection()

Parameters

None

Return Value

ECLConnection *

Pointer to the first ECLConnection object in the list. If there are no connections in the list, null is returned.

Example

```
//-------------------------------------------------------------------
// ECLConnection::GetFirstConnection
//
// Iterate over list of connections and display information about
// each one.
//-------------------------------------------------------------------
void Sample21() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object
char TypeString[21];     // Type of connection

for (Info = ConnList.GetFirstConnection();       // Get first one
     Info != NULL;                               // While there is one
     Info = ConnList.GetNextConnection(Info)) {  // Get next one

  ECLBase::ConvertTypeToString(Info->GetConnType(), TypeString);
  printf("Connection %c is a %s type connection.\n",
         Info->GetName(), TypeString);
}

} // end sample
```

GetNextConnection

This method returns a pointer to the next connection information object in the ECLConnList collection given a connection in the list. The application supplies a pointer to a connection previously returned by this function or GetFirstConnection. See ECLConnection Class on page 754 for details on its contents. The returned pointer is not valid after the next ECLConnList Refresh() call, or the ECLConnList object is destroyed. A NULL pointer is returned if there is an attempt to read past the end of the list. Successive calls to this method (supplying the prior pointer on each call) iterates over the list of connections. After the last connection is returned, subsequent calls return a NULL pointer. The first connection in the list can be obtained by supplying NULL for the previous connection.

Prototype

ECLConnection *GetNext Connection (ECLConnection *Prev)

Parameters

ECLConnection *Prev

Pointer returned by prior call to this function, GetFirstConnection(), or NULL.

Return Value

ECLConnection *

This is the pointer to the next ECLConnection object, or NULL if end of list.

Example

```
//-------------------------------------------------------------------
// ECLConnection::GetNextConnection
//
// Iterate over list of connections and display information about
// each one.
//-------------------------------------------------------------------
void Sample22() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object
char TypeString[21];     // Type of connection

for (Info = ConnList.GetFirstConnection();       // Get first one
     Info != NULL;                               // While there is one
     Info = ConnList.GetNextConnection(Info)) {  // Get next one

  ECLBase::ConvertTypeToString(Info->GetConnType(), TypeString);
  printf("Connection %c is a %s type connection.\n",
         Info->GetName(), TypeString);
}

} // end sample
```

FindConnection

This method searches the current connection list for the connection specified. The desired connection can be specified by handle or by name. There are two signatures for the FindConnection method. If the specified connection is found, a pointer to the ECLConnection object is returned. If the specified connection is not in the list, NULL is returned. The list is not automatically refreshed by this function; if a new connection has started since the list was constructed or refreshed it is not found. The returned pointer is to an object in the connection list maintained by the ECLConnList object. The returned pointer is invalid after the next ECLConnList::Refresh call or the ECLConnList object is destroyed.

Prototype

ECLConnection *FindConnection(Long ConnHandle),

ECLConnection *FindConnection(char ConnName)

Parameters

Long ConnHandle

Handle of the connection to find in the list.

char ConnName

Name of the connection to find in the list.

Return Value

ECLConnection *

Pointer to the requested ECLConnection object. If the specified connection is not in the list, NULL is returned.

Example

```
//-------------------------------------------------------------------
// ECLConnection::FindConnection
//
// Find connection 'B' in the list of connections.  If found, display
// its type.
//-------------------------------------------------------------------
void Sample23() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object
char TypeString[21];     // Type of connection

Info = ConnList.FindConnection('B');  // Find connection by name
if (Info != NULL) {

  ECLBase::ConvertTypeToString(Info->GetConnType(), TypeString);
  printf("Connection 'B' is a %s type connection.\n",
          TypeString);
}
else printf("Connection 'B' not found.\n");

} // end sample
```

GetCount

This method returns the number of connections currently in the ECLConnList collection.

Prototype

ULONG GetCount()

Parameters

None

Return Value

ULONG

Number of connections in the collection.

Example

```
//-------------------------------------------------------------------
// ECLConnList::GetCount
//
// Dynamically construct a connection list object, display number
// of connections in the list, then delete the list.
//-------------------------------------------------------------------
void Sample24() {

ECLConnList *pConnList;  // Pointer to connection list object

try {
  pConnList = new ECLConnList();
  printf("There are %lu connections in the connection list.\n",
         pConnList->GetCount());

  delete pConnList;  // Call destructor
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

Refresh

This method updates the ECLConnList collection with a list of all currently known connections in the system. All pointers previously returned by GetNextConnection, GetFirstConnection and FindConnection become invalid.

Prototype

void Refresh()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnection::Refresh
//
// Loop-and-wait until connection 'B' is started.
//-------------------------------------------------------------------
void Sample25() {

ECLConnection *Info;     // Pointer to connection object
ECLConnList ConnList;    // Connection list object
int i;

printf("Waiting up to 60 seconds for connection B to start...\n");
for (i=0; i<60; i++) {   // Limit wait to 60 seconds
  ConnList.Refresh();    // Refresh the connection list
  Info = ConnList.FindConnection('B');
  if ((Info != NULL) && (Info->IsStarted())) {
    printf("Connection B is now started.\n");
    return;
  }
  Sleep(1000L);          // Wait 1 second and try again
}

printf("Connection 'B' not started after 60 seconds.\n");

} // end sample
```

ECLConnMgr Class

ECLConnMgr manages all Z and I Emulator for Windows connections on a given machine. It provides methods relating to the management of connections such as starting and stopping connections. It also creates an ECLConnList object to enumerate the list of all known connections on the system (see ECLConnList Class on page 769).

Derivation

ECLBase > ECLConnMgr

ECLConnMgr Methods

The following shows the methods that are valid with the ECLConnMgr class.

ECLConnMgr()

~ECLConnMgr()

ECLConnList * GetConnList()

void StartConnection(char *ConfigParms)

void StopConnection(Long ConnHandle, char *StopParms)

void RegisterStartEvent(ECLStartNotify *NotifyObject)

void UnregisterStartEvent(ECLStartNotify *NotifyObject)

ECLConnMgr Constructor

This method constructs an ECLConnMgr object.

Prototype

ECLConnMgr()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::ECLConnMgr          (Constructor)
//
// Create a connection mangager object, start a new connection,
// then delete the manager.
//-------------------------------------------------------------------
void Sample26() {

ECLConnMgr   *pCM; // Pointer to connection manager object

try {
  pCM = new ECLConnMgr();  // Create connection manager
  pCM->StartConnection("profile=coax connname=e");
  printf("Connection 'E' started with COAX profile.\n");
  delete pCM;              // Delete connection manager
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLConnMgr Deconstructor

This method destroys an ECLConnMgr object.

Prototype

~ECLConnMgr()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::~ECLConnMgr          (Destructor)
//
// Create a connection mangager object, start a new connection,
// then delete the manager.
//-------------------------------------------------------------------
void Sample27() {

ECLConnMgr   *pCM; // Pointer to connection manager object

try {
  pCM = new ECLConnMgr();  // Create connection manager
  pCM->StartConnection("profile=coax connname=e");
  printf("Connection 'E' started with COAX profile.\n");
  delete pCM;              // Delete connection manager
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetConnList

This method returns a pointer to an ECLConnList object. See ECLConnList Class on page 769 for more information. The ECLConnList object is destroyed when the ECLConnMgr object is destroyed.

Prototype

ECLConnList * GetConnList()

Parameters

None

Return Value

ECLConnList *

Pointer to an ECLConnList object

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::GetConnList
//
// Use connection manager's connection list object to display
// number of connections (see also ECLConnList::GetCount).
//-------------------------------------------------------------------
void Sample28() {

ECLConnMgr   CM; // Connection manager object

printf("There are %lu connections in the connection list.\n",
       CM.GetConnList()->GetCount());

} // end sample
```

StartConnection

This method starts a new Z and I Emulator for Windows emulator connection. The ConfigParms string contains connection configuration information as explained under Usage Notes on page 781.

Prototype

void StartConnection(char *ConfigParms)

Parameters

char *ConfigParms

Null terminated connection configuration string.

Return Value

None

Usage Notes

The connection configuration string is implementation-specific. Different implementations of the Host Access Class Library may require different formats or information in the configuration string. This call is asynchronous in nature; the new connection may not yet be started when this call returns. An application can use the RegisterStartEvent function to be notified when a connection starts.

For Z and I Emulator for Windows, the configuration string has the following format:

```
PROFILE=[\"]<filename>[\"] [CONNNAME=<c>] [WINSTATE=<MAX|MIN|RESTORE|HIDE>]
```

Optional parameters are enclosed in square brackets []. The parameters are separated by at least one blank. Parameters may be in upper, lower, or mixed case and may appear in any order. The meaning of each parameter is as follows:

PROFILE=<filename>

Names the Z and I Emulator for Windows workstation profile (.WS file) that contains the connection configuration information. This parameter is not optional; a profile name must be supplied. If the file name contains blanks, the name must be enclosed in double quotation marks. The <filename> value may be either the profile name with no extension, the profile name with the .WS extension, or the fully-qualified profile name path.

CONNNAME=<c>

Specifies the connection name (EHLLAPI short session ID) of the new connection. This value must be a single, alphabetic character (A-Z or a-z). If this value is not specified, the next available connection name is assigned automatically. If a connection already exists with the specified name an error is thrown (ERRMAJ_INVALID_SESSION).

WINSTATE=<MAX|MIN|RESTORE|HIDE>

Specifies the initial state of the emulator window. The default if this parameter is not specified is RESTORE.

Note: Due to the asynchronous nature of this call, it is possible for this function to return without error, but the connection fails to start. For example, if two connections are started in a short period of time with the same connection name the second StartConnection does not fail because the first connection has not yet started. However, when the second connection finally attempts to register its name it does fail to start because the name is already in use by the first connection. To minimize this possibility, connections should be started without specifying the CONNNAME parameter if possible.

Example

The following is an example of the StartConnection method.

```
ECLConnMgr Manager;  // Connection manager object

// Start a host connection "E" and check for errors
```

```
try {
  Manager.StartConnection("profile=coax connname=e");
}
catch (ECLErr Error) {
  MessageBox(NULL, Error.GetMsgText(), "Session start error!", MB_OK);
}
```

StopConnection

This method stops (terminates) the emulator connection identified by the connection handle. See Usage Notes on page 782 for contents of the StopParms string.

Prototype

void StopConnection(Long ConnHandle, char *StopParms)

Parameters

Long ConnHandle

Handle of the connection to be stopped.

char * StopParms

Null terminated connection stop parameter string.

Return Value

None

Usage Notes

The connection stop parameter string is implementation-specific. Different implementations of the Host Access Class Library may require a different format and contents of the parameter string. For Z and I Emulator for Windows the string has the following format:

```
[SAVEPROFILE=<YES|NO|DEFAULT>]
```

Optional parameters are enclosed in square brackets []. The parameters are separated by at least one blank. Parameters may be in upper, lower, or mixed case and may appear in any order. The meaning of the SAVEPROFILE parameter is as follows:

SAVEPROFILE=<YES|NO|DEFAULT> controls the saving of the current connection configuration back to the workstation profile (.WS file). This causes the profile to be updated with any configuration changes you may have made during the connection. If NO is specified, the connection is stopped and the profile is not updated. If YES is specified, the connection is stopped and the profile is updated with the current (possibly changed) configuration. If DEFAULT is specified, the update option is controlled by the **File->Save On Exit** emulator menu option. If this parameter is not specified, DEFAULT is used.

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::StopConnection
//
// Stop the first connection in the connection list.
//-------------------------------------------------------------------
void Sample29() {

ECLConnMgr   CM; // Connection manager object

if (CM.GetConnList()->GetCount() > 0) {

  printf("Stopping connection %c.\n",
         CM.GetConnList()->GetFirstConnection()->GetName());

  CM.StopConnection(
    CM.GetConnList()->GetFirstConnection()->GetHandle(),
    "saveprofile=no");
}
else printf("No connections to stop.\n");


} // end sample
```

RegisterStartEvent

This method registers an application object to receive notification of all connection start and stop events. To use this function, the application must create an object derived from the ECLStartNotify class. A pointer to that object is then passed to this registration function. *Implementation Restriction:* An application can register only one object for connection start or stop notification.

After a notify object has been registered with this function, it is called whenever a Z and I Emulator for Windows connection is started or stopped. The object receives notification for all connections whether they are started by the StartConnection function or explicitly by you. This event should not be confused with the start/stop Communication event, which is triggered when a connection connects or disconnects from a host system.

See ECLStartNotify Class on page 922 for more information.

Prototype

void RegisterStartEvent(ECLStartNotify *NotifyObject)

Parameters

ECLStartNotify *NotifyObject

Pointer to object derived from the ECLStartNotify class.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::RegisterStartEvent
//
// See ECLStartNotify Class on page 922 for example of this method.
//---------------------------------------------------------
```

UnregisterStartEvent

This method unregisters an application object previously registered for connection start or stop events with the RegisterStartEvent function. A registered application notify object should not be destroyed without first calling this function to unregister it. If there is no notify object currently registered, or the registered object is not the NotifyObject passed in, this function does nothing (no error is thrown).

When a notify object is unregistered, its NotifyStop method is called.

See ECLStartNotify Class on page 922 for more information.

Prototype

void UnregisterStartEvent(ECLStartNotify *NotifyObject)

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLConnMgr::UnregisterStartEvent
//
// See ECLStartNotify Class on page 922 for example of this method.
//-------------------------------------------------------------------
```

ECLCommNotify Class

ECLCommNotify is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLCommNotify. The application must implement the NotifyEvent() member function in its derived class. It may also optionally implement NotifyError() and NotifyStop() member functions.

The ECLCommNotify class is used to allow an application to be notified of communications connect/disconnect events on a ZIEWin connection. Connect/disconnect events are generated whenever a ZIEWin connection (window) is connected or disconnected from a host system.

To be notified of communications connect/disconnect events, the application must perform the following steps:

1. Define a class derived from ECLCommNotify.
2. Implement the derived class and implement the NotifyEvent() member function.
3. Optionally implement the NotifyError() function, NotifyStop() function or both.
4. Create an instance of the derived class.
5. Register the instance with the ECLConnection::RegisterCommEvent() function.

The example shown demonstrates how this may be done. When the above steps are complete, each time a connection's communications link is connected or disconnected from a host, the applications NotifyEvent() member function will be called.

If an error is detected during event generation, the NotifyError() member function is called with an ECLErr object. Events may or may not continue to be generated after an error, depending on the nature of the error. When event generation terminates (either due to an error, by calling the ECLConnection::UnregisterCommEvent, or by destruction of the ECLConnection object) the NotifyStop() member function is called. However event notification is terminated, the NotifyStop() member function is always called, and the application object is unregistered.

If the application does not provide an implementation of the NotifyError() member function, the default implementation is used (a simple message box is displayed to the user). The application can override the default behavior by implementing the NotifyError() function in the applications derived class. Likewise, the default NotifyStop() function is used if the application does not provide this function (the default behavior is to do nothing).

Note that the application can also choose to provide its own constructor and destructor for the derived class. This can be useful if the application wants to store some instance-specific data in the class and pass that information as a parameter on the constructor. For example, the application may want to post a message to an application window when a communications event occurs. Rather than define the window handle as a global variable (so it would be visible to the NotifyEvent() function), the application can define a constructor for the class which takes the window handle and stores it in the class member data area.

The application must not destroy the notification object while it is registered to receive events.

Implementation Restriction: Currently the ECLConnection object allows only one notification object to be registered for communications event notification. The ECLConnection::RegisterCommEvent will throw an error if a notify object is already registered for that ECLConnection object.

Derivation

ECLBase > ECLNotify > ECLCommNotify

Example

```
//-------------------------------------------------------------------
// ECLCommNotify class
//
// This sample demonstrates the use of:
//
// ECLCommNotify::NotifyEvent
// ECLCommNotify::NotifyError
// ECLCommNotify::NotifyStop
// ECLConnection::RegisterCommEvent
// ECLConnection::UnregisterCommEvent
//-------------------------------------------------------------------


 //...................................................................
// Define a class derived from ECLCommNotify
//...................................................................
class MyCommNotify: public ECLCommNotify
{
public:
  // Define my own constructor to store instance data
  MyCommNotify(HANDLE DataHandle);

  // We have to implement this function
  void NotifyEvent(ECLConnection *ConnObj, BOOL Connected);

  // We choose to implement this function
  void NotifyStop (ECLConnection *ConnObj, int Reason);

  // We will take the default behaviour for this so we
  // don't implement it in our class:
  // void NotifyError (ECLConnection *ConnObj, ECLErr ErrObject);

private:
  // We will store our application data handle here
  HANDLE MyDataH;
};
```

```
//...................................................................
void MyCommNotify::NotifyEvent(ECLConnection *ConnObj,
                           BOOL Connected)
//
// This function is called whenever the communications link
// with the host connects or disconnects.
//
// For this example, we will just write a message.  Note that we
// have access the the MyDataH handle which could have application
// instance data if we needed it here.
//
// The ConnObj pointer is to the ECLConnection object upon which
// this event was registered.
```

```
//...................................................................
{
  if (Connected)
    printf("Connection %c is now connected.\n", ConnObj->GetName());
  else
    printf("Connection %c is now disconnected.\n", ConnObj->GetName());

  return;
}

 //...................................................................
MyCommNotify::MyCommNotify(HANDLE DataHandle)   // Constructor
//...................................................................
{
  MyDataH = DataHandle;  // Save data handle for later use
}

//...................................................................
void MyCommNotify::NotifyStop(ECLConnection *ConnObj,
                         int Reason)
//...................................................................
{
  // When notification ends, display message
  printf("Comm link monitoring for %c stopped.\n", ConnObj->GetName());
}

 //...................................................................
// Create the class and start notification on connection 'A'.
//...................................................................
void Sample30() {

ECLConnection *Conn;    // Ptr to connection object
MyCommNotify *Event;    // Ptr to my event handling object
HANDLE InstData;      // Handle to application data block (for example)

try {
  Conn = new ECLConnection('A');         // Create connection obj
  Event = new MyCommNotify(InstData);    // Create event handler

  Conn->RegisterCommEvent(Event);        // Register for comm events

  // At this point, any comm link event will cause the
  // MyCommEvent::NotifyEvent() function to execute.  For
  // this sample, we put this thread to sleep during this
  // time.

  printf("Monitoring comm link on 'A' for 60 seconds...\n");
  Sleep(60000);


  // Now stop event generation.  This will cause the NotifyStop
  // member to be called.
  Conn->UnregisterCommEvent(Event);

  delete Event;  // Don't delete until after unregister!
  delete Conn;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLCommNotify Methods

The following section describes the methods that are valid for the ECLCommNotify class:

ECLCommNotify()
~ECLCommNotify()
virtual void NotifyEvent (ECLConnection *ConnObj, BOOL Connected) = 0
virtual void NotifyError (ECLConnection *ConnObj, ECLErr ErrObject)
virtual void NotifyStop (ECLConnection *ConnObj, int Reason)

NotifyEvent

This method is a “pure virtual” member function (the application *must* implement this function in classes derived from ECLCommNotify). This function is called whenever a connection starts or stops and the object is registered for start/stop events. The Connected BOOL is TRUE if the communications link is connected, or FALSE if it is not connected to the host.

Prototype

virtual void NotifyEvent (ECLConnection *ConnObj, BOOL Connected)

Parameters

ECLConnection *ConnObj

This is the pointer to ECLConnection object where the event occurred.

BOOL Connected

This is TRUE if comm link is connected and FALSE if disconnected.

Return Value

None

NotifyError

This method is called whenever the ECLConnection object detects an error during event generation. The error object contains information about the error (see ECLErr Class on page 789). Events may continue to be generated after the error, depending on the nature of the error. If the event generation stops due to an error, the NotifyStop() function is called. An application can choose to implement this function or allow the ECLCommNotify base class to handle the error. The base class will display the error in a message box using the text supplied by the ECLErr::GetMsgText() function. If the application implements this function in its derived class, it will override the base class function.

Prototype

virtual void NotifyError (ECLConnection *ConnObj, ECLErr ErrObject)

Parameters

ECLConnection *ConnObj

This is the pointer to ECLConnection object in which the error occurred.

ECLErr ErrObject

This is the ECLErr object describing the error.

Return Value

None

NotifyStop

This method is called when event generation is stopped for any reason (for example, due to an error condition or a call to ECLConnection::UnregisterCommEvent, etc.).

Implementation Note: the reason code is currently unused and will be zero.

Prototype

virtual void NotifyStop (ECLConnection *ConnObj, int Reason)

Parameters

ECLConnection *ConnObj

This is the ptr to ECLConnection object that is stopping notification.

int Reason

This is unused (zero).

Return Value

None

ECLErr Class

The ECLErr class provides a method of returning run-time error information from Host Access Class Library classes. In error situations, ECLErr objects are created and populated with error and diagnostic information. The ECLErr objects are then thrown as C++ exceptions. The error and diagnostic information can then be queried from the caught ECLErr object.

Applications should not create or throw ECLErr objects directly.

Derivation

ECLBase > ECLErr

ECLErr Methods

The following section describes the methods that are valid for the ECLErr class.

const int GetMsgNumber()

const int GetReasonCode()

const char *GetMsgText()

GetMsgNumber

This method returns the message number that was set when this ECLErr object was created. Error message numbers are described in ERRORIDS.HPP.

Prototype

const int GetMsgNumber()

Parameters

None

Return Value

const int

The error message number.

Example

```
//-------------------------------------------------------------------
// ECLErr::GetMsgNumber
//
// Cause an 'invalid parameters' error and tryp the ECL exception.
// The extract the error number and language-sensative text.
```

```
//-------------------------------------------------------------------
void Sample31() {

ECLPS   *PS = NULL;

try {
  PS = new ECLPS('A');
  PS->SetCursorPos(999,999);  // Invalid parameters
}
catch (ECLErr ErrObj) {
  printf("The following ECL error was trapped:\n");
  printf("%s \nError number: %lu\nReason code: %lu\n",
    ErrObj.GetMsgText(),
    ErrObj.GetMsgNumber(),
    ErrObj.GetReasonCode());
}

if (PS != NULL)
  delete PS;

} // end sample
```

GetReasonCode

This method gets the reason code (sometimes referred to as the secondary or minor return code) from the ECLErr object. This code is generally used for debugging and diagnostic purposes. It is subject to change in future versions of the Host Access Class Library and should not be used programmatically. Descriptions of the reason codes can be found in ERRORIDS.HPP.

Prototype

const int GetReasonCode()

Parameters

None

Return Value

const int

The ECLErr reason code.

Example

```
//-------------------------------------------------------------------
// ECLErr::GetReasonCode
//
// Cause an 'invalid parameters' error and tryp the ECL exception.
// The extract the error number and language-sensative text.
//-------------------------------------------------------------------
```

```
void Sample32() {

ECLPS   *PS = NULL;

try {
  PS = new ECLPS('A');
  PS->SetCursorPos(999,999);  // Invalid parameters
}
catch (ECLErr ErrObj) {
  printf("The following ECL error was trapped:\n");
  printf("%s \nError number: %lu\nReason code: %lu\n",
    ErrObj.GetMsgText(),
    ErrObj.GetMsgNumber(),
    ErrObj.GetReasonCode());
}

if (PS != NULL)
  delete PS;

} // end sample
```

GetMsgText

This method returns the message text associated with the error code used to create this ECLErr object. The message text is returned in the language for which Z and I Emulator for Windows is currently installed.

Note: The returned pointer is invalid after the ECLErr object is deleted.

Prototype

const char *GetMsgText()

Parameters

None

Return Value

char *

The message text associated with the error code that is part of this ECLErr object.

Example

```
//-------------------------------------------------------------------
// ECLErr::GetMsgText
//
// Cause an 'invalid parameters' error and tryp the ECL exception.
// The extract the error number and language-sensative text.
//-------------------------------------------------------------------
```

```
void Sample33() {

ECLPS   *PS = NULL;

try {
  PS = new ECLPS('A');
  PS->SetCursorPos(999,999);  // Invalid parameters
}
catch (ECLErr ErrObj) {
  printf("The following ECL error was trapped:\n");
  printf("%s \nError number: %lu\nReason code: %lu\n",
    ErrObj.GetMsgText(),
    ErrObj.GetMsgNumber(),
    ErrObj.GetReasonCode());
}

if (PS != NULL)
  delete PS;

} // end sample
```

Usage Notes

The message text is retrieved from the Z and I Emulator for Windows message facility.

ECLField Class

ECLField contains information for a given field in an ECLFieldList object contained by an ECLPS object. An application should not create an object of this type directly. ECLField objects are created indirectly by the ECLFieldList object.

An ECLField object describes a single field of the host presentation space. It has methods for querying various attributes of the field and for updating the text of the field (for example, modifying the field text). Field attributes cannot be modified.

Derivation

ECLBase > ECLField

Copy-Constructor and Assignment Operator

This object supports copy-construction and assignment. This is useful for an application that wants to easily capture fields on a host screen for later processing. Rather than allocate text buffers and copy the string contents of the field, the application can simply store the field in a private ECLField object. The stored copy retains all the function of an ECLField object including the field's text value, attributes, starting position, length, etc. For example, suppose an application wanted to capture the first input field of the screen. Table 92: Copy-Construction and Assignment Examples on page 794 shows two ways this could be accomplished.

Table 92. Copy-Construction and Assignment Examples

<table>
<tr><th>Save the field as a string</th><th>Save the field as an ECLField object</th></tr>
<tr><td><pre>
#include "eclall.hpp"

{
   char *SavePtr; // Ptr to saved string
   ECLPS Ps('A'); // PS object
   ECLFieldList *List;
   ECLField   *Fld;

   // Get fld list and rebuild it
   List = Ps->GetFieldList();
   List->Refresh();

   // See if there is an input field
   Fld = List->GetFirstField(GetUnmodified);
   if (Fld !=NULL) {
      // Copy the field's text value
      SavePtr=malloc(Fld->Length() + 1);
      Fld->GetScreen(SavePtr, Fld->Length()+1);
  }

  // We now have captured the field text
</pre></td><td><pre>
#include "eclall.hpp"

{
   ECLField SaveFld; // Saved field
   ECLPS Ps('A');   // PS object
   ECLFieldList *List;
   ECLField   *Fld;

   // Get fld list and rebuild it
   List = Ps->GetFieldList();
   List->Refresh();

   // See if there is an input field
   Fld = List->GetFirstField(GetUnmodified);
   if (Fld !=NULL) {
      // Copy the field object
      SaveFld = *Fld;
  }

  // We now have captured the field text
  // including text, position, attrib
</pre></td></tr>
</table>

There are several advantages to using an ECLField object instead of a string to store a field:

- The ECLField object does all storage management of the field's text buffer; the application does not have to allocate or free text buffers or calculate the size of the buffer required.
- The saved field retains all of the characteristics of the original field including its attributes and starting position. All of the usual ECLField member functions can be used on the stored field except SetText(). Note that the stored field is a copy of the original — its values are not updated when the host screen changes or when the ECLFieldList::Refresh() function is called. As a result, the field can be stored and used later in the application.

Assignment operator overrides are also provided for character strings and long integer value types. These overrides make it easy to assign new string or numeric values to unprotected fields. For example, the following sets the first two input fields of the screen:

```
ECLField *Fld1; //Ptr to 1st unprotected field in field list
ECLField *Fld2; // PTR to 2nd unprotected field in field list

Fld1 = FieldList->GetFirstField(GetUnprotected);
Fld2 = FieldList->GetNextField(Fld1, GetUnprotected);
if ((Fld1 == NULL) || (Fld2 == NULL)) return;

*Fld1 = "Easy string assignment";
*Fld2 = 1087;
```

Notes:

1. ECLField objects initialized by copy-construction or assignment are read-only copies of the original field object. The SetText() method is invalid for such an object and will cause an ECLErr exception to be thrown. Because the objects are copies, they are not updated or deleted when the original field object is updated or deleted. The application is responsible for deleting copies of field objects when they are no longer needed.
2. Calling any method on an unitialized ECLField object will return undefined results.
3. An ECLField object created by the application can be reassigned any number of times.
4. Assignments can only be made from another ECLField object, a character string, or a long integer value. Assigning any other data type to an ECLField object is invalid.
5. If an assignment is made to an ECLField object that currently is part of an ECLFieldList, the effect is to update only the field's text value. This is allowed only if the field object is an unprotected field. For example, the following will modify the 2nd input field of the screen by copying the value from the 1st input field:

   ```
   ECLField *Fld1;   // Ptr to 1st unprotected field in field list
   ECLField *Fld2;   // Ptr to 2nd unprotected field in field list

   Fld1 = FieldList->GetFirstField(GetUnprotected);
   Fld2 = FieldList->GetNextField(Fld1, GetUnprotected);
   if ((Fld1 == NULL) || (Fld2 == NULL)) return;

   // Update the 2nd input field using text from the first
   FLD2 = * Fld1;
   ```

   Because Fld2 is part of an ECLFieldList, the above assignment is identical to:

   ```
   { char temp[Fld1->GetLength()+1];
     Fld1->GetText(temp, Fld1->GetLength()+1);
     Fld2->SetText(temp);
     delete []temp;
   }
   ```

   Note that this will throw an ECLErr exception if Fld2 is protected. Also note that only the text of Fld2 is updated, not its attributes, position, or length.
6. Assigning a string to a field object is equivalent to calling the SetText() method. You can also assign numeric values without first converting to strings:

   ```
   *Field = 1087;
   ```

   This is equivalent to converting the number to a string and then calling the SetText() method.

ECLField Methods

The following section describes the methods that are valid for the ECLField class.

```
ULONG GetStart()
void GetStart(ULONG *RowULONG *Col)
ULONG GetStartRow()
ULONG GetStartCol()
ULONG GetEnd()
void GetEnd(ULONG *RowULONG *Col)
ULONG GetEndRow()
ULONG GetEndCol()
ULONG GetLength()
ULONG GetScreen(char *Buff, ULONG BuffLen, PS_PLANE Plane = TextPlane)
void SetText(char *text)
BOOL IsModified()
BOOL IsProtected()
BOOL IsNumeric()
BOOL IsHighIntensity()
BOOL IsPenDetectable()
BOOL IsDisplay()
unsigned charGetAttribute()
```

The following methods are valid for the ECLField class :

```
ULONG GetScreen(WCHAR *Buff, ULONG BuffLen, PS_PLANE Plane = TextPlane)
void SetText(WCHAR *text)
```

GetStart

This method returns the position in the presentation space of the first character of the field. There are two signatures for the GetStart method. ULONG GetStart returns the position as a linear value with the upper left corner of the presentation space being “1”. void GetStart(ULONG *Row, ULONG *Col) returns the position as a row and column coordinate.

Prototype

ULONG GetStart(),

void GetStart(ULONG *Row, ULONG *Col)

Parameters

ULONG *Row

This output parameter is a pointer to the row value to be updated.

ULONG *Col

This output parameter is a pointer to the column value to be updated.

Return Value

ULONG

Position in the presentation space represented as a linear array.

Example

The following example shows how to return the position in the presentation space of the first character of the field.

```
/-------------------------------------------------------------------
// ECLField::GetStart
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)
     Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(),
      pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetStartRow

This method returns the starting row position of a given field in the ECLFieldList collection for the connection associated with the ECLPS object.

Prototype

ULONG GetStartRow()

Parameters

None

Return Value

ULONG

This is the starting row of a given field.

Example

```
/-------------------------------------------------------------------
// ECLField::GetStartRow
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)  Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(), pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetStartCol

This method return the starting column position of a given field in the ECLFieldList collection for the connection associated with the ECLPS object.

Prototype

ULONG GetStartCol()

Parameters

None

Return Value

ULONG

This is the starting column of a given field.

Example

```
/-------------------------------------------------------------------
// ECLField::GetStartCol
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)
      Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(),
      pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetEnd

This method returns the position in the presentation space of the last character of the field. There are two signatures for the GetEnd method. ULONG GetEnd returns the position as a linear value with the upper left corner of the presentation space being “1”. void GetEnd(ULONG *Row, ULONG *Col) returns the position as a row and column coordinate.

Prototype

ULONG GetEnd()

void GetEnd(ULONG *Row, ULONG *Col)

Parameters

ULONG *Row

This output parameter is a pointer to the row value to be updated.

ULONG *Col

This output parameter is a pointer to the column value to be updated.

Return Value

ULONG

Position in the presentation space represented as a linear array.

Example

The following example shows how to return the position in the presentation space of the last character of the field.

```
/-------------------------------------------------------------------
// ECLField::GetEnd
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'
  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)
     Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(),
      pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetEndRow

This method returns the ending row position of the field.

Prototype

ULONG GetEndRow()

Parameters

None

Return Value

ULONG

This is the ending row in a given field.

Example

```
/-------------------------------------------------------------------
// ECLField::GetEndRow
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)
     Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(),
      pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetEndCol

This method returns the ending column position of a field.

Prototype

ULONG GetEndCol()

Parameters

None

Return Value

ULONG

This is the ending row in a given field.

Example

```
/-------------------------------------------------------------------
// ECLField::GetEndCol
//
```

```
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)
     Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(),
      pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetLength

This method returns the length of the field. The length includes the entire field even if it spans multiple lines of the presentation space. It does not include the field attribute character that starts the field.

Prototype

ULONG GetLength()

Parameters

None

Return Value

ULONG

Length of the field.

Example

The following example shows how to return the length of the field.

```
/-------------------------------------------------------------------
// ECLField::GetLength
//
// Iterate over list of fields and print each field
// starting pos, row, col, and ending pos, row, col.
//-------------------------------------------------------------------
void Sample34() {

ECLPS        *pPS;           // Pointer to PS object
ECLFieldList *pFieldList;    // Pointer to field list object
ECLField     *pField;        // Pointer to field object

try {
  pPS = new ECLPS('A');                // Create PS object for 'A'

  pFieldList = pPS->GetFieldList();    // Get pointer to field list
  pFieldList->Refresh();               // Build the field list

  printf("Start(Pos,Row,Col)  End(Pos,Row,Col) Length(Len)\n");
  for (pField = pFieldList->GetFirstField();      // First field
    pField != NULL;                               // While more
    pField = pFieldList->GetNextField(pField)) {  // Next field

    printf("Start(%04lu,%04lu,%04lu)  End(%04lu,%03lu,%04lu)  Length(%04lu)\n",
      pField->GetStart(), pField->GetStartRow(), pField->GetStartCol(),
      pField->GetEnd(), pField->GetEndRow(),
      pField->GetEndCol(), pField->GetLength());
  }
  delete pPS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetScreen

The GetScreen method fills an application-supplied buffer with data from the field. The type of data copied to the buffer is selected with the optional Plane parameter. The default is to return the text plane data. The data returned is the field as it existed at the time this field object was created; it will not reflect the current contents of the field if it has been updated since the ECLFieldList::Refresh function was called.

The length of the data returned is the length of the field (see GetLength on page 803). When the TextPlane is copied, an additional null terminating byte is added after the last data byte. Therefore, the application should provide a buffer that is at least 1 byte more than the field length when getting the text plane. If the application buffer is too small the returned data is truncated. The number of bytes of copied to the application buffer is returned as the function result (not including the null terminator for copies of the text plane).

The FieldPlane cannot be obtained with this function. The ECLField::GetAttribute can be used to obtain the field attribute value.

Prototype

```
ULONG GetScreen(char *Buff, ULONG BuffLen, PS_PLANE Plane=TextPlane)
```

Parameters

char * Buff

Pointer to application buffer to be filled with field data.

ULONG BuffLen

Length of application buffer.

PS_PLANE Plane

Optional parameter. Enumeration which indicates what plane of field data is to be retrieved. Must be one of TextPlane, ColorPlane, or ExtendedFieldPlane.

Return Value

ULONG

Number of bytes copied to application buffer, not including trailing null character for TextPlane data.

Example

The following example shows how to return a pointer to the field data indicated by the Plane parameter.

```
/-------------------------------------------------------------------
// ECLField::GetScreen
//
// Iterate over list of fields and print each fields text contents.
//-------------------------------------------------------------------
void Sample35() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object
char         *Buff;         // Screen data buffer
ULONG        BuffLen;

try {
  PS = new ECLPS('A');                // Create PS object for 'A'
  BuffLen = PS->GetSize() + 1;        // Make big enough for entire screen
  Buff = new char[BuffLen];           // Allocate screen buffer

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  for (Field = FieldList->GetFirstField();     // First field
    Field != NULL;                             // While more
    Field = FieldList->GetNextField(Field)) {  // Next field

      Field->GetScreen(Buff, BuffLen); // Get this fields text
      printf("%02lu,%02lu: %s\n",      // Print "row,col: text"
             Field->GetStartRow(),
             Field->GetStartCol(),
             Buff);
  }
  delete []Buff;
  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetText

This method populates a given field in the presentation space with the character string passed in as text. If the text exceeds the length of the field, the text is truncated. If the text is shorter than the field, the field is padded with nulls.

Prototype

void SetText(char *text)

Parameters

char *text

Null terminated string to set in field.

Return Value

None

Example

The following example shows how to populate a given field in the presentation space with the character string passed in as text.

```
//-------------------------------------------------------------------
// ECLField::SetText
```

```
//
// Set the field that contains row 2, column 10 to a value.
//-------------------------------------------------------------------
void Sample36() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'
  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  // If the field at row 2 col 10 is an input field, set
  // it to a new value.
  Field = FieldList->FindField(2, 10);    // Find field at this location
  if (Field != NULL) {
    if (!Field->IsProtected())            // Make sure its an input field
      Field->SetText("Way cool!");        // Assign new field text
    else
      printf("Position 2,10 is protected.\n");
  }
  else printf("Cannot find field at position 2,10.\n");

  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

IsModified, IsProtected, IsNumeric, IsHighIntensity, IsPenDetectable, IsDisplay

This method determines if a given field in the presentation space has a particular attribute. The method returns a TRUE value if the field has the attribute or a FALSE value if the field does not have the attribute.

Prototype

BOOL IsModified()

BOOL IsProtected()

BOOL IsNumeric()

BOOL IsHighIntensity()

BOOL IsPenDetectable()

BOOL IsDisplay()

Parameters

None

Return Value

BOOL

Returns a TRUE value if the attribute is present; a FALSE value if the attribute is not present.

Example

The following example shows how to determine if a given field has an attribute.

```
//-------------------------------------------------------------------
// ECLField::IsModified
// ECLField::IsProtected
// ECLField::IsNumeric
// ECLField::IsHighIntensity
// ECLField::IsPenDetectable
// ECLField::IsDisplay
//
// Iterate over list of fields and print each fields attributes.
//-------------------------------------------------------------------
void Sample37() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  for (Field = FieldList->GetFirstField();     // First field
    Field != NULL;                             // While more
    Field = FieldList->GetNextField(Field)) {  // Next field

      printf("Field at %02lu,%02lu is: ",
             Field->GetStartRow(), Field->GetStartCol());

      if (Field->IsProtected())
        printf("Protect ");
      else
        printf("Input   ");

      if (Field->IsModified())
        printf("Modified   ");
      else
        printf("Unmodified ");

      if (Field->IsNumeric())
        printf("Numeric  ");
      else
```

```
        printf("Alphanum ");

      if (Field->IsHighIntensity())
        printf("HiIntensity ");
      else
        printf("Normal      ");

      if (Field->IsPenDetectable())
        printf("Penable ");
      else
        printf("NoPen   ");

      if (Field->IsDisplay())
        printf("Display \n");
      else
        printf("Hidden  \n");
  }
  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample

//-------------------------------------------------------------------
```

GetAttribute

This method returns the attribute of the field. The value returned contains the bit flags for each of the possible field attributes (modified, protected, numeric, high intensity, pen, and display). See ECL Planes — Format and Content on page 1159 for more details on these bits. There is a method provided for each type of attribute (for example, IsModified or IsHighIntensity). This method can be used to obtain complete attribute information in a single call.

Prototype

unsigned char GetAttribute()

Parameters

None

Return Value

unsigned char

Attribute bits of the field.

Example

The following example shows how to return the attribute of the field.

```
/ ECLField::GetAttribute
//
// Iterate over list of fields and print each fields attribute
// value.
//-------------------------------------------------------------------
void Sample38() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  for (Field = FieldList->GetFirstField();     // First field
    Field != NULL;                             // While more
    Field = FieldList->GetNextField(Field)) {  // Next field

      printf("Attribute value for field at %02lu,%02lu is: 0x%02x\n",
             Field->GetStartRow(), Field->GetStartCol(),
             Field->GetAttribute());
  }
  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLFieldList Class

The ECLFieldList class performs operations on a list of fields in a host presentation space. An application should not create an ECLFieldList object directly, but only indirectly by creating an ECLPS object.

ECLFieldList contains a collection of all the fields in the presentation space. Each element of the collection is an ECLField object. See ECLField Class on page 793 for details on its properties and methods.

An ECLFieldList object provides a static snapshot of what the presentation space contained when the Refresh method was called. If the presentation space is updated after the call to Refresh(), the field list does not reflect those changes. An application must explicitly call Refresh to refresh the field list.

Once an application has called Refresh it can begin walking through the collection of fields using GetFirstField and GetNextField. If the location of a field is known, FindField can be used to locate it in the list directly.

Note: All ECLField object pointers returned by GetFirstField, GetNextField, and FindField become invalid when Refresh is called or the ECLFieldList object is destroyed.

Derivation

ECLBase > ECLFieldList

Properties

None

ECLFieldList Methods

The following section describes the methods that are valid for the ECLFieldList class.

```
void Refresh(PS_PLANE Planes)
ULONG GetFieldCount()
ECLField * GetFirstField()
ECLField *GetNextField(ECLField *Prev)
ECLField * FindField(ULONG Pos)
ECLField * FindField(ULONG Row, ULONG Col)
ECLField *FindField(char* text, PS_DIR DIR=SrchForward);
ECLField *FindField(char* text, ULONG Pos, PS_DIR DIR=SrchForward);
ECLField *FindField(char* text, ULONG Row, ULONG Col, PS_DIR DIR=SrchForward);
```

Refresh

This method gets a snapshot of all the fields currently in the presentation space. All ECLField object pointers previously returned by this object become invalid. To improve performance, the field data can be limited to the planes of interest. Note that the TextPlane and FieldPlane are always obtained.

Prototype

void Refresh(PS_PLANE Planes=TextPlane)

Parameters

PS_PLANE Planes

Plane for which fields are built. Valid values are **TextPlane**, **ColorPlane**, **FieldPlane**, **ExfieldPlane**, and **AllPlanes** (to build for all). This is an enumeration defined in ECLPS.HPP. This optional parameter defaults to TextPlane.

Return Value

None

Example

The following example shows how to use the Refresh method to get a snapshot of all the fields currently in the presentation space.

```
///-------------------------------------------------------------------
// ECLFieldList::Refresh
//
// Display number of fields on the screen.
//-------------------------------------------------------------------
void Sample39() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  printf("There are %lu fields on the screen of connection %c.\n",
    FieldList->GetFieldCount(), PS->GetName());

  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample

----------------------------------------
```

GetFieldCount

This method returns the number of fields present in the ECLFieldList collection (based on the most recent call to the Refresh method).

Prototype

ULONG GetFieldCount()

Parameters

None

Return Value

ULONG

Number of fields in the ECLFieldList collection.

Example

The following example shows how to use the GetFieldCount method to return the number of fields present in the ECLFieldList collection.

```
//-------------------------------------------------------------------
// ECLFieldList::GetFieldCount
//
// Display number of fields on the screen.
//-------------------------------------------------------------------
void Sample40() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  printf("There are %lu fields on the screen of connection %c.\n",
    FieldList->GetFieldCount(), PS->GetName());

  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetFirstField

This method returns a pointer to the first ECLField object in the collection. ECLFieldList contains a collection of ECLField objects. See ECLField Class on page 793 for more information. The method returns a NULL pointer if there are no fields in the collection.

Prototype

ECLField * GetFirstField();

Parameters

None

Return Value

ECLField *

Pointer to an ECLField object. If there are no fields in the connection, a null is returned.

Example

The following example shows how to use the GetFirstField method to return a pointer to the first ECLField object in the collection.

```
/-------------------------------------------------------------------
// ECLFieldList::GetFirstField
//
// Display starting position of every input (unprotected) field.
//-------------------------------------------------------------------
void Sample41() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  // Interate over (only) unprotected fields
  printf("List of input fields:\n");
  for (Field = FieldList->GetFirstField(GetUnprotected);
    Field != NULL;
    Field = FieldList->GetNextField(Field, GetUnprotected)) {

    printf("Input field starts at %02lu,%02lu\n",
            Field->GetStartRow(), Field->GetStartCol());
  }
  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetNextField

This method returns the next ECLField object in the collection after a given object. If there are no more objects in the collection after the given object, a NULL pointer is returned. An application can make repeated calls to this method to iterate over the ECLField objects in the collection.

Prototype

ECLField *GetNextField(ECLField *Prev)

Parameters

ECLField *Prev

A pointer to any ECLField object in the collection. The returned pointer will be the next object after this one. If this value is NULL a pointer to the first object in the collection is returned. This pointer is a pointer returned by the GetFirstField, GetNextField, or FindField member functions.

Return Value

ECLField *

A pointer to the next object in the collection. If there are no more objects in the collection after the Prev object, NULL is returned.

Example

The following example shows how to use the GetNextFieldInfo method to return a pointer to the next ECLField object in the collection.

```
///-------------------------------------------------------------------
// ECLFieldList::GetNextField
//
// Display starting position of every input (unprotected) field.
//-------------------------------------------------------------------
void Sample42() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  // Interate over (only) unprotected fields
  printf("List of input fields:\n");
  for (Field = FieldList->GetFirstField(GetUnprotected);
    Field != NULL;
    Field = FieldList->GetNextField(Field, GetUnprotected)) {

    printf("Input field starts at %02lu,%02lu\n",
            Field->GetStartRow(), Field->GetStartCol());
  }
  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

FindField

This method finds a field in the ECLFieldList collection using either text or a position. The position can be either a linear position or a row, column position. If a field contains the text or the position, a pointer to an ECLField object for that field is returned. The returned pointer is to an object in the field list collection. NULL is returned if the field is not found. When searching for text, the search begins at row1 column1 unless you specify a starting position. Also for text, this method will search forward in the list as a default; however, you can specify the direction to search explicitly.

Note: A search for text will be successful even if the text spans multiple fields. The field object returned will be the field where the found text begins.

Prototype

ECLField *FindField(ULONG Pos);
ECLField *FindField(ULONG Row, ULONG Col);
ECLField *FindField(char* text, PS_DIR DIR=SrchForward);
ECLField *FindField(char* text, ULONG Pos, PS_DIR DIR=SrchForward);
ECLField *FindField(char* text, ULONG Row, ULONG Col, PS_DIR DIR=SrchForward);

Parameters

ULONG Pos

Linear position to search for OR linear position to begin text search.

ULONG Row

Row position to search for OR row to begin text search.

ULONG Col

Column position to search for OR column to begin text search.

char *text

String to search

PS_DIR Dir

Direction to search

Return Value

ECLField *

Pointer to an ECLField object if field is found. NULL if field is not found. Returned pointer is invalid after the next call to Refresh.

Example

The following is an example of the FindField method.

```
//-------------------------------------------------------------------
// ECLFieldList::FindField
//
// Display the field which contains row 2 column 10.  Also find
// the first field containing a particular string.
//-------------------------------------------------------------------
void Sample43() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object
ECLField     *Field;        // Pointer to field object
char         Buff[4000];

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  // Find by row,column coordinate

  Field = FieldList->FindField(2, 10);
  if (Field != NULL) {
    Field->GetText(Buff, sizeof(Buff));
    printf("Field at 2,10: %s\n", Buff);
  }
  else printf("No field found at 2,10.\n");

  // Find by text.  Note that text may span fields, this
  // will find the field in which the text starts.

  Field = FieldList->FindField("HCL");
  if (Field != NULL) {
    printf("String 'HCL' found in field that starts at %lu,%lu.\n",
            Field->GetStartRow(), Field->GetStartCol());
  }
  else printf("String 'HCL' not found.\n");

  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

```
//---------------------------
```

ECLKeyNotify Class

ECLKeyNotify is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLKeyNotify. The application must implement the NotifyEvent() member function in its derived class. It may also optionally implement NotifyError() and NotifyStop() member functions.

The ECLKeyNotify class is used to allow an application to be notified of keystroke events. The application can also choose to filter (remove) the keystrokes so they are not sent to the host screen, or replace them with other keystrokes. Keystroke notifications are queued so that the application will always receive a notification for each and every keystroke. Only keystrokes made by the real physical keyboard are detected by this object; keystrokes sent to the host by other ECL objects (such as ECLPS::SendKeys) do not cause keystroke notification events.

To be notified of keystroke events, the application must perform the following steps:

1. Define a class derived from ECLKeyNotify.
2. Implement the derived class and implement the NotifyEvent() member function.
3. Optionally implement the NotifyError() and/or NotifyStop() functions.
4. Create an instance of the derived class.
5. Register the instance with the ECLPS::RegisterKeyEvent() function.

The example shown demonstrates how this may be done. When the above steps are complete, each keystroke in the emulator window will cause the applications NotifyEvent() member function to be called. The function is passed parameters indicating the type of keystroke (plain ASCII key, or special function key), and the value of the key (a single ASCII character, or a keyword representing a function key). The application may perform any functions required in the NotifyEvent() procedure, including calling other ECL functions such as ECLPS::SendKeys(). The application returns a value from NotifyEvent() to indicate if the keystroke is to be filtered or not (return 1 to filter (discard) the keystroke, return 0 to have it processed normally).

If an error is detected during keystroke event generation, the NotifyError() member function is called with an ECLErr object. Keystroke events may or may not continue to be generated after an error, depending on the nature of the error. When event generation terminates (either due to an error, by calling ECLPS::UnregisterKeyEvent, or by destruction of the ECLPS object) the NotifyStop() member function is called. However event notification is terminated, the NotifyStop() member function is always called, and the application object is unregistered.

If the application does not provide an implementation of the NotifyError() member function, the default implementation is used (a simple message box is displayed to the user). The application can override the default behavior by implementing the NotifyError() function in the applications derived class. Likewise, the default NotifyStop() function is used if the application does not provide this function (the default behavior is to do nothing).

Note that the application can also choose to provide its own constructor and destructor for the derived class. This can be useful if the application wants to store some instance-specific data in the class and pass that information as a parameter on the constructor. For example, the application may want to post a message to an application window when a keystroke occurs. Rather than define the window handle as a global variable (so it would be visible to the NotifyEvent() function), the application can define a constructor for the class which takes the window handle and stores it in the class member data area.

The application must not destroy the notification object while it is registered to receive events.

The same instance of a keystroke notification object can be registered with multiple ECLPS objects to receive keystrokes for multiple connections. Thus an application can use a single instance of this object to process keystrokes on any number of sessions. The member functions are passed a pointer to the ECLPS object for which the event occurred so an application can distinguish between events on different connections. The sample shown uses the same object to process keystrokes on two connections.

Implementation Restriction: Currently the ECLPS object allows only one notification object to be registered for a given connection. The ECLPS::RegisterKeyEvent will throw an error if a notify object is already registered for that ECLPS object.

Derivation

ECLBase > ECLNotify > ECLKeyNotify

Example

The following is an example of how to construct and use an ECLKeyNotify object.

```
// ECLKeyNotify class
//
// This sample demonstrates the use of:
//
// ECLKeyNotify::NotifyEvent
// ECLKeyNotify::NotifyError
// ECLKeyNotify::NotifyStop
// ECLPS::RegisterKeyEvent
// ECLPS::UnregisterKeyEvent
//-------------------------------------------------------------------

//...................................................................
// Define a class derived from ECLKeyNotify
//...................................................................
class MyKeyNotify: public ECLKeyNotify
{
public:
  // Define my own constructor to store instance data
  MyKeyNotify(HANDLE DataHandle);

  // We have to implement this function
  virtual int NotifyEvent(ECLPS *PSObj, char const KeyType[2],
                          const char * const KeyString);

  // We choose to implement this function
  void NotifyStop (ECLPS *PSObj, int Reason);

  // We will take the default behaviour for this so we
  // don't implement it in our class:
```

```
  // void NotifyError (ECLPS *PSObj, ECLErr ErrObject);

private:
  // We will store our application data handle here
  HANDLE MyDataH;
};

 //..................................................................
MyKeyNotify::MyKeyNotify(HANDLE DataHandle)   // Constructor
//...................................................................
{
  MyDataH = DataHandle;  // Save data handle for later use
}

//...................................................................
int MyKeyNotify::NotifyEvent(ECLPS *PSObj,
                             char const KeyType[2],
                             const char * const KeyString)
```

```
//...................................................................

{
  // This function is called whenever a keystroke occurs.  We will
  // just do something simple: when the user presses PF1 we will
  // send a PF2 to the host instead.  All other keys will be unchanged.

  if (KeyType[0] == 'M') {             // Is this a mnemonic keyword?
    if (!strcmp(KeyString, "[pf1]")) { // Is it a PF1 key?
      PSObj->SendKeys("[pf2]");        // Send PF2 instead
      printf("Changed PF1 to PF2 on connection %c.\n",
             PSObj->GetName());
      return 1;                        // Discard this PF1 key
    }
  }

  return 0;                            // Process key normally
}
```

```
//..................................................................
void MyKeyNotify::NotifyStop (ECLPS *PSObj, int Reason)
//...................................................................
{
  // When notification ends, display message
  printf("Keystroke intercept for connection %c stopped.\n", PSObj->GetName());
}

//...................................................................
// Create the class and start keystroke processing on A and B.
//...................................................................
void Sample44() {

ECLPS *PSA, *PSB;       // PS objects
MyKeyNotify *Event;     // Ptr to my event handling object
HANDLE InstData;      // Handle to application data block (for example)

try {
  PSA = new ECLPS('A');               // Create PS objects
  PSB = new ECLPS('B');
  Event = new MyKeyNotify(InstData);  // Create event handler

  PSA->RegisterKeyEvent(Event);       // Register for keystroke events
  PSB->RegisterKeyEvent(Event);       // Register for keystroke events
```

```
  // At this point, any keystrokes on A or B will cause the
  // MyKeyEvent::NotifyEvent() function to execute.  For
  // this sample, we put this thread to sleep during this
  // time.

  printf("Processing keystrokes for 60 seconds on A and B...\n");
  Sleep(60000);

  // Now stop event generation.  This will cause the NotifyStop
  // member to be called.
  PSA->UnregisterKeyEvent(Event);
  PSB->UnregisterKeyEvent(Event);

  delete Event;  // Don't delete until after unregister!
  delete PSA;
  delete PSB;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample

//-------------------------------------------------------------------
```

ECLKeyNotify Methods

The following section describes the methods that are valid for the ECLKeyNotify class.

virtual int NotifyEvent (ECLPS *PSObj, char const KeyType [2],
   const char * const KeyString) =0
virtual void NotifyError (ELLPS *PSobj, ECLErr ErrObject)
virtual void NotifyStop (ELLPS *PSObj, int Reason)

NotifyEvent

This method is a“pure virtual” member function (the application *must* implement this function in classes derived from ECLKeyNotify). This function is called whenever a keystroke event occurs and the object is registered for keystroke events. The return value indicates the disposition of the keystroke (return 1 to discard, 0 to process).

Prototype

virtual int NotifyEvent (ECLPS *PSObj, char const KeyType [2], const char * const KeyString) =0

Parameters

ECLPS *PSObj

This is a ptr to ECLPS object in which the event occurred.

char const KeyType[2]

This is a null terminated 1–char string indicating the type of key:

A = Plain ASCII keystroke

M = Mnemonic keyword

const char * const KeyString

This is a null terminated string containing the keystroke or mnemonic keyword. Keywords will always be in lowercase (for example, "[enter]"). See Sendkeys Mnemonic Keywords on page 1156 for a list of mnemonic keywords.

Return Value

int

This is the filter indicator.

1 = Filter (discard) keystroke

0 = Process keystroke (send to host)

NotifyError

This method is called whenever the ECLPS object detects an error during keystroke event generation. The error object contains information about the error (see ECLErr Class on page 789). Keystroke events may continue to be generated after the error, depending on the nature of the error. If keystroke event generation stops due to an error, the NotifyStop() function will be called.

Prototype

virtual void NotifyError (ELLPS *PSobj, ECLErr ErrObject)

Parameters

ECLPS *PSObj

This is the ptr to ECLPS object in which the error occurred.

ECLErr ErrObject

This is the ECLErr object describing the error.

Return Value

None

NotifyStop

This method is called when keystroke event generation is stopped for any reason (for example, due to an error condition, a call to ECLPS::UnregisterKeyEvent, destruction of the ECLPS object, etc.).

Prototype

virtual void NotifyStop (ELLPS *PSObj, int Reason)

Parameters

ECLPS *PSObj

This is the ptr to ECLPS object in which events are stopping.

int Reason

This is unused (zero).

Return Value

None

ECLListener Class

ECLListener is the base class for all HACL "listener" objects. Listeners are objects which are registered to receive particular types of asynchronous events. Methods on the listener objects are called when events occur or errors are detected.

There are no public methods on the ECLListener class.

Derivation

ECLBase > ECLListener

Usage Notes

Applications do not use this class directly, but create instances of classes which are derived from it (for example, ECLPSListener).

ECLOIA Class

ECLOIA provides Operator Information Area (OIA) services.

Because ECLOIA is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

The ECLOIA object is created for the connection identified upon construction. You may create an ECLOIA object by passing either the connection name (a single, alphabetic character from A-Z or a-z) or the connection handle, which is usually obtained from the ECLConnList object. There can be only one Z and I Emulator for Windows connection with a given name or handle open at a time.

Derivation

ECLBase > ECLConnection > ECLOIA

Usage Notes

The ECLSession class creates an instance of this object. If the application does not need other services, this object may be created directly. Otherwise, consider using an ECLSession object to create all the objects needed.

ECLOIA Methods

The following section describes the methods that are valid for the ECLOIA class.

ECLOIA(char ConnName)

ECLOIA(long ConnHandle)

~ECLOIA()

BOOL IsAlphanumeric()

BOOL IsAPL()

BOOL IsUpperShift()

BOOL IsNumeric()

BOOL IsCapsLock()

BOOL IsInsertMode()

BOOL IsCommErrorReminder()

BOOL IsMessageWaiting()

BOOL WaitForInputReady(long nTimeOut = INFINITE)

BOOL WaitForAppAvailable(long nTimeOut = INFINITE)

BOOL WaitForSystemAvailable(long nTimeOut = INFINITE)

BOOL WaitForTransition(BYTE nIndex = 0xFF, long nTimeOut = INFINITE)

INHIBIT_REASON InputInhibited()

ULONG GetStatusFlags()

ECLOIA Constructor

This method creates an ECLOIA object from a connection name (a single, alphabetic character from A-Z or a-z) or a connection handle. There can be only one Z and I Emulator for Windows connection started with a given name.

Prototype

ECLOIA(char ConnName)

ECLOIA(long ConnHandle)

Parameters

char ConnName

One-character short name of the connection (A-Z or a-z).

long ConnHandle

Handle of an ECL connection.

Return Value

None

Example

The following example shows how to create an ECLOIA object using the connection name.

```
// ECLOIA::ECLOIA              (Constructor)
//
// Build an OIA object from a name, and another from a handle.
//-------------------------------------------------------------------
void Sample45() {

ECLOIA *OIA1, *OIA2;    // Pointer to OIA objects
ECLConnList ConnList;   // Connection list object

try {
  // Create OIA object for connection 'A'
  OIA1 = new ECLOIA('A');

  // Create OIA object for first connection in conn list
  OIA2 = new ECLOIA(ConnList.GetFirstConnection()->GetHandle());

  printf("OIA #1 is for connection %c, OIA #2 is for connection %c.\n",
         OIA1->GetName(), OIA2->GetName());
  delete OIA1;
  delete OIA2;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

```
-------------------------------------------
```

IsAlphanumeric

This method checks to determine if the OIA indicates that the cursor is at an alphanumeric location.

Prototype

BOOL IsAlphanumeric()

Parameters

None

Return Value

BOOL

TRUE if the keyboard is in alphanumeric mode; FALSE if the keyboard is not in alphanumeric mode.

Example

The following example shows how to determine if the OIA indicates that the keyboard is in alphanumeric mode.

```
//-------------------------------------------------------------------
// ECLOIA::IsAlphanumeric
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample46() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsAlphanumeric())
  printf("Alphanumeric.\n");
else
  printf("Not Alphanumeric.\n");

} // end sample
```

IsAPL

This method checks to determine if the OIA indicates that the keyboard is in APL mode.

Prototype

BOOL IsAPL()

Parameters

None

Return Value

BOOL

TRUE if the keyboard is in APL mode; FALSE if the keyboard is not in APL mode.

Example

The following example shows how to determine if the OIA indicates that the keyboard is in APL mode.

```
//-------------------------------------------------------------------
// ECLOIA::IsAPL
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample47() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsAPL())
  printf("APL.\n");
else
  printf("Not APL.\n");

} // end sample

//------------------------
```

IsUpperShift

This method checks to determine if the OIA indicates that the keyboard is in upper shift mode.

Prototype

BOOL IsUpperShift()

Parameters

None

Return Value

BOOL

TRUE if the keyboard is in upper shift mode; FALSE if the keyboard is not in upper shift mode.

Example

The following example shows how to determine if the OIA indicates that the keyboard is in upper shift mode.

```
//-------------------------------------------------------------------
// ECLOIA::IsUpperShift
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample51() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsUpperShift())
  printf("UpperShift.\n");
else
  printf("Not UpperShift.\n");

} // end sample
```

IsNumeric

This method checks to determine if the OIA indicates that the cursor is at a numeric-only location.

Prototype

BOOL IsNumLock()

Parameters

None

Return Value

BOOL

TRUE if Numeric is on; FALSE if not Numeric.

Example

The following example shows how to determine if the OIA indicates that the cursor is at a numeric location.

```
//-------------------------------------------------------------------
// ECLOIA::IsNumeric
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample52() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsNumeric())
```

```
  printf("Numeric.\n");
else
  printf("Not Numeric.\n");

} // end sample
```

IsCapsLock

This method checks to determine if the OIA indicates that the keyboard has Caps Lock on.

Prototype

BOOL IsCapsLock()

Parameters

None

Return Value

BOOL

TRUE if Caps Lock is on; FALSE if Caps Lock is not on.

Example

The following example shows how to determine if the OIA indicates that the keyboard has Caps Lock on.

```
//-------------------------------------------------------------------
// ECLOIA::IsCapsLock
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample53() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsCapsLock())
  printf("CapsLock.\n");
else
  printf("Not CapsLock.\n");

} // end sample
```

IsInsertMode

This method checks to determine if the OIA indicates that the keyboard is in insert mode.

Prototype

BOOL IsInsertMode()

Parameters

None

Return Value

BOOL

TRUE if the keyboard is in insert mode; FALSE if the keyboard is not in insert mode.

Example

The following example shows how to determine if the OIA indicates that the keyboard is in insert mode.

```
//-------------------------------------------------------------------
// ECLOIA::IsInsertMode
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample54() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsInsertMode())
  printf("InsertMode.\n");
else
  printf("Not InsertMode.\n");

} // end sample
```

IsCommErrorReminder

This method checks to determine if the OIA indicates that a communications error reminder condition exists.

Prototype

BOOL IsCommErrorReminder()

Parameters

None

Return Value

BOOL

TRUE if a condition exists; FALSE if a condition does not exist.

Example

The following example shows how to determine if the OIA indicates that a communications error reminder condition exists.

```
//-------------------------------------------------------------------
// ECLOIA::IsCommErrorReminder
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample55() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsCommErrorReminder())
  printf("CommErrorReminder.\n");
else
  printf("Not CommErrorReminder.\n");

} // end sample

//
```

IsMessageWaiting

This method checks to determine if the OIA indicates that the message waiting indicator is on. This can only occur for 5250 connections.

Prototype

BOOL IsMessageWaiting()

Parameters

None

Return Value

BOOL

TRUE if the message waiting indicator is on; FALSE if the indicator is not on.

Example

The following example shows how to determine if the OIA indicates that the message waiting indicator is on.

```
-------------------------------------------------------------------
// ECLOIA::IsMessageWaiting
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample56() {

ECLOIA OIA('A');   // OIA object for connection A

if (OIA.IsMessageWaiting())
  printf("MessageWaiting.\n");
else
  printf("Not MessageWaiting.\n");

} // end sample
```

WaitForInputReady

The WaitForInputReady method waits until the OIA of the connection associated with the autECLOIA object indicates that the connection is able to accept keyboard input.

Prototype

BOOL WaitForInputReady(long nTimeOut = INFINITE)

Parameters

long nTimeOut

The maximum length of time to wait in milliseconds, this parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

WaitForSystemAvailable

The WaitForSystemAvailable method waits until the OIA of the session connected with the ECLOIA object indicates that session is connected to a host system.

Prototype

BOOL WaitForSystemAvailable(long nTimeOut = INFINITE)

Parameters

long nTimeOut

The maximum length of time to wait in milliseconds, this parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

WaitForAppAvailable

The WaitForAppAvailable method waits while the OIA of the connected session indicates that the application is initialized and ready for use.

Prototype

BOOL WaitForAppAvailable(long nTimeOut = INFINITE)

Parameters

long nTimeOut

The maximum length of time to wait in milliseconds, this parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

WaitForTransition

The WaitForTransition method waits for the value at the specified position in the OIA of the connected session to change.

Prototype

BOOL WaitForTransition(BYTE nIndex = 0xFF, long nTimeOut = INFINITE)

Parameters

BYTE nIndex

The 1 byte Hex position of the OIA to monitor. This parameter is optional. The default is 3.

long nTimeOut

The maximum length of time to wait in milliseconds, this parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

InputInhibited

This method returns an enumerated value that indicates whether input is inhibited or not. If input is inhibited, the reason for the inhibit can be determined. If input is inhibited for more than one reason the highest value enumeration is returned (for example, if there is a communications error and a protocol programming error, the ProgCheck value is returned).

Prototype

INHIBIT_REASON InputInhibited ()

Parameters

None

Return Value

INHIBIT_REASON

Returns one of the INHIBIT_REASON values as defined in ECLOIA.HPP. The value NotInhibited is returned if input is currently not inhibited.

Example

The following example shows how to determine whether input is inhibited or not.

```
//-------------------------------------------------------------------
// ECLOIA::InputInhibited
//
// Determine status of connection 'A' OIA indicator
//-------------------------------------------------------------------
void Sample57() {

ECLOIA OIA('A');   // OIA object for connection A

switch (OIA.InputInhibited()) {
case NotInhibited:
  printf("Input not inhibited.\n");
  break;
case SystemWait:
  printf("Input inhibited for SystemWait.\n");
  break;
case CommCheck:
  printf("Input inhibited for CommCheck.\n");
  break;
case ProgCheck:
  printf("Input inhibited for ProgCheck.\n");
  break;
case MachCheck:
  printf("Input inhibited for MachCheck.\n");
  break;
case OtherInhibit:
  printf("Input inhibited for OtherInhibit.\n");
  break;
default:
  printf("Input inhibited for unknown reason.\n");
  break;
}
} // end sample
```

GetStatusFlags

This method returns a set of status bits that represent various OIA indicators. This method can be used to collect a set of OIA indicators in a single call rather than making calls to several different IsXXX methods. Each bit returned represents a single OIA indicator where a value of 1 means the indicator is on (TRUE), and 0 means it is off (FALSE). A set of bitmask constants are defined in the ECLOIA.HPP header file for isolating individual indicators in the returned 32–bit value.

Prototype

ULONG GetStatusFlags()

Parameters

None

Return Value

ULONG

Set of bit flags defined as follows:

Bit Position	Mask Constant	Description
31 (msb)	OIAFLAG_ALPHANUM	IsAlphanumeric
30	OIAFLAG_APL	IsAPL
26	OIAFLAG_UPSHIFT	IsUpperShift
25	OIAFLAG_NUMERIC	IsNumeric
24	OIAFLAG_CAPSLOCK	IsCapsLock
23	OIAFLAG_INSERT	IsInsertMode
22	OIAFLAG_COMMERR	IsCommErrorReminder
21	OIAFLAG_MSGWAIT	IsMessageWaiting
20	OIAFLAG_ENCRYPTED	IsConnectionEncrypted
19-4		<reserved>

Bit Position	Mask Constant	Description
3-0	OIAFLAG_INHIBMASK	InputInhibited: 0=NotInhibited 1=SystemWait 2=CommCheck 3=ProgCheck 4=MachCheck 5=OtherInhibit

RegisterOIAEvent

This member function registers an application object to receive notifications of OIA update events. To use this function the application must create an object derived from ECLOIANotify. A pointer to that object is then passed to this registration function. Any number of notify objects may be registered at the same time. The order in which multiple listeners receive events is not defined and should not be assumed.

After an ECLOIANotify object is registered with this function, its NotifyEvent() method will be called whenever a update to the OIA occurs. Multiple updates to the OIA in a short time period may be aggregated into a single event.

The application must unregister the notify object before destroying it. The object will automatically be unregistered if the ECLOIA object is destroyed.

Prototype

void RegisterOIAEvent(ECLOIANotify * notify)

Parameters

ECLOIANotify *

Pointer to the ECLOIANotify object to be registered.

Return Value

None

UnregisterOIAEvent

This member function unregisters an application object previously registered with the RegisterOIAEvent function. An object registered to receive events should not be destroyed without first calling this function to unregister it. If the specific object is not currently registered, no action is taken and no error occurs.

When an ECLOIANotify object is unregistered its NotifyStop() method is called.

Prototype

void UnregisterOIAEvent(ECLOIANotify * notify)

Parameters

ECLPSNotify *

Pointer to the ECLOIANotify object to be unregistered.

Return Value

None

ECLOIANotify Class

ECLOIANotify is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLOIANotify. The application must implement the NotifyEvent() member function in its derived class. It may also optionally implement NotifyError() and NotifyStop() member functions.

The ECLOIANotify class is used to allow an application to be notified of updates to the Operator Information Area. Events are generated whenever any indicator on the OIA is updated.

Derivation

ECLBase > ECLNotify > ECLOIANotify

Usage Notes

To be notified of OIA updates using this class, the application must perform the following steps:

1. Define a class derived from ECLOIANotify.
2. Implement the NotifyEvent method of the ECLOIANotify-derived class.
3. Optionally implement other member functions of ECLOIANotify.
4. Create an instance of the derived class.
5. Register the instance with the ECLOIA::RegisterOIAEvent() method.

After registration is complete, updates to the OIA indicators will cause the NotifyEvent() method of the ECLOIANotify-derived class to be called.

Note that multiple OIA updates which occur in a short period of time may be aggregated into a single event notification.

An application can choose to provide its own constructor and destructor for the derived class. This can be useful if the application needs to store some instance-specific data in the class and pass that information as a parameter on the constructor.

If an error is detected during event registration, the NotifyError() member function is called with an ECLErr object. Events may or may not continue to be generated after an error. When event generation terminates (due to an error or some other reason) the NotifyStop() member function is called. The default implementation of NotifyError() will present a message box to the user showing the text of the error messages retrieved from the ECLErr object.

When event notification stops for any reason (error or a call the ECLOIA::UnregisterOIAEvent) the NotifyStop() member function is called. The default implementation of NotifyStop() does nothing.

ECLOIANotify Methods

The following section describes the methods that are valid for the ECLOIANotify class and all classes derived from it.

ECLOIANotify()
~ECLOIANotify()
virtual void NotifyEvent(ECLOIA * OIAObj) = 0
virtual void NotifyError(ECLOIA * OIAObj, ECLErr ErrObj)
virtual void NotifyStop(ECLOIA * OIAObj, int Reason)

NotifyEvent

This method is a *pure virtual* member function (the application **must** implement this function in classes derived from ECLOIANotify). This method is called whenever the OIA is updated and this object is registered to receive update events.

Multiple OIA updates may be aggregated into a single event causing only a single call to this method.

Prototype

virtual void NotifyEvent(ECLOIA * OIAObj) = 0

Parameters

ECLOIA *

Pointer to the ECLOIA object which generated this event.

Return Value

None

NotifyError

This method is called whenever the ECLOIA object detects an error during event generation. The error object contains information about the error (see the ECLErr class description). Events may continue to be generated after the error depending on the nature of the error. If the event generation stops due to an error, the NotifyStop() method is called.

An application can choose to implement this function or allow the base ECLOIANotify class handle it. The default implementation will display the error in a message box using text supplied by the ECLErr::GetMsgText() method. If the application implements this function in its derived class it overrides this behavior.

Prototype

virtual void NotifyError(ECLOIA * OIAObj, ECLErr ErrObj)

Parameters

ECLOIA *

Pointer to the ECLOIA object which generated this event.

ECLErr

An ECLErr object which describes the error.

Return Value

None

NotifyStop

This method is called when event generation is stopped for any reason (for example, due to an error condition or a call to ECLOIA::UnregisterOIAEvent).

The reason code parameter is currently unused and will be zero.

The default implementation of this function does nothing.

Prototype

virtual void NotifyStop(ECLOIA * OIAObj, int Reason)

Parameters

ECLOIA *

Pointer to the ECLOIA object which generated this event.

int

Reason event generation has stopped (currently unused and will be zero).

Return Value

None

ECLPS Class

The ECLPS class performs operations on a host presentation space.

The ECLPS object is created for the connection identified upon construction. You may create an ECLPS object by passing either the connection name (a single, alphabetic character from A-Z) or the connection handle, which is usually obtained from an ECLConnection object. There can be only one Z and I Emulator for Windows connection with a given name or handle open at a time.

Derivation

ECLBase > ECLConnection > ECLPS

Properties

None

Usage Notes

The ECLSession class creates an instance of this object. If the application does not need other services, this object may be created directly. Otherwise, you may want to consider using an ECLSession object to create all the objects needed.

ECLPS Methods

The following section describes the methods available for ECLPS.

ECLPS(char ConnName)
ECLPS(char ConnName)
ECLPS(long ConnHandle)
~ECLPS()
int GetPCCodePage()
int GetHostCodePage()
int GetOSCodePage()
void GetSize(ULONG *Rows, ULONG *Cols) ULONG GetSize()
ULONG GetSizeCols() ULONG GetSizeRows()
void GetCursorPos(ULONG *Row, ULONG *Col) ULONG GetCursorPos()
ULONG GetCursorPosRow()

ULONG GetCursorPosCol()
void SetCursorPos(ULONG pos),
void SetCursorPos(ULONG Row, ULONG Col)
void SendKeys(Char *text, ULONG AtPos),
void SendKeys(Char * text),
void SendKeys(Char *text, ULONG AtRow, ULONG AtCol)
ULONG SearchText(const char * const text, PS_DIR Dir=SrchForward,
BOOL FoldCase=FALSE)
ULONG SearchText(const char * const text,
ULONG StartPos, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE)
ULONG SearchText(const char char * const text, ULONG StartRow,
ULONG StartCol, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE)
ULONG GetScreen(char * Buff, ULONG BuffLen, PS_PLANE Plane=TextPlane)
ULONG GetScreen(char * Buff, ULONG BuffLen, ULONG StartPos,
ULONG Length, PS_PLANE Plane=TextPlane)
ULONG GetScreen(char * Buff, ULONG BuffLen, ULONG StartRow,
ULONG StartCol, ULONG Length, PS_PLANE Plane=TextPlane)
ULONG GetScreenRect(char * Buff, ULONG BuffLen, ULONG StartPos,
ULONG EndPos, PS_PLANE Plane=TextPlane)
ULONG StartCol, ULONG EndRow, ULONG EndCol,
ULONG GetScreenRect(char * Buff, ULONG BuffLen, ULONG StartRow,
ULONG StartCol, ULONG EndRow, ULONG EndCol,
PS_PLANE Plane=TextPlane)
void SetText(char *text);
void SetText(char *text, ULONG AtPos);
void SetText(char *text, ULONG AtRow, ULONG AtCol);
void CopyText ();
void CopyText (ULONG Long Len);
void CopyText (ULONG AtPos, ULONG Long Len);
void CopyText (ULONG AtRow, ULONG AtCol, ULONG Long Len);
void PasteText ();
void PasteText (ULONG Long Len);
void PasteText (ULONG AtPos, ULONG Long Len);
void PasteText (ULONG AtRow, ULONG AtCol, ULONG Long Len);
void ConvertPosToRowCol(ULONG pos, ULONG *row, ULONG *col)
ULONG ConvertRowColToPos(ULONG row, ULONG col)
ULONG ConvertPosToRow(ULONG Pos)
ULONG ConvertPosToCol(ULONG Pos)
void RegisterKeyEvent(ECLKeyNotify *NotifyObject)
virtual UnregisterKeyEvent(ECLKeyNotify *NotifyObject)

ECLFieldList *GetFieldList()
BOOL WaitForCursor(int Row, int Col, long nTimeOut=INFINITE,
BOOL bWaitForIR=TRUE)
BOOL WaitWhileCursor(int Row, int Col, long nTimeOut=INFINITE,
BOOL bWaitForIR=TRUE)
BOOL WaitForString(char* WaitString, int Row=0, int Col=0,
long nTimeOut=INFINITE, BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
BOOL WaitWhileString(char* WaitString, int Row=0, int Col=0,
long nTimeOut=INFINITE, BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
BOOL WaitForStringInRect(char* WaitString, int sRow, int sCol,
int eRow,int eCol, long nTimeOut=INFINITE,
BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
BOOL WaitWhileStringInRect(char* WaitString, int sRow, int sCol,
int eRow,int eCol, long nTimeOut=INFINITE, BOOL bWaitForIR=TRUE,
BOOL bCaseSens=TRUE)
BOOL WaitForAttrib(int Row, int Col, unsigned char AttribDatum,
unsigned char MskDatum = 0xFF, PS_PLANE plane = FieldPlane,
long TimeOut = INFINITE, BOOL bWaitForIR = TRUE)
BOOL WaitWhileAttrib(int Row, int Col, unsigned char AttribDatum,
unsigned char MskDatum = 0xFF, PS_PLANE plane = FieldPlane,
long TimeOut = INFINITE, BOOL bWaitForIR = TRUE)
BOOL WaitForScreen(ECLScreenDesc* screenDesc, long TimeOut = INFINITE)
BOOL WaitWhileScreen(ECLScreenDesc* screenDesc, long TimeOut = INFINITE)
void RegisterPSEvent(ECLPSNotify * notify)
void RegisterPSEvent(ECLPSListener * listener)
void RegisterPSEvent(ECLPSListener * listener, int type)
void StartMacro(String MacroName)
void UnregisterPSEvent(ECLPSNotify * notify)
void UnregisterPSEvent(ECLPSListener * listener)
void UnregisterPSEvent(ECLPSListener * listener, int type)

The following methods are available for ECLPS :

void SendKeys(WCHAR * text), void SendKeys(WCHAR *text, ULONG AtPos), void SendKeys(WCHAR *text, ULONG AtRow, ULONG AtCol) ULONG SearchText(const WCHAR * const text, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE) ULONG SearchText(const WCHAR * const text, ULONG StartPos, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE) ULONG SearchText(const WCHAR * const text, ULONG StartRow,

ULONG StartCol, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE)
ULONG GetScreen(WCHAR * Buff, ULONG BuffLen, PS_PLANE Plane=TextPlane)
ULONG GetScreen(WCHAR * Buff, ULONG BuffLen, ULONG StartPos,
ULONG Length, PS_PLANE Plane=TextPlane)
ULONG GetScreen(WCHAR * Buff, ULONG BuffLen, ULONG StartRow,
ULONG StartCol, ULONG Length, PS_PLANE Plane=TextPlane)

ECLPS Constructor

This method uses a connection name or handle to create an ECLPS object.

Prototype

ECLPS(char ConnName)
ECLPS(long ConnHandle)

Parameters

char ConnName

One-character short name of the connection (A-Z or a-z).

long ConnHandle

Handle of an ECL connection.

Return Value

None

Example

The following example shows how to use a connection name to create an ECLPS object.

```
//-------------------------------------------------------------------
// ECLPS::ECLPS              (Constructor)
//
// Build a PS object from a name, and another from a handle.
//-------------------------------------------------------------------
void Sample58() {

ECLPS *PS1, *PS2;       // Pointer to PS objects
ECLConnList ConnList;   // Connection list object

try {
  // Create PS object for connection 'A'
  PS1 = new ECLPS('A');

  // Create PS object for first connection in conn list
  PS2 = new ECLPS(ConnList.GetFirstConnection()->GetHandle());
  printf("PS #1 is for connection %c, PS #2 is for connection %c.\n",
         PS1->GetName(), PS2->GetName());
  delete PS1;
  delete PS2;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

ECLPS Destructor

This method destroys the ECLPS object.

Prototype

~ECLPS()

Parameters

None

Return Value

None

Example

The following example shows how to destroy an ECLPS object.

```
ULONG  RowPos, ColPos;
ECLPS  *pPS;

try {
      pPS = new ECLPS('A');
      RowPos = pPS->ConvertPosToRow(544);
      ColPos = pPS->ConvertPosToCol(544);
      printf("PS position is at row %lu column %lu.",
      RowPos, ColPos);
      // Done with PS object so kill it
      delete pPS;
}
catch (ECLErr HE) {
  // Just report the error text in a message box
  MessageBox( NULL, HE.GetMsgText(), "Error!", MB_OK );
}
```

GetPCCodePage

The GetPCCodePage method retrieves the number designating the code page in force for the personal computer.

Prototype

int GetPCCodePage()

Parameters

None

Return Value

int

Number of the code page.

GetHostCodePage

The GetHostCodePage method retrieves the number designating the code page in force for the host computer.

Prototype

int GetHostCodePage()

Parameters

None

Return Value

int

Number of the code page.

GetOSCodePage

The GetOSCodePage method retrieves the number designating the code page in force for the operating system on the personal computer.

Prototype

int GetOSCodePage()

Parameters

None

Return Value

int

Number of the code page.

GetSize

This method returns the size of the presentation space for the connection associated with the ECLPS object. There are two signatures of the GetSize method. Using ULONG GetSize(), the size is returned as a linear value and represents the total number of characters in the presentation space. With void GetSize(ULONG *Rows, ULONG *Cols), the number of rows and columns of the presentation space is returned.

Prototype

ULONG GetSize()

void GetSize(ULONG *Rows, ULONG *Cols)

Parameters

ULONG *Rows

This output parameter is the number of rows in the presentation space.

ULONG *Cols

This output parameter is the number of columns in the presentation space.

Return Value

ULONG

Size of the presentation space as a linear value.

Example

The following is an example of using the GetSize method.

```
//-------------------------------------------------------------------
// ECLPS::GetSize
//
// Display dimensions of connection 'A'
//-------------------------------------------------------------------
void Sample59() {

ECLPS PS('A');       // PS object for connection A
ULONG Rows, Cols, Len;

PS.GetSize(&Rows, &Cols);   // Get num of rows and cols
// Could also write as:
Rows = PS.GetSizeRows();    // Redundant
Cols = PS.GetSizeCols();    // Redundant

Len = PS.GetSize();         // Get total size

printf("Connection A has %lu rows and %lu columns (%lu total length)\n",
       Rows, Cols, Len);

} // end sample
```

GetSizeRows

This method returns the number of rows in the Presentation Space for the connection associated with the ECLPS object.

Prototype

ULONG GetSizeRows()

Parameters

None

Return Value

ULONG

This is the number of rows in the Presentation Space.

Example

The following is an example of using the GetSizeRows method.

```
//-------------------------------------------------------------------
// ECLPS::GetSizeRows
//
// Display dimensions of connection 'A'
//-------------------------------------------------------------------
void Sample59() {

ECLPS PS('A');       // PS object for connection A
ULONG Rows, Cols, Len;

PS.GetSize(&Rows, &Cols);   // Get num of rows and cols
// Could also write as:
Rows = PS.GetSizeRows();    // Redundant
Cols = PS.GetSizeCols();    // Redundant

Len = PS.GetSize();         // Get total size

printf("Connection A has %lu rows and %lu columns (%lu total length)\n",
       Rows, Cols, Len);
} // end sample
```

GetSizeCols

This method returns the number of columns in the Presentation Space for the connection associated with the ECLPS object.

Prototype

ULONG GetSizeCols()

Parameters

None

Return Value

ULONG

This is the number of columns in the Presentation Space.

Example

The following is an example of using the GetSizeCols method.

```
//-------------------------------------------------------------------
// ECLPS::GetSizeCols
//
// Display dimensions of connection 'A'
//-------------------------------------------------------------------
void Sample59() {

ECLPS PS('A');       // PS object for connection A
ULONG Rows, Cols, Len;

PS.GetSize(&Rows, &Cols);   // Get num of rows and cols
// Could also write as:
Rows = PS.GetSizeRows();    // Redundant
Cols = PS.GetSizeCols();    // Redundant

Len = PS.GetSize();         // Get total size

printf("Connection A has %lu rows and %lu columns (%lu total length)\n",
       Rows, Cols, Len);

} // end sample
```

GetCursorPos

This method returns the position of the cursor in the presentation space for the connection associated with the ECLPS object. There are two signatures for the GetCursorPos method. Using ULONG GetCursorPos(), the position is returned as a linear (1-based) position. With void GetCursorPos(ULONG *Row, ULONG * Col), the position is returned as a row and column coordinate.

Prototype

ULONG GetCursorPos()
void GetCursorPos(ULONG *Row, ULONG *Col)

Parameters

ULONG *Row

This output parameter is the row coordinate of the host cursor.

ULONG *Col

This output parameter is the column coordinate of the host cursor.

Return Value

ULONG

Cursor position represented as a linear value.

Example

The following is an example of using the GetCursorPos method.

```
//-------------------------------------------------------------------
// ECLPS::GetCursorPos
//
// Display position of host cursor in connection 'A'
//-------------------------------------------------------------------
void Sample60() {

ECLPS PS('A');       // PS object for connection A
ULONG Row, Col, Pos;

PS.GetCursorPos(&Row, &Col);   // Get row/col position
// Could also write as:
Row = PS.GetCursorPosRow();    // Redundant
Col = PS.GetCursorPosCol();    // Redundant

Pos = PS.GetCursorPos();       // Get linear position

printf("Host cursor of connection A is at row %lu column %lu
 (linear position %lu)\n", Row, Col, Pos);

} // end sample
```

```
/
```

GetCursorPosRow

This method returns the row position of the cursor in the Presentation Space for the connection associated with the ECLPS object.

Prototype

ULONG GetCursorPosRow()

Parameters

None

Return Value

ULONG

This is the row position of the cursor in the Presentation Space.

Example

The following is an example of using the GetCursorPosRow method.

```
//-------------------------------------------------------------------
// ECLPS::GetCursorPosRow
//
// Display position of host cursor in connection 'A'
//-------------------------------------------------------------------
void Sample60() {

ECLPS PS('A');       // PS object for connection A
ULONG Row, Col, Pos;

PS.GetCursorPos(&Row, &Col);   // Get row/col position
// Could also write as:
Row = PS.GetCursorPosRow();    // Redundant
Col = PS.GetCursorPosCol();    // Redundant

Pos = PS.GetCursorPos();       // Get linear position

printf("Host cursor of connection A is at row %lu column %lu
 (linear position %lu)\n", Row, Col, Pos);

} // end sample
```

GetCursorPosCol

This method returns the column position of the cursor in the Presentation Space for the connection associated with the ECLPS object.

Prototype

ULONG GetCursorPosCol()

Parameters

None

Return Value

ULONG

This is the column position of the cursor in the Presentation Space.

Example

The following is an example of using the GetCursorPosCol method.

```
//-------------------------------------------------------------------
// ECLPS::GetCursorPosCol
//
// Display position of host cursor in connection 'A'
//-------------------------------------------------------------------
void Sample60() {

ECLPS PS('A');       // PS object for connection A
ULONG Row, Col, Pos;

PS.GetCursorPos(&Row, &Col);   // Get row/col position
// Could also write as:
Row = PS.GetCursorPosRow();    // Redundant
Col = PS.GetCursorPosCol();    // Redundant

Pos = PS.GetCursorPos();       // Get linear position

printf("Host cursor of connection A is at row %lu column %lu
 (linear position %lu)\n", Row, Col, Pos);

} // end sample

//-----------------------------------------------------------------
```

SetCursorPos

The SetCursorPos method sets the position of the cursor in the presentation space for the connection associated with the ECLPS object. There are two signatures for the SetCursorPos method. The position can be specified as a linear (1-based) position using void SetCursorPos(ULONG pos), or as a row and column coordinate using void SetCursorPos(ULONG Row, ULONG Col).

Prototype

void SetCursorPos(ULONG pos),

void SetCursorPos(ULONG Row, ULONG Col)

Parameters

ULONG pos

Cursor position as a linear position.

ULONG Row

Cursor row coordinate.

ULONG Col

Cursor column coordinate.

Return Value

None

Example

The following is an example of using the SetCursorPos method.

```
--
// ECLPS::SetCursorPos
//
// Set host cursor to row 2 column 1.
//-------------------------------------------------------------------
void Sample61() {

ECLPS PS('A');          // PS object for connection A

PS.SetCursorPos(2, 1);  // Put cursor at row 2, column 1
printf("Cursor of connection A set to row 2 column 1.\n");

} // end sample

/
```

SendKeys

The SendKeys method sends a null-terminated string of keys to the presentation space for the connection associated with the ECLPS object. There are three signatures for the SendKeys method. If no position is specified, the keystrokes are entered starting at the current host cursor position. A position may be specified (in linear or row and column coordinates), in which case the host cursor is first moved to the given position.

The text string may contain plain text characters, which are written to the presentation space exactly as given. In addition, the string can contain imbedded keywords (mnemonics) that represent various control keystrokes such as 3270 Enter keys and 5250 PageUp keys. Keywords are enclosed in square brackets (for example, [enter]). When such a keyword is encountered in the string it is translated into the proper emulator command and sent. A text string may contain any number of plain characters and imbedded keywords. The keywords are processed from left to right until the end of the string is reached. For example, the following string would cause the characters ABC to be typed at the current cursor position, followed by a 3270 Erase-end-of-field keystroke, followed by a 3270 Tab keystroke, followed by XYZ and a PF1 key:

```
ABC[eraseeof][tab]XYZ[pf1]
```

Note: Blank characters in the string are written to the host presentation space like any other plain text character. Therefore, blanks should not be used to separate keywords or text.

To send a left or right square bracket character to the host, it must be doubled in the text string (for example, it must occur twice to cause a single bracket to be written). The following example causes the string “A [:]” to be written to the presentation space.

```
A[[:]]
```

If you attempt to write keystrokes to a protected position on the screen, the keyboard locks and the remainder of the keystrokes are discarded.

Refer to Sendkeys Mnemonic Keywords on page 1156 for a list of keywords.

Prototype

void SendKeys(char * text),
void SendKeys(char * text, ULONG AtPos),
void SendKeys(char * text, ULONG AtRow, ULONG AtCol)

Parameters

Char *text

String of keys to send to the presentation space.

ULONG AtPos

Position at which to start writing keystrokes.

ULONG AtRow

Row at which to start writing keystrokes.

ULONG AtCol

Column at which to start writing keystrokes.

Return Value

None

Example

The following is an example of using the SendKeys method.

```
//-------------------------------------------------------------------
// ECLPS::SendKeys
//
// Sends a series of keystrokes, including 3270 function keys, to
// the host on connection A.
//-------------------------------------------------------------------
void Sample62() {

ECLPS PS('A');         // PS object for connection A

// The following key string will erase from the current cursor
// position to the end of the field, and then type the given
// characters into the field.
char  SendStr[] = "[eraseeof]ZIEWin is really cool";

// Note that an ECL error is thrown if we try to send keys to
// a protected field.

try {
  PS.SendKeys(SendStr);        // Do it at the current cursor position
  PS.SendKeys(SendStr, 3, 10); // Again at row 3 column 10
}
catch (ECLErr Err) {
  printf("Failed to send keys: %s\n", Err.GetMsgText());
}

} // end sample
```

SearchText

The SearchText method searches for text in the presentation space of the connection associated with the ECLPS object. The method returns the linear position at which the text is found, or zero if the text is not found. The search may be made in the forward (left to right, top to bottom) or backward (right to left, bottom to top) directions using the optional Dir parameter. The search can be case-sensitive or case folded (insensitive) using the optional FoldCase parameter.

If no starting position is given, the search starts at the beginning of the screen for forward searches, or at the end of the screen for backward searches. A starting position may be given in terms of a linear position or row and column coordinates. If a starting position is given it indicates the position at which to begin the search. Forward searches search from the starting position (inclusive) to the last character of the screen. Backward searches search from the starting position (inclusive) to the first character of the screen.

The search string must exist completely within the search area for the search to be successful (for example, if the search string spans over the specified starting position it will not be found).

The returned linear position may be converted to row and column coordinates using the base class ConvertPosToRowCol method.

Prototype

```
ULONG SearchText(const char * const text, PS_DIR Dir=SrchForward,
    BOOL FoldCase=FALSE)
ULONG SearchText(const char * const text,
    ULONG StartPos, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE)
ULONG SearchText(const char char * const text, ULONG StartRow,
    ULONG StartCol, PS_DIR Dir=SrchForward, BOOL FoldCase=FALSE)
```

Parameters

char *text

Null-terminated string to search for.

PS_DIR Dir

Optional parameter indicating the direction in which to search. If specified, must be one of **SrchForward** or **SrchBackward**. The default is **SrchForward**.

BOOL FoldCase

Optional parameter indicating the case-sensitivity of the search. If specified as FALSE the text string must exactly match the presentation space including the use of uppercase and lowercase characters. If specified as TRUE, the text string will be found without regard to uppercase or lowercase. The default is FALSE.

ULONG StartPos

Indicates the starting linear position of the search. This position will be included in the search.

ULONG StartRow

Indicates the row in which to start the search.

ULONG StartCol

Indicates the column in which to start the search.

Return Value

ULONG

Linear position of the found string, or zero if not found.

Example

The following is an example of using the SearchText method.

```
/-------------------------------------------------------------------
// ECLPS::SearchText
//
// Search for a string in various parts of the screen.
//-------------------------------------------------------------------
void Sample63() {

ECLPS PS('A');            // PS object
char  FindStr[] = "HCL";  // String to search for
ULONG LastOne;            // Position of search result

// Case insensative search of entire screen

printf("Searching for '%s'...\n", FindStr);
printf("  Anywhere, any case: ");
if (PS.SearchText(FindStr, TRUE) != 0)
  printf("Yes\n");
else
  printf("No\n");

// Backward, case sensative search on line 1

printf("  Line 1, exact match: ");
if (PS.SearchText(FindStr, 1, 80, SrchBackward) != 0)
  printf("Yes\n");
else
  printf("No\n");

// Backward, full screen search

LastOne = PS.SearchText(FindStr, SrchBackward, TRUE);
if (LastOne != 0)
  printf("  Last occurance on the screen is at row %lu, column %lu.\n",
         PS.ConvertPosToRow(LastOne), PS.ConvertPosToCol(LastOne));

} // end sample
```

GetScreen

This method retrieves data from the presentation space of the connection associated with the ECLPS object. The data is returned as a linear array of byte values, one byte per presentation space character position. The array is not null terminated except when data is retrieved from the TextPlane, in which case a single null termination byte is appended.

The application must supply a buffer for the returned data, and the length of the buffer. If the requested data does not fit into the buffer it is truncated. For TextPlane data, the buffer must include at least one extra byte for the terminating null. The method returns the number of bytes copied to the application buffer (not including the terminating null for TextPlane copies).

The application must specify the number of bytes of data to retrieve from the presentation space. If the starting position plus this length exceeds the size of the presentation space an error is thrown. Data is returned starting at the given starting position or row 1, column 1 if no starting position is specified. Returned data is copied from the presentation space in a linear fashion from left to right, top to bottom spanning multiple rows up to the length specified. If the application wants to get screen data for a rectangular area of the screen, the GetScreenRect method should be used.

The application can specify any plane for which to retrieve data. If no plane is specified, the TextPlane is retrieved. See ECL Planes — Format and Content on page 1159 for details on the different ECL planes.

Prototype

```
ULONG GetScreen(char * Buff, ULONG BuffLen, PS_PLANE Plane=TextPlane)
ULONG GetScreen(char * Buff, ULONG BuffLen, ULONG StartPos, ULONG Length,
    PS_PLANE Plane=TextPlane)
ULONG GetScreen(char * Buff, ULONG BuffLen, ULONG StartRow, ULONG StartCol,
    ULONG Length, PS_PLANE Plane=TextPlane)
```

Parameters

char *Buff

Pointer to application supplied buffer of at least BuffLen size.

ULONG BuffLen

Number of bytes in the supplied buffer.

ULONG StartPos

Linear position in the presentation space at which to start the copy.

ULONG StartRow

Row in the presentation space at which to start the copy.

ULONG StartCol

Column in the presentation space at which to start the copy.

ULONG Length

Linear number of bytes to copy from the presentation space.

PS_PLANE plane

Optional parameter specifying which presentation space plane is to be copied. If specified, must be one of **TextPlane**, **ColorPlane**, **FieldPlane**, and **ExfieldPlane**. The default is **TextPlane**. See ECL Planes — Format and Content on page 1159 for the content and format of the different ECL planes.

Return Value

ULONG

Number of data bytes copied from the presentation space. This value does not include the trailing null byte for TextPlane copies.

Example

The following is an example of using the GetScreen method.

```
//-------------------------------------------------------------------
// ECLPS::GetScreen
//
// Get text and other planes of data from the presentation space.
//-------------------------------------------------------------------
void Sample64() {

ECLPS PS('A');            // PS object
char  *Text;              // Text plane data
char  *Field;             // Field plane data
ULONG Len;                // Size of PS

Len = PS.GetSize();

// Note text buffer needs extra byte for null terminator

Text  = new char[Len + 1];
Field = new char[Len];

PS.GetScreen(Text, Len+1);                // Get entire screen (text)
PS.GetScreen(Field, Len, FieldPlane);     // Get entire field plane
PS.GetScreen(Text, Len+1, 1, 1, 80);      // Get line 1 of text

printf("Line 1 of the screen is:\n%s\n", Text);

delete []Text;
delete []Field;

} // end sample
```

GetScreenRect

This method retrieves data from the presentation space of the connection associated with the ECLPS object. The data is returned as a linear array of byte values, one byte per presentation space character position. The array is not null terminated.

The application supplies a starting and ending coordinate in the presentation space. These coordinates form the opposing corner points of a rectangular area. The presentation space within the rectangular area is copied to the application buffer as a single linear array. The starting and ending points may be in any spatial relationship to each other. The copy is defined to start from the row containing the uppermost point to the row containing the lowermost point, and from the left-most column to the right-most column. Both coordinates must be within the bounds of the size of the presentation space or an error is thrown. The coordinates may be specified in terms of linear position or row and column numbers.

The supplied application buffer must be at least large enough to contain the number of bytes in the rectangle. If the buffer is too small, no data is copied and zero is returned as the method result. Otherwise the method returns the number of bytes copied.

The application can specify any plane for which to retrieve data. If no plane is specified, the TextPlane is retrieved. See ECL Planes — Format and Content on page 1159 for details on the different ECL planes.

Prototype

```
ULONG GetScreenRect(char * Buff, ULONG BuffLen,
    ULONG StartPos, ULONG EndPos, PS_PLANE Plane=TextPlane)
ULONG GetScreenRect(char * Buff, ULONG BuffLen,
    ULONG StartRow, ULONG StartCol, ULONG EndRow,
    ULONG EndCol, PS_PLANE Plane=TextPlane)
```

Parameters

char *Buff

Pointer to application supplied buffer of at least BuffLen size.

ULONG BuffLen

Number of bytes in the supplied buffer.

ULONG StartPos

Linear position in the presentation space of one corner of the copy rectangle.

ULONG EndPos

Linear position in the presentation space of one corner of the copy rectangle.

ULONG StartRow

Row in the presentation space of one corner of the copy rectangle.

ULONG StartCol

Column in the presentation space of one corner of the copy rectangle.

ULONG EndRow

Row in the presentation space of one corner of the copy rectangle.

ULONG EndCol

Column in the presentation space of one corner of the copy rectangle.

PS_PLANE plane

Optional parameter specifying which presentation space plane is to be copied. If specified, must be one of **TextPlane**, **ColorPlane**, **FieldPlane**, or **ExfieldPlane**. The default is **TextPlane**. See ECL Planes — Format and Content on page 1159 for the content and format of the different ECL planes.

Return Value

ULONG

Number of data bytes copied from the presentation space.

Example

The following is an example of using the GetScreenRect method.

```
------------------------------
// ECLPS::GetScreenRect
//
// Get rectangular parts of the host screen.
//-------------------------------------------------------------------
void Sample66() {

ECLPS PS('A');         // PS object for connection A
char Buff[4000];       // Big buffer

// Get first 2 lines of the screen text
PS.GetScreenRect(Buff, sizeof(Buff), 1, 1, 2, 80);

// Get last 2 lines of the screen
PS.GetScreenRect(Buff, sizeof(Buff),
                 PS.GetSizeRows()-1,
                 1,
                 PS.GetSizeRows(),
                 PS.GetSizeCols());

// Get just a part of the screen (VM main menu calendar)
PS.GetScreenRect(Buff, sizeof(Buff),
                 5,  51,
                 13, 76);

// Same as previous (specify any 2 oposite corners of the rectangle)
PS.GetScreenRect(Buff, sizeof(Buff),
                 13, 51,
                 5,  76);

// Note results are placed in buffer end-to-end with no line delimiters
printf("Contents of rectangular screen area:\n%s\n", Buff);

} // end sample
```

SetText

The SetText method sends a character array to the Presentation Space for the connection associated with the ECLPS object. Although this is similar to the SendKeys method, it is different in that it does not send mnemonic keystrokes (for example, [enter] or [pf1]).

If a position is not specified, the text is written starting at the current cursor position.

Prototype

void SetText(char *text);

void SetText(char *text, ULONG AtPos);

void SetText(char *text, ULONG AtRow, ULONG AtCol);

Parameters

char *text

Null terminated string of characters to copy to the presentation space.

ULONG AtPos

Linear position in the presentation space at which to begin the copy.

ULONG AtRow

Row in the presentation space of which to begin the copy.

ULONG AtCol

Column in the presentation space at which to begin the copy.

Return Value

None

Example

The following is an example of using the SetText method.

```
//-------------------------------------------------------------------
// ECLPS::SetText
//
// Update various input fields of the screen.
//-------------------------------------------------------------------
void Sample65() {

ECLPS PS('A');         // PS object for connection A

// Note that an ECL error is thrown if we try to write to
// a protected field.

try {
  // Update first 2 input fields of the screen.  Note
  // fields are not erased before update.
  PS.SendKeys("[home]");
  PS.SetText("Field 1");
  PS.SendKeys("[tab]");
  PS.SetText("Field 2");
  // Note: Above 4 lines could also be written as:
  // PS.SendKeys("[home]Field 1[tab]Field 2");
  // But SetText() is faster, esp for long strings
}
catch (ECLErr Err) {
  printf("Failed to send keys: %s\n", Err.GetMsgText());
}

} // end sample

//-------------------------------------
```

CopyText

This method copies the text from a given location in presentation space of a specified length to clipboard. The length of the text copied will be the length specified, if no length is specified the text till the end of presentation space is copied. If the location is not specified, the text copied is from the current cursor position in presentation space. In case of no parameters, whole presentation space is copied to clipboard.

Prototype

void CopyText ();

void CopyText (ULONG Long Len);

void CopyText (ULONG AtPos, ULONG Long Len);

void CopyText (ULONG AtRow, ULONG AtCol, ULONG Long Len);

Parameters

ULONG Long Len

Linear number of bytes to copy from the presentation space.

ULONG AtPos

Linear position in the presentation space at which to begin the copy.

ULONG AtRow

Row in the presentation space of which to begin the copy.

ULONG AtCol

Column in the presentation space at which to begin the copy.

Return Value

None

Example

The following is an example of using the CopyText method.

```
//-------------------------------------------------------------------
// ECLPS::CopyText
//
// Copy text from Presentation Space to clipboard.
//-------------------------------------------------------------------
void Sample126() {

ECLPS PS('A');         // PS object for connection A
long row, col, length2copy;

// Note that an ECL error is thrown if we try to write to
// a protected field.
try {
   printf("Please enter the position and length to copy from PS [row col length2copy] \n");

   scanf("%ld %ld %ld", &row, &col, &length2copy);
  PS.CopyText(row, col, length2copy);
  }
catch (ECLErr Err) {
   printf("Failed to copy text: %s\n", Err.GetMsgText());
}
} // end sample
//-------------------------------------
```

PasteText

This method pastes the text of specified length from clipboard to a given location in presentation space. The length of the text pasted is the length specified, if no length is specified the whole text in clipboard is pasted until it reaches the end of presentation space. If the location is not specified, the text is pasted at the current cursor position in presentation space. If the presentation space is field formatted and while pasting the clipboard content, when there is a tab character '\t,' the remaining paste content is moved to the next writable field.

Prototype

void PasteText ();

void PasteText (ULONG Long Len);

void PasteText (ULONG AtPos, ULONG Long Len);

void PasteText (ULONG AtRow, ULONG AtCol, ULONG Long Len);

Parameters

ULONG Long Len

Linear number of bytes to paste from the presentation space.

ULONG AtPos

Linear position in the presentation space at which to begin the paste.

ULONG AtRow

Row in the presentation space of which to begin the paste.

ULONG AtCol

Column in the presentation space at which to begin the paste.

Return Value

None

Example

The following is an example of using the PasteText method.

```
//-------------------------------------------------------------------
// ECLPS::PasteText
//
// Paste text to Presentation Space from clipboard.
//-------------------------------------------------------------------
void Sample127() {

ECLPS PS('A');         // PS object for connection A
long row, col, length2paste;

// Note that an ECL error is thrown if we try to write to
// a protected field.
try {
   printf("Please enter the position and length to paste from clipboard [row col length2paste] \n");
   scanf("%ld %ld %ld", &row, &col, &length2paste);
  PS.PasteText(row, col, length2paste);
}
catch (ECLErr Err) {
  printf("Failed to paste text: %s\n", Err.GetMsgText());
}
} // end sample
//----------------------------------------------
```

ConvertPosToRowCol

The ConvertPosToRowCol method converts a position in the presentation space represented as a linear array to a position in the presentation space given in row and column coordinates. The position converted is in the presentation space for the connection associated with the ECLPS object.

Prototype

void ConvertPosToRowCol(ULONG pos, ULONG *row, ULONG *col)

Parameters

ULONG pos

Position to convert in the presentation space represented as a linear array.

ULONG *row

Converted row coordinate in the presentation space.

ULONG *col

Converted column coordinate in the presentation space.

Return Value

None

Example

The following example shows how to convert a position in the presentation space represented as a linear array to a position shown in row and column coordinates.

```
///-------------------------------------------------------------------
// ECLPS::ConvertPosToRowCol
//
// Find a string in the presentation space and display the row/column
// coordinate of its location.
//-------------------------------------------------------------------
void Sample67() {

ECLPS PS('A');            // PS Object
ULONG FoundPos;           // Linear position
ULONG FoundRow,FoundCol;


FoundPos = PS.SearchText("HCL", TRUE);
if (FoundPos != 0) {
  PS.ConvertPosToRowCol(FoundPos, &FoundRow, &FoundCol);
  // Another way to do the same thing:
  FoundRow = PS.ConvertPosToRow(FoundPos);
  FoundCol = PS.ConvertPosToCol(FoundPos);

  printf("String found at row %lu column %lu (position %lu)\n",
         FoundRow, FoundCol, FoundPos);
}
else printf("String not found.\n");

} // end sample
```

ConvertRowColToPos

The ConvertRowColToPos method converts a position in the presentation space in row and column coordinates to a position in the presentation space represented as a linear array. The position converted is in the presentation space for the connection associated with the ECLPS object.

Prototype

ULONG ConvertRowColToPos(ULONG row, ULONG col)

Parameters

ULONG row

Row coordinate to convert in the presentation space.

ULONG col

Column coordinate to convert in the presentation space.

Return Value

ULONG

Converted position in the presentation space represented as a linear array.

Example

The following example shows how to convert a position in the presentation space shown in row and column coordinates to a linear array position.

```
///-------------------------------------------------------------------
// ECLPS::ConvertRowColToPos
//
// Find a string in the presentation space and display the row/column
// coordinate of its location.
//-------------------------------------------------------------------
void Sample67() {

ECLPS PS('A');             // PS Object
ULONG FoundPos;            // Linear position
ULONG FoundRow,FoundCol;


FoundPos = PS.SearchText("HCL", TRUE);
if (FoundPos != 0) {
  PS.ConvertPosToRowCol(FoundPos, &FoundRow, &FoundCol);
  // Another way to do the same thing:
  FoundRow = PS.ConvertPosToRow(FoundPos);
  FoundCol = PS.ConvertPosToCol(FoundPos);

  printf("String found at row %lu column %lu (position %lu)\n",
         FoundRow, FoundCol, FoundPos);
}
else printf("String not found.\n");
} // end sample
```

ConvertPosToRow

This method takes a linear position value in the Presentation Space and returns the row in which it resides for the connection associated with the ECLPS object.

Prototype

ULONG ConvertPosToRow(ULONG Pos)

Parameters

ULONG Pos

This is the linear position in the Presentation Space to convert.

Return Value

ULONG

This is the row position for the linear position.

Example

The following is an example of using the ConvertPosToRow method.

```
///-------------------------------------------------------------------
// ECLPS::ConvertPosToRow
//
// Find a string in the presentation space and display the row/column
// coordinate of its location.
//-------------------------------------------------------------------
void Sample67() {

ECLPS PS('A');            // PS Object
ULONG FoundPos;           // Linear position
ULONG FoundRow,FoundCol;


FoundPos = PS.SearchText("HCL", TRUE);
if (FoundPos != 0) {
  PS.ConvertPosToRowCol(FoundPos, &FoundRow, &FoundCol);
  // Another way to do the same thing:
  FoundRow = PS.ConvertPosToRow(FoundPos);
  FoundCol = PS.ConvertPosToCol(FoundPos);

  printf("String found at row %lu column %lu (position %lu)\n",
         FoundRow, FoundCol, FoundPos);
}
else printf("String not found.\n");
} // end sample
```

ConvertPosToCol

This method takes a linear position value in the Presentation Space and returns the column in which it resides for the connection associated with the ECLPS object.

Prototype

ULONG ConvertPosToCol(ULONG Pos)

Parameters

ULONG Pos

This is the linear position in the Presentation Space to convert.

Return Value

ULONG

This is the column position for the linear position.

Example

The following is an example of using the ConvertPosToCol method.

```
///-------------------------------------------------------------------
/// ECLPS::ConvertPosToCol
//
// Find a string in the presentation space and display the row/column
// coordinate of its location.
//-------------------------------------------------------------------
void Sample67() {

ECLPS PS('A');             // PS Object
ULONG FoundPos;            // Linear position
ULONG FoundRow,FoundCol;


FoundPos = PS.SearchText("HCL", TRUE);
if (FoundPos != 0) {
  PS.ConvertPosToRowCol(FoundPos, &FoundRow, &FoundCol);
  // Another way to do the same thing:
  FoundRow = PS.ConvertPosToRow(FoundPos);
  FoundCol = PS.ConvertPosToCol(FoundPos);

  printf("String found at row %lu column %lu (position %lu)\n",
         FoundRow, FoundCol, FoundPos);
}
else printf("String not found.\n");
} // end sample
```

RegisterKeyEvent

The RegisterKeyEvent function registers an application-supplied object to receive notification of operator keystroke events. The application must construct an object derived from the ECLKeyNotify abstract base class. When an operator keystroke occurs, the NotifyEvent() method of the application supplied object is called. The application can choose to have the keystroke filtered or passed on and processed in the usual way. See ECLKeyNotify Class on page 818 for more details.

Implementation Restriction: Only one object may be registered to receive keystroke events at a time.

Prototype

void RegisterKeyEvent(ECLKeyNotify *NotifyObject)

Parameters

ECLKeyNotify *NotifyObject

Application object derived from ECLKeyNotify class.

Return Value

None

Example

The following example shows how to register an application-supplied object to receive notification of operator keystroke events. See the ECLKeyNotify Class on page 818 for a RegisterKeyEvent example.

```
// This is the declaration of your class derived from ECLKeyNotify....
class MyKeyNotify: public ECLKeyNotify
{
public:
  // App can put parms on constructors if needed
  MyKeyNotify();          // Constructor
  MyKeyNotify();         // Destructor

  // App must define the NotifyEvent method
  int NotifyEvent(char KeyType[2], char KeyString[7]);  // Keystroke callback

private:
  // Whatever you like...
};
// this is the implementation of app methods...

int MyKeyNotify::NotifyEvent( ECLPS *, char *KeyType, char *Keystring )
{
  if (...) {
    ...
    return 0;  // Remove keystroke (filter)
  }
  else
    ...
    return 1;  // Pass keystroke to emulator as usual
  }
}

// this would be the code in say, WinMain...

ECLPS *pPS;            // Pointer to ECLPS object
MyKeyNotify *MyKeyNotifyObject; // My key notification object,derived
                                   // from ECLKeyNotify

try {
  pPS = new ECLPS('A');              // Create PS object for 'A' session

  // Register for keystroke events
  MyKeyNotifyObject = new MyKeyNotify();
  pPS->RegisterKeyEvent(MyKeyNotifyObject);

  // After this, MyKeyNotifyObject->NotifyEvent() will be called
  // for each operator keystroke...
}
catch (ECLErr HE) {
  // Just report the error text in a message box
  MessageBox( NULL, HE.GetMsgText(), "Error!", MB_OK );
}
```

UnregisterKeyEvent

The UnregisterKeyEvent method unregisters an application object previously registered for keystroke events with the RegisterKeyEvent function. A registered application notify object should not be destroyed without first calling this function to unregister it. If there is no notify object currently registered, or the registered object is not the NotifyObject passed in, this function does nothing (no error is thrown).

Prototype

virtual UnregisterKeyEvent(ECLKeyNotify *NotifyObject)

Parameters

ECLKeyNotify *NotifyObject

Object currently registered for keystroke events.

Return Value

None

Example

See the ECLKeyNotify Class on page 818 for a UnregisterKeyEvent example.

GetFieldList

This method returns a pointer to an ECLFieldList object. The field list object can be used to iterate over the list of fields in the host presentation space. The ECLFieldList object returned by this function is automatically destroyed when the ECLPS object is destroyed. See ECLFieldList Class on page 810 for more information about this object.

Prototype

ECLFieldList *GetFieldList()

Parameters

None

Return Value

ECLFieldList *

Pointer to ECLFieldList object.

Example

The following example shows how to return a pointer to an ECLFieldList object.

```
// ECLPS::GetFieldList
//
// Display number of fields on the screen.
//-------------------------------------------------------------------
void Sample68() {

ECLPS        *PS;           // Pointer to PS object
ECLFieldList *FieldList;    // Pointer to field list object

try {
  PS = new ECLPS('A');                // Create PS object for 'A'

  FieldList = PS->GetFieldList();     // Get pointer to field list
  FieldList->Refresh();               // Build the field list

  printf("There are %lu fields on the screen of connection %c.\n",
    FieldList->GetFieldCount(), PS->GetName());

  delete PS;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

WaitForCursor

The WaitForCursor method waits for the cursor in the presentation space of the connection associated with the ECLPS object to be located at a specified position.

Prototype

BOOL WaitForCursor(int Row, int Col, long nTimeOut=INFINITE,
    BOOL bWaitForIR=TRUE)

Parameters

int Row

Row position of the cursor. If negative, this value indicates the Row position from the bottom of the PS.

int Col

Column position of the cursor. If negative, this value indicates the Cursor position from the edge of the PS.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
int TimeOut = 5000;
BOOL waitOK = ps.WaitForCursor(23,1,TimeOut, TRUE);

// do the processing for the screen
```

WaitWhileCursor

The WaitWhileCursor method waits while the cursor in the presentation space of the connection associated with the ECLPS object is located at a specified position.

Prototype

BOOL WaitWhileCursor(int Row, int Col, long nTimeOut=INFINITE,
    BOOL bWaitForIR=TRUE)

Parameters

int Row

Row position of the cursor. If negative, this value indicates the Row position from the bottom of the PS.

int Col

Column position of the cursor. If negative, this value indicates the Cursor position from the edge of the PS.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
int TimeOut = 5000;
BOOL waitOK = ps.WaitWhileCursor(23,1,TimeOut, TRUE);

// do the processing for when the screen goes away
```

WaitForString

The WaitForString method waits for the specified string to appear in the presentation space of the connection associated with the ECLPS object. If the optional Row and Column parameters are used, the string must begin at the specified position. If 0,0 are passed for Row,Col the method searches the entire PS.

Prototype

```
BOOL WaitForString( char* WaitString, int Row=0, int Col=0, long nTimeOut=INFINITE,
    BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
```

Parameters

char* WaitString

The string which will be the subject of the wait.

int Row

Row position of the cursor. If negative, this value indicates the Row position from the bottom of the PS. The default is zero.

int Col

Column position of the cursor. If negative, this value indicates the Cursor position from the edge of the PS. The default is zero.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

BOOL bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitForString("LOGON");

// do the processing for the screen
```

WaitWhileString

The WaitWhileString method waits while the specified string is in the presentation space of the connection associated with the ECLPS object. If the optional Row and Column parameters are used, the string must begin at the specified position. If 0,0 are passed for Row,Col the method searches the entire PS.

Prototype

```
BOOL WaitWhileString(char* WaitString, int Row=0, int Col=0,
                     long nTimeOut=INFINITE,
                     BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
```

Parameters

char* WaitString

The string which will be the subject of the wait.

int Row

Start Row position of the string. If negative, this value indicates the Row position from the bottom of the PS. The default is zero.

int Col

Start Column position of the string. If negative, this value indicates the Cursor position from the edge of the PS. The default is zero.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

BOOL bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitWhileString("LOGON");

// do the processing for when the screen goes away
```

WaitForStringInRect

The WaitForStringInRect method waits for the specified string to appear in the presentation space of the connection associated with the ECLPS object in the specified Rectangle.

Prototype

```
BOOL WaitForStringInRect(char* WaitString, int sRow, int sCol, int eRow,int eCol,
    long nTimeOut=INFINITE, BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
```

Parameters

char* WaitString

The string which will be the subject of the wait.

int Row

Start Row position of the rectangle.

int Col

Start Column position of the rectangle.

int eRow

Ending row position of the search rectangle.

int eCol

Ending column position of the search rectangle.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

BOOL bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitForStringInRect("LOGON",1,1,23,80);

// do the processing for the screen
```

WaitWhileStringInRect

The WaitWhileStringInRect method waits while the specified string is in the presentation space of the connection associated with the ECLPS object in the specified Rectangle.

Prototype

```
BOOL WaitWhileStringInRect(char* WaitString, int sRow, int sCol, int eRow,int eCol,
    long nTimeOut=INFINITE, BOOL bWaitForIR=TRUE, BOOL bCaseSens=TRUE)
```

Parameters

char* WaitString

The string which will be the subject of the wait.

int Row

Start Row position of the rectangle.

int Col

Start Column position of the rectangle.

int eRow

Ending row position of the search rectangle.

int eCol

Ending column position of the search rectangle.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

BOOL bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitWhileStringInRect("LOGON",1,1,23,80);

// do the processing for when the screen goes away
```

WaitForAttrib

The WaitForAttrib method will wait until the specified Attribute value appears in the presentation space of the connection associated with the ECLPS object at the specified Row/Column position. The optional MaskData parameter can be used to control which values of the attribute you are looking for. The optional plane parameter allows you to select any of the four PS planes.

Prototype

```
BOOL WaitForAttrib(int Row, int Col, unsigned char AttribDatum,
    unsigned char MskDatum= 0xFF, PS_PLANE plane = FieldPlane,
    long TimeOut = INFINITE, BOOL bWaitForIR = TRUE)
```

Parameters

int Row

Row position of the attribute.

int Col

Column position of the attribute.

unsigned char AttribDatum

The 1 byte HEX value of the attribute to wait for.

unsigned char MskDatum

The 1 byte HEX value to use as a mask with the attribute. This parameter is optional. The default value is 0xFF.

PS_PLANE plane

The plane of the attribute to get. The plane can have the following values: **TextPlane**, **ColorPlane**, **FieldPlane**, and **ExfieldPlane**. See ECL Planes — Format and Content on page 1159 for the content and format of the different ECL planes.

This parameter is optional. The default is FieldPlane.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitForAttrib(10, 16, 0xE0, 0xFF, FieldPlane, INFINITE, FALSE);

// do the processing for when the screen goes away
```

WaitWhileAttrib

The WaitWhileAttrib method waits while the specified Attribute value appears in the presentation space of the connection associated with the ECLPS object at the specified Row/Column position. The optional MaskData parameter can be used to control which values of the attribute you are looking for. The optional plane parameter allows you to select any of the four PS planes.

Prototype

```
BOOL WaitWhileAttrib(int Row, int Col, unsigned char AttribDatum,
    unsigned char MskDatum= 0xFF, PS_PLANE plane = FieldPlane,
    long TimeOut = INFINITE, BOOL bWaitForIR = TRUE)
```

Parameters

int Row

Row position of the attribute.

int Col

Column position of the attribute unsigned.

char AttribDatum

The 1 byte HEX value of the attribute to wait for.

unsigned char MskDatum

The 1 byte HEX value to use as a mask with the attribute. This parameter is optional. The default value is 0xFF.

PS_PLANE plane

The plane of the attribute to get. The plane can have the following values: **TextPlane**, **ColorPlane**, **FieldPlane**, and **ExfieldPlane**. See ECL Planes — Format and Content on page 1159 for the content and format of the different ECL planes.

This parameter is optional. The default is FieldPlane.

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

BOOL bWaitForIR

If this value is true, after meeting the wait condition the function will wait until the OIA indicates the PS is ready to accept input. This parameter is optional and is defaulted to TRUE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// do the wait
BOOL waitOK = ps.WaitWhileAttrib(10, 16, 0xE0, 0xFF, FieldPlane, INFINITE, FALSE);

// do the processing for when the screen goes away
```

WaitForScreen

Synchronously waits for the screen described by the ECLScreenDesc parameter to appear in the Presentation Space.

Prototype

BOOL WaitForScreen(ECLScreenDesc* screenDesc, long TimeOut = INFINITE)

Parameters

ECLScreenDesc

screenDesc Object that describes the screen (see ECLScreenDesc Class on page 899).

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddCursorPos(23,1);
eclSD.AddString("LOGON");

// do the wait
int TimeOut = 5000;
BOOL waitOK = ps.WaitForScreen(eclSD, timeInt.intValue());

// do processing for the screen
```

WaitWhileScreen

Synchronously waits until the screen described by the ECLScreenDesc parameter is no longer in the Presentation Space.

Prototype

BOOL WaitWhileScreen(ECLScreenDesc* screenDesc, long TimeOut = INFINITE)

Parameters

ECLScreenDesc

screenDesc Object that describes the screen (see ECLScreenDesc Methods on page 900).

long nTimeOut

The maximum length of time in milliseconds to wait. This parameter is optional. The default is INFINITE.

Return Value

The method returns TRUE if the condition is met, or FALSE if nTimeOut (in milliseconds) has elapsed.

Note: This method will block if nTimeOut is default value (INFINITE) when the test condition would return FALSE.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');
```

```
// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddCursorPos(23,1);
eclSD.AddString("LOGON");

// do the wait
int TimeOut = 5000;
BOOL waitOK = ps.WaitWhileScreen(eclSD, timeInt.intValue());

// do processing for when the screen goes away
```

RegisterPSEvent

This member function registers an application object to receive notifications of PS update events. To use this function the application must create an object derived from either ECLPSNotify or ECLPSListener. A pointer to that object is then passed to this registration function. Any number of notify or listener objects may be registered at the same time. The order in which multiple listeners receive events is not defined and should not be assumed.

Different prototypes for this function allow different types of update events to be generated, and different levels of detail about the updates. The simplest update event is registered with an ECLPSNotify object. The type of registration produces an event for every PS update. No information about the update is generated. See the description of the ECLPSNotify object for more information.

For applications with need more information about the update, the ECLPSListener object can be registered. Registration of this object gives the application the ability to ignore some types of updates (for example, local terminal functions such as keystrokes) and to determine the region of the screen which was updated. See the description of the ECLPSListener object for more information. When registering an ECLPSListener object, the application can optionally specify the type of updates which are to cause events.

After an ECLPSNotify or ECLPSListener object is registered with this function, it's NotifyEvent() method will be called whenever a update to the presentation space occurs. Multiple updates to the PS in a short time period may be aggregated into a single event.

The application must unregister the notify/listener object before destroying it. The object will automatically be unregistered if the ECLPS object is destroyed.

Prototype

void RegisterPSEvent(ECLPSNotify * notify)
void RegisterPSEvent(ECLPSListener * listener)
void RegisterPSEvent(ECLPSListener * listener, int type)

Parameters

ECLPSNotify *

Pointer to the ECLPSNotify object to be registered.

ECLPSListener *

Pointer to the ECLPSListener object to be registered.

int

Type of updates which will cause events:

- USER_EVENTS (local terminal functions)
- HOST_EVENTS (host updates)
- ALL_EVENTS (all updates)

Return Value

None

StartMacro

The StartMacro method runs the Z and I Emulator for Windows macro file indicated by the MacroName parameter.

Prototype

void StartMacro(String MacroName)

Parameters

String MacroName

Name of macro file located in the Z and I Emulator for Windows user-class application data directory (specified at installation), without the file extension. This method does not support long file names.

Return Value

None

Usage Notes

You must use the short file name for the macro name. This method does not support long file names.

Example

The following example shows how to start a macro.

```
Dim PS as Object

Set PS = CreateObject("ZIEWin.autECLPS")
PS.StartMacro "mymacro"
```

UnregisterPSEvent

This member function unregisters an application object previously registered with the RegisterPSEvent function. An object registered to receive events should not be destroyed without first calling this function to unregister it. If the specific object is not currently registered, no action is taken and no error occurs.

When an ECLPSNotify or ECLPSListener object is unregistered its NotifyStop() method is called.

Prototype

void UnregisterPSEvent(ECLPSNotify * notify)
void UnregisterPSEvent(ECLPSListener * listener)
void UnregisterPSEvent(ECLPSListener * listener, int type)

Parameters

ECLPSNotify *

Pointer to the ECLPSNotify object to be unregistered.

ECLPSListener *

Pointer to the ECLPSListener object to be unregistered.

int

Type of updates which where registered:

- USER_EVENTS (local terminal functions)
- HOST_EVENTS (host updates)
- ALL_EVENTS (all updates)

Return Value

None

ECLPSEvent Class

ECLPSEvent objects are passed to ECLListener objects when the presentation space has been updated. This event object represents the presentation space update event and contains information about the update.

There are two sets of functions an application can use to determine the region of the presentation space which was updated. The GetStart() and GetEnd() methods return a linear position indicating the starting position and ending position of the update region in the presentation space. Linear addressing starts at 1 for the upper-left-most character and proceeds left-to-right wrapping from row to row. A corresponding set of functions (GetStartRow, GetStartCol, GetEndRow, GetEndCol) return the same information in row/column coordinates.

The update region includes all PS characters from the starting character to the ending character (inclusive). If the start and end position are not on the same row then the update region wraps from the end of one row to the first column of the next row. Note that the update region is (generally) not rectangular. If the starting position is greater than the ending position, the update region starts at the starting position, wraps from the last character of the screen to the first, and continues to the ending position.

Note that the update region may encompass more than the actual changed portion of the presentation space, but it is guaranteed to cover at least the changed area. When multiple PS updates occur in a short period of time the changes may be aggregated into a single event in which the update region spans the sum of all the updates.

Derivation

ECLBase > ECLEvent > ECLPSEvent

Usage Notes

Applications do not use this class directly. Applications create ECLListener-derived objects which receive ECLPSEvent objects on the ECLListener::NotifyEvent method.

ECLPSEvent Methods

The following section describes the methods that are valid for the ECLPSEvent class and all classes derived from it.

ECLPS * GetPS()

int GetType()

ULONG GetStart()

ULONG GetEnd()

ULONG GetStartRow()

ULONG GetStartCol()

ULONG GetEndRow()

ULONG GetEndCol()

GetPS

This method returns the ECLPS object which generated this event.

Prototype

ECLPS * GetPS()

Parameters

None

Return Value

ECLPS *

Pointer to ECLPS object which generated the event.

GetType

This method returns the type of presentation space update which generated this event. The return value is on of USER_EVENTS or HOST_EVENTS. User events are defined as any PS update which occurs as a local terminal function (for example, keystrokes entered by the user or by a programming API). Host events are PS updates which occur from host outbound datastreams.

Prototype

int GetType()

Parameters

None

Return Value

int

Returns USER_EVENTS or HOST_EVENTS constants.

GetStart

This method returns the linear location in the presentation space of the start of the update region. Note that the row/column coordinate of this location is dependant on the number of columns currently defined for the presentation space. If this value is greater than that returned by GetEnd(), then the update region starts at this location, wraps at the end of the screen to the beginning of the screen, and continues to the ending position.

Prototype

ULONG GetStart()

Parameters

None

Return Value

ULONG

Linear position of start of the update region.

GetEnd

This method returns the linear location in the presentation space of the end of the update region. Note that the row/column coordinate of this location is dependant on the number of columns currently defined for the presentation space. If this value is less than that returned by GetStart(), then the update region starts at the GetStart() location, wraps at the end of the screen to the beginning of the screen, and continues to this position.

Prototype

ULONG GetEnd()

Parameters

None

Return Value

ULONG

Linear position of end of the update region.

GetStartRow

This method returns the row number in the presentation space of the start of the update region. If the starting row/column position is greater than that of the ending row/column position, then the update region starts at this location, wraps at the end of the screen to the beginning of the screen, and continues to the ending position.

Prototype

ULONG GetStartRow()

Parameters

None

Return Value

ULONG

Row number of start of the update region.

GetStartCol

This method returns the column number in the presentation space of the start of the update region. If the starting row/column position is greater than that of the ending row/column position, then the update region starts at the starting row/column, wraps at the end of the screen to the beginning of the screen, and continues to the ending position.

Prototype

ULONG GetStartCol()

Parameters

None

Return Value

ULONG

Column number of start of the update region.

GetEndRow

This method returns the row number in the presentation space of the end of the update region. If the starting row/column position is greater than that of the ending row/column position, then the update region starts at the starting row/column, wraps at the end of the screen to the beginning of the screen, and continues to the ending row/column.

Prototype

ULONG GetEndRow()

Parameters

None

Return Value

ULONG

Row number of end of the update region.

GetEndCol

This method returns the column number in the presentation space of the end of the update region. If the starting row/column position is greater than that of the ending row/column position, then the update region starts at the starting row/column, wraps at the end of the screen to the beginning of the screen, and continues to the ending row/column.

Prototype

ULONG GetEndCol()

Parameters

None

Return Value

ULONG

Column number of end of the update region.

ECLPSListener Class

ECLPSListener is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLPSListener. The application must implement all the methods in this class.

The ECLPSListener class is used to allow an application to be notified of updates to the presentation space. Events are generated whenever the host screen is updated (any data in the presentation space is changed in any plane).

This class is similar to the ECLPSNotify class in that it is used to receive notifications of PS updates. It differs however in that it receives much more information about the cause and scope of the update than the ECLPSNotify class. In general using this class will be more expensive in terms of processing time and memory since more information has to be generated with each event. For applications which need to efficiently update a visual representation of the host screen this class may be more efficient than redrawing the representation each time an update occurs. Using this class the application can update only the portion of the visual representation that has changed.

This class also differs from ECLPSNotify in that all the methods are pure virtual and therefor must be implemented by the application (there is no default implementation of any methods).

Derivation

ECLBase > ECLListener > ECLPSListener

Usage Notes

To be notified of PS updates using this class, the application must perform the following steps:

1. Define a class derived from ECLPSListener.
2. Implement all methods of the ECLPSListener-derived class.
3. Create an instance of the derived class.
4. Register the instance with the ECLPS::RegisterPSEvent() method.

After registration is complete, updates to the presentation space will cause the NotifyEvent() method of the ECLPSListener-derived class to be called. The application can then used the ECLPSEvent object supplied on the method call to determine what caused the PS update and the region of the screen affected.

Note that multiple PS updates which occurred in a short period of time may be aggregated into a single event notification.

An application can choose to provide its own constructor and destructor for the derived class. This can be useful if the application needs to store some instance-specific data in the class and pass that information as a parameter on the constructor.

If an error is detected during event registration, the NotifyError() member function is called with an ECLErr object. Events may or may not continue to be generated after an error. When event generation terminates (due to an error or some other reason) the NotifyStop() member function is called.

ECLPSListener Methods

The following section describes the methods that are valid for the ECLPSListener class and all classes derived from it. Note that all methods except the constructor and destructor are pure virtual methods.

ECLPSListener()
ECLPSListener()
virtual void NotifyEvent(ECLPSEvent * event) = 0
virtual void NotifyError(ECLPS * PSObj, ECLErr ErrObj) = 0
virtual void NotifyStop(ECLPS * PSObj, int Reason) = 0

NotifyEvent

This method is a *pure virtual* member function (the application *must* implement this function in classes derived from ECLPSListener). This method is called whenever the presentation space is updated and this object is registered to receive update events. The ECLPSEvent object passed as a parameter contains information about the event including the region of the screen that was modified. See ECLPSEvent Class on page 885 for details.

Multiple PS updates may be aggregated into a single event causing only a single call to this method. The changed region contained in the ECLPSEvent object will encompass the sum of all the modifications.

Events may be restricted to only a particular type of PS update by supplying the appropriate parameters on the ECLPS::RegisterPSEvent() method. For example the application may choose to be notified only for updates from the host and not for local keystrokes.

Prototype

virtual void NotifyEvent(ECLPSEvent * event) = 0

Parameters

ECLPSEvent *

Pointer to an ECLPSEvent object which represents the PS update.

Return Value

None

NotifyError

This method is called whenever the ECLPS object detects an error during event generation. The error object contains information about the error (see ECLErr Class on page 789). Events may continue to be generated after the error depending on the nature of the error. If the event generation stops due to an error, the NotifyStop() method is called.

This is a *pure virtual* method which the application must implement.

Prototype

virtual void NotifyError(ECLPS * PSObj, ECLErr ErrObj) = 0

Parameters

ECLPS *

Pointer to the ECLPS object which generated this event.

ECLErr

An ECLErr object which describes the error.

Return Value

None

NotifyStop

This method is called when event generation is stopped for any reason (for example, due to an error condition or a call to ECLPS::UnregisterPSEvent).

This is a *pure virtual* method which the application must implement.

The reason code parameter is currently unused and will be zero.

Prototype

virtual void NotifyStop(ECLPS * PSObj, int Reason) = 0

Parameters

ECLPS *

Pointer to the ECLPS object which generated this event.

int

Reason event generation has stopped (currently unused and will be zero).

Return Value

None

ECLPSNotify Class

ECLPSNotify is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLPSNotify. The application must implement the NotifyEvent() member function in its derived class. It may also optionally implement NotifyError() and NotifyStop() member functions.

The ECLPSNotify class is used to allow an application to be notified of updates to the presentation space. Events are generated whenever the host screen is updated (any data in the presentation space is changed in any plane).

This class is similar to the ECLPSListener class in that it is used to receive notifications of PS updates. It differs however in that it receives no information about the cause and scope of the update than the ECLPSNotify class. In general using this class will be more efficient in terms of processing time and memory since no information has to be generated with each event. This class may be used for applications which only need notification of updates and do not need the details of what caused the event or what part of the screen was updated.

This class also differs from ECLPSListener in that default implementations are provided for the NotifyError() and NotifyStop() methods.

Derivation

ECLBase > ECLNotify > ECLPSNotify

Usage Notes

To be notified of PS updates using this class, the application must perform the following steps:

1. Define a class derived from ECLPSNotify.
2. Implement the NotifyEvent method of the ECLPSNotify-derived class.
3. Optionally implement other member functions of ECLPSNotify.
4. Create an instance of the derived class.
5. Register the instance with the ECLPS::RegisterPSEvent() method.

After registration is complete, updates to the presentation space will cause the NotifyEvent() method of the ECLPSNotify-derived class to be called.

Note that multiple PS updates which occur in a short period of time may be aggregated into a single event notification.

An application can choose to provide its own constructor and destructor for the derived class. This can be useful if the application needs to store some instance-specific data in the class and pass that information as a parameter on the constructor.

If an error is detected during event registration, the NotifyError() member function is called with an ECLErr object. Events may or may not continue to be generated after an error. When event generation terminates (due to an error or some other reason) the NotifyStop() member function is called. The default implementation of NotifyError() will present a message box to the user showing the text of the error messages retrieved from the ECLErr object.

When event notification stops for any reason (error or a call the ECLPS::UnregisterPSEvent) the NotifyStop() member function is called. The default implementation of NotifyStop() does nothing.

ECLPSNotify Methods

The following section describes the methods that are valid for the ECLPSNotify class and all classes derived from it.

ECLPSNotify()=0
~ECLPSNotify()
virtual void NotifyEvent(ECLPS * PSObj)
virtual void NotifyError(ECLPS * PSObj, ECLErr ErrObj)
virtual void NotifyStop(ECLPS * PSObj, int Reason)

NotifyEvent

This method is a *pure virtual* member function (the application **must** implement this function in classes derived from ECLPSNotify). This method is called whenever the presentation space is updated and this object is registered to receive update events.

Multiple PS updates may be aggregated into a single event causing only a single call to this method.

Prototype

virtual void NotifyEvent(ECLPS * PSObj)

Parameters

ECLPS *

Pointer to the ECLPS object which generated this event.

Return Value

None

NotifyError

This method is called whenever the ECLPS object detects an error during event generation. The error object contains information about the error (see ECLErr Class on page 789). Events may continue to be generated after the error depending on the nature of the error. If the event generation stops due to an error, the NotifyStop() method is called.

An application can choose to implement this function or allow the base ECLPSNotify class handle it. The default implementation will display the error in a message box using text supplied by the ECLErr::GetMsgText() method. If the application implements this function in its derived class it overrides this behavior.

Prototype

virtual void NotifyError(ECLPS * PSObj, ECLErr ErrObj) = 0

Parameters

ECLPS *

Pointer to the ECLPS object which generated this event.

ECLErr

An ECLErr object which describes the error.

Return Value

None

NotifyStop

This method is called when event generation is stopped for any reason (for example, due to an error condition or a call to ECLPS::UnregisterPSEvent).

The reason code parameter is currently unused and will be zero.

The default implementation of this function does nothing.

Prototype

virtual void NotifyStop(ECLPS * PSObj, int Reason) = 0

Parameters

ECLPS *

Pointer to the ECLPS object which generated this event.

int

Reason event generation has stopped (currently unused and will be zero).

Return Value

None

ECLRecoNotify Class

ECLRecoNotify can be used to implement an object which will receive and handle ECLScreenReco events. Events are generated whenever any screen in the PS is matched to an ECLScreenDesc object in ECLScreenReco. Special events are generated when event generation stops and when errors occur during event generation.

To be notified of ECLScreenReco events, the application must perform the following steps:

1. Define a class which derives from the ECLRecoNotify class.
2. Implement the NotifyEvent(), NotifyStop(), and NotifyError() methods.
3. Create an instance of the new class.
4. Register the instance with the ECLScreenReco::RegisterScreen() method.

See ECLScreenReco Class on page 908 for an example.

Derivation

ECLBase > ECLNotify > ECLRecoNotify

ECLRecoNotify Methods

Valid methods for ECLRecoNotify are listed below:

ECLRecoNotify()
~ECLRecoNotify()
void NotifyEvent(ECLPS *ps, ECLScreenDesc *sd)

void NotifyStop(ECLPS *ps, ECLScreenDesc *sd)
void NotifyError(ECLPS *ps, ECLScreenDesc *sd, ECLErr e)

ECLRecoNotify Constructor

Creates an empty instance of ECLRecoNotify.

Prototype

ECLRecoNotify()

Parameters

None

Return Value

None

Example

See ECLScreenReco Class on page 908 for an example.

ECLRecoNotify Destructor

Destroys the instance of ECLRecoNotify

Prototype

~ECLRecoNotify()

Parameters

None

Return Value

None

Example

See ECLScreenReco Class on page 908 for an example.

NotifyEvent

Called when the ECLScreenDesc registered with the ECLRecoNotify object on ECLScreenReco appears in the presentation space.

Prototype

void NotifyEvent(ECLPS *ps, ECLScreenDesc *sd)

Parameters

ECLPS ps

The ECLPS object that you registered.

ECLScreenDesc sd

ECLScreenDesc that you registered.

Return Value

None

Example

See ECLScreenReco Class on page 908 for an example.

NotifyStop

Called when the ECLScreenReco object stops monitoring its ECLPS objects for the registered ECLScreenDesc objects.

Prototype

void NotifyStop(ECLPS *ps, ECLScreenDesc *sd)

Parameters

ECLPS ps

The ECLPS object that you registered.

ECLScreenDesc sd

ECLScreenDesc that you registered.

Return Value

None

Example

See ECLScreenReco Class on page 908 for an example.

NotifyError

Called when the ECLScreenReco object encounters an error.

Prototype

void NotifyError(ECLPS *ps, ECLScreenDesc *sd, ECLErr e)

Parameters

ECLPS ps

The ECLPS object that you registered.

ECLScreenDesc sd

ECLScreenDesc that you registered.

ECLErr e

ECLErr object that contains the error information.

Return Value

None

Example

See ECLScreenReco Class on page 908 for an example.

ECLScreenDesc Class

ECLScreenDesc is the class that is used to *describe* a screen for the Host Access Class Library screen recognition technology. It uses all four major planes of the presentation space to describe it (TEXT,FIELD,EXFIELD, COLOR), as well as the cursor position.

Using the methods provided on this object, the programmer can set up a detailed description of what a given screen looks like in a host side application. Once an ECLScreenDesc object is created and set, it may be passed to either the synchronous WaitFor... methods provided on ECLPS, or it may be passed to ECLScreenReco, which fires an asynchronous event if the screen matching the ECLScreenDesc object appears in the PS.

Derivation

ECLBase > ECLScreenDesc

ECLScreenDesc Methods

Valid methods for ECLScreenDesc are listed below:

```
ECLScreenDesc()
~ECLScreenDesc()
void AddAttrib(BYTE attrib, UINT pos, PS_PLANE plane=FieldPlane);
void AddAttrib(BYTE attrib, UINT row, UINT col, PS_PLANE plane=FieldPlane);
void AddCursorPos(uint row, uint col)
void AddNumFields(uint num)
void AddNumInputFields(uint num)
void AddOIAInhibitStatus(OIAStatus type=NOTINHIBITED)
void AddString(LPCSTR s, UINT row, UINT col, BOOL caseSensitive=TRUE)
void AddStringInRect(char * str, int Top, int Left, int Bottom, int Right,
                    BOOL caseSense=TRUE)
void Clear()
```

ECLScreenDesc Constructor

Creates an empty instance of ECLScreenDesc.

Prototype

ECLScreenDesc()

Parameters

None

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddCursorPos(23,1);
eclSD.AddString("LOGON");
```

```
// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

ECLScreenDesc Destructor

Destroys the instance of ECLScreenDesc.

Prototype

~ ECLScreenDesc()

Parameters

None

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddCursorPos(23,1);
eclSD.AddString("LOGON");

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
// destroy the descriptor
delete eclSD;
```

AddAttrib

Adds an attribute value at the given position to the screen description.

Prototype

```
void AddAttrib(BYTE attrib, UINT pos, PS_PLANE plane=FieldPlane);
void AddAttrib(BYTE attrib, UINT row, UINT col, PS_PLANE plane=FieldPlane);
```

Parameters

BYTE attrib

Attribute value to add.

int row

Row position.

int col

Column position.

PS_PLANE plane

Plane in which attribute resides. Valid values are: TextPlane, ColorPlane, FieldPlane, Exfield Plane, GridPlane.**TextPlane**, **ColorPlane**, **FieldPlane**, and **ExfieldPlane**. See ECL Planes — Format and Content on page 1159 for the content and format of the different ECL planes.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddCursorPos

Sets the cursor position for the screen description to the given position.

Prototype

void AddCursorPos(uint row, uint col)

Parameters

uint row

Row position.

uint col

Column position.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddNumFields

Adds the number of input fields to the screen description.

Prototype

void AddNumFields(uint num)

Parameters

uint num

Number of fields.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddNumInputFields

Adds the number of input fields to the screen description.

Prototype

void AddNumInputFields(uint num)

Parameters

uint num

Number of input fields.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;
```

```
// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddOIAInhibitStatus

Sets the type of OIA monitoring for the screen description.

Prototype

void AddOIAInhibitStatus(OIAStatus type=NOTINHIBITED)

Parameters

OIAStatus type

Type of OIA status. Current valid values are DONTCARE and NOTINHIBITED.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddString

Adds a string at the given location to the screen description. If row and column are not provided, string may appear anywhere in the PS.

Note: Negative values are absolute positions from the bottom of the PS. For example, row=-2 is row 23 out of 24 rows.

Prototype

void AddString(LPCSTR s, UINT row, UINT col, BOOL caseSensitive=TRUE)

Parameters

LPCSTR s

String to add.

uint row

Row position.

uint col

Column position.

BOOL caseSense

If this value is TRUE, the strings are added as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

AddStringInRect

Adds a string in the given rectangle to the screen description.

Prototype

void AddStringInRect(char * str, int Top, int Left, int Bottom, int Right,
BOOL caseSense=TURE)

Parameters

char * str

String to add.

int Top

Upper left row position. This parameter is optional. The default is the first row.

int Left

Upper left column position. This parameter is optional. The default is the first column.

int Bottom

Lower right row position. This parameter is optional. The default is the last row.

int Right

Lower right column position. This parameter is optional. The default is the last column.

BOOL caseSense

If this value is TRUE, the strings are added as case-sensitive. This parameter is optional. The default is TRUE.

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
```

```
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());
```

Clear

Removes all description elements from the screen description.

Prototype

void Clear()

Parameters

None

Return Value

None

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45) ;
eclSD.AddNumInputFields(17) ;
AddOIAInhibitStatus(NOTINHIBITED) ;
eclSD.AddString("LOGON"., 23, 11, TRUE) ;
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE) ;

// do the wait
int TimeOut = 5000;
BOOL waitOK = eclPS.WaitForScreen(eclSD, timeInt.intValue());

// do processing for the screen

eclSD.Clear() // start over for a new screen
```

ECLScreenReco Class

The ECLScreenReco class is the engine for the Host Access Class Library screen recognition system. It contains the methods for adding and removing descriptions of screens. It also contains the logic for recognizing those screens and for asynchronously calling back to your handler code for those screens.

Think of an object of the ECLScreenReco class as a unique *recognition set*. The object can have multiple ECLPS objects that it watches for screens, multiple screens to look for, and multiple callback points to call when it sees a screen in any of the ECLPS objects.

All you need to do is set up your ECLScreenReco objects at the start of your application, and when any screen appears in any ECLPS that you want to monitor, your code will get called by ECLScreenReco. You do absolutely no legwork in monitoring screens!

Here's an example of a common implementation:

```
class MyApp {
ECLPS myECLPS('A'); // My main HACL PS object
ECLScreenReco myScreenReco(); // My screen reco object
ECLScreenDesc myScreenDesc(); // My screen descriptor
MyRecoCallback myCallback(); // My GUI handler

MyApp() {
// Save the number of fields for below
ECLFieldList *fl = myECLPS.GetFieldList()
Fl->Refresh();
int numFields = fl->GetFieldCount();

// Set up my HACL screen description object. Say the screen
// is identified by a cursor position, a key word, and the
// number of fields
myScreenDesc.AddCursorPos(23,1);
myScreenDesc.AddString("LOGON");
myScreenDesc.AddNumFields(numFields);
```

```

// Set up HACL screen reco object, it will begin monitoring here
myScreenReco.AddPS(myECLPS);
myScreenReco.RegisterScreen(&myScreenDesc, &myCallback);
}

 MyApp() {
myScreenReco.UnregisterScreen(&myScreenDesc, &myCallback);
myScreenReco.RemovePS(&eclPS);
}

public void showMainGUI() {
// Show the main application GUI, this is just a simple example
}


// ECLRecoNotify-derived inner class (the "callback" code)
class MyRecoCallback public: ECLRecoNotify {
public: void NotifyEvent(ECLScreenDesc *sd, ECLPS *ps) {
// GUI code here for the specific screen
// Maybe fire a dialog that front ends the screen
}

public void NotifyError(ECLScreenDesc *sd, ECLPS *ps, ECLErr e) {
// Error handling
}
```

```
public void NotifyStop(ECLScreenDesc *sd, ECLPS *ps, int Reason) {
// Possible stop monitoring, not essential
}
}

}

int main() {
MyApp app = new MyApp();
app.showMainGUI();
}
```

Derivation

ECLBase > ECLScreenReco

ECLScreenReco Methods

The following methods are valid for ECLScreenReco:

ECLScreenReco()

~ECLScreenReco()

AddPS(ECLPS*)

IsMatch(ECLPS*, ECLScreenDesc*)

RegisterScreen(ECLScreenDesc*, ECLRecoNotify*)

RemovePS(ECLPS*)

UnregisterScreen(ECLScreenDesc*)

ECLScreenReco Constructor

Creates an empty instance of ECLScreenReco

Prototype

ECLScreenReco()

Parameters

None

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

ECLScreenReco Destructor

Destroys the instance of ECLScreenReco

Prototype

~ECLScreenReco()

Parameters

None

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

AddPS

Adds Presentation Space object to monitor.

Prototype

AddPS(ECLPS*)

Parameters

ECLPS*

PS object to monitor.

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

IsMatch

Static member method that allows for passing an ECLPS object and an ECLScreenDesc object and determining if the screen description matches the PS. It is provided as a static method so any routine can call it without creating an ECLScreenReco object.

Prototype

IsMatch(ECLPS*, ECLScreenDesc*)

Parameters

ECLPS*

ECLPS object to compare.

ECLScreenDesc*

ECLScreenDesc object to compare.

Return Value

TRUE if the screen in PS matches, FALSE otherwise.

Example

```
// set up PS
ECLPS ps = new ECLPS('A');

// set up screen description
ECLScreenDesc eclSD = new ECLScreenDesc();
eclSD.AddAttrib(0xe8, 1, 1, ColorPlane);
eclSD.AddCursorPos(23,1);
eclSD.AddNumFields(45);
eclSD.AddNumInputFields(17);
AddOIAInhibitStatus(NOTINHIBITED);
eclSD.AddString("LOGON"., 23, 11, TRUE);
eclSD.AddStringInRect("PASSWORD", 23, 1, 24, 80, FALSE);
if(ECLScreenReco::IsMatch(ps,eclSD)) {
        // Handle Screen Match here . . .
}
```

RegisterScreen

Begins monitoring all ECLPS objects added to the screen recognition object for the given screen description. If the screen appears in the PS, the NotifyEvent method on the ECLRecoNotify object will be called.

Prototype

RegisterScreen(ECLScreenDesc*, ECLRecoNotify*)

Parameters

ECLScreenDesc*

Screen description object to register.

ECLRecoNotify*

Object that contains the callback code for the screen description.

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

RemovePS

Removes the ECLPS object from screen recognition monitoring.

Prototype

RemovePS(ECLPS*)

Parameters

ECLPS*

ECLPS object to remove.

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

UnregisterScreen

Removes the screen description and its callback code from screen recognition monitoring.

Prototype

UnregisterScreen(ECLScreenDesc*)

Parameters

ECLScreenDesc*

Screen description object to remove.

Return Value

None

Example

See the example of a common implementation provided in ECLScreenReco Class on page 908.

ECLSession Class

ECLSession provides general emulator connection-related services and contains pointers to instances of other objects in the Host Access Class Library.

Derivation

ECLBase > ECLConnection > ECLSession

Properties

None

Usage Notes

Because ECLSession is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

Although the objects ECLSession contains are capable of standing on their own, pointers to them exist in the ECLSession class. When an ECLSession object is created, ECLPS, ECLOIA, ECLXfer, and ECLWinMetrics objects are also created.

ECLSession Methods

The following section describes the methods that are valid for the ECLSession class:

```
ECLSession(char Name)
ECLSession(Long Handle)
~ECLSession()
ECLPS *GetPS()
ECLOIA *GetOIA()
ECLXfer *GetXfer()
ECLWinMetrics *GetWinMetrics()
void RegisterUpdateEvent(UPDATETYPE Type, ECLUpdateNotify *UpdateNotifyClass,
```

BOOL InitEvent)
void UnregisterUpdateEvent(ECLUpdateNotify *UpdateNotifyClass,)

ECLSession Constructor

This method creates an ECLSession object from a connection name (a single, alphabetic character from A-Z or a-z) or a connection handle. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can only be one connection "A" open at a time.

Prototype

ECLSession(char Name)

ECLSession(long Handle)

Parameters

char Name

One-character short name of the connection (A-Z or a-z).

long Handle

Handle of an ECL connection.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLSession::ECLSession      (Constructor)
//
// Build PS object from name.
//-------------------------------------------------------------------
void Sample73() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLPS      *PS;         // PS object pointer

try {
  Sess = new ECLSession('A');

  PS = Sess->GetPS();
  printf("Size of presentation space is %lu.\n", PS->GetSize());

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

ECLSession Destructor

This method destroys an ECLSession object.

Prototype

~ECLSession();

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLSession::~ECLSession      (Destructor)
//
// Build PS object from name and then delete it.
//-------------------------------------------------------------------
void Sample74() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLPS      *PS;         // PS object pointer

try {
  Sess = new ECLSession('A');

  PS = Sess->GetPS();
  printf("Size of presentation space is %lu.\n", PS->GetSize());

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

GetPS

This method returns a pointer to the ECLPS object contained in the ECLSession object. Use this method to access the ECLPS object methods. See ECLPS Class on page 840 for more information.

Prototype

ECLPS *GetPS()

Parameters

None

Return Value

ECLPS *

ECLPS object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetPS
//
// Get PS object from session object and use it.
//-------------------------------------------------------------------
void Sample69() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLPS      *PS;         // PS object pointer

try {
  Sess = new ECLSession('A');

  PS = Sess->GetPS();
  printf("Size of presentation space is %lu.\n", PS->GetSize());

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

GetOIA

This method returns a pointer to the ECLOIA object contained in the ECLSession object. Use this method to access the ECLOIA methods. See ECLOIA Class on page 823 for more information.

Prototype

ECLOIA *GetOIA()

Parameters

None

Return Value

ECLOIA *

ECLOIA object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetOIA
//
// Get OIA object from session object and use it.
//-------------------------------------------------------------------
void Sample70() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLOIA     *OIA;        // OIA object pointer

try {
  Sess = new ECLSession('A');

  OIA = Sess->GetOIA();
  if (OIA->InputInhibited() == NotInhibited)
    printf("Input is not inhibited.\n");
  else
    printf("Input is inhibited.\n");

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

GetXfer

This method returns a pointer to the ECLXfer object contained in the ECLSession object. Use this method to access the ECLXfer methods. See ECLXfer Class on page 950 for more information.

Prototype

ECLXfer *GetXfer()

Parameters

None

Return Value

ECLXfer *

ECLXfer object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetXfer
//
// Get OIA object from session object and use it.
//-------------------------------------------------------------------
void Sample71() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLXfer    *Xfer;       // Xfer object pointer

try {
  Sess = new ECLSession('A');

  Xfer = Sess->GetXfer();
  Xfer->SendFile("c:\\autoexec.bat", "AUTOEXEC BAT A", "(ASCII CRLF");

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

GetWinMetrics

This method returns a pointer to the ECLWinMetrics object contained in the ECLSession object. Use this method to access the ECLWinMetrics methods. See ECLWinMetrics Class on page 927 for more information.

Prototype

ECLWinMetrics *GetWinMetrics()

Parameters

None

Return Value

ECLWinMetrics *

ECLWinMetrics object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetWinMetrics
//
// Get WinMetrics object from session object and use it.
//-------------------------------------------------------------------
void Sample72() {

ECLSession *Sess;       // Pointer to Session object for connection A
ECLWinMetrics *Metrics; // WinMetrics object pointer

try {
  Sess = new ECLSession('A');

  Metrics = Sess->GetWinMetrics();
  printf("Window height is %lu pixels.\n", Metrics->GetHeight());

  delete Sess;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}
} // end sample
```

GetPageSettings

This method returns a pointer to the ECLPageSettings object contained in the ECLSession object. Use this method to access the ECLPageSettings methods. See ECLPageSettings Class on page 956 for more information.

Prototype

ECLPageSettings *GetPageSettings() const;

Parameters

None

Return Value

ECLPageSettings *

ECLPageSettings object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetPageSettings
//
// Get PageSettings object from session object and use it.
//-------------------------------------------------------------------
void Sample124() {
   ECLSession *Sess;       // Pointer to Session object for connection A
   ECLPageSettings *PgSet; // PageSettings object pointer

   try {
      Sess = new ECLSession('A');
      PgSet = Sess->GetPageSettings();
      printf("FaceName = %s\n", PgSet->GetFontFaceName());
      delete Sess;
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPrinterSettings

This method returns a pointer to the ECLPrinterSettings object contained in the ECLSession object. Use this method to access the ECLPrinterSettings methods. See ECLPageSettings Class on page 956 for more information.

Prototype

ECLPrinterSettings *GetPrinterSettings() const;

Parameters

None

Return Value

ECLPrinterSettings *

ECLPrinterSettings object pointer.

Example

```
//-------------------------------------------------------------------
// ECLSession::GetPrinterSettings
//
// Get PrinterSettings object from session object and use it.
//-------------------------------------------------------------------
void Sample125() {
   ECLSession *Sess;        // Pointer to Session object for connection A
   ECLPrinterSettings *PrSet; // PrinterSettings object pointer
   try {
      Sess = new ECLSession('A');
      PrSet = Sess->GetPrinterSettings();
      if (PrSet->IsPDTMode())
         printf("PDTMode\n");
      else
         printf("Not PDTMode\n");
      delete Sess;
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

RegisterUpdateEvent

Deprecated. See ECLPS::RegisterPSEvent in RegisterPSEvent on page 883.

UnregisterUpdateEvent

Deprecated. See ECLPS::UnregisterPSEvent in UnregisterPSEvent on page 884.

ECLStartNotify Class

ECLStartNotify is an abstract base class. An application cannot create an instance of this class directly. To use this class, the application must define its own class which is derived from ECLStartNotify. The application must implement the NotifyEvent() member function in its derived class. It may also optionally implement NotifyError() and NotifyStop() member functions.

The ECLStartNotify class is used to allow an application to be notified of the starting and stopping of ZIEWin connections. Start/stop events are generated whenever a ZIEWin connection (window) is started or stopped by any means, including the ECLConnMgr start/stop methods.

To be notified of start/stop events, the application must perform the following steps:

1. Define a class derived from ECLStartNotify.
2. Implement the derived class and implement the NotifyEvent() member function.
3. Optionally implement the NotifyError() and/or NotifyStop() functions.
4. Create an instance of the derived class.
5. Register the instance with the ECLConnMgr::RegisterStartEvent() function.

The example shown demonstrates how this may be done. When the above steps are complete, each time a connection is started or stopped the applications NotifyEvent() member function will be called. The function is passed two parameters giving the handle of the connection, and a BOOL start/stop indicator. The application may perform any functions required in the NotifyEvent() procedure, including calling other ECL functions. Note that the application cannot prevent the stopping of a connection; the notification is made after the session is already stopped.

If an error is detected during event generation, the NotifyError() member function is called with an ECLErr object. Events may or may not continue to be generated after an error, depending on the nature of the error. When event generation terminates (either due to an error, by calling the ECLConnMgr::UnregisterStartEvent, or by destruction of the ECLConnMgr object) the NotifyStop() member function is called. However event notification is terminated, the NotifyStop() member function is always called, and the application object is unregistered.

If the application does not provide an implementation of the NotifyError() member function, the default implementation is used (a simple message box is displayed to the user). The application can override the default behavior by implementing the NotifyError() function in the applications derived class. Likewise, the default NotifyStop() function is used if the application does not provide this function (the default behavior is to do nothing).

Note that the application can also choose to provide its own constructor and destructor for the derived class. This can be useful if the application wants to store some instance-specific data in the class and pass that information as a parameter on the constructor. For example, the application may want to post a message to an application window when a start/stop event occurs. Rather than define the window handle as a global variable (so it would be visible to the NotifyEvent() function), the application can define a constructor for the class which takes the window handle and stores it in the class member data area.

The application must not destroy the notification object while it is registered to receive events.

*Implementation Restriction:*Currently, the ECLConnMgr object allows only one notification object to be registered for a start/stop event notification. The ECLConnMgr::RegisterStartEvent will throw an error if a notify object is already registered for that ECLConnMgr object.

Derivation

ECLBase > ECLNotify > ECLStartNotify

Example

```
//-------------------------------------------------------------------
// ECLStartNotify class
//
// This sample demonstrates the use of:
//
// ECLStartNotify::NotifyEvent
// ECLStartNotify::NotifyError
// ECLStartNotify::NotifyStop
// ECLConnMgr::RegisterStartEvent
// ECLConnMgr::UnregisterStartEvent
//-------------------------------------------------------------------

//...................................................................
// Define a class derived from ECLStartNotify
//...................................................................
class MyStartNotify: public ECLStartNotify
{
public:
  // Define my own constructor to store instance data
  MyStartNotify(HANDLE DataHandle);
  // We have to implement this function
  void NotifyEvent(ECLConnMgr *CMObj, long ConnHandle,
                    BOOL Started);

  // We will take the default behaviour for these so we
  // don't implement them in our class:
  // void NotifyError (ECLConnMgr *CMObj, long ConnHandle, ECLErr ErrObject);
  // void NotifyStop (ECLConnMgr *CMObj, int Reason);
```

```
private:
  // We will store our application data handle here
  HANDLE MyDataH;
};

 //..................................................................
MyStartNotify::MyStartNotify(HANDLE DataHandle)   // Constructor
//..................................................................
{
  MyDataH = DataHandle;  // Save data handle for later use
}

//..................................................................
void MyStartNotify::NotifyEvent(ECLConnMgr *CMObj, long ConnHandle,
                    BOOL Started)
//..................................................................
{
  // This function is called whenever a connection start or stops.

  if (Started)
    printf("Connection %c started.\n", CMObj->ConvertHandle2ShortName(ConnHandle));
  else
    printf("Connection %c stopped.\n", CMObj->ConvertHandle2ShortName(ConnHandle));

  return;
}

 //..................................................................
// Create the class and begin start/stop monitoring.
//..................................................................
void Sample75() {

ECLConnMgr    CMgr;     // Connection manager object
MyStartNotify *Event;   // Ptr to my event handling object
HANDLE InstData;      // Handle to application data block (for example)

try {
  Event = new MyStartNotify(InstData);  // Create event handler

  CMgr.RegisterStartEvent(Event);       // Register to get events
```

```
  // At this point, any connection start/stops will cause the
  // MyStartEvent::NotifyEvent() function to execute.  For
  // this sample, we put this thread to sleep during this
  // time.

  printf("Monitoring connection start/stops for 60 seconds...\n");
  Sleep(60000);

  // Now stop event generation.
  CMgr.UnregisterStartEvent(Event);
  printf("Start/stop monitoring ended.\n");

  delete Event;  // Don't delete until after unregister!
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLStartNotify Methods

The following section describes the methods that are valid for the ECLStartNotify class.

ECLStartNotfiy()
ECLStartNotify()
virtual int NotifyEvent (ECLConnMgr *CMObj, long ConnHandle,
                    BOOL Started) = 0
virtual void NotifyError (ECLConnMgr *CMObj, long ConnHandle,
                    ECLErr ErrObject)
virtual void NotifyStop (ECLConnMgr *CMObj int Reason)

NotifyEvent

This method is a pure virtual member function (the application *must* implement this function in classes derived from ECLStartNotify). This function is called whenever a connection starts or stops and the object is registered for start/stop events. The Started BOOL is TRUE if the connection is started, or FALSE if is stopped.

Prototype

virtual int NotifyEvent (ECLConnMgr *CMObj, long ConnHandle,
                    BOOL Started) = 0

Parameters

ECLConnMgr *CMObj

This is the pointer to ECLConnMgr object in which the event occurred.

long ConnHandle

This is the handle of the connection that started or stopped.

BOOL Started

This is TRUE if the connection is started, or FALSE if the connection is stopped.

Return Value

None

NotifyError

This method is called whenever the ECLConnMgr object detects an error event generation. The error object contains information about the error (see the ECLErr class description). Events may continue to be generated after the error, depending on the nature of the error. If event generation stops due to an error, the NotifyStop() function is called.

The ConnHandle contains the handle of the connection that is related to the error. This value may be zero if the error is not related to any specific connection.

An application can choose to implement this function or allow the ECLStartNotify base class to handle the error. The base class will display the error in a message box using the text supplied by the ECLErr::GetMsgText() function. If the application implements this function in its derived class it will override the base class function.

Prototype

```
virtual void NotifyError  (ECLConnMgr  *CMObj, long ConnHandle,
          ECLErr ErrObject)
```

Parameters

ECLConnMgr *CMObj

This is the ptr to ECLConnMgr object in which the error occurred.

long ConnHandle

This is the handle of the connection related to the error or zero.

ECLErr ErrObject

This is theECLErr object describing the error.

Return Value

None

NotifyStop

This method is called when event generation is stopped for any reason (for example, due to an error condition or a call to ECLConnMgr::UnregisterStartEvent).

Prototype

virtual void NotifyStop (ECLConnMgr *CMObj int Reason)

Parameters

ECLConnMgr *CMObj

This is the ptr to ECLConnMgr object that is stopping notification.

int Reason

This is the unused zero.

Return Value

None

ECLUpdateNotify Class

Deprecated. See the class descriptions in ECLPSListener Class on page 890 and ECLOIA Class on page 823.

ECLWinMetrics Class

The ECLWinMetrics class performs operations on a Z and I Emulator for Windows connection window. It allows you to perform window rectangle and position manipulation (for example, SetWindowRect, GetXpos or SetWidth), as well as window state manipulation (for example, SetVisible or IsRestored).

Derivation

ECLBase > ECLConnection > ECLWinMetrics

Properties

None

Usage Notes

Because ECLWinMetrics is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

The ECLWinMetrics object is created for the connection identified upon construction. You may create an ECLWinMetrics object by passing either the connection ID (a single, alphabetical character from A-Z or a-z) or the connection handle, which is usually obtained from the ECLConnection object. There can be only one Z and I Emulator for Windows connection with a given name or handle open at a time.

Note: There is a pointer to the ECLWinMetrics object in the ECLSession class. If you just want to manipulate the connection window, create ECLWinMetrics on its own. If you want to do more, you may want to create an ECLSession object.

ECLWinMetrics Methods

The following methods apply to the ECLWinMetrics class.

ECLWinMetrics(char Name)

ECLWinMetrics(long Handle)

~ECLWinMetrics()

const char *GetWindowTitle()

void SetWindowTitle(char *NewTitle)

long GetXpos()

void SetXpos(long NewXpos)

long GetYpos()

void SetYpos(long NewYpos)

long GetWidth()

void SetWidth(long NewWidth)

long GetHeight()

void SetHeight(long NewHeight)

void GetWindowRect(Long *left, Long *top, Long *right, Long *bottom)

void SetWindowRect(Long left, Long top, Long right, Long bottom)

BOOL IsVisible()

void SetVisible(BOOL SetFlag)

BOOL Active()

void SetActive(BOOL SetFlag)

BOOL IsMinimized()

void SetMinimized()

BOOL IsMaximized()

void SetMaximized()

BOOL IsRestored()

void SetRestored()

ECLWinMetrics Constructor

This method creates an ECLWinMetrics object from a connection name or connection handle. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time.

Prototype

ECLWinMetrics(char Name)

ECLWinMetrics(long Handle)

Parameters

char Name

One-character short name of the connection (A-Z or a-z).

long Handle

Handle of an ECL connection.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::ECLWinMetrics   (Constructor)
//
// Build WinMetrics object from name.
//-------------------------------------------------------------------
void Sample77() {

ECLWinMetrics *Metrics;    // Ptr to object

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  printf("Window of connection A is %lu pixels wide.\n",
    Metrics->GetWidth());

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLWinMetrics Destructor

This method destroys a ECLWinMetrics object.

Prototype

~ECLWinMetrics()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::ECLWinMetrics   (Destructor)
//
// Build WinMetrics object from name.
//-------------------------------------------------------------------
void Sample78() {

ECLWinMetrics *Metrics;    // Ptr to object

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  printf("Window of connection A is %lu pixels wide.\n",
    Metrics->GetWidth());

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetWindowTitle

The GetWindowTitle method returns a pointer to a null terminate string containing the title that is currently in the title bar for the connection associated with the ECLWinMetrics object. Do not assume that the string returned is persistent over time. You must either make a copy of the string or make a call to this method each time you need it.

Prototype

const char *GetWindowTitle()

Parameters

None

Return Value

Pointer to null terminated string that contains the title.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::GetWindowTitle
//
// Display current window title of connection A.
//-------------------------------------------------------------------
void Sample79() {

ECLWinMetrics *Metrics;    // Ptr to object

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  printf("Title of connection A is: %s\n",
    Metrics->GetWindowTitle());

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetWindowTitle

The SetWindowTitle method changes the title currently in the title bar for the connection associated with the ECLWinMetrics object to the title passed in the input parameter. A null string can be used to reset the title to the default title.

Prototype

void SetWindowTitle(char *NewTitle)

Parameters

char *NewTitle

Null-terminated title string.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetWindowTitle
//
// Change current window title of connection A.
//-------------------------------------------------------------------
void Sample80() {

ECLWinMetrics *Metrics;    // Ptr to object

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  // Get current title
  printf("Title of connection A is: %s\n", Metrics->GetWindowTitle());

  // Set new title
  Metrics->SetWindowTitle("New Title");
  printf("New title is: %s\n", Metrics->GetWindowTitle());

  // Reset back to original title
  Metrics->SetWindowTitle("");
  printf("Returned title to: %s\n", Metrics->GetWindowTitle());

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

Usage Notes

If NewTitle is a null string, SetWindowTitle will restore the window title to its original setting.

GetXpos

The GetXpos method returns the *x* position of the upper left point of the connection window rectangle.

Prototype

long GetXpos()

Parameters

None

Return Value

long

x position of connection window.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::GetXpos
//
// Move window 10 pixels.
//-------------------------------------------------------------------
void Sample81() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot move minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetXpos();
    Y = Metrics->GetYpos();
    Metrics->SetXpos(X+10);
    Metrics->SetYpos(Y+10);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetXpos

The SetXpos method sets the *x* position of the upper left point of the connection window rectangle.

Prototype

void SetXpos(long NewXpos)

Parameters

long NewXpos

The new *x* coordinate of the window rectangle.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetXpos
//
// Move window 10 pixels.
//-------------------------------------------------------------------
void Sample83() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot move minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetXpos();
    Y = Metrics->GetYpos();
    Metrics->SetXpos(X+10);
    Metrics->SetYpos(Y+10);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetYpos

The GetYpos method returns the *y* position of the upper left point of the connection window rectangle.

Prototype

long GetYpos()

Parameters

None

Return Value

long

y position of the connection window.

Example

```
a//-------------------------------------------------------------------
// ECLWinMetrics::GetYpos
//
// Move window 10 pixels.
//-------------------------------------------------------------------
void Sample82() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot move minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetXpos();
    Y = Metrics->GetYpos();
    Metrics->SetXpos(X+10);
    Metrics->SetYpos(Y+10);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetYpos

The SetYpos method sets the *y* position of the upper left point of the connection window rectangle.

Prototype

void SetYpos(long NewYpos)

Parameters

long NewYpos

New *y* coordinate of the window rectangle.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetYpos
//
// Move window 10 pixels.
//-------------------------------------------------------------------
void Sample84() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot move minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetXpos();
    Y = Metrics->GetYpos();
    Metrics->SetXpos(X+10);
    Metrics->SetYpos(Y+10);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetWidth

This method returns the width of the connection window rectangle.

Prototype

long GetWidth()

Parameters

None

Return Value

long

Width of the connection window.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::GetWidth
//
// Make window 1/2 its current size.  Depending on display settings
// (Appearance->Display Setup menu) it may snap to a font that is
// not exactly the 1/2 size we specify.
//-------------------------------------------------------------------
void Sample85() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot size minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetWidth();
    Y = Metrics->GetHeight();
    Metrics->SetWidth(X/2);
    Metrics->SetHeight(Y/2);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetWidth

The SetWidth method sets the width of the connection window rectangle.

Prototype

void SetWidth(long NewWidth)

Parameters

long NewWidth

New width of the window rectangle.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetWidth
//
// Make window 1/2 its current size.  Depending on display settings
// (Appearance->Display Setup menu) it may snap to a font that is
// not exactly the 1/2 size we specify.
//-------------------------------------------------------------------
void Sample87() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot size minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetWidth();
    Y = Metrics->GetHeight();
    Metrics->SetWidth(X/2);
    Metrics->SetHeight(Y/2);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

GetHeight

The GetHeight method returns the height of the connection window rectangle.

Prototype

long GetHeight()

Parameters

None

Return Value

long

Height of the connection window.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::GetHeight
//
// Make window 1/2 its current size.  Depending on display settings
// (Appearance->Display Setup menu) it may snap to a font that is
// not exactly the 1/2 size we specify.
//-------------------------------------------------------------------
void Sample86() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot size minimized or maximized window.\n");
  }
  else {
    X = Metrics->GetWidth();
    Y = Metrics->GetHeight();
    Metrics->SetWidth(X/2);
    Metrics->SetHeight(Y/2);
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetHeight

This method sets the height of the connection window rectangle.

Prototype

void SetHeight(Long NewHeight)

Parameters

long NewHeight

New height of the window rectangle.

Return Value

None

Example

The following example shows how to use the SetHeight method to set the height of the connection window rectangle.

```
ECLWinMetrics   *pWM;
ECLConnList  ConnList();

// Create using connection handle of first connection in the list of
// active connections
try {
  if ( ConnList.Count() != 0 ) {
    pWM = new ECLWinMetrics(ConnList.GetFirstSession()->GetHandle());

    // Set the height
    pWM->SetHeight(6081);
  }
}
catch (ECLErr ErrObj) {
  // Just report the error text in a message box
  MessageBox( NULL, ErrObj.GetMsgText(), "Error!", MB_OK );
}
```

GetWindowRect

This method returns the bounding points of the connection window rectangle.

Prototype

void GetWindowRect(Long *left, Long *top, Long *right, Long *bottom)

Parameters

long *left

This output parameter is set to the left coordinate of the window rectangle.

long *top

This output parameter is set to the top coordinate of the window rectangle.

long *right

This output parameter is set to the right coordinate of the window rectangle.

long *bottom

This output parameter is set to the bottom coordinate of the window rectangle.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::GetWindowRect
//
// Make window 1/2 its current size.  Depending on display settings
// (Appearance->Display Setup menu) it may snap to a font that is
// not exactly the 1/2 size we specify.  Also move the window.
//-------------------------------------------------------------------
void Sample88() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y, Width, Height;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot size/move minimized or maximized window.\n");
  }
  else {
    Metrics->GetWindowRect(&X, &Y, &Width, &Height);
    Metrics->SetWindowRect(X+10, Y+10,         // Move window
                           Width/2, Height/2); // Size window
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SetWindowRect

This method sets the bounding points of the connection window rectangle.

Prototype

void SetWindowRect(long left, long top, long right, long bottom)

Parameters

long left

The left coordinate of the window rectangle.

long top

The top coordinate of the window rectangle.

long right

The right coordinate of the window rectangle.

long bottom

The bottom coordinate of the window rectangle.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetWindowRect
//
// Make window 1/2 its current size.  Depending on display settings
// (Appearance->Display Setup menu) it may snap to a font that is
// not exactly the 1/2 size we specify.  Also move the window.
//-------------------------------------------------------------------
void Sample89() {

ECLWinMetrics *Metrics;    // Ptr to object
long X, Y, Width, Height;

try {
  Metrics = new ECLWinMetrics('A');  // Create for connection A

  if (Metrics->IsMinimized() || Metrics->IsMaximized()) {
    printf("Cannot size/move minimized or maximized window.\n");
  }
  else {
    Metrics->GetWindowRect(&X, &Y, &Width, &Height);
    Metrics->SetWindowRect(X+10, Y+10,         // Move window
                           Width/2, Height/2); // Size window
  }

  delete Metrics;
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

IsVisible

This method returns the visibility state of the connection window.

Prototype

BOOL IsVisible()

Parameters

None

Return Value

Visibility state. TRUE value if the window is visible, FALSE value if the window is not visible.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::IsVisible
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample90() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsVisible(); // Get state
Metrics.SetVisible(!CurrState);  // Set state

} // end sample
```

SetVisible

This method sets the visibility state of the connection window.

Prototype

void SetVisible(BOOL SetFlag)

Parameters

BOOL SetFlag

TRUE for visible, FALSE for invisible.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetVisible
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample91() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsVisible(); // Get state
Metrics.SetVisible(!CurrState);  // Set state

} // end sample

//-------------------------------------------------------------------
```

IsActive

This method returns the focus state of the connection window.

Prototype

BOOL Active()

Parameters

None

Return Value

BOOL

Focus state. TRUE if active, FALSE if not active.

Example

```

// ECLWinMetrics::IsActive
```

```
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample92() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsActive();  // Get state
Metrics.SetActive(!CurrState);   // Set state

} // end sample
```

SetActive

This method sets the focus state of the connection window.

Prototype

void SetActive(BOOL SetFlag)

Parameters

Bool SetFlag

New state. TRUE for active, FALSE for inactive.

Return Value

None

Example

The following is an example of the SetActive method.

```
ECLWinMetrics *pWM;
ECLConnList  ConnList();

// Create using connection handle of first connection in the list of
// active connections
try {
  if ( ConnList.Count() != 0 ) {
    pWM = new ECLWinMetrics(ConnList.GetFirstSession()->GetHandle());

    // Set to inactive if active
    if ( pWM->Active() )
   pWM->SetActive(FALSE);
  }
}
catch (ECLErr ErrObj) {
  // Just report the error text in a message box
  MessageBox( NULL, ErrObj.GetMsgText(), "Error!", MB_OK );
}
```

IsMinimized

This method returns the minimize state of the connection window.

Prototype

BOOL IsMinimized()

Parameters

None

Return Value

BOOL

Minimize state. TRUE value returned if the window is minimized; FALSE value returned if the window is not minimized.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::IsMinimized
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample93() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsMinimized();  // Get state
if (!CurrState)
  Metrics.SetMinimized();           // Set state
else
  Metrics.SetRestored();

} // end sample
```

SetMinimized

This method sets the connection window to minimized

Prototype

void SetMinimized()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetMinimized
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample94() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsMinimized();  // Get state
if (!CurrState)
  Metrics.SetMinimized();           // Set state
else
  Metrics.SetRestored();

} // end sample
```

IsMaximized

This method returns the maximize state of the connection window.

Prototype

BOOL IsMaximized()

Parameters

None

Return Value

BOOL

Maximize state. TRUE value if the window is maximized; FALSE value if the window is not maximized.

Example

```
// ECLWinMetrics::IsMaximized
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample97() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsMaximized();  // Get state
if (!CurrState)
  Metrics.SetMaximized();           // Set state
else
  Metrics.SetMinimized();

} // end sample
```

SetMaximized

This method sets the connection window to maximized.

Prototype

void SetMaximized()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetMaximized
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample98() {
```

```
ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsMaximized();  // Get state
if (!CurrState)
  Metrics.SetMaximized();           // Set state
else
  Metrics.SetMinimized();

} // end sample
```

IsRestored

This method returns the restore state of the connection window.

Prototype

BOOL IsRestored()

Parameters

None

Return Value

BOOL

Restore state. TRUE value if the window is restored; FALSE value if the window is not restored.

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::IsRestored
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample95() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsRestored();  // Get state
if (!CurrState)
  Metrics.SetRestored();           // Set state
else
  Metrics.SetMinimized();

} // end sample
```

SetRestored

The SetRestored method sets the connection window to restored.

Prototype

void SetRestored()

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLWinMetrics::SetRestored
//
// Get current state of window, and then toggle it.
//-------------------------------------------------------------------
void Sample96() {

ECLWinMetrics Metrics('A');      // Window metrics class
BOOL  CurrState;

CurrState = Metrics.IsRestored();  // Get state
if (!CurrState)
  Metrics.SetRestored();           // Set state
else
  Metrics.SetMinimized();

} // end sample

//-------------------------------------------------------------------
```

ECLXfer Class

ECLXfer provides file transfer services.

Derivation

ECLBase > ECLConnection > ECLXfer

Properties

None

Usage Notes

Because ECLXfer is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

The ECLXfer object is created for the connection identified upon construction. You may create an ECLXfer object by passing either the connection ID (a single, alphabetic character from A-Z or a-z) or the connection handle, which is usually obtained from the ECLConnList object. There can be only one Z and I Emulator for Windows connection with a given name or handle open at a time.

Note: There is a pointer to the ECLXfer object in the ECLSession class. If you only want to manipulate the connection window, create an ECLXfer object on its own. If you want to do more, you may want to create an ECLSession object.

ECLXfer Methods

The following section describes the methods that are valid for the ECLXfer class:

ECLXfer(char Name)
ECLXfer(long Handle)
~ECLXfer()
int SendFile(char *PCFile, char *HostFile, char *Options)
int ReceiveFile(char *PCFile, char *HostFile, char *Options)

ECLXfer Constructor

This method creates an ECLXfer object from a connection ID (a single, alphabetic character from A-Z or a-z) or a connection handle. There can be only one Z and I Emulator for Windows connection open with a given ID. For example, there can be only one connection “A” open at a time.

Prototype

ECLXfer(char Name)

ECLXfer(long Handle)

Parameters

char Name

One-character short name of the connection (A-Z or a-z).

long Handle

Handle of an ECL connection.

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLXfer::ECLXfer          (Constructor)
//
// Build ECLXfer object from a connection name.
//-------------------------------------------------------------------
void Sample99() {

ECLXfer  *Xfer;             // Pointer to Xfer object

try {
  Xfer = new ECLXfer('A');  // Create object for connection A
  printf("Created ECLXfer for connection %c.\n", Xfer->GetName());

  delete Xfer;              // Delete Xfer object
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

ECLXfer Destructor

This method destroys an ECLXfer object.

Prototype

~ECLXfer();

Parameters

None

Return Value

None

Example

```
//-------------------------------------------------------------------
// ECLXfer::~ECLXfer          (Destructor)
//
// Build ECLXfer object from a connection name.
//-------------------------------------------------------------------
void Sample100() {

ECLXfer  *Xfer;             // Pointer to Xfer object

try {
  Xfer = new ECLXfer('A');  // Create object for connection A
  printf("Created ECLXfer for connection %c.\n", Xfer->GetName());

  delete Xfer;              // Delete Xfer object
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

SendFile

This method sends a file from the workstation to the host.

Prototype

int SendFile(char *PCFile, char *HostFile, char *Options)

Parameters

char *PCFile

Pointer to a string containing the workstation file name to be sent to the host.

char *HostFile

Pointer to a string containing the host file name to be created or updated on the host.

char *Options

Pointer to a string containing the options to be used during the transfer.

Return Value

int

EHLLAPI return code as documented in *Emulator Programming* for the SendFile EHLLAPI function.

Example

```
//-------------------------------------------------------------------
// ECLXfer::SendFile
//
// Send a file to a VM/CMS host with ASCII translation.
//-------------------------------------------------------------------
void Sample101() {

ECLXfer  *Xfer;             // Pointer to Xfer object
int Rc;

try {
  Xfer = new ECLXfer('A');  // Create object for connection A

  printf("Sending file...\n");
  Rc = Xfer->SendFile("c:\\autoexec.bat", "autoexec bat a", "(ASCII CRLF QUIET");
  switch (Rc) {
  case 2:
    printf("File transfer failed, error in parameters.\n", Rc);
    break;
  case 3:
    printf("File transfer sucessfull.\n");
    break;
  case 4:
    printf("File transfer sucessfull, some records were segmented.\n");
    break;
  case 5:
    printf("File transfer failed, workstation file not found.\n");
    break;
  case 27:
    printf("File transfer cancelled or timed out.\n");
    break;
  default:
    printf("File transfer failed, code %u.\n", Rc);
    break;
  } // case

  delete Xfer;              // Delete Xfer object
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

Usage Notes

File transfer options are host-dependent. The following is a list of some of the valid host options for a VM/CMS host:

ASCII
CRLF
APPEND
LRECL
RECFM
CLEAR/NOCLEAR
PROGRESS
QUIET

Refer to *Emulator Programming* for the list of supported hosts and associated file transfer options.

ReceiveFile

This method receives a file from the host and sends the file to the workstation.

Prototype

int ReceiveFile(char *PCFile, char *HostFile, char *Options)

Parameters

char *PCFile

Pointer to a string containing the workstation file name to be sent to the host.

char *HostFile

Pointer to a string containing the host file name to be created or updated on the host.

char *Options

Pointer to a string containing the options to be used during the transfer.

Return Value

int

EHLLAPI return code as documented in *Emulator Programming* for the ReceiveFile EHLLAPI function.

Example

```
//-------------------------------------------------------------------
// ECLXfer::ReceiveFile
//
// Receive file from a VM/CMS host with ASCII translation.
```

```
//-------------------------------------------------------------------
void Sample102() {

ECLXfer  *Xfer;             // Pointer to Xfer object
int Rc;

try {
  Xfer = new ECLXfer('A');  // Create object for connection A

  printf("Receiving file...\n");
  Rc = Xfer->ReceiveFile("c:\\temp.txt", "temp text a", "(ASCII CRLF QUIET");
  switch (Rc) {
  case 2:
    printf("File transfer failed, error in parameters.\n", Rc);
    break;
  case 3:
    printf("File transfer sucessfull.\n");
    break;
  case 4:
    printf("File transfer sucessfull, some records were segmented.\n");
    break;
  case 27:
    printf("File transfer cancelled or timed out.\n");
    break;
  default:
    printf("File transfer failed, code %u.\n", Rc);
    break;
  } // case

  delete Xfer;              // Delete Xfer object
}
catch (ECLErr Err) {
  printf("ECL Error: %s\n", Err.GetMsgText());
}

} // end sample
```

Usage Notes

File transfer options are host-dependent. The following is a list of some of the valid host options for a VM/CMS host:

ASCII
CRLF
APPEND
LRECL
RECFM
CLEAR/NOCLEAR
PROGRESS
QUIET

Refer to *Emulator Programming* for the list of supported hosts and associated file transfer options.

ECLPageSettings Class

The ECLPageSettings class performs operations on the session page settings. It enables you to retrieve and configure the **File → Page Setup** dialog settings, such as CPI, LPI, and Face Name. Only the settings in the **Text** tab of the dialog are supported.

Derivation

ECLBase > ECLConnection > ECLPageSettings

Properties

None

Restrictions

The connection associated with each method must be in a particular state for the method to succeed. If the restrictions are not met, an appropriate exception is raised.

The following restrictions apply when any method of the ECLPageSettings class is invoked. If the restrictions are not met, an exception is thrown.

- The connection **Page Setup** and **Printer Setup** dialogs must not be in use.
- The connection must not be printing.
- The associated connection must not be in PDT mode.

Additional restrictions might apply for each specific method.

Usage Notes

Because ECLPageSettings is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

The ECLPageSettings object is created for the connection identified upon construction. You can create an ECLPageSettings object by passing the connection ID (a single alphabetical character from **A** to **Z**) or the connection handle (usually obtained from the ECLConnection object). There can be only one Z and I Emulator for Windows connection with a given name or handle open at one time.

The ECLSession class creates an instance of this object. If the application does not need other services provided by ECLSession, you can create this object independently. Otherwise, consider creating an ECLSession object and use the objects created by ECLSession. See ECLSession Class on page 914 for more information.

Each method supports only certain connection types of the connection associated with the ECLPageSettings object. The supported connection types are provided in each method section. If a method is called on an unsupported connection, an exception is thrown. Use the method GetConnType to determine the connection type.

CPI, LPI and FontSize are dependent on the property FaceName. Therefore, if CPI, LPI, and FontSize are set before the FaceName is set, and if the values are not valid for the FaceName property, then different CPI, LPI, or FontSize values might be reconfigured in the connection. You should set the FaceName value before setting the CPI, LPI, or FontSize. Or you can query CPI, LPI, and FontSize each time you set FaceName to ensure that they use the desired values.

ECLPageSettings Methods

The following sections describe the methods that are valid for the ECLPageSettings class.

```
ECLPageSettings(char Name)
ECLPageSettings(long Handle)
~ECLPageSettings()
void SetCPI(ULONG CPI=FONT_CPI)
ULONG GetCPI() const
BOOL  IsFontCPI()
void SetLPI(ULONG LPI=FONT_LPI)
ULONG GetLPI() const
BOOL  IsFontLPI()
void SetFontFaceName(const char *const FaceName)
const char *GetFontFaceName() const
void SetFontSize(ULONG FontSize)
ULONG GetFontSize()
void SetMaxLinesPerPage(ULONG MPL)
ULONG GetMaxLinesPerPage() const
void SetMaxCharsPerLine(ULONG MPP)
ULONG GetMaxCharsPerLine() const
void RestoreDefaults(ULONG Tabs=PAGE_TEXT) const
```

Connection types

The valid connection types for the ECLPageSettings methods are as follows:

Connection Type	String Value
3270 display	HOSTTYPE_3270DISPLAY
5250 display	HOSTTYPE_5250DISPLAY
3270 printer	HOSTTYPE_3270PRINTER
VT (ASCII) emulation	HOSTTYPE_VT

ECLPageSettings Constructor

This method uses a connection name or handle to create an ECLPageSettings object.

Prototype

ECLPageSettings(char Name)

ECLPageSettings(long Handle)

Parameters

char Name

One-character short name of the connection. Valid values are A–Z.

long Handle

Handle of an ECL connection.

Return Value

None

Example

The following example shows how to create an ECLPageSettings object using the connection name and the connection handle.

```
void Sample108() {

   ECLPageSettings *PgSet1, *PgSet2; // Pointer to ECLPageSettings objects
   ECLConnList ConnList; // Connection list object

   try {
      // Create ECLPageSettings object for connection 'A'
      PgSet1 = new ECLPageSettings('A');
      // Create ECLPageSettings object for first connection in conn list
      ECLConnection *Connection = ConnList.GetFirstConnection();
      if (Connection != NULL) {
         PgSet2 = new ECLPageSettings(Connection->GetHandle());
         printf("PgSet#1 is for connection %c, PgSet #2 is for connection %c.\n",
                PgSet1->GetName(), PgSet2->GetName());
         delete PgSet1;
         delete PgSet2;
      }
      else
         printf("No connections to create PageSettings object.\n");
   } catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetCPI

This method sets the CPI (characters per inch) value in the connection. If this method is called without any arguments, it sets the Font CPI in the connection.

Prototype

void SetCPI(ULONG CPI=FONT_CPI);

Parameters

ULONG CPI

Characters per inch. This parameter is optional. The default value is FONT_CPI.

Return Value

None

Example

```
void Sample109() {

   ECLPageSettings PgSet('A');

   PgSet.SetCPI(10);
   ULONG cpi = PgSet.GetCPI();
   printf("CPI = %ld\n", cpi);
   if (PgSet.IsFontCPI())
      printf("FontCPI\n");
   else
      printf("Not FontCPI\n");
} // end sample
```

GetCPI

This method returns the CPI (characters per inch) value of the connection. Even if **Font CPI** is selected in the associated connection, this method returns the value of the CPI selected for the font in the associated connection.

If **Font CPI** is configured in the connection, this method does not return the constant FONT_CPI . Use the IsFontCPI method to determine whether **Font CPI** is set in the connection.

Prototype

ULONG GetCPI() const;

Parameters

None

Return Value

ULONG CPI

Characters per inch.

Example

```
void Sample109() {

   ECLPageSettings PgSet('A');

   PgSet.SetCPI(10);
   ULONG cpi = PgSet.GetCPI();
   printf("CPI = %ld\n", cpi);
   if (PgSet.IsFontCPI())
      printf("FontCPI\n");
   else
      printf("Not FontCPI\n");
} // end sample
```

IsFontCPI

This method returns an indication of whether **Font CPI** is set in the connection.

Prototype

BOOL IsFontCPI();

Parameters

None

Return Value

BOOL

Possible values are as follows:

- TRUE if **Font CPI** is set in the connection.
- FALSE if **Font CPI** is not set in the connection.

Example

```
void Sample109() {

   ECLPageSettings PgSet('A');

   PgSet.SetCPI(10);
   ULONG cpi = PgSet.GetCPI();
   printf("CPI = %ld\n", cpi);
   if (PgSet.IsFontCPI())
      printf("FontCPI\n");
   else
      printf("Not FontCPI\n");
} // end sample
```

SetLPI

This method sets the LPI (lines per inch) value in the connection. If this method is called without any arguments, it sets the Font LPI in the connection.

Prototype

void SetLPI(ULONG LPI=FONT_LPI);

Parameters

ULONG LPI

Lines per inch. This parameter is optional. The default value is FONT_LPI.

Return Value

None

Example

```
void Sample110() {

   ECLPageSettings PgSet('A');

   PgSet.SetLPI(10);
   ULONG lpi = PgSet.GetLPI();
   printf("LPI = %ld\n", lpi);
   if (PgSet.IsFontLPI())
      printf("FontLPI\n");
   else
      printf("Not FontLPI\n");
} // end sample
```

GetLPI

This method returns the LPI (lines per inch) value of the connection. Even if **Font LPI** is selected in the associated connection, this method returns the value of the LPI selected for the font in the associated connection.

If **Font LPI** is configured in the connection, this method does not return the constant FONT_LPI. Use the IsFontLPI method to determine whether **Font LPI** is set in the connection.

Prototype

ULONG GetLPI() const;

Parameters

None

Return Value

ULONG LPI

Lines per inch.

Example

```
void Sample110() {

   ECLPageSettings PgSet('A');

   PgSet.SetLPI(10);
   ULONG lpi = PgSet.GetLPI();
   printf("LPI = %ld\n", lpi);
   if (PgSet.IsFontLPI())
      printf("FontLPI\n");
   else
      printf("Not FontLPI\n");
} // end sample
```

IsFontLPI

This method returns an indication of whether **Font LPI** is set in the associated connection.

Prototype

BOOL IsFontLPI();

Parameters

None

Return Value

BOOL

Possible values are as follows:

- TRUE if **Font LPI** is set in the connection.
- FALSE if **Font LPI** is not set in the connection.

Example

```
void Sample110() {

   ECLPageSettings PgSet('A');

   PgSet.SetLPI(10);
   ULONG lpi = PgSet.GetLPI();
   printf("LPI = %ld\n", lpi);
   if (PgSet.IsFontLPI())
      printf("FontLPI\n");
   else
      printf("Not FontLPI\n");
} // end sample
```

SetFontFaceName

This method sets the font face in the connection.

Prototype

void SetFontFaceName(const char *const FaceName);

Parameters

char *FaceName

A null-terminated string that contains the font face name.

Return Value

None

Example

```
void Sample111() {

   ECLPageSettings PgSet('A');
   const char *Face;

   PgSet.SetFontFaceName("Courier New");
   Face = PgSet.GetFontFaceName();
   printf("FaceName = %s\n", Face);
} // end sample
```

GetFontFaceName

This method returns a pointer to a null-terminated string. The string contains the face name of the font that is currently chosen in the page settings for the connection that is associated with the ECLPageSettings object. The method might not return the same string each time.

The string is valid only for the lifetime of the object. You must either make a copy of the string or make a call to this method each time you need it.

Prototype

const char *GetFontFaceName() const;

Parameters

None

Return Value

char *

A pointer to a null-terminated string that contains the face name of the font.

Example

```
void Sample111() {

   ECLPageSettings PgSet('A');
   const char *Face;

   PgSet.SetFontFaceName("Courier New");
   Face = PgSet.GetFontFaceName();
   printf("FaceName = %s\n", Face);
} // end sample
```

SetFontSize

This method sets the size of the font.

Prototype

void SetFontSize(ULONG FontSize);

Parameters

ULONG FontSize

Size of the font to set in the connection.

Return Value

None

SetMaxLinesPerPage

This method sets the maximum number of lines that can be printed per page.

Prototype

void SetMaxLinesPerPage(ULONG MPL);

Parameters

ULONG MPL

The maximum lines per page (Maximum Print Lines). Valid values are in the range 1–255.

Return Value

None

Example

```
void Sample113() {

   ECLPageSettings PgSet('A');

   PgSet.SetMaxLinesPerPage(40);
   ULONG MPL = PgSet.GetMaxLinesPerPage();
   printf("MaxLinesPerPage = %ld\n", MPL);
} // end sample
```

GetMaxLinesPerPage

This method returns the maximum number of lines that can be printed per page.

Prototype

ULONG GetMaxLinesPerPage() const;

Parameters

None

Return Value

ULONG

The maximum lines per page (Maximum Print Lines).

Example

```
void Sample113() {

   ECLPageSettings PgSet('A');

   PgSet.SetMaxLinesPerPage(40);
   ULONG MPL = PgSet.GetMaxLinesPerPage();
   printf("MaxLinesPerPage = %ld\n", MPL);
} // end sample
```

SetMaxCharsPerLine

This method sets the maximum number of characters that can be printed per line.

Prototype

void SetMaxCharsPerLine(ULONG MPP);

Parameters

ULONG MPP

The maximum number of characters that can be printed per line (Maximum Print Position). Valid values are in the range 1–255.

Return Value

None

Example

```
void Sample114() {

   ECLPageSettings PgSet('A');

   PgSet.SetMaxCharsPerLine(50);
   ULONG MPP = PgSet.GetMaxCharsPerLine();
   printf("MaxCharsPerLine=%ld\n", MPP);
} // end sample
```

GetMaxCharsPerLine

This method returns the maximum number of characters that can be printed per line.

Prototype

ULONG GetMaxCharsPerLine() const;

Parameters

None

Return Value

ULONG

The maximum number of characters that can be printed per line (Maximum Print Position).

Example

```
void Sample114() {

   ECLPageSettings PgSet('A');

   PgSet.SetMaxCharsPerLine(50);
   ULONG MPP = PgSet.GetMaxCharsPerLine();
   printf("MaxCharsPerLine=%ld\n", MPP);
} // end sample
```

RestoreDefaults

This method restores the system default values of the property pages specified in the nFlags field of the PageSetup panel. This is equivalent to clicking the **Default** button in the connection **Page Setup** dialog property pages.

Prototype

void RestoreDefaults(ULONG Flags=PAGE_TEXT) const;

Parameters

ULONG Flags

This parameter is optional. The following flag describes the name of the specified **Page Setup** dialog property page. This flag can be bitwise ORed to restore the property page (defined in PCSAPI32.H).

PAGE_TEXT

This flag describes the Text property page. This is the only property page currently supported.

Return Value

None

Example

```
void Sample115() {

   ECLPageSettings PgSet('A');

   PgSet.RestoreDefaults(PAGE_TEXT);
} // end sample
```

ECLPrinterSettings Class

The ECLPrinterSettings class performs operations on the printer settings of the Z and I Emulator for Windows connection. It enables you to retrieve and configure the **File → Printer Setup** dialog settings, such as Printer and PDT Mode.

Derivation

ECLBase > ECLConnection > ECLPrinterSettings

Properties

None

Restrictions

The connection associated with each method must be in a particular state for the method to succeed. If the restrictions are not met, an appropriate exception is raised.

The following restrictions apply when any method of the ECLPrinterSettings class is invoked. If the restrictions are not met, an exception is thrown.

- The connection **Page Setup** and **Printer Setup** dialogs must not be in use.
- The connection must not be printing.

Additional restrictions might apply for each specific method.

Usage Notes

Because ECLPrinterSettings is derived from ECLConnection, you can obtain all the information contained in an ECLConnection object. See ECLConnection Class on page 754 for more information.

The ECLPrinterSettings object is created for the connection identified upon construction. You can create an ECLPrinterSettings object by passing either the connection ID (a single alphabetical character from **A** to **Z**) or the connection handle (usually obtained from the ECLConnection object). There can be only one Z and I Emulator for Windows connection with a given name or handle open at one time.

The ECLSession class creates an instance of this object. If the application does not need other services provided by ECLSession, you can create this object independently. Otherwise, consider creating an ECLSession object and use the objects created by ECLSession. See ECLSession Class on page 914 for more information.

ECLPrinterSettings Methods

The following sections describe the methods that are valid for the ECLPrinterSettings class.

```
ECLPrinterSettings(char Name)
ECLPrinterSettings(long Handle)
~ECLPrinterSettings()
void SetPDTMode(BOOL PDTMode=TRUE, const char*const PDTFile = NULL)
const char *GetPDTFile() const
BOOL IsPDTMode() const
ECLPrinterSettings::PrintMode GetPrintMode() const
void SetPrtToDskAppend(const char *const FileName = NULL)
const char *GetPrtToDskAppendFile()
void SetPrtToDskSeparate(const char *const FileName = NULL)
const char *GetPrtToDskSeparateFile()
void SetSpecificPrinter(const char *const PrinterName)
void SetWinDefaultPrinter()
const char*GetPrinterName()
void SetPromptDialog(BOOL Prompt=TRUE)
BOOL IsPromptDialogEnabled()
```

ECLPrinterSettings Constructor

This method uses a connection name or handle to create an ECLPrinterSettings object.

Prototype

ECLPrinterSettings(char Name)

ECLPrinterSettings(long Handle)

Parameters

char Name

One-character short name of the connection. Valid values are A–Z.

long Handle

Handle of an ECL connection.

Return Value

None

Example

The following example shows how to create an ECLPrinterSettings object using the connection name and the connection handle.

```
void Sample116() {
   ECLPrinterSettings *PrSet1, *PrSet2; // Pointer to ECLPrinterSettings objects
   ECLConnList ConnList; // Connection list object

   try {
      // Create ECLPrinterSettings object for connection 'A'
      PrSet1 = new ECLPrinterSettings('A');
      // Create ECLPrinterSettings object for first connection in conn list
      ECLConnection *Connection = ConnList.GetFirstConnection();
      if (Connection != NULL) {
         PrSet2 = new ECLPrinterSettings(Connection->GetHandle());
         printf("PrSet#1 is for connection %c, PrSet #2 is for connection %c.\n",
            PrSet1->GetName(), PrSet2->GetName());
      delete PrSet1;
      delete PrSet2;
   } else
      printf("No connections to create PageSettings object.\n");
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetPDTMode

This method sets the connection in PDT mode with the given PDT file, or it sets the connection in non-PDT mode (GDI mode).

Note: If this method is called with PDTMode set to FALSE, PrintMode of the associated connection must already be SpecificPrinter or WinDefaultPrinter.

Prototype

void SetPDTMode(BOOL PDTMode=TRUE, const char *const PDTFile = NULL);

Parameters

BOOL PDTMode

This parameter is optional. Possible values are as follows:

- **TRUE** to set the connection to PDT mode. This is the default value.
- **FALSE** to set the connection in non-PDT mode.

char *PDTFile

Null-terminated string containing the name of the PDT file.

This parameter is optional. It is used only if PDTMode is TRUE. The parameter is ignored if PDTMode is FALSE.

Possible values are as follows:

- NULL

  The PDT file configured in the connection is used. If there is no PDT file already configured in the connection, this method fails with an exception. This is the default value.
- File name without the path

  PDTFile in the PDFPDT subfolder in the Z and I Emulator for Windows installation path is used.
- Fully qualified path name of the file

  If PDTFile does not exist, this method fails with an exception.

Return Value

None

Example

```
void Sample117() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPDTMode(TRUE, "epson.pdt");
      const char *PDTFile = PrSet.GetPDTFile();
      printf("PDT File = %s\n", PDTFile);
      if (PrSet.IsPDTMode())
         printf("PDTMode\n");
      else
         printf("Not PDTMode\n");
      PrSet.SetPDTMode(FALSE);
      PrSet.SetPDTMode(TRUE);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPDTFile

This method returns the PDT file configured in the connection. The method might not return the same string each time.

The string is valid only for the lifetime of the object. You must either make a copy of the string or make a call to this method each time you need it.

Prototype

const char *GetPDTFile() const;

Parameters

None

Return Value

char *

Possible values are as follows:

- A null-terminated string containing the fully qualified path name of the PDT file of the connection.
- **NULL** if no PDT file is configured in the connection.

Example

```
void Sample117() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPDTMode(TRUE, "epson.pdt");
      const char *PDTFile = PrSet.GetPDTFile();
      printf("PDT File = %s\n", PDTFile);
      if (PrSet.IsPDTMode())
         printf("PDTMode\n");
      else
```

```
         printf("Not PDTMode\n");
      PrSet.SetPDTMode(FALSE);
      PrSet.SetPDTMode(TRUE);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

IsPDTMode

This method returns the state of the PDT mode of the connection.

Prototype

BOOL IsPDTMode() const;

Parameters

None

Return Value

BOOL

Possible values are as follows:

- TRUE if the connection is in PDT mode.
- FALSE if the connection is not in PDT mode.

Example

```
void Sample117() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPDTMode(TRUE, "epson.pdt");
      const char *PDTFile = PrSet.GetPDTFile();
      printf("PDT File = %s\n", PDTFile);
      if (PrSet.IsPDTMode())
         printf("PDTMode\n");
      else
         printf("Not PDTMode\n");
      PrSet.SetPDTMode(FALSE);
      PrSet.SetPDTMode(TRUE);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPrintMode

This method returns an enumerated value that indicates the PrintMode of the connection. The enum data type ECLPrinterSettings::PrintMode is defined in ECLPRSET.HPP.

PrintMode can be one of the following:

- **PrtToDskAppend** (**Print to Disk-Append** mode)

  This is equivalent to selecting the **Append** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **PrtToDskSeparate** (**Print to Disk-Separate** mode)

  This is equivalent to selecting the **Separate** option in the host session **Printer Setup → Printer → Print to Disk** dialog.
- **WinDefaultPrinter** (**Windows Default Printer** mode)

  This is equivalent to selecting the **Use Windows Default Printer** option in the host session **Printer Setup** dialog.
- **SpecificPrinter** (**Specific Printer** mode)

  This is equivalent to selecting a printer in the host session **Printer Setup** dialog, while leaving **Use Windows Default Printer** unchecked.

Prototype

ECLPrinterSettings::PrintMode GetPrintMode() const;

Parameters

None

Return Value

ECLPrinterSettings::PrintMode

One of the PrintMode values defined in ECLPRSET.HPP.

Example

```
void Sample118() {

   ECLPrinterSettings PrSet('A');

   ECLPrinterSettings::PrintMode PrtMode;
   PrtMode = PrSet.GetPrintMode();
   switch (PrtMode) {
   case ECLPrinterSettings::PrtToDskAppend:
      printf("PrtToDskAppend mode\n");
```

```
         break;
      case ECLPrinterSettings::PrtToDskSeparate:
         printf("PrtToDskSeparate mode\n");
         break;
      case ECLPrinterSettings::SpecificPrinter:
         printf("SpecificPrinter mode\n");
         break;
      case ECLPrinterSettings::WinDefaultPrinter:
         printf("WinDefaultPrinter mode\n");
         break;
   }
} // end sample
```

SetPrtToDskAppend

This method sets the PrintMode to **Print to Disk-Append** mode and sets the appropriate file for this mode.

Note:

1. The associated connection must be in PDT mode.
2. The folder where this file is to be set must have write access. If it does not, this method fails with an exception.
3. If the file exists, it will be used. Otherwise, it will be created when printing is complete.

Prototype

void SetPrtToDskAppend(const char *const FileName = NULL);

Parameters

char *FileName

Null-terminated string containing the name of the **Print to Disk-Append** file. This parameter is optional. Possible values are as follows:

- NULL

  The file that is currently configured for this PrintMode in the connection is used. If there is no file already configured in the connection, the method fails with an exception. This is the default value.
- File name, without the path

  The user-class application data directory path will be used to locate the file.
- Fully qualified path name of the file

  The directory must exist in the path, or the method will fail with an exception. It is not necessary that the file exist in the path.

Return Value

None

Example

```
void Sample119() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPrtToDskAppend("dskapp.txt");
      const char *DskAppFile = PrSet.GetPrtToDskAppendFile();
      printf("Print to Disk-Append File = %s\n", DskAppFile);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPrtToDskAppendFile

This method returns the file configured for **Print to Disk-Append** mode. This file is called the ***Print to Disk-Append*** file. The method might not return the same string each time.

The string is valid only for the lifetime of the object. You must either make a copy of the string or make a call to this method each time you need it.

Prototype

const char *GetPrtToDskAppendFile();

Parameters

None

Return Value

char *

Possible values are as follows:

- A null-terminated string that contains the fully qualified path name of the **Print to Disk-Append** file of the connection.
- NULL if the **Print to Disk-Append** file is not configured in the connection.

Example

```
void Sample119() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPrtToDskAppend("dskapp.txt");
      const char *DskAppFile = PrSet.GetPrtToDskAppendFile();
      printf("Print to Disk-Append File = %s\n", DskAppFile);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetPrtToDskSeparate

This method sets the connection in **Print to Disk-Separate** mode and sets the appropriate file for this mode.

Note:

1. The associated connection must be in PDT mode.
2. The folder where this file is to be set must have write access. If it does not, this method fails with an exception.
3. The file name must not contain an extension. If it contains an extension, the method fails with an exception.

Prototype

void SetPrtToDskSeparate(const char *const FileName = NULL);

Parameters

char *FileName

Null-terminated string containing the name of the **Print to Disk-Separate** file. This parameter is optional. Possible values are as follows:

- NULL

  The file that is currently configured for this PrintMode in the connection is used. If there is no file already configured in the connection, the method fails with an exception. This is the default value.
- File name, without the path

The user-class application data directory path will be used to locate the file.

- Fully qualified path name of the file

  The directory must exist in the path, or the method will fail with an exception. It is not necessary that the file exist in the path.

Return Value

None

Example

```
void Sample120() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPrtToDskSeparate("dsksep");
      const char *DskSepFile = PrSet.GetPrtToDskSeparateFile();
      printf("Print to Disk-Separate File = %s\n", DskSepFile);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPrtToDskSeparateFile

This method returns the file configured for **Print to Disk-Separate** mode. This file is called the ***Print to Disk-Separate*** file. The method might not return the same string each time.

The string is valid only for the lifetime of the object. You must either make a copy of the string or make a call to this method each time you need it.

Prototype

const char *GetPrtToDskSeparateFile();

Parameters

None

Return Value

char *

Possible values are as follows:

- A null-terminated string that contains the fully qualified path name of the **Print to Disk-Separate** file.
- NULL, if no **Print to Disk-Separate** file is configured in the connection.

Example

```
void Sample120() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPrtToDskSeparate("dsksep");
      const char *DskSepFile = PrSet.GetPrtToDskSeparateFile();
      printf("Print to Disk-Separate File = %s\n", DskSepFile);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetSpecificPrinter

This method sets the connection in SpecificPrinter mode with the printer specified in the Printer parameter.

Prototype

void SetSpecificPrinter(const char *const Printer);

Parameters

char *Printer

A null-terminated string that contains the printer name and the port name. If the printer does not exist, this method fails with an exception.
The value must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `HP LaserJet 4050 Series PCL 6 on LPT1`

Return Value

None

Example

```
void Sample121() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetSpecificPrinter("HCL InfoPrint 40 PS on Network Port");
      const char *Printer = PrSet.GetPrinterName();
      printf("Printer = %s\n", Printer);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetWinDefaultPrinter

This method sets the connection in WinDefaultPrinter mode—that is, the connection is made to use the Windows® default printer. If no Windows default printer is configured in the machine, the method fails with an exception.

Prototype

void SetWinDefaultPrinter();

Parameters

None

Return Value

None

Example

```
void Sample122() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetWinDefaultPrinter();
      const char *Printer = PrSet.GetPrinterName();
   printf("Windows Default Printer = %s\n", Printer);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

GetPrinterName

This method returns NULL or the name of the printer configured in the connection. The method might not return the same string each time.

The string is valid only for the lifetime of the object. You must either make a copy of the string or make a call to this method each time you need it.

PrinterName must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `HP LaserJet 4050 Series PCL 6 on LPT1`

Prototype

const char *GetPrinterName();

Parameters

None

Return Value

char *

Possible values are as follows:

- A null-terminated string that contains the name of the specific printer, if the PrintMode of the connection is SpecificPrinter.
- A null-terminated string that contains the name of the Windows default printer, if the PrintMode of the connection is WinDefaultPrinter.
- NULL if no Printer is configured in the connection, or if the PrintMode of the connection is PrtToDskAppend or PrtToDskSeparate.

Example

```
void Sample122() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetWinDefaultPrinter();
      const char *Printer = PrSet.GetPrinterName();
   printf("Windows Default Printer = %s\n", Printer);
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

SetPromptDialog

This method sets or resets the option to show the Printer Setup dialog before printing.

Prototype

void SetPromptDialog(BOOL bPrompt=TRUE);

Parameters

BOOL bPrompt

This parameter is optional. Possible values are as follows:

- TRUE to show the Printer Setup dialog before printing. This is the default value.
- FALSE to not show the Printer Setup dialog before printing.

Return Value

None

Example

```
void Sample123() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPromptDialog();
      if (PrSet.IsPromptDialogEnabled())
         printf("Prompt Dialog before Printing - Enabled\n");
      else
         printf("Prompt Dialog before Printing - Disabled\n");
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

IsPromptDialogEnabled

This method checks whether the Printer Setup dialog is shown before printing or not.

Prototype

BOOL IsPromptDialogEnabled();

Parameters

None

Return Value

BOOL

Possible values are as follows:

- TRUE if the Printer Setup dialog is shown before printing.
- FALSE if the Printer Setup dialog is not shown before printing.

Example

```
void Sample123() {

   ECLPrinterSettings PrSet('A');

   try {
      PrSet.SetPromptDialog();
      if (PrSet.IsPromptDialogEnabled())
         printf("Prompt Dialog before Printing - Enabled\n");
      else
         printf("Prompt Dialog before Printing - Disabled\n");
   }
   catch (ECLErr Err) {
      printf("ECL Error: %s\n", Err.GetMsgText());
   }
} // end sample
```

Host Access Class Library Automation Objects

The Host Access Class Library Automation Objects allow the Z and I Emulator for Windows product to support Microsoft® COM-based automation technology (formerly known as OLE automation). The ECL Automation Objects are a series of automation servers that allow automation controllers, for example, Microsoft® Visual Basic®, to programmatically access Z and I Emulator for Windows data and functionality.

An example of this would be sending keys to Z and I Emulator for Windows presentation space. This can be accomplished by manually typing keys in the Z and I Emulator for Windows window, but it can also be automated through the appropriate Z and I Emulator for Windows automation server (autECLPS in this case). Using Visual Basic® you can create the autECLPS object and then call the SendKeys method in that object with the string that is to be placed in the presentation space.

In other words, applications that are enabled for controlling the automation protocol (automation controller) can control some Z and I Emulator for Windows operations (automation server). Z and I Emulator for Windows supports

Visual Basic® Script, which uses ECL Automation objects. Refer to the Z and I Emulator for Windows Macro/Script support for more details.

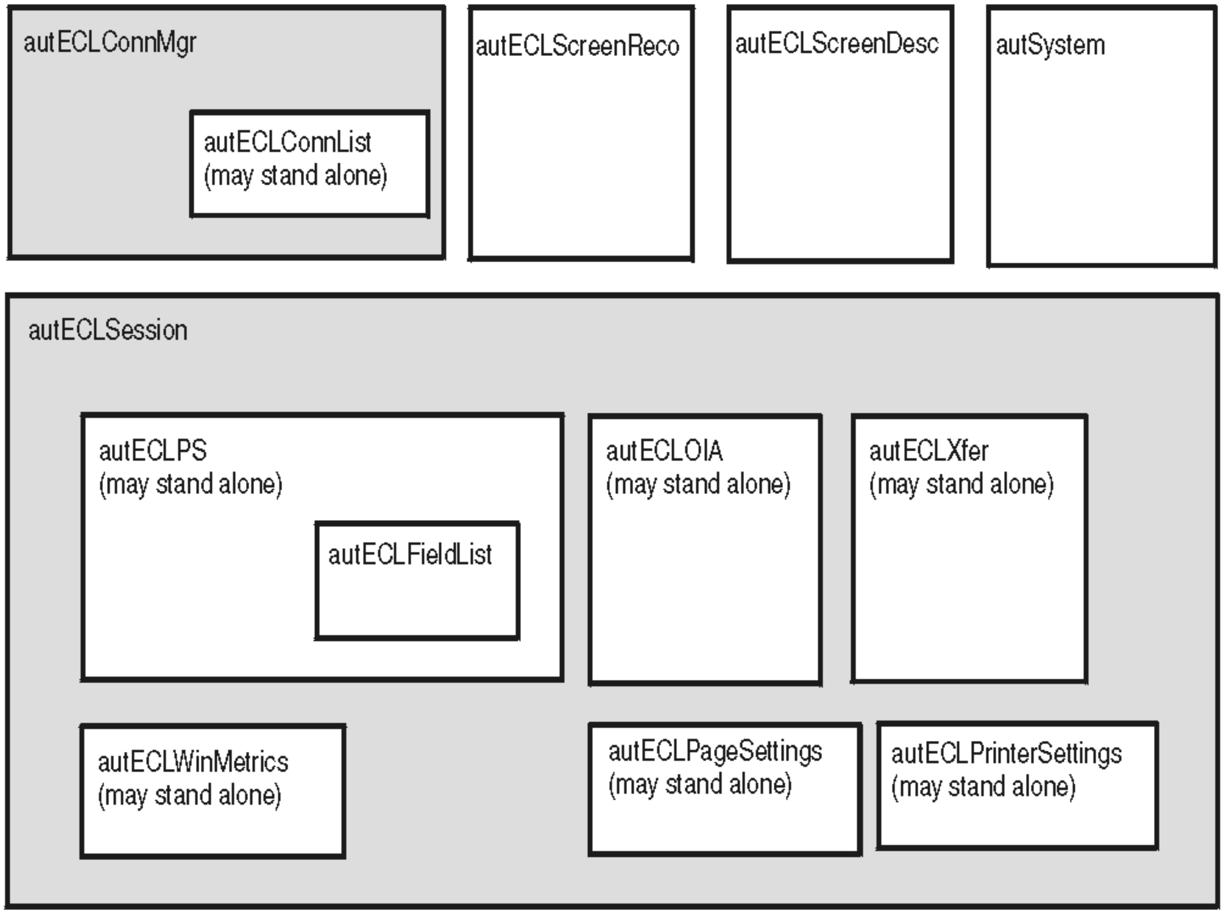
Z and I Emulator for Windows offers several automation servers to accomplish this. These servers are implemented as real-world, intuitive objects with methods and properties that control Z and I Emulator for Windows operability. Each object begins with autECL, for automation Host Access Class Library. The objects are as follows:

- autECLConnList, Connection List, on page autECLConnList Class on page 987, contains a list of Z and I Emulator for Windows connections for a given system. This is contained by autECLConnMgr, but may be created independently of autECLConnMgr.
- autECLConnMgr, Connection Manager, on page autECLConnMgr Class on page 994, provides methods and properties to manage Z and I Emulator for Windows connections for a given system. A connection in this context is a Z and I Emulator for Windows window.
- autECLFieldList, Field List, on page autECLFieldList Class on page 1000, performs operations on fields in an emulator presentation space.
- autECLOIA, Operator Information Area, on page autECLOIA Class on page 1011, provides methods and properties to query and manipulate the Operator Information Area. This is contained by autECLSession, but may be created independently of autECLSession.
- autECLPS, Presentation Space, on page autECLPS Class on page 1029, provides methods and properties to query and manipulate the presentation space for the related Z and I Emulator for Windows connection. This contains a list of all the fields in the presentation space. It is contained by autECLSession, but may be created independently of autECLSession.
- autECLScreenDesc, Screen Description, on page autECLScreenDesc Class on page 1069, provides methods and properties to describe a screen. This may be used to wait for screens on the autECLPS object or the autECLScreenReco object.
- autECLScreenReco, Screen Recognition, on page autECLScreenReco Class on page 1077, provides the engine of the HACL screen recognition system.
- autECLSession, Session, on page autECLSession Class on page 1083, provides general session-related functionality and information. For convenience, it contains the autECLPS, autECLOIA, autECLXfer, autECLWinMetrics, autECLPageSettings, and autECLPrinterSettings objects.
- autECLWinMetrics, Window Metrics, on page autECLWinMetrics Class on page 1096, provides methods to query the window metrics of the Z and I Emulator for Windows session associated with this object. For example, use this object to minimize or maximize a Z and I Emulator for Windows window. This is contained by autECLSession, but may be created independently of autECLSession.
- autECLXfer, File Transfer, on page autECLXfer Class on page 1112, provides methods and properties to transfer files between the host and the workstation over the Z and I Emulator for Windows connection associated with this file transfer object. This is contained by autECLSession, but may be created independently of autECLsession.
- autECLPageSettings, Page Settings, on page autECLPageSettings Class on page 1126, provides methods and properties to query and manipulate commonly used settings such as CPI, LPI, and Face Name of the session Page Setup dialog. This is contained by autECLSession, but may be created independently of autECLSession.

- autECLPrinterSettings, Printer Settings, on page autECLPrinterSettings Class on page 1137, provides methods and properties to query and manipulate settings such as the Printer and PDT modes of the session Printer Setup dialog. This is contained by autECLSession, but may be created independently of autECLSession.

Figure 15: Host Access Class Library Automation Objects on page 986 is a graphical representation of the autECL objects:

Figure 15. Host Access Class Library Automation Objects



This chapter describes each object's methods and properties in detail and is intended to cover all potential users of the automation object. Because the most common way to use the object is through a scripting application such as Visual Basic®, all examples are shown using a Visual Basic® format.

autSystem Class

The autSystem Class provides two utility functions that may be useful for use with some programming languages. See autSystem Class on page 1124 for more information.

autECLConnList Class

autECLConnList contains information about all started connections. Its name in the registry is ZIEWin.autECLConnList.

The autECLConnList object contains a collection of information about connections to a host. Each element of the collection represents a single connection (emulator window). A connection in this list may be in any state (for example, stopped or disconnected). All started connections appear in this list. The list element contains the state of the connection.

An autECLConnList object provides a static snapshot of current connections. The list is not dynamically updated as connections are started and stopped. The Refresh method is automatically called upon construction of the autECLConnList object. If you use the autECLConnList object right after its construction, your list of connections is current. However, you should call the Refresh method in the autECLConnList object before accessing its other methods if some time has passed since its construction to ensure that you have current data. Once you have called Refresh you may begin walking through the collection

Properties

This section describes the properties for the autECLConnList object.

Type	Name	Attributes
Long	Count	Read-only

The following table shows Collection Element Properties, which are valid for each item in the list.

Type	Name	Attributes
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

Count

This is the number of connections present in the autECLConnList collection for the last call to the Refresh method. The Count property is a Long data type and is read-only. The following example uses the Count property.

```
Dim autECLConnList as Object
Dim Num as Long
```

```
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

autECLConnList.Refresh
Num = autECLConnList.Count
```

Name

This collection element property is the connection name string of the connection. Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example uses the Name collection element property.

```
Dim  Str as String
Dim autECLConnList as Object
Dim Num as Long

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

autECLConnList.Refresh
Str = autECLConnList(1).Name
```

Handle

This collection element property is the handle of the connection. There can be only one Z and I Emulator for Windows connection open with a given handle. Handle is a Long data type and is read-only. The following example uses the Handle property.

```
Dim autECLConnList as Object
Dim Hand as Long

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

autECLConnList.Refresh
Hand = autECLConnList(1).Handle
```

ConnType

This collection element property is the connection type. This type may change over time. ConnType is a String data type and is read-only. The following example shows the ConnType property.

```
Dim  Type as String
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

autECLConnList.Refresh
Type = autECLConnList(1).ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This collection element property is the code page of the connection. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows the CodePage property.

```
Dim  CodePage as Long
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

autECLConnList.Refresh
CodePage = autECLConnList(1).CodePage
```

Started

This collection element property indicates whether the emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows the Started property.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
autECLConnList.Refresh

' This code segment checks to see if is started.
' The results are sent to a text box called Result.
If Not autECLConnList(1).Started Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This collection element property indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows the CommStarted property.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
```

```
autECLConnList.Refresh

' This code segment checks to see if communications are connected
' The results are sent to a text box called CommConn.
If Not autECLConnList(1).CommStarted Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This collection element property indicates whether the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File -> API Settings**). The value is True if the emulator is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows the APIEnabled property.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
autECLConnList.Refresh

' This code segment checks to see if API is enabled.
' The results are sent to a text box called Result.
If Not autECLConnList(1).APIEnabled Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This collection element property indicates whether the emulator window is started, API-enabled, and connected. This property checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows the Ready property.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
autECLConnList.Refresh

' This code segment checks to see if X is ready.
' The results are sent to a text box called Result.
If Not autECLConnList(1).Ready Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

autECLConnList Methods

The following section describes the methods that are valid for the autECLConnList object.

void Refresh()

Object FindConnectionByHandle(Long Hand)

Object FindConnectionByName(String Name)

Collection Element Methods

The following collection element methods are valid for each item in the list.

void StartCommunication()

void StopCommunication()

Refresh

The Refresh method gets a snapshot of all the started connections.

Note: You should call this method before accessing the autECLConnList collection to ensure that you have current data.

Prototype

void Refresh()

Parameters

None

Return Value

None

Example

The following example shows how to use the Refresh method to get a snapshot of all the started connections.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
```

FindConnectionByHandle

This method finds an element in the autECLConnList object for the handle passed in the **Hand** parameter. This method is commonly used to see if a given connection is alive in the system.

Prototype

Object FindConnectionByHandle(Long Hand)

Parameters

Long Hand

Handle to search for in the list.

Return Value

Object

Collection element dispatch object.

Example

The following example shows how to find an element by the connection handle.

```
Dim Hand as Long
Dim autECLConnList as Object
Dim ConnObj as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the collection
autECLConnList.Refresh
' Assume Hand obtained earlier
Set ConnObj = autECLConnList.FindConnectionByHandle(Hand)
Hand = ConnObj.Handle
```

FindConnectionByName

This method finds an element in the autECLConnList object for the name passed in the **Name** parameter. This method is commonly used to see if a given connection is alive in the system.

Prototype

Object FindConnectionByName(String Name)

Parameters

String Name

Name to search for in the list.

Return Value

Object

Collection element dispatch object.

Example

The following example shows how to find an element in the autECLConnList object by the connection name.

```
Dim Hand as Long
Dim autECLConnList as Object
Dim ConnObj as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the collection
autECLConnList.Refresh
' Assume Hand obtained earlier
Set ConnObj = autECLConnList.FindConnectionByName("A")
Hand = ConnObj.Handle
```

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

'Start the first session
autECLConnList.Refresh
autECLConnList(1).StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to disconnect a ZIEWin emulator session from the host.

```
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

'Start the first session
autECLConnList.Refresh
autECLConnList(1).StartCommunication()
'
'Interact programmatically with host
'
autECLConnList.Refresh
'Stop the first session
autECLConnList(1).StartCommunication()
```

autECLConnMgr Class

autECLConnMgr manages all Z and I Emulator for Windows connections on a given machine. It contains methods relating to the connection management such as starting and stopping connections. It also creates an autECLConnList object to enumerate the list of all known connections on the system (see autECLConnList Class on page 987). Its name in the registry is ZIEWin.autECLConnMgr.

Properties

This section describes the properties for the autECLConnMgr object.

Type	Name	Attributes
autECLConnList Object	autECLConnList	Read-only

autECLConnList

The autECLConnMgr object contains an autECLConnList object. See autECLConnList Class on page 987 for details on its methods and properties. The property has a value of autECLConnList, which is an autECLConnList dispatch object. The following example shows this property.

```
Dim Mgr as Object
Dim Num as Long

Set Mgr = CreateObject("ZIEWin.autECLConnMgr ")

Mgr.autECLConnList.Refresh
Num = Mgr.autECLConnList.Count
```

autECLConnMgr Methods

The following section describes the methods that are valid for autECLConnMgr.

void RegisterStartEvent()
void UnregisterStartEvent()
void StartConnection(String ConfigParms)
void StopConnection(Variant Connection, [optional] String StopParms)

RegisterStartEvent

This method registers an autECLConnMgr object to receive notification of start events in sessions.

Prototype

void RegisterStartEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1000 for an example.

UnregisterStartEvent

Ends Start Event Processing

Prototype

void UnregisterStartEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1000 for an example.

StartConnection

This member function starts a new Z and I Emulator for Windows emulator window. The ConfigParms string contains connection configuration information as explained under Usage Notes on page 996.

Prototype

void StartConnection(String ConfigParms)

Parameters

String ConfigParms

Configuration string.

Return Value

None

Usage Notes

The configuration string is implementation-specific. Different implementations of the autECL objects may require different formats or information in the configuration string. The new emulator is started upon return from this call, but it may or may not be connected to the host.

For Z and I Emulator for Windows, the configuration string has the following format:

```
PROFILE=[']<filename>['] [CONNNAME=<c>] [WINSTATE=<MAX|MIN|RESTORE|HIDE>]
```

Optional parameters are enclosed in square brackets []. The parameters are separated by at least one blank. Parameters may be in upper, lower, or mixed case and may appear in any order. The meaning of each parameter is as follows:

- PROFILE=<filename>: Names the Z and I Emulator for Windows workstation profile (.WS file), which contains the configuration information. This parameter is not optional; a profile name must be supplied. If the file name contains blanks the name must be enclosed in single quotation marks. The <filename> value may be either the profile name with no extension, the profile name with the .WS extension, or the fully qualified profile name path.
- CONNNAME=<c> specifies the short ID of the new connection. This value must be a single, alphabetic character (A-Z or a-z). If this value is not specified, the next available connection ID is assigned automatically.
- WINSTATE=<MAX|MIN|RESTORE|HIDE> specifies the initial state of the emulator window. The default if this parameter is not specified is RESTORE.

Example

The following example shows how to start a new Z and I Emulator for Windows emulator window.

```
Dim Mgr as Object
Dim Obj as Object
Dim Hand as Long

Set Mgr = CreateObject("ZIEWin.autECLConnMgr ")
Mgr.StartConnection("profile=coax connname=e")
```

StopConnection

The StopConnection method stops (terminates) the emulator window identified by the connection handle. See Usage Notes on page 998 for contents of the StopParms string.

Prototype

void StopConnection(Variant Connection, [optional] String StopParms)

Parameters

Variant Connection

Connection name or handle. Legal types for this variant are short, long, BSTR, short by reference, long by reference, and BSTR by reference.

String StopParms

Stop parameters string. See usage notes for format of string. This parameter is optional.

Return Value

None

Usage Notes

The stop parameter string is implementation-specific. Different implementations of the autECL objects may require a different format and contents of the parameter string. For Z and I Emulator for Windows, the string has the following format:

```
[SAVEPROFILE=<YES|NO|DEFAULT>]
```

Optional parameters are enclosed in square brackets []. The parameters are separated by at least one blank. Parameters may be in upper, lower, or mixed case and may appear in any order. The meaning of each parameter is as follows:

- SAVEPROFILE=<YES|NO|DEFAULT> controls the saving of the current configuration back to the workstation profile (.WS file). This causes the profile to be updated with any configuration changes you may have made. If NO is specified, the connection is stopped and the profile is not updated. If YES is specified, the connection is stopped and the profile is updated with the current (possibly changed) configuration. If DEFAULT is specified, the update option is controlled by the **File->Save On Exit** emulator menu option. If this parameter is not specified, DEFAULT is used.

Example

The following example shows how to stop the emulator window identified by the connection handle.

```
Dim Mgr as Object
Dim Hand as Long

Set Mgr = CreateObject("ZIEWin.autECLConnMgr ")

' Assume we've got connections open and the Hand parm was obtained earlier
Mgr.StopConnection Hand, "saveprofile=no"
'or
Mgr.StopConnection "B", "saveprofile=no"
```

autECLConnMgr Events

The following events are valid for autECLConnMgr:

void NotifyStartEvent(By Val Handle As Variant, By Val Started As Boolean)
NotifyStartError(By Val ConnHandle As Variant)
void NotifyStartStop(Long Reason)

NotifyStartEvent

A Session has started or stopped.

Prototype

void NotifyStartEvent(By Val Handle As Variant, By Val Started As Boolean)

Note: Visual Basic will create this subroutine correctly.

Parameters

By Val Handle As Variant

Handle of the Session that started or stopped.

By Val Started As Boolean

True if the Session is started, False otherwise.

Example

See Event Processing Example on page 1000 for an example.

NotifyStartError

This event occurs when an error occurs in Event Processing.

Prototype

NotifyStartError(By Val ConnHandle As Variant)

Note: Visual Basic will create this subroutine correctly.

Parameters

None

Example

See Event Processing Example on page 1000 for an example.

NotifyStartStop

This event occurs when event processing stops.

Prototype

void NotifyStartStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently, this will always be 0.

Event Processing Example

The following is a short example of how to implement Start Events:

```
Option Explicit
Private WithEvents mCmgr As autECLConnMgr 'AutConnMgr added as reference
dim mSess as object

sub main()
'Create Objects
Set mCmgr = New autECLConnMgr
Set mSess = CreateObject("ZIEWin.autECLSession")
mCmgr.RegisterStartEvent 'register for PS Updates

' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
call DisplayGUI()
mCmgr.UnregisterStartEvent
set mCmgr = Nothing
set mSess = Nothing
End Sub

'This sub will get called when a session is started or stopped
Private Sub mCmgr_NotifyStartEvent(Handle as long, bStarted as Boolean)
' do your processing here
if (bStarted) then
mSess.SetConnectionByHandle Handle
end if
End Sub

'This event occurs if an error happens
Private Sub mCmgr_NotifyStartError()
'Do any error processing here
End Sub

Private Sub mCmgr_NotifyStartStop(Reason As Long)
'Do any stop processing here
End Sub
```

autECLFieldList Class

autECLFieldList performs operations on fields in an emulator presentation space. This object does not stand on its own. It is contained by autECLPS, and can only be accessed through an autECLPS object. autECLPS can stand alone or be contained by autECLSession.

autECLFieldList contains a collection of all the fields on a given presentation space. Each element of the collection contains the elements shown in Collection Element Properties on page 1001.

An autECLFieldList object provides a static snapshot of what the presentation space contained when the Refresh method was called.

Note: You should call the Refresh method in the autECLFieldList object before accessing its elements to ensure that you have current field data. Once you have called Refresh, you may begin walking through the collection.

Properties

This section describes the properties and the collection element properties for the autECLFieldList object.

Type	Name	Attributes
Long	Count	Read-only

The following properties are collection element properties and are valid for each item in the list.

Type	Name	Attributes
Long	StartRow	Read-only
Long	StartCol	Read-only
Long	EndRow	Read-only
Long	EndCol	Read-only
Long	Length	Read-only
Boolean	Modified	Read-only
Boolean	Protected	Read-only
Boolean	Numeric	Read-only
Boolean	HighIntensity	Read-only
Boolean	PenDetectable	Read-only
Boolean	Display	Read-only

Count

This property is the number of fields present in the autECLFieldList collection for the last call to the Refresh method. Count is a Long data type and is read-only. The following example shows this property.

```
Dim NumFields as long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
```

```
' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
NumFields = autECLPSObj.autECLFieldList.Count
```

StartRow

This collection element property is the row position of the first character in a given field in the autECLFieldList collection. StartRow is a Long data type and is read-only. The following example shows this property.

```
Dim StartRow as Long
Dim StartCol as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  StartRow = autECLPSObj.autECLFieldList(1).StartRow
  StartCol = autECLPSObj.autECLFieldList(1).StartCol
Endif
```

StartCol

This collection element property is the column position of the first character in a given field in the autECLFieldList collection. StartCol is a Long data type and is read-only. The following example shows this property.

```
Dim StartRow as Long
Dim StartCol as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  StartRow = autECLPSObj.autECLFieldList(1).StartRow
  StartCol = autECLPSObj.autECLFieldList(1).StartCol
Endif
```

EndRow

This collection element property is the row position of the last character in a given field in the autECLFieldList collection. EndRow is a Long data type and is read-only. The following example shows this property.

```
Dim EndRow as Long
Dim EndCol as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  EndRow = autECLPSObj.autECLFieldList(1).EndRow
  EndCol = autECLPSObj.autECLFieldList(1).EndCol
Endif
```

EndCol

This collection element property is the column position of the last character in a given field in the autECLFieldList collection. EndCol is a Long data type and is read-only. The following example shows this property.

```
Dim EndRow as Long
Dim EndCol as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  EndRow = autECLPSObj.autECLFieldList(1).EndRow
  EndCol = autECLPSObj.autECLFieldList(1).EndCol
Endif
```

Length

This collection element property is the length of a given field in the autECLFieldList collection. Length is a Long data type and is read-only. The following example shows this property.

```
Dim Len as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
```

```
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  Len = autECLPSObj.autECLFieldList(1).Length
Endif
```

Modified

This collection element property indicates if a given field in the autECLFieldList collection has a modified attribute. Modified is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).Modified ) Then
    ' do whatever
  Endif
Endif
```

Protected

This collection element property indicates if a given field in the autECLFieldList collection has a protected attribute. Protected is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).Protected ) Then
    ' do whatever
  Endif
Endif
```

Numeric

This collection element property indicates if a given field in the autECLFieldList collection has a numeric input only attribute. Numeric is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).Numeric ) Then
    ' do whatever
  Endif
Endif
```

HighIntensity

This collection element property indicates if a given field in the autECLFieldList collection has a high intensity attribute. HighIntensity is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).HighIntensity ) Then
    ' do whatever
  Endif
Endif
```

PenDetectable

This collection element property indicates if a given field in the autECLFieldList collection has a pen detectable attribute. PenDetectable is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).PenDetectable ) Then
    ' do whatever
  Endif
Endif
```

Display

This collection element property indicates whether a given field in the autECLFieldList collection has a display attribute. Display is a Boolean data type and is read-only. The following example shows this property.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh(1)
If (Not autECLPSObj.autECLFieldList.Count = 0 ) Then
  If ( autECLPSObj.autECLFieldList(1).Display ) Then
    ' do whatever
  Endif
Endif
```

autECLFieldList Methods

The following section describes the methods that are valid for the autECLFieldList object.

void Refresh()
Object FindFieldByRowCol(Long Row, Long Col)
Object FindFieldByText(String text, [optional] Long Direction, [optional] Long StartRow, [optional] Long StartCol)

Collection Element Methods

The following collection element methods are valid for each item in the list.

String GetText()

void SetText(String Text)

Refresh

The Refresh method gets a snapshot of all the fields.

Note: You should call the Refresh method before accessing the field collection to ensure that you have current field data.

Prototype

void Refresh()

Parameters

None

Return Value

None

Example

The following example shows how to get a snapshot of all the fields for a given presentation space.

```
Dim NumFields as long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and get the number of fields
autECLPSObj.autECLFieldList.Refresh()
NumFields = autECLPSObj.autECLFieldList.Count
```

FindFieldByRowCol

This method searches the autECLFieldList object for a field containing the given row and column coordinates. The value returned is a collection element object in the autECLFieldList collection.

Prototype

Object FindFieldByRowCol(Long Row, Long Col)

Parameters

Long Row

Field row to search for.

Long Col

Field column to search for.

Return Value

Object

Dispatch object for the autECLFieldList collection item.

Example

The following example shows how to search the autECLFieldList object for a field containing the given row and column coordinates.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Dim FieldElement as Object

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList)

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and search for field at row 2 col 1
autECLPSObj.autECLFieldList.Refresh(1)
Set FieldElement = autECLPSObj.autECLFieldList.FindFieldByRowCol( 2, 1 )
FieldElement.SetText("HCL")
```

FindFieldByText

This method searches the autECLFieldList object for a field containing the string passed in as **Text**. The value returned is a collection element object in the autECLFieldList collection.

Prototype

Object FindFieldByText(String Text, [optional] Long Direction, [optional] Long StartRow, [optional] Long StartCol)

Parameters

String Text

The text string to search for.

Long StartRow

Row position in the presentation space at which to begin the search.

Long StartCol

Column position in the presentation space at which to begin the search.

Long Direction

Direction in which to search. Values are **1** for search forward, **2** for search backward

Return Value

Object

Dispatch object for the autECLFieldList collection item.

Example

The following example shows how to search the autECLFieldList object for a field containing the string passed in as text.

```
Dim autECLPSObj as Object
Dim autECLConnList as Object
Dim FieldElement as Object

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and search for field with text
autECLPSObj.autECLFieldList.Refresh(1)
set FieldElement = autECLPSObj.autECLFieldList.FindFieldByText "HCL"

' Or... search starting at row 2 col 1
set FieldElement = autECLPSObj.autECLFieldList.FindFieldByText "HCL", 2, 1
' Or... search starting at row 2 col 1 going backwards
set FieldElement = autECLPSObj.autECLFieldList.FindFieldByText "HCL", 2, 2, 1

FieldElement.SetText("Hello.")
```

GetText

The collection element method GetText retrieves the characters of a given field in an autECLFieldList item.

Prototype

String GetText()

Parameters

None

Return Value

String

Field text.

Example

The following example shows how to use the GetText method.

```
Dim autECLPSObj as Object
Dim TextStr as String

' Initialize the connection
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

autECLPSObj.autECLFieldList.Refresh()
TextStr = autECLPSObj.autECLFieldList(1).GetText()
```

SetText

This method populates a given field in an autECLFieldList item with the character string passed in as text. If the text exceeds the length of the field, the text is truncated.

Prototype

void SetText(String Text)

Parameters

String text

String to set in field

Return Value

None

Example

The following example shows how to populate the field in an autECLFieldList item with the character string passed in as text.

```
Dim NumFields as Long
Dim autECLPSObj as Object
Dim autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the list and set the first field with some text
autECLPSObj.autECLFieldList.Refresh(1)
autECLPSObj.autECLFieldList(1).SetText("HCL is a cool company")
```

autECLOIA Class

The autECLOIA object retrieves status from the Host Operator Information Area. Its name in the registry is ZIEWin.autECLOIA.

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection may be set only once. After the connection is set, any further calls to the set connection methods cause an exception. If you do not set the connection and try to access a property or method, an exception is also raised.

Note: The autECLOIA object in the autECLSession object is set by the autECLSession object.

The following example shows how to create and set the autECLOIA object in Visual Basic.

```
DIM  autECLOIA as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
autECLOIA.SetConnectionByName("A")
```

Properties

This section describes the properties for the autECLOIA object.

Type	Name	Attributes
Boolean	Alphanumeric	Read-only
Boolean	APL	Read-only
Boolean	UpperShift	Read-only
Boolean	Numeric	Read-only

Type	Name	Attributes
Boolean	CapsLock	Read-only
Boolean	InsertMode	Read-only
Boolean	CommErrorReminder	Read-only
Boolean	MessageWaiting	Read-only
Long	InputInhibited	Read-only
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only
Boolean	NumLock	Read-only

Alphanumeric

This property queries the operator information area to determine whether the field at the cursor location is alphanumeric. Alphanumeric is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

If autECLOIA.Alphanumeric Then...
```

APL

This property queries the operator information area to determine whether the keyboard is in APL mode. APL is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)
```

```
' Check if the keyboard is in APL mode
if autECLOIA.APL Then...
```

Katakana

This property queries the operator information area to determine whether Katakana characters are enabled. Katakana is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if Katakana characters are available
if autECLOIA.Katakana Then...
```

Hiragana

This property queries the operator information area to determine whether Hiragana characters are enabled. Hiragana is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if Hiragana characters are available
if autECLOIA.Hiragana Then...
```

UpperShift

This property queries the operator information area to determine whether the keyboard is in uppershift mode. Uppershift is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
```

```
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if the keyboard is in uppershift mode
If autECLOIA.UpperShift then...
```

Numeric

This property queries the operator information area to determine whether the field at the cursor location is numeric. Numeric is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if the cursor location is a numeric field
If autECLOIA.Numeric Then...
```

CapsLock

This property queries the operator information area to determine if the keyboard CapsLock key is on. CapsLock is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if the caps lock
If autECLOIA.CapsLock Then...
```

InsertMode

This property queries the operator information area to determine whether if the keyboard is in insert mode. InsertMode is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
```

```
' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if in insert mode
If autECLOIA.InsertMode Then...
```

CommErrorReminder

This property queries the operator information area to determine whether a communications error reminder condition exists. CommErrorReminder is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if comm error
If autECLOIA.CommErrorReminder Then...
```

MessageWaiting

This property queries the operator information area to determine whether the message waiting indicator is on. This can only occur for 5250 connections. MessageWaiting is a Boolean data type and is read-only. The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object
Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if message waiting
If autECLOIA.MessageWaiting Then...
```

InputInhibited

This property queries the operator information area to determine whether keyboard input is inhibited. InputInhibited is a Long data type and is read-only. The following table shows valid values for InputInhibited.

Name	Value
Not Inhibited	0
System Wait	1
Communication Check	2
Program Check	3
Machine Check	4
Other Inhibit	5

The following example shows this property.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object
Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)

' Check if input inhibited
If not autECLOIA.InputInhibited = 0 Then...
```

Name

This property is the connection name string of the connection for which autECLOIA was set. Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example shows this property.

```
DIM  Name as String
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the name
Name = Obj.Name
```

Handle

This is the handle of the connection for which the autECLOIA object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time. Handle is a Long data type and is read-only. The following example shows this property.

```
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")
```

```
' Save the handle
Hand = Obj.Handle
```

ConnType

This is the connection type for which autECLOIA was set. This type may change over time. ConnType is a String data type and is read-only. The following example shows this property.

```
DIM  Type as String
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the type
Type = Obj.ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This is the code page of the connection for which autECLOIA was set. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows this property.

```
DIM  CodePage as Long
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the code page
CodePage = Obj.CodePage
```

Started

This indicates whether the emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object
```

```
Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If Obj.Started = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if communications are connected
' for A.  The results are sent to a text box called
' CommConn.
If Obj.CommStarted = False Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This indicates whether the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File -> API Settings**). The value is True if the emulator is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is API enabled.
' The results are sent to a text box called Result.
```

```
If Obj.APIEnabled = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This indicates whether the emulator window is started, API-enabled, and connected. This property checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If Obj.Ready = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

NumLock

This property queries the operator information area to determine if the keyboard NumLock key is on. NumLock is a Boolean data type and is read-only. The following example shows this property.

```
DIM autECLOIA as Object
    DIM autECLConnList as Object

    Set autECLOIA = CreateObject ("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject ("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByFHandle (autECLConnList (1) .Handle)

' Check if the num lock is on
If autECLOIA.NumLock Then . . .
```

autECLOIA Methods

The following section describes the methods that are valid for autECLOIA.

void RegisterOIAEvent()
void UnregisterOIAEvent()

void SetConnectionByName (String Name)

void SetConnectionByHandle (Long Handle)

void StartCommunication()

void StopCommunication()

Boolean WaitForInputReady([optional] Variant TimeOut)

Boolean WaitForSystemAvailable([optional] Variant TimeOut)

Boolean WaitForAppAvailable([optional] Variant TimeOut)

Boolean WaitForTransition([optional] Variant Index, [optional] Variant timeout)

void CancelWaits()

RegisterOIAEvent

This method registers an object to receive notification of all OIA events.

Prototype

void RegisterOIAEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1028 for an example.

UnregisterOIAEvent

Ends OIA event processing.

Prototype

void UnregisterOIAEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1028 for an example.

SetConnectionByName

The SetConnectionByName method uses the connection name to set the connection for a newly created autECLOIA object. In Z and I Emulator for Windows this connection name is the short connection ID (character A-Z or a-z). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time.

Note: Do not call this if using the autECLOIA object in autECLSession.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection (A-Z or a-z).

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLOIA object.

```
DIM  autECLOIA as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")

' Initialize the connection
autECLOIA.SetConnectionByName("A")
' For example, see if its num lock is on
If ( autECLOIA.NumLock = True ) Then
  'your logic here...
Endif
```

SetConnectionByHandle

The SetConnectionByHandle method uses the connection handle to set the connection for a newly created autECLOIA object. In Z and I Emulator for Windows this connection handle is a Long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time.

Note: Do not call this if using the autECLOIA object in autECLSession.

```
DIM  autECLOIA as Object
DIM  autECLConnList as Object

Set autECLOIA = CreateObject("ZIEWin.autECLOIA")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLOIA.SetConnectionByHandle(autECLConnList(1).Handle)
' For example, see if its num lock is on
If ( autECLOIA.NumLock = True ) Then
  'your logic here...
Endif
```

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autELCOIA object.

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

None

```
Dim OIAObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set OIAObj = CreateObject("ZIEWin.autECLOIA")

' Initialize the session
autECLConnList.Refresh
OIAObj.SetConnectionByHandle(autECLConnList(1).Handle)

OIAObj.StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim OIAObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set OIAObj = CreateObject("ZIEWin.autECLOIA")
```

```
' Initialize the session
autECLConnList.Refresh
OIAObj.SetConnectionByHandle(autECLConnList(1).Handle)

OIAObj.StopCommunication()
```

WaitForInputReady

The WaitForInputReady method waits until the OIA of the connection associated with the autECLOIA object indicates that the connection is able to accept keyboard input.

Prototype

Boolean WaitForInputReady([optional] Variant TimeOut)

Parameters

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLOIAObj as Object

Set autECLOIAObj = CreateObject("ZIEWin.autECLOIA")
autECLOIAObj.SetConnectionByName("A")

if (autECLOIAObj.WaitForInputReady(10000)) then
msgbox "Ready for input"
else
msgbox "Timeout Occurred"
end if
```

WaitForSystemAvailable

The WaitForSystemAvailable method waits until the OIA of the connection associated with the autECLOIA object indicates that the connection is connected to a host system.

Prototype

Boolean WaitForSystemAvailable([optional] Variant TimeOut)

Parameters

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLOIAObj as Object

Set autECLOIAObj = CreateObject("ZIEWin.autECLOIA")
autECLOIAObj.SetConnectionByName("A")

if (autECLOIAObj.WaitForSystemAvailable(10000)) then
msgbox "System Available"
else
msgbox "Timeout Occurred"
end if
```

WaitForAppAvailable

The WaitForAppAvailable method waits while the OIA of the connection associated with the autECLOIA object indicates that the application is being worked with.

Prototype

Boolean WaitForAppAvailable([optional] Variant TimeOut)

Parameters

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLOIAObj as Object

Set autECLOIAObj = CreateObject("ZIEWin.autECLOIA")
autECLOIAObj.SetConnectionByName("A")

if (autECLOIAObj.WaitForAppAvailable (10000)) then
```

```
msgbox "Application is available"
else
msgbox "Timeout Occurred"
end if
```

WaitForTransition

The WaitForTransition method waits for the OIA position specified of the connection associated with the autECLOIA object to change.

Prototype

Boolean WaitForTransition([optional] Variant Index, [optional] Variant timeout)

Parameters

Variant Index

The 1 byte Hex position of the OIA to monitor. This parameter is optional. The default is 3.

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLOIAObj as Object
Dim Index

Index = 03h

Set autECLOIAObj = CreateObject("ZIEWin.autECLOIA")
autECLOIAObj.SetConnectionByName("A")

if (autECLOIAObj.WaitForTransition(Index,10000)) then
    msgbox "Position " " Index " " of the OIA Changed"
else
    msgbox "Timeout Occurred"
end if
```

CancelWaits

Cancels any currently active wait methods.

Prototype

void CancelWaits()

Parameters

None

Return Value

None

autECLOIA Events

The following events are valid for autECLOIA:

void NotifyOIAEvent()

void NotifyOIAError()

void NotifyOIAStop(Long Reason)

NotifyOIAEvent

A given OIA has occurred.

Prototype

void NotifyOIAEvent()

Parameters

None

Example

See Event Processing Example on page 1028 for an example.

NotifyOIAError

This event occurs when an error occurs in the OIA.

Prototype

void NotifyOIAError()

Parameters

None

Example

See Event Processing Example on page 1028 for an example.

NotifyOIAStop

This event occurs when event processing stops.

Prototype

void NotifyOIAStop(Long Reason)

Parameters

Long Reason

Long Reason code for the stop. Currently, this will always be 0.

Event Processing Example

The following is a short example of how to implement OIA Events

```
Option Explicit
Private WithEvents myOIA As autECLOIA 'AutOIA added as reference

sub main()
'Create Objects
Set myOIA = New AutOIA

Set myConnMgr = New AutConnMgr

myOIA.SetConnectionByName ("B") 'Monitor Session B for OIA Updates

myOIA.RegisterOIAEvent 'register for OIA Notifications

' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
call DisplayGUI()

'Clean up
myOIA.UnregisterOIAEvent

Private Sub myOIA_NotifyOIAEvent()
' do your processing here
End Sub
Private Sub myOIA_NotifyOIAError()
' do your processing here
End Sub
```

```
 'This event occurs when Communications Status Notification ends
Private Sub myOIA_NotifyOIAStop(Reason As Long)
'Do any stop processing here
End Sub
```

autECLPS Class

autECLPS performs operations on a presentation space. Its name in the registry is ZIEWin.autECLPS.

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection may be set only once. After the connection is set, any further calls to the SetConnection methods cause an exception. If you do not set the connection and try to access a property or method, an exception is also raised.

Note:

1. In the presentation space, the first row coordinate is row 1 and the first column coordinate is column 1. Therefore, the top, left position has a coordinate of row 1, column 1.
2. The autECLPS object in the autECLSession object is set by the autECLSession object.

The following is an example of how to create and set the autECLPS object in Visual Basic.

```
DIM  autECLPSObj as Object
DIM  NumRows as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
' Initialize the connection
autECLPSObj .SetConnectionByName("A")
' For example, get the number of rows in the PS
NumRows = autECLPSObj.NumRows
```

Properties

This section describes the properties of the autECLPS object.

Type	Name	Attributes
Object	autECLFieldList	Read-only
Long	NumRows	Read-only
Long	NumCols	Read-only
Long	CursorPosRow	Read-only
Long	CursorPosCol	Read-only
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only

Type	Name	Attributes
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

autECLFieldList

This is the field collection object for the connection associated with the autECLPS object. See autECLFieldList Class on page 1000 for details. The following example shows this object.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)

' Build the field list
CurPosCol = autECLPSObj.autECLFieldList.Refresh(1)
```

NumRows

This is the number of rows in the presentation space for the connection associated with the autECLPS object. NumRows is a Long data type and is read-only. The following example shows this property.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Dim  Rows as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
Rows = autECLPSObj.NumRows
```

NumCols

This is the number of columns in the presentation space for the connection associated with the autECLPS object. NumCols is a Long data type and is read-only. The following example shows this property.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Dim  Cols as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
```

```
' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
Cols = autECLPSObj.NumCols
```

CursorPosRow

This is the current row position of the cursor in the presentation space for the connection associated with the autECLPS object. CursorPosRow is a Long data type and is read-only. The following example shows this property.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Dim  CurPosRow as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
CurPosRow = autECLPSObj.CursorPosRow
```

CursorPosCol

This is the current column position of the cursor in the presentation space for the connection associated with the autECLPS object. CursorPosCol is a Long data type and is read-only. The following example shows this property.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Dim  CurPosCol as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
CurPosCol = autECLPSObj.CursorPosCol
```

Name

This is the connection name string of the connection for which autECLPS was set. Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example shows this property.

```
DIM  Name as String
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")
```

```
' Save the name
Name = Obj.Name
```

Handle

This is the handle of the connection for which the autECLPS object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time. Handle is a Long data type and is read-only. The following example shows this property.

```
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the connection handle
Hand = Obj.Handle
```

ConnType

This is the connection type for which autECLPS was set. This connection type may change over time. ConnType is a String data type and is read-only. The following example shows this property.

```
DIM  Type as String
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the type
Type = Obj.ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This is the code page of the connection for which autECLPS was set. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows this property.

```
DIM  CodePage as Long
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the code page
CodePage = Obj.CodePage
```

Started

This indicates if the connection emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If Obj.Started = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if communications are connected
' for A.  The results are sent to a text box called
' CommConn.
If Obj.CommStarted = False Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This indicates if the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File -> API Settings**). The value is True if API is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is API enabled.
' The results are sent to a text box called Result.
If Obj.APIEnabled = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This indicates whether the emulator window is started, API enabled and connected. This checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If Obj.Ready = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

autECLPS Methods

The following section describes the methods that are valid for the autECLPS object.

void RegisterPSEvent()
void RegisterKeyEvent()

```
void RegisterCommEvent()
void UnregisterPSEvent()
void UnregisterKeyEvent()
void UnregisterCommEvent()
void SetConnectionByName (String Name)
void SetConnectionByHandle (Long Handle)
void SetCursorPos(Long Row, Long Col)
void SendKeys(String text, [optional] Long row, [optional] Long col)
Boolean SearchText(String text, [optional] Long Dir, [optional] Long row,
    [optional] Long col)
String GetText([optional] Long row, [optional] Long col, [optional] Long lenToGet)
void SetText(String Text, [optional] Long Row, [optional] Long Col)
void CopyText([optional] Long Row, [optional] Long Col, [optional] Long LenToGet)
void PasteText([optional] Long Row, [optional] Long Col, [optional] Long LenToGet)
String GetTextRect(Long StartRow, Long StartCol, Long EndRow, Long EndCol )
void StartCommunication()
void StopCommunication()
void StartMacro(String MacroName)
void Wait(milliseconds as Long)
Boolean WaitForCursor(Variant Row, Variant Col, [optional]Variant TimeOut,
    [optional] Boolean bWaitForIr)
Boolean WaitWhileCursor(Variant Row, Variant Col, [optional]Variant TimeOut,
    [optional] Boolean bWaitForIr)
Boolean WaitForString(Variant WaitString, [optional] Variant Row,
    [optional] Variant Col, [optional] Variant TimeOut, [optional] Boolean bWaitForIr,
    [optional] Boolean bCaseSens)
Boolean WaitWhileString(Variant WaitString, [optional] Variant Row,
    [optional] Variant Col, [optional] Variant TimeOut, [optional] Boolean bWaitForIr,
    [optional] Boolean bCaseSens)
Boolean WaitForStringInRect(Variant WaitString, Variant sRow, Variant sCol,
    Variant eRow, Variant eCol, [optional] Variant nTimeOut,
    [optional] Boolean bWaitForIr, [optional] Boolean bCaseSens)
Boolean WaitWhileStringInRect(Variant WaitString, Variant sRow, Variant sCol,
    Variant eRow, Variant eCol, [optional] Variant nTimeOut,
    [optional] Boolean bWaitForIr, [optional] Boolean bCaseSens)
Boolean WaitForAttrib(Variant Row, Variant Col, Variant WaitData,
    [optional] Variant MaskData, [optional] Variant plane, [optional] Variant TimeOut,
    [optional] Boolean bWaitForIr)
Boolean WaitWhileAttrib(Variant Row, Variant Col, Variant WaitData,
    [optional] Variant MaskData, [optional] Variant plane,
    [optional] Variant TimeOut, [optional] Boolean bWaitForIr)
Boolean WaitForScreen(Object screenDesc, [optional] Variant TimeOut)
Boolean WaitWhileScreen(Object screenDesc, [optional] Variant TimeOut)
void CancelWaits()
```

RegisterPSEvent

This method registers an autECLPS object to receive notification of all changes to the PS of the connected session.

Prototype

void RegisterPSEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1067 for an example.

RegisterKeyEvent

Begins Keystroke Event Processing.

Prototype

void RegisterKeyEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1067 for an example.

RegisterCommEvent

This method registers an object to receive notification of all communication link connect/disconnect events.

Prototype

void RegisterCommEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1067 for an example.

UnregisterPSEvent

Ends PS Event Processing.

Prototype

void UnregisterPSEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1067 for an example.

UnregisterKeyEvent

Ends Keystroke Event Processing.

Prototype

void UnregisterKeyEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1067 for an example.

UnregisterCommEvent

Ends Communications Link Event Processing.

Prototype

void UnregisterCommEvent()

Parameters

None

Return Value

None

SetConnectionByName

This method uses the connection name to set the connection for a newly created autECLPS object. In Z and I Emulator for Windows this connection name is the short ID (character A-Z or a-z). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time.

Note: Do not call this if using the autECLPS object in autECLSession.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection (A-Z or a-z).

Return Value

None

Example

The following example shows how to set the connection for a newly created autECLPS object using the connection name.

```
DIM  autECLPSObj as Object
DIM  NumRows as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")

' Initialize the connection
autECLPSObj.SetConnectionByName("A")
' For example, get the number of rows in the PS
NumRows = autECLPSObj.NumRows
```

SetConnectionByHandle

This method uses the connection handle to set the connection for a newly created autECLPS object. In Z and I Emulator for Windows this connection handle is a Long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection "A" open at a time.

Note: Do not call this if using the autECLPS object in autECLSession.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to set the connection for a newly created autECLPS object using the connection handle.

```
DIM  autECLPSObj as Object
DIM  autECLConnList as Object
DIM  NumRows as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection with the first in the list
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
' For example, get the number of rows in the PS
NumRows = autECLPSObj.NumRows
```

SetCursorPos

The SetCursorPos method sets the position of the cursor in the presentation space for the connection associated with the autECLPS object. The position set is in row and column units.

Prototype

void SetCursorPos(Long Row, Long Col)

Parameters

Long Row

The row position of the cursor in the presentation space.

Long Col

The column position of the cursor in the presentation space.

Return Value

None

Example

The following example shows how to set the position of the cursor in the presentation space for the connection associated with the autECLPS object.

```
DIM  autECLPSObj as Object
DIM  autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection with the first in the list
autECLConnList.Refresh
```

```
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
autECLPSObj.SetCursorPos 2, 1
```

SendKeys

The SendKeys method sends a string of keys to the presentation space for the connection associated with the autECLPS object. This method allows you to send mnemonic keystrokes to the presentation space. See Sendkeys Mnemonic Keywords on page 1156 for a list of these keystrokes.

Prototype

void SendKeys(String text, [optional] Long row, [optional] Long col)

Parameters

String text

String of keys to send to the presentation space.

Long Row

Row position to send keys to the presentation space. This parameter is optional. The default is the current cursor row position. If row is specified, col must also be specified.

Long Col

Column position to send keys to the presentation space. This parameter is optional. The default is the current cursor column position. If col is specified, row must also be specified.

Return Value

None

Example

The following example shows how to use the SendKeys method to send a string of keys to the presentation space for the connection associated with the autECLPS object.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
autECLPSObj.SendKeys "HCL is a really cool company", 3, 1
```

SearchText

The SearchText method searches for the first occurrence of text in the presentation space for the connection associated with the autECLPS object. The search is case-sensitive. If text is found, the method returns a TRUE value. It returns a FALSE value if no text is found. If the optional row and column parameters are used, **row** and **col** are also returned, indicating the position of the text if it was found.

Prototype

boolean SearchText(String text, [optional] Long Dir, [optional] Long Row, [optional] Long Col)

Parameters

String text

String to search for.

Long Dir

Direction in which to search. Must either be **1** for search forward or **2** for search backward. This parameter is optional. The default is 1 for Forward.

Long Row

Row position at which to start the search in the presentation space. The row of found text is returned if the search is successful. This parameter is optional. If row is specified, col must also be specified.

Long Col

Column position at which to start the search in the presentation space. The column of found text is returned if the search is successful. This parameter is optional. If col is specified, row must also be specified.

Return Value

TRUE if text is found, FALSE if text is not found.

Example

The following example shows how to search for text in the presentation space for the connection associated with the autECLPS object.

```
Dim  autECLPSObj as Object
Dim  autECLConnList as Object
Dim  Row as Long
Dim  Col as Long
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
```

```
// Search forward in the PS from the start of the PS.  If found
// then call a hypothetical found routine, if not found, call a hypothetical

// not found routine.
row = 3
col = 1
If ( autECLPSObj.SearchText "HCL", 1, row, col) Then
  Call FoundFunc (row, col)
Else
  Call NotFoundFunc
Endif
```

GetText

The GetText method retrieves characters from the presentation space for the connection associated with the autECLPS object.

Prototype

String GetText([optional] Long Row, [optional] Long Col, [optional] Long LenToGet)

Parameters

Long Row

Row position at which to start the retrieval in the presentation space. This parameter is optional.

Long Col

Column position at which to start the retrieval in the presentation space. This parameter is optional.

Long LenToGet

Number of characters to retrieve from the presentation space. This parameter is optional. The default is the length of the array passed in as BuffLen.

Return Value

String

Text from the PS.

Example

The following example shows how to retrieve a string from the presentation space for the connection associated with the autECLPS object.

```
Dim  autECLPSObj as Object
Dim  PSText as String

' Initialize the connection
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")
```

```
PSText = autECLPSObj.GetText(2,1,50)
```

SetText

The SetText method sends a string to the presentation space for the connection associated with the autECLPS object. Although this method is similar to the SendKeys method, this method does not send mnemonic keystrokes (for example, [enter] or [pf1]).

Prototype

void SetText(String Text, [optional] Long Row, [optional] Long Col)

Parameters

String Text

Character array to send.

Long Row

The row at which to begin the retrieval from the presentation space. This parameter is optional. The default is the current cursor row position.

Long Col

The column position at which to begin the retrieval from the presentation space. This parameter is optional. The default is the current cursor column position.

Return Value

None

Example

The following example shows how to search for text in the presentation space for the connection associated with the autECLPS object.

```
Dim autECLPSObj as Object

'Initialize the connection
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")
autECLPSObj.SetText"HCL is great", 2, 1
```

CopyText

This method copies the text from a given location in presentation space of a specified length to clipboard. The length of the text copied will be the length specified, if no length is specified the text till the end of presentation space is copied. If the location is not specified, the text copied is from the current cursor position in presentation space. In case of no parameters, whole presentation space is copied to clipboard.

Prototype

void CopyText([optional] Long Row, [optional] Long Col, [optional] Long LenToGet)

Parameters

Long LenToGet

Number of characters to copy from the presentation space. This parameter is optional. The default is the length of the array passed in as BuffLen.

Long Row

The row at which to begin the copy from the presentation space. This parameter is optional. The default is the current cursor row position.

Long Col

The column position at which to begin the copy from the presentation space. This parameter is optional. The default is the current cursor column.

Return Value

None

Example

The following example shows how to copy text in the presentation space to clipboard for the connection associated with the autECLPS object.

```
Dim autECLPSObj as Object
'Initialize the connection

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")

autECLPSObj.SetConnectionByName("A")
autECLPSObj.CopyText 6, 53, 10
```

PasteText

This method pastes the text of specified length from clipboard to a given location in presentation space. The length of the text pasted is the length specified, if no length is specified the whole text in clipboard is pasted until it reaches the end of presentation space. If the location is not specified, the text is pasted at the current cursor position in presentation space. If the presentation space is field formatted and while pasting the clipboard content, when there is a tab character '\t,' the remaining paste content is moved to the next writable field.

Prototype

void PasteText([optional] Long Row, [optional] Long Col, [optional] Long LenToGet)

Parameters

Long LenToGet

Number of characters to paste from clipboard to the presentation space. This parameter is optional. The default is the length of the text in clipboard.

Long Row

The row at which to begin the paste from clipboard to the presentation space. This parameter is optional. The default is the current cursor row position.

Long Col

The column position at which to begin the paste from clipboard to the presentation space. This parameter is optional. The default is the current cursor column position.

Return Value

None

Example

The following example shows how to paste text from clipboard to presentation space the connection associated with the autECLPS object.

```
Dim autECLPSObj as Object
’Initialize the connection

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")

autECLPSObj.SetConnectionByName("A")
autECLPSObj.PasteText 8, 53, 10
```

GetTextRect

The GetTextRect method retrieves characters from a rectangular area in the presentation space for the connection associated with the autECLPS object. No wrapping takes place in the text retrieval; only the rectangular area is retrieved.

Prototype

String GetTextRect(Long StartRow, Long StartCol, Long EndRow, Long EndCol)

Parameters

Long StartRow

Row at which to begin the retrieval in the presentation space.

Long StartCol

Column at which to begin the retrieval in the presentation space.

Long EndRow

Row at which to end the retrieval in the presentation space.

Long EndCol

Column at which to end the retrieval in the presentation space.

Return Value

String

PS Text.

Example

The following example shows how to retrieve characters from a rectangular area in the presentation space for the connection associated with the autECLPS object.

```
Dim  autECLPSObj as Object
Dim  PSText String

' Initialize the connection
Set autECLPSObj = CreateObject ("ZIEWin.autELCPS")
autECLPSObj.SetConnectionByName("A")

PSText = GetTextRect(1,1,2,80)
```

SetTextRect

The SetTextRect method sets characters to a rectangular area in the presentation space for the connection associated with the autECLPS object. No wrapping takes place in the text setting; only the rectangular area is set.

Prototype

SetTextRect(String Text, Long StartRow, Long StartCol, Long EndRow, Long EndCol)

Parameters

String Text

Character array to set on presentation space.

Long StartRow

Row at which to begin the setting in the presentation space.

Long StartCol

Column at which to begin the setting in the presentation space.

Long EndRow

Row at which to end the setting in the presentation space.

Long EndCol

Column at which to end the setting in the presentation space.

Return Value

None

Example

The following example shows how to set characters to a rectangular area in the presentation space for the connection associated with the autECLPS object.

```
Dim autECLPSObj as Object
Dim PSText String

’ Initialize the connection
Set autECLPSObj = CreateObject ("ZIEWin.autELCPS")
autECLPSObj.SetConnectionByName("A")

SetTextRect "HCL is great company to collaborate with", 1, 1, 4, 8
```

If we want to use parentheses then we can use SetTextRect as

```
call SetTextRect("HCL is great company to collaborate with", 1, 1, 4, 8)
```

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim PSObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set PSObj = CreateObject("ZIEWin.autECLPS")

' Initialize the session
autECLConnList.Refresh
PSObj.SetConnectionByHandle(autECLConnList(1).Handle)

PSObj.StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim PSObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set PSObj = CreateObject("ZIEWin.autECLPS")

' Initialize the session
autECLConnList.Refresh
PSObj.SetConnectionByHandle(autECLConnList(1).Handle)
PSObj.StopCommunication()
```

StartMacro

The StartMacro method runs the Z and I Emulator for Windows macro file indicated by the MacroName parameter.

Prototype

void StartMacro(String MacroName)

Parameters

String MacroName

Name of macro file located in the Z and I Emulator for Windows user-class application data directory (specified at installation), without the file extension. This method does not support long file names.

Return Value

None

Usage Notes

You must use the short file name for the macro name. This method does not support long file names.

Example

The following example shows how to start a macro.

```
Dim PS as Object

Set PS = CreateObject("ZIEWin.autECLPS")
PS.StartMacro "mymacro"
```

Wait

The Wait method waits for the number of milliseconds specified by the milliseconds parameter

Prototype

void Wait(milliseconds as Long)

Parameters

Long milliseconds

The number of milliseconds to wait.

Return Value

None

Example

```
Dim autECLPSObj as Object

Set autECLPSObj = CreateObject ("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName ("A")

' Wait for 10 seconds
autECLPSObj.Wait(10000)
```

WaitForCursor

The WaitForCursor method waits for the cursor in the presentation space of the connection associated with the autECLPS object to be located at a specified position.

Prototype

Boolean WaitForCursor(Variant Row, Variant Col, [optional]Variant TimeOut,
    [optional] Boolean bWaitForIr)

Parameters

Variant Row

Row position of the cursor.

Variant Col

Column position of the cursor.

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col
```

```
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

Row = 20
Col = 16

if (autECLPSObj.WaitForCursor(Row,Col,10000)) then
    msgbox "Cursor is at " " Row " "," " Col
else
    msgbox "Timeout Occurred"
end if
```

WaitWhileCursor

The WaitWhileCursor method waits while the cursor in the presentation space of the connection associated with the autECLPS object is located at a specified position.

Prototype

Boolean WaitWhileCursor(Variant Row, Variant Col, [optional]Variant TimeOut,
[optional] Boolean bWaitForIr)

Parameters

Variant Row

Row position of the cursor.

Variant Col

Column position of the cursor.

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")
```

```
Row = 20
Col = 16

if (autECLPSObj.WaitWhileCursor(Row,Col,10000)) then
     msgbox "Cursor is no longer at " " Row " "," " Col
else
     msgbox "Timeout Occurred"
end if
```

WaitForString

The WaitForString method waits for the specified string to appear in the presentation space of the connection associated with the autECLPS object. If the optional Row and Column parameters are used, the string must begin at the specified position. If 0,0 are passed for Row,Col the method searches the entire PS.

Prototype

```
Boolean WaitForString(Variant WaitString, [optional] Variant Row,
    [optional] Variant Col, [optional] Variant TimeOut, [optional] Boolean bWaitForIr,
    [optional] Boolean bCaseSens)
```

Parameters

Variant WaitString

The string for which to wait.

Variant Row

Row position that the string will begin. This parameter is optional. The default is 0.

Variant Col

Column position that the string will begin. This parameter is optional. The default is 0.

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Boolean bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col, WaitString

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

WaitString = "Enter USERID"
Row = 20
Col = 16

if (autECLPSObj.WaitForString(WaitString,Row,Col,10000))  then
    msgbox WaitString " " found at " " Row " "," " Col
else
    msgbox "Timeout Occurred"
end if
```

WaitWhileString

The WaitWhileString method waits while the specified string appears in the presentation space of the connection associated with the autECLPS object. If the optional Row and Column parameters are used, the string must begin at the specified position. If 0,0 are passed for Row,Col the method searches the entire PS.

Prototype

```
Boolean WaitWhileString(Variant WaitString, [optional] Variant Row,
    [optional] Variant Col, [optional] Variant TimeOut, [optional] Boolean bWaitForIr,
    [optional] Boolean bCaseSens)
```

Parameters

Variant WaitString

The method waits while this string value exists.

Variant Row

Row position that the string will begin. This parameter is optional. The default is 0.

Variant Col

Column position that the string will begin. This parameter is optional. The default is 0.

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Boolean bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col, WaitString

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

WaitString = "Enter USERID"
Row = 20
Col = 16

if (autECLPSObj.WaitWhileString(WaitString,Row,Col,10000))  then
    msgbox WaitString " " was found at " " Row " "," " Col
else
    msgbox "Timeout Occurred"
end if
```

WaitForStringInRect

The WaitForStringInRect method waits for the specified string to appear in the presentation space of the connection associated with the autECLPS object in the specified rectangle.

Prototype

Boolean WaitForStringInRect(Variant WaitString, Variant sRow, Variant sCol,
Variant eRow, Variant eCol, [optional] Variant nTimeOut,
[optional] Boolean bWaitForIr, [optional] Boolean bCaseSens)

Parameters

Variant WaitString

The string for which to wait.

Variant sRow

Starting row position of the search rectangle.

Variant sCol

Starting column position of the search rectangle.

Variant eRow

Ending row position of the search rectangle.

Variant eCol

Ending column position of the search rectangle

Variant nTimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Boolean bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim sRow, sCol, eRow, eCol, WaitString

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

WaitString = "Enter USERID"
sRow = 20
sCol = 16
eRow = 21
eCol = 31

if (autECLPSObj.WaitForStringInRect(WaitString,sRow,sCol,eRow,eCol,10000))  then
    msgbox WaitString " " found in rectangle"
else
    msgbox "Timeout Occurred"
end if
```

WaitWhileStringInRect

The WaitWhileStringInRect method waits while the specified string appears in the presentation space of the connection associated with the autECLPS object in the specified rectangle.

Prototype

Boolean WaitWhileStringInRect(Variant WaitString, Variant sRow, Variant sCol,
Variant eRow, Variant eCol, [optional] Variant nTimeOut,
[optional] Boolean bWaitForIr, [optional] Boolean bCaseSens)

Parameters

Variant WaitString

The method waits while this string value exists.

Variant sRow

Starting row position of the search rectangle.

Variant sCol

Starting column position of the search rectangle.

Variant eRow

Ending row position of the search rectangle.

Variant eCol

Ending column position of the search rectangle.

Variant nTimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Boolean bCaseSens

If this value is True, the wait condition is verified as case-sensitive. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim sRow, sCol, eRow, eCol, WaitString

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

WaitString = "Enter USERID"
sRow = 20
```

```
sCol = 16
eRow = 21
eCol = 31

if (autECLPSObj.WaitWhileStringInRect(WaitString,sRow,sCol,eRow,eCol,10000))  then
    msgbox WaitString " " no longer in rectangle"
else
    msgbox "Timeout Occurred"
end if
```

WaitForAttrib

The WaitForAttrib method will wait until the specified Attribute value appears in the presentation space of the connection associated with the autECLPS object at the specified Row/Column position. The optional MaskData parameter can be used to control which values of the attribute you are looking for. The optional plane parameter allows you to select any of the four PS planes.

Prototype

```
Boolean WaitForAttrib(Variant Row, Variant Col, Variant WaitData,
    [optional] Variant MaskData, [optional] Variant plane, [optional] Variant TimeOut,
    [optional] Boolean bWaitForIr)
```

Parameters

Variant Row

Row position of the attribute.

Variant Col

Column position of the attribute.

Variant WaitData

The 1-byte HEX value of the attribute to wait for.

Variant MaskData

The 1-byte HEX value to use as a mask with the attribute. This parameter is optional. The default value is 0xFF.

Variant plane

The plane of the attribute to get. The plane can have the following values:

1

Text Plane

2

Color Plane

3

Field Plane (default)

4

Extended Field Plane

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col, WaitData, MaskData, plane

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

Row = 20
Col = 16
WaitData = E8h
MaskData = FFh
plane = 3

if (autECLPSObj.WaitForAttrib(Row, Col, WaitData, MaskData, plane, 10000))  then
    msgbox "Attribute " " WaitData " " found"
else
    msgbox "Timeout Occurred"
end if
```

WaitWhileAttrib

The WaitWhileAttrib method waits while the specified Attribute value appears in the presentation space of the connection associated with the autECLPS object at the specified Row/Column position. The optional MaskData parameter can be used to control which values of the attribute you are looking for. The optional plane parameter allows you to select any of the four PS planes.

Prototype

```
Boolean WaitWhileAttrib(Variant Row, Variant Col, Variant WaitData,
    [optional] Variant MaskData, [optional] Variant plane, [optional] Variant TimeOut,
    [optional] Boolean bWaitForIr)
```

Parameters

Variant Row

Row position of the attribute.

Variant Col

Column position of the attribute.

Variant WaitData

The 1 byte HEX value of the attribute to wait for.

Variant MaskData

The 1 byte HEX value to use as a mask with the attribute. This parameter is optional. The default value is 0xFF.

Variant plane

The plane of the attribute to get. The plane can have the following values:

1

Text Plane

2

Color Plane

3

Field Plane (default)

4

Extended Field Plane

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Boolean bWaitForIr

If this value is true, after meeting the wait condition the function will wait until the OIA is ready to accept input. This parameter is optional. The default is False.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim Row, Col, WaitData, MaskData, plane

Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

Row = 20
Col = 16
WaitData = E8h
MaskData = FFh
plane = 3

if (autECLPSObj.WaitWhileAttrib(Row, Col, WaitData, MaskData, plane, 10000))  then
         msgbox "Attribute " " WaitData " " No longer exists"
else
         msgbox "Timeout Occurred"
end if
```

WaitForScreen

Synchronously waits for the screen described by the autECLScreenDesc parameter to appear in the Presentation Space.

Note: The wait for OIA input flag is set on the autECLScreenDesc object, it is not passed as a parameter to the wait method.

Prototype

Boolean WaitForScreen(Object screenDesc, [optional] Variant TimeOut)

Parameters

Object screenDesc

autECLScreenDesc object that describes the screen (see autECLScreenDesc Class on page 1069).

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object
```

```
Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

autECLScreenDesObj.AddCursorPos 23, 1

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000))  then
    msgbox "Screen found"
else
    msgbox "Timeout Occurred"
end if
```

WaitWhileScreen

Synchronously waits until the screen described by the autECLScreenDesc parameter is no longer in the Presentation Space.

Note: The wait for OIA input flag is set on the autECLScreenDesc object, it is not passed as a parameter to the wait method.

Prototype

Boolean WaitWhileScreen(Object screenDesc, [optional] Variant TimeOut)

Parameters

Object ScreenDesc

autECLScreenDesc object that describes the screen (see autECLScreenDesc Class on page 1069).

Variant TimeOut

The maximum length of time in milliseconds to wait, this parameter is optional. The default is Infinite.

Return Value

The method returns True if the condition is met, or False if the Timeout value is exceeded.

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName("A")

autECLScreenDesObj.AddCursorPos 23, 1

if (autECLPSObj.WaitWhileScreen(autECLScreenDesObj, 10000))  then
```

```
    msgbox "Screen exited"
else
    msgbox "Timeout Occurred"
end if
```

CancelWaits

Cancels any currently active wait methods.

Prototype

void CancelWaits()

Parameters

None

Return Value

None

autECLPS Events

The following events are valid for autECLPS:

void NotifyPSEvent()

void NotifyKeyEvent(string KeyType, string KeyString, PassItOn as Boolean)

void NotifyCommEvent(boolean bConnected)

void NotifyPSError()

void NotifyKeyError()

void NotifyCommError()

void NotifyPSStop(Long Reason)

void NotifyKeyStop(Long Reason)

void NotifyCommStop(Long Reason)

NotifyPSEvent

A given PS has been updated.

Prototype

void NotifyPSEvent()

Parameters

None

Example

See Event Processing Example on page 1067 for an example.

NotifyKeyEvent

A keystroke event has occurred and the key information has been supplied. This function can be used to intercept keystrokes to a given PS. The Key information is passed to the event handler and can be passed on, or another action can be performed.

Note: Only one object can have keystroke event handling registered to a given PS at a time.

Prototype

void NotifyKeyEvent(string KeyType, string KeyString, PassItOn as Boolean)

Parameters

String KeyType

Type of key intercepted.

M

Mnemonic keystroke

A

ASCII

String KeyString

Intercepted keystroke

Boolean PassItOn

Flag to indicate if the keystroke should be echoed to the PS.

TRUE

Allows the keystroke to be passed on to the PS.

FALSE

Prevents the keystroke from being passed to the PS.

Example

See Event Processing Example on page 1067 for an example.

NotifyCommEvent

A given communications link as been connected or disconnected.

Prototype

void NotifyCommEvent(boolean bConnected)

Parameters

boolean bConnected

True if Communications Link is currently Connected, False otherwise.

Example

See Event Processing Example on page 1067 for an example.

NotifyPSError

This event occurs when an error occurs in event processing.

Prototype

void NotifyPSError()

Parameters

None

Example

See Event Processing Example on page 1067 for an example.

NotifyKeyError

This event occurs when an error occurs in event processing.

Prototype

void NotifyKeyError()

Parameters

None

Example

See Event Processing Example on page 1067 for an example.

NotifyCommError

This event occurs when an error occurs in event processing.

Prototype

void NotifyCommError()

Parameters

None

Example

See Event Processing Example on page 1067 for an example.

NotifyPSStop

This event occurs when event processing stops.

Prototype

void NotifyPSStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Example

See Event Processing Example on page 1067 for an example.

NotifyKeyStop

This event occurs when event processing stops.

Prototype

void NotifyKeyStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Example

See Event Processing Example on page 1067 for an example.

NotifyCommStop

This event occurs when event processing stops.

Prototype

void NotifyCommStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Event Processing Example

The following is a short example of how to implement PS Events

```
Option Explicit
Private WithEvents mPS As autECLPS 'AutPS added as reference
Private WithEvents Mkey as autECLPS

sub main()
'Create Objects
Set mPS = New autECLPS
Set mkey = New autECLPS
mPS.SetConnectionByName "A" 'Monitor Session A for PS Updates
mPS.SetConnectionByName "B" 'Intercept Keystrokes intended for Session B

mPS.RegisterPSEvent 'register for PS Updates
mPS.RegisterCommEvent ' register for Communications Link updates for session A
mkey.RegisterKeyEvent 'register for Key stroke intercept

' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
call DisplayGUI()
```

```
mPS.UnregisterPSEvent
mPS.UnregisterCommEvent
mkey.UnregisterKeyEvent
```

```

set mPS = Nothing
set mKey = Nothing
End Sub

'This sub will get called when the PS of the Session registered
'above changes
Private Sub mPS_NotifyPSEvent()
' do your processing here
End Sub

'This sub will get called when Keystrokes are entered into Session B
Private Sub mkey_NotifyKeyEvent(string KeyType, string KeyString, PassItOn as Boolean)
' do your keystroke filtering here
If (KeyType = "M") Then
'handle mnemonics here
if (KeyString = "[PF1]" then 'intercept PF1 and send PF2 instead
mkey.SendKeys "[PF2]"
set PassItOn = false
end if
end if
```

```

End Sub

'This event occurs if an error happens in PS event processing
Private Sub mPS_NotifyPSError()
'Do any error processing here
End Sub

'This event occurs when PS Event handling ends
Private Sub mPS_NotifyPSStop(Reason As Long)
'Do any stop processing here
End Sub

'This event occurs if an error happens in Keystroke processing
Private Sub mkey_NotifyKeyError()
'Do any error processing here
End Sub

'This event occurs when key stroke event handling ends
Private Sub mkey_NotifyKeyStop(Reason As Long)
'Do any stop processing here
End Sub
```

```

'This sub will get called when the Communication Link Status of the registered
'connection changes
Private Sub mPS_NotifyCommEvent()
' do your processing here
End Sub

```

```
'This event occurs if an error happens in Communications Link event processing
Private Sub mPS_NotifyCommError()
'Do any error processing here
```

```
End Sub

'This event occurs when Communications Status Notification ends
Private Sub mPS_NotifyCommStop()
'Do any stop processing here
End Sub
```

autECLScreenDesc Class

autECLScreenDesc is the class that is used to describe a screen for HCL's Host Access Class Library Screen Recognition Technology. It uses all four major planes of the presentation space to describe it (text, field, extended field, and color planes), as well as the cursor position.

Using the methods provided on this object, the programmer can set up a detailed description of what a given screen looks like in a host side application. Once an autECLScreenDesc object is created and set, it may be passed to either the synchronous WaitFor... methods provided on autECLPS, or it may be passed to autECLScreenReco, which fires an asynchronous event if the screen matching the autECLScreenDesc object appears in the PS.

autECLScreenDesc Methods

The following section describes the methods that are valid for autECLScreenDesc.

```
void AddAttrib(Variant attrib, Variant row, Variant col, Variant plane)
void AddCursorPos(Variant row, Variant col)
void AddNumFields(Variant num)
void AddNumInputFields(Variant num)
void AddOIAInhibitStatus(Variant type)
void AddString(String str, Variant row, Variant col, [optional] Boolean caseSense)
void AddStringInRect(String str,  Variant sRow,  Variant sCol,
    Variant eRow,  Variant eCol, [optional] Variant caseSense)
void Clear()
```

AddAttrib

Adds an attribute value at the given position to the screen description.

Prototype

void AddAttrib(Variant attrib, Variant row, Variant col, Variant plane)

Parameters

Variant attrib

The 1 byte HEX value of the attribute.

Variant row

Row position.

Variant col

Column position.

Variant plane

The plane of the attribute to get. The plane can have the following values:

0. All Planes
1. Text Plane
2. Color Plane
3. Field Plane
4. Extended Field Plane

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
     msgbox "Screen reached"
 else
     msgbox "Timeout Occurred"
end if
```

AddCursorPos

Sets the cursor position for the screen description to the given position.

Prototype

void AddCursorPos(Variant row, Variant col)

Parameters

Variant row

Row position.

Variant col

Column position.

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
     msgbox "Screen reached"
else
     msgbox "Timeout Occurred"
end if
```

AddNumFields

Adds the number of fields to the screen description.

Prototype

void AddNumFields(Variant num)

Parameters

Variant num

Number of fields.

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
    msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if
```

AddNumInputFields

Adds the number of fields to the screen description.

Prototype

void AddNumInputFields(Variant num)

Parameters

Variant num

Number of input fields.

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if
```

AddOIAInhibitStatus

Sets the type of OIA monitoring for the screen description.

Prototype

void AddOIAInhibitStatus(Variant type)

Parameters

Variant type

Type of OIA status. Valid values are as follows:

0. Don't Care
1. Not Inhibited

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
    msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if
```

AddString

Adds a string at the given location to the screen description.

Prototype

void AddString(String str, Variant row, Variant col, [optional] Boolean caseSense)

Parameters

String str

String to add.

Variant row

Row position.

Variant col

Column position.

Boolean caseSense

If this value is True, the strings are added as case-sensitive. This parameter is optional. The default is True.

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object
```

```
Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
    msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if
```

AddStringInRect

Adds a string in the given rectangle to the screen description.

Prototype

void AddStringInRect(String str, Variant sRow, Variant sCol,
    Variant eRow, Variant eCol, [optional] Variant caseSense)

Parameters

String str

String to add

Variant sRow

Upper left row position.

Variant sCol

Upper left column position.

Variant eRow

Lower right row position.

Variant eCol

Lower right column position.

Variant caseSense

If this value is True, the strings are added as case-sensitive. This parameter is optional. The default is True.

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
    msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if
```

Clear

Removes all description elements from the screen description.

Prototype

void Clear()

Parameters

None

Return Value

None

Example

```
Dim autECLPSObj as Object
Dim autECLScreenDescObj as Object

Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autECLPSObj = CreateObject("ZIEWin.autECLPS")
```

```
autECLPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddCursorPos 23,1
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLPSObj.WaitForScreen(autECLScreenDesObj, 10000)) then
    msgbox "Screen reached"
else
    msgbox "Timeout Occurred"
end if

autECLScreenDesObj.Clear // start over for a new screen
```

autECLScreenReco Class

The autECLScreenReco class is the engine for the Host Access Class Library screen recognition system. It contains the methods for adding and removing descriptions of screens. It also contains the logic for recognizing those screens and for asynchronously calling back to your event handler code for those screens.

Think of an object of the autECLScreenReco class as a unique recognition set. The object can have multiple autECLPS objects that it watches for screens, and multiple screens to look for, and when it sees a registered screen in any of the added autECLPS objects it will fire event handling code defined in your application.

All you need to do is set up your autECLScreenReco objects at the start of your application, and when any screen appears in any autECLPS that you want to monitor, your event code will get called by autECLScreenReco. You do absolutely no legwork in monitoring screens.

See Event Processing Example on page 1082 for an example.

autECLScreenReco Methods

The following section describes the methods that are valid for autECLScreenReco.

void AddPS(autECLPS ps)

Boolean IsMatch(autECLPS ps, AutECLScreenDesc sd)

void RegisterScreen(AutECLScreenDesc sd)

void RemovePS(autECLPS ps)

void UnregisterScreen(AutECLScreenDesc sd)

AddPS

Adds an autECLPS object to monitor to the autECLScreenReco Object.

Prototype

void AddPS(autECLPS ps)

Parameters

autECLPS ps

PS object to monitor.

Return Value

None

Example

See Event Processing Example on page 1082 for an example.

IsMatch

Allows for passing an autECLPS object and an AutECLScreenDesc object and determining if the screen description matches the current state of the PS. The screen recognition engine uses this logic, but is provided so any routine can call it.

Prototype

Boolean IsMatch(autECLPS ps, AutECLScreenDesc sd)

Parameters

autECLPS ps

autPS object to compare.

AutECLScreenDesc sd

autECLScreenDesc object to compare.

Return Value

True if the AutECLScreenDesc object matches the current screen in the PS, False otherwise.

Example

```
Dim autPSObj as Object
Dim autECLScreenDescObj as Object
```

```
Set autECLScreenDescObj = CreateObject("ZIEWin.autECLScreenDesc")
Set autPSObj = CreateObject("ZIEWin.autECLPS")
autPSObj.SetConnectionByName "A"

autECLScreenDesObj.AddCursorPos 23, 1
autECLScreenDesObj.AddAttrib E8h, 1, 1, 2
autECLScreenDesObj.AddNumFields 45
autECLScreenDesObj.AddNumInputFields 17
autECLScreenDesObj.AddOIAInhibitStatus 1
autECLScreenDesObj.AddString "LOGON", 23, 11, True
autECLScreenDesObj.AddStringInRect "PASSWORD", 23, 1, 24, 80, False

if (autECLScreenReco.IsMatch(autPSObj, autECLScreenDesObj)) then
    msgbox "matched"
else
    msgbox "no match"
end if
```

RegisterScreen

Begins monitoring all autECLPS objects added to the screen recognition object for the given screen description. If the screen appears in the PS, a NotifyRecoEvent will occur.

Prototype

void RegisterScreen(AutECLScreenDesc sd)

Parameters

AutECLScreenDesc sd

Screen description object to register.

Return Value

None

Example

See Event Processing Example on page 1082 for an example.

RemovePS

Removes the autECLPS object from screen recognition monitoring.

Prototype

void RemovePS(autECLPS ps)

Parameters

autECLPS ps

autECLPS object to remove.

Return Value

None

Example

See Event Processing Example on page 1082 for an example.

UnregisterScreen

Removes the screen description from screen recognition monitoring.

Prototype

void UnregisterScreen(AutECLScreenDesc sd)

Parameters

AutECLScreenDesc sd

Screen description object to remove.

Return Value

None

Example

See Event Processing Example on page 1082 for an example.

autECLScreenReco Events

The following events are valid for autECLScreenReco:

void NotifyRecoEvent(AutECLScreenDesc sd, autECLPS ps)
void NotifyRecoError()
void NotifyRecoStop(Long Reason)

NotifyRecoEvent

This event occurs when a Registered Screen Description appears in a PS that was added to the autECLScreenReco object.

Prototype

void NotifyRecoEvent(AutECLScreenDesc sd, autECLPS ps)

Parameters

AutECLScreenDesc sd

Screen Description object that had its criteria met.

autECLPS ps

PS object in which the match occurred.

Example

See Event Processing Example on page 1082 for an example.

NotifyRecoError

This event occurs when an error occurs in Event Processing.

Prototype

void NotifyRecoError()

Parameters

None

Example

See Event Processing Example on page 1082 for an example.

NotifyRecoStop

This event occurs when event processing stops.

Prototype

void NotifyRecoStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Event Processing Example

The following is a short example of how to implement Screen Recognition Events:

```
Dim myPS as Object
Dim myScreenDesc as Object
Dim WithEvents reco as autECLScreenReco  'autECLScreenReco added as reference

Sub Main()
  ' Create the objects
  Set reco= new autECLScreenReco
  myScreenDesc = CreateObject("ZIEWin.autECLScreenDesc")
  Set myPS = CreateObject("ZIEWin.autECLPS")
  myPS.SetConnectionByName "A"

  ' Set up the screen description
  myScreenDesc.AddCursorPos 23, 1
  myScreenDesc.AddString "LOGON"
  myScreenDesc.AddNumFields 59

  ' Add the PS to the reco object (can add multiple PS's)
  reco.addPS myPS

  ' Register the screen (can add multiple screen descriptions)
  reco.RegisterScreen myScreenDesc

  ' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
  call DisplayGUI()

  ' Clean up
  reco.UnregisterScreen myScreenDesc
  reco.RemovePS myPS
  set myPS = Nothing
  set myScreenDesc = Nothing
  set reco = Nothing
End Sub

'This sub will get called when the screen Description registered above appears in
'Session A.  If multiple PS objects or screen descriptions were added,  you can
'determine which screen and which PS via the parameters.

Sub reco_NotifyRecoEvent(autECLScreenDesc SD, autECLPS PS)
  If (reco.IsMatch(PS,myScreenDesc)) Then
    ' do your processing for your screen here
  End If
End Sub

Sub reco_NotifyRecoError
   'do your error handling here
```

```
End sub

Sub reco_NotifyRecoStop(Reason as Long)
    'Do any stop processing here
End sub
```

autECLSession Class

The autECLSession object provides general emulator related services and contains pointers to other key objects in the Host Access Class Library. Its name in the registry is ZIEWin.autECLSession.

Although the objects that autECLSession contains are capable of standing on their own, pointers to them exist in the autECLSession class. When an autECLSession object is created, autECLPS, autECLOIA, autECLXfer, autECLWindowMetrics, autECLPageSettings, and autECLPrinterSettings objects are also created. Refer to them as you would any other property.

Note:

1. The current version of this object is 1.2. There are two versions of this object; their ProgIDs in the registry are ZIEWin.autECLSession.1 and ZIEWin.autECLSession.2. The version-independent ProgID is ZIEWin.autECLSession. The ZIEWin.autECLSession.1 object does not support the properties autECLPageSettings and autECLPrinterSettings.
2. You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection can be set only once. After the connection is set, any further calls to the SetConnection methods cause an exception. If you do not set the connection and try to access an autECLSession property or method, an exception is also raised.

The following example shows how to create and set the autECLSession object in Visual Basic.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example, set the host window to minimized
SessObj.autECLWinMetrics.Minimized = True
```

Properties

This section describes the properties for the autECLSession object.

Type	Name	Attributes
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only

Type	Name	Attributes
Long	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only
Object	autECLPS	Read-only
Object	autECLOIA	Read-only
Object	autECLXfer	Read-only
Object	autECLWinMetrics	Read-only
Object	autECLPageSettings	Read-only
Object	autECLPrinterSettings	Read-only

Name

This property is the connection name string of the connection for which autECLSession was set. Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example shows this property.

```
DIM  Name as String
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")

' Save the name
Name = SessObj.Name
```

Handle

This is the handle of the connection for which the autECLSession object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time. Handle is a Long data type and is read-only. The following example shows this property.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")

' Save the session handle
Hand = SessObj.Handle
```

ConnType

This is the connection type for which autECLXfer was set. This type may change over time. ConnType is a String data type and is read-only. The following example shows this property.

```
DIM  Type as String
DIM  SessObj as Object

Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' Save the type
Type = SessObj.ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This is the code page of the connection for which autECLXfer was set. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows this property.

```
DIM  CodePage as Long
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' Save the code page
CodePage = SessObj.CodePage
```

Started

This indicates whether the emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  SessObj as Object

Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If SessObj.Started = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  SessObj as Object

Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")

' This code segment checks to see if communications are connected
' for session A.  The results are sent to a text box called
' CommConn.
If SessObj.CommStarted = False Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This indicates whether the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File -> API Settings**). The value is True if the emulator is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  SessObj as Object

Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")

' This code segment checks to see if A is API enabled.
' The results are sent to a text box called Result.
If SessObj.APIEnabled = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This indicates whether the emulator window is started, API-enabled, and connected. This property checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  SessObj as Object

Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If SessObj.Ready = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

autECLPS object

The autECLPS object allows you to access the methods contained in the ZIEWin.autECLPS class. See autECLPS Class on page 1029 for more information. The following example shows this object.

```
DIM  SessObj as Object
DIM  PSSize as Long
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example, get the PS size
PSSize = SessObj.autECLPS.GetSize()
```

autECLOIA object

The autECLOIA object allows you to access the methods contained in the ZIEWin.autECLOIA class. See autECLOIA Class on page 1011 for more information. The following example shows this object.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example, set the host window to minimized
If (SessObj.autECLOIA.Katakana) Then
 'whatever
Endif
```

autECLXfer object

The autECLXfer object allows you to access the methods contained in the ZIEWin.autECLXfer class. See autECLXfer Class on page 1112 for more information. The following example shows this object.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example
SessObj.Xfer.Sendfile "c:\temp\filename.txt",
       "filename text a0",
       "CRLF ASCII"
```

autECLWinMetrics object

The autECLWinMetrics object allows you to access the methods contained in the ZIEWin.autECLWinMetrics class. See autECLWinMetrics Class on page 1096 for more information. The following example shows this object.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example, set the host window to minimized
SessObj.autECLWinMetrics.Minimized = True
```

autECLPageSettings object

The autECLPageSettings object enables you to access the methods contained in the ZIEWin.autECLPageSettings class. See autECLPageSettings Class on page 1126 for more information.

The following example shows the autECLPageSettings object.

```
DIM SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")
'Initialize the session
SessObj.SetConnectionByName("A")

'For example, set the FaceName
SessObj.autECLPageSettings.FaceName = "Courier New"
```

The autECLPageSettings object is also supported in VBSCRIPT. The following example shows how to use VBSCRIPT.

```
sub test_()
  autECLSession.SetConnectionByName(ThisSessionName)
  autECLSession.autECLPageSettings.FaceName="Courier"
 end sub
```

autECLPrinterSettings object

The autECLPrinterSettings object enables you to access the methods contained in the ZIEWin.autECLPrinterSettings class. See autECLPageSettings Class on page 1126 for more information.

The following example shows the autECLPageSettings object.

```
DIM SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")
' Initialize the session
SessObj.SetConnectionByName("A")

'For example, set the Windows default printer
SessObj.autECLPrinterSettings.SetWinDefaultPrinter
```

The autECLPrinterSettings object is also supported in VBSCRIPT. The following example shows how to use VBSCRIPT.

```
sub test_()
  autECLSession.SetConnectionByName(ThisSessionName)
  autECLSession.autECLPrinterSettings.SetWinDefaultPrinter
end sub
```

autECLSession Methods

The following section describes the methods that are valid for the autECLSession object.

void RegisterSessionEvent(Long updateType)

void RegisterCommEvent()

void UnregisterSessionEvent()

void UnregisterCommEvent()

void SetConnectionByName (String Name)

void SetConnectionByHandle (Long Handle)

void StartCommunication()

void StopCommunication()

RegisterSessionEvent

This method registers an autECLSession object to receive notification of specified Session events.

Note: This method is not supported and is not recommended for use.

Prototype

void RegisterSessionEvent(Long updateType)

Parameters

Long updateType

Type of update to monitor for:

1. PS Update
2. OIA Update
3. PS or OIA Update

Return Value

None

Example

See Event Processing Example on page 1095 for an example.

RegisterCommEvent

This method registers an object to receive notification of all communication link connect/disconnect events.

Prototype

void RegisterCommEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1095 for an example.

UnregisterSessionEvent

Ends Session Event processing.

Note: This method is not supported and is not recommended for use.

Prototype

void UnregisterSessionEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1095 for an example.

UnregisterCommEvent

Ends Communications Link Event processing.

Prototype

void UnregisterCommEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1095 for an example.

SetConnectionByName

This method uses the connection name to set the connection for a newly created autECLSession object. In Z and I Emulator for Windows this connection name is the short ID (character A-Z or a-z). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection (A-Z or a-z).

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLSession object.

```
DIM  SessObj as Object
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
SessObj.SetConnectionByName("A")
' For example, set the host window to minimized
SessObj.autECLWinMetrics.Minimized = True
```

SetConnectionByHandle

This method uses the connection handle to set the connection for a newly created autECLSession object. In Z and I Emulator for Windows this connection handle is a long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autECLSession object.

```
Dim  SessObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
autECLConnList.Refresh
autECLPSObj.SetConnectionByHandle(autECLConnList(1).Handle)
```

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim SessObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
autECLConnList.Refresh
SessObj.SetConnectionByHandle(autECLConnList(1).Handle)

SessObj.StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim SessObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set SessObj = CreateObject("ZIEWin.autECLSession")

' Initialize the session
autECLConnList.Refresh
SessObj.SetConnectionByHandle(autECLConnList(1).Handle)

SessObj.StopCommunication()
```

autECLSession Events

The following events are valid for autECLSession:

void NotifyCommEvent(boolean bConnected)
void NotifyCommError()
void NotifyCommStop(Long Reason)

NotifyCommEvent

A given communications link has been connected or disconnected.

Prototype

void NotifyCommEvent(boolean bConnected)

Parameters

boolean bConnected

TRUE if communications link is currently connected. FALSE otherwise.

Example

See Event Processing Example on page 1095 for an example.

NotifyCommError

This event occurs when an error occurs in event processing.

Prototype

void NotifyCommError()

Parameters

None

Example

See Event Processing Example on page 1095 for an example.

NotifyCommStop

This event occurs when event processing stops.

Prototype

void NotifyCommStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently, this will always be 0.

Event Processing Example

The following is a short example of how to implement Session Events

```
Option Explicit
Private WithEvents mSess As autECLSession   'AutSess added as reference
```

```
sub main()
   'Create Objects
   Set mSess = New autECLSession
   mSess.SetConnectionByName "A"
    mSess.RegisterCommEvent          'register for communication link notifications
   ' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
   call DisplayGUI()
   mSess.UnregisterCommEvent
   set mSess = Nothing
End Sub

'This sub will get called when the Communication Link Status of the registered
'connection changes
Private Sub mSess_NotifyCommEvent()
    ' do your processing here
End Sub

'This event occurs if an error happens in Communications Link event processing
Private Sub mSess_NotifyCommError()
   'Do any error processing here
End Sub

'This event occurs when Communications Status Notification ends
Private Sub mSess_NotifyCommStop()
   'Do any stop processing here
End Sub
```

autECLWinMetrics Class

The autECLWinMetrics object performs operations on an emulator window. It allows you to perform window rectangle and position manipulation (for example, SetWindowRect, Ypos and Width), as well as window state manipulation (for example, Visible or Restored). Its name in the registry is ZIEWin.autECLWinMetrics.

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection may be set only once. After the connection is set, any further calls to the set connection methods cause an exception. If you do not set the connection and try to access a property or method, an exception is also raised.

Note: The autECLSession object in the autECL object is set by the autECL object.

The following example shows how to create and set the autECLWinMetrics object in Visual Basic.

```
DIM  autECLWinObj as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
autECLWinObj.SetConnectionByName("A")
' For example, set the host window to minimized
autECLWinObj.Minimized = True
```

Properties

This section describes the properties for the autECLWinMetrics object.

Type	Name	Attributes
String	WindowTitle	Read/Write
Long	Xpos	Read/Write
Long	Ypos	Read/Write
Long	Width	Read/Write
Long	Height	Read/Write
Boolean	Visible	Read/Write
Boolean	Active	Read/Write
Boolean	Minimized	Read/Write
Boolean	Maximized	Read/Write
Boolean	Restored	Read/Write
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

WindowTitle

This is the title that is currently in the title bar for the connection associated with the autECLWinMetrics object. This property may be both changed and retrieved. WindowTitle is a String data type and is read/write enabled. The following example shows this process. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  WinTitle as String
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

WinTitle = autECLWinObj.WindowTitle 'get the window title

' or...

autECLWinObj.WindowTitle = "Flibberdeejibbet" 'set the window title
```

Usage Notes

If WindowTitle is set to blank, the window title of the connection is restored to its original setting.

Xpos

This is the *x* position of the upper left point of the emulator window rectangle. This property may be both changed and retrieved. Xpos is a Long data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  x as Long
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

x = autECLWinObj.Xpos 'get the x position

' or...

autECLWinObj.Xpos = 6081 'set the x position
```

Ypos

This is the *y* position of the upper left point of the emulator window rectangle. This property may be both changed and retrieved. Ypos is a Long data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  y as Long
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

y = autECLWinObj.Ypos 'get the y position

' or...

autECLWinObj.Ypos = 6081 'set the y position
```

Width

This is the width of the emulator window rectangle. This property may be both changed and retrieved. Width is a Long data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  cx as Long
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

cx = autECLWinObj.Width 'get the width

' or...

autECLWinObj.Width = 6081 'set the width
```

Height

This is the height of the emulator window rectangle. This property may be both changed and retrieved. Height is a Long data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  cy as Long
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

cy = autECLWinObj.Height 'get the height

' or...

autECLWinObj.Height = 6081 'set the height
```

Visible

This is the visibility state of the emulator window. This property may be both changed and retrieved. Visible is a Boolean data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
```

```
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

' Set to Visible if not, and vice versa
If ( autECLWinObj.Visible) Then
 autECLWinObj.Visible = False
Else
 autECLWinObj.Visible = True
End If
```

Active

This is the focus state of the emulator window. This property may be both changed and retrieved. Active is a Boolean data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

' Set to Active if not, and vice versa
If ( autECLWinObj.Active) Then
 autECLWinObj.Active = False
Else
 autECLWinObj.Active = True
End If
```

Minimized

This is the minimize state of the emulator window. This property may be both changed and retrieved. Minimized is a Boolean data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

' Set to minimized if not, if minimized set to maximized
If ( autECLWinObj.Minimized) Then
```

```
 autECLWinObj.Maximized = True
Else
 autECLWinObj.Minimized = True
End If
```

Maximized

This is the maximize state of the emulator window. This property may be both changed and retrieved. Maximized is a Boolean data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)

' Set to maximized if not, if maximized set to minimized
If ( autECLWinObj.Maximized) Then
 autECLWinObj.Minimized = False
Else
 autECLWinObj.Maximized = True
End If
```

Restored

This is the restore state of the emulator window. Restored is a Boolean data type and is read/write enabled. However, if the connection you are attached to is an inplace, embedded object, this property is read-only. The following example shows this property.

```
Dim  autECLWinObj as Object
Dim  SessList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set SessList = CreateObject("ZIEWin.autECLConnList")

' Initialize the session
SessList.Refresh
autECLWinObj.SetSessionByHandle(SessList(1).Handle)

' Set to restored if not, if restored set to minimized
If ( autECLWinObj.Restored) Then
 autECLWinObj.Minimized = False
Else
 autECLWinObj.Restored = True
End If
```

Name

This property is the connection name string of the connection for which autECLWinMetrics was set. Currently, Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example shows this property.

```
DIM  Name as String
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the name
Name = Obj.Name
```

Handle

This is the handle of the connection for which the autECLWinMetrics object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time. Handle is a Long data type and is read-only. The following example shows this property.

```
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the handle
Hand = Obj.Handle
```

ConnType

This is the connection type for which autECLWinMetrics was set. This type may change over time. ConnType is a String data type and is read-only. The following example shows this property.

```
DIM  Type as String
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the type
Type = Obj.ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display

String Returned	Meaning
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This is the code page of the connection for which autECLWinMetrics was set. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows this property.

```
DIM  CodePage as Long
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the code page
CodePage = Obj.CodePage
```

Started

This indicates whether the emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWinZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If Obj.Started = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object
```

```
Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if communications are connected
' for A.  The results are sent to a text box called
' CommConn.
If Obj.CommStarted = False Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This indicates whether the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File ->API Settings**). The value is True if the emulator is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is API enabled.
' The results are sent to a text box called Result.
If Obj.APIEnabled = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This indicates whether the emulator window is started, API enabled, and connected. This property checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
```

```
If Obj.Ready = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

autECLWinMetrics Methods

The following section describes the methods that are valid for the autECLWinMetrics object.

void RegisterCommEvent()

void UnregisterCommEvent()

void SetConnectionByName(String Name)

void SetConnectionByHandle(Long Handle)

void GetWindowRect(Variant Left, Variant Top, Variant Right, Variant Bottom)

void SetWindowRect(Long Left, Long Top, Long Right, Long Bottom)

void StartCommunication()

void StopCommunication()

RegisterCommEvent

This method registers an object to receive notification of all communication link connect/disconnect events.

Prototype

void RegisterCommEvent()

Parameters

None

Return Value

None

Example

See Event Processing Example on page 1111 for an example.

UnregisterCommEvent

Ends Communications Link Event Processing.

Prototype

void UnregisterCommEvent()

Parameters

None

Return Value

None

SetConnectionByName

This method uses the connection name to set the connection for a newly created autECLWinMetrics object. In Z and I Emulator for Windows this connection name is the short ID (character A-Z or a-z). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection "A" open at a time.

Note: Do not call this if using the autECLWinMetrics object in autECLSession.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection (A-Z or a-z).

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLWinMetrics object.

```
DIM  autECLWinObj as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the connection
autECLWinObj.SetConnectionByName("A")
' For example, set the host window to minimized
autECLWinObj.Minimized = True
```

SetConnectionByHandle

This method uses the connection handle to set the connection for a newly created autECLWinMetrics object. In Z and I Emulator for Windows this connection handle is a long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time.

Note: Do not call this if using the autECLWinMetrics object in autECLSession.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autECLWinMetrics object.

```
DIM  autECLWinObj as Object
DIM  ConnList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)
' For example, set the host window to minimized
autECLWinObj.Minimized = True
```

GetWindowRect

The GetWindowRect method returns the bounding points of the emulator window rectangle.

Prototype

void GetWindowRect(Variant Left, Variant Top, Variant Right, Variant Bottom)

Parameters

Variant Left, Top, Right, Bottom

Bounding points of the emulator window.

Return Value

None

Example

The following example shows how to return the bounding points of the emulator window rectangle.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Dim  left
Dim  top
Dim  right
Dim  bottom
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)
autECLWinObj.GetWindowRect left, top, right, bottom
```

SetWindowRect

The SetWindowRect method sets the bounding points of the emulator window rectangle.

Prototype

void SetWindowRect(Long Left, Long Top, Long Right, Long Bottom)

Parameters

Long Left, Top, Right, Bottom

Bounding points of the emulator window.

Return Value

None

Example

The following example shows how to set the bounding points of the emulator window rectangle.

```
Dim  autECLWinObj as Object
Dim  ConnList as Object
Set autECLWinObj = CreateObject("ZIEWin.autECLWinMetrics")
Set ConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection
ConnList.Refresh
autECLWinObj.SetConnectionByHandle(ConnList(1).Handle)
autECLWinObj.SetWindowRect 0, 0, 6081, 6081
```

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim WinObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set WinObj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the session
autECLConnList.Refresh
WinObj.SetConnectionByHandle(autECLConnList(1).Handle)

WinObj.StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim WinObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set WinObj = CreateObject("ZIEWin.autECLWinMetrics")

' Initialize the session
autECLConnList.Refresh
WinObj.SetConnectionByHandle(autECLConnList(1).Handle)

WinObj.StopCommunication()
```

autECL WinMetrics Events

The following events are valid for autECL WinMetrics:

void NotifyCommEvent(boolean bConnected)

NotifyCommError()

void NotifyCommStop(Long Reason)

NotifyCommEvent

A given communications link as been connected or disconnected.

Prototype

void NotifyCommEvent(boolean bConnected)

Parameters

boolean bConnected

True if Communications Link is currently Connected, False otherwise.

Example

See Event Processing Example on page 1111 for an example.

NotifyCommError

This event occurs when an error occurs in Event Processing.

Prototype

NotifyCommError()

Parameters

None

Example

See Event Processing Example on page 1111 for an example.

NotifyCommStop

This event occurs when event processing stops.

Prototype

void NotifyCommStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Event Processing Example

The following is a short example of how to implement WinMetrics Events.

```
Option Explicit
Private WithEvents mWmet As autECLWinMetrics   'AutWinMetrics added as reference

sub main()
   'Create Objects
   Set mWmet = New autECLWinMetrics
   mWmet.SetConnectionByName "A"  'Monitor Session A

   mWmet.RegisterCommEvent ' register for Communications Link updates for session A
   ' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
   call DisplayGUI()

   mWmet.UnregisterCommEvent

   set mWmet = Nothing
End Sub

'This sub will get called when the Communication Link Status of the registered
'connection changes
Private Sub mWmet _NotifyCommEvent()
    ' do your processing here
End Sub

'This event occurs if an error happens in Communications Link event processing
Private Sub mWmet _NotifyCommError()
   'Do any error processing here
End Sub

'This event occurs when Communications Status Notification ends
Private Sub mWmet _NotifyCommStop()
   'Do any stop processing here
End Sub
```

autECLXfer Class

The autECLXfer object provides file transfer services. Its name in the registry is ZIEWin.autECLXfer.

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection may be set only once. After the connection is set, any further calls to the SetConnection methods cause an exception. If you do not set the connection and try to access an autECLXfer property or method, an exception is also raised. The following shows how to create and set the autECLXfer object in Visual Basic.

```
DIM  XferObj as Object

Set XferObj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
XferObj.SetConnectionByName("A")
```

Properties

This section describes the properties for the autECLXfer object.

Type	Name	Attribute
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only
Boolean	Started	Read-only

Type	Name	Attribute
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

Name

This property is the connection name string of the connection for which autECLXfer was set. Z and I Emulator for Windows only returns the short character ID (A-Z or a-z) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection “A” open at a time. Name is a String data type and is read-only. The following example shows this property.

```
DIM  Name as String
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the name
Name = Obj.Name
```

Handle

This is the handle of the connection for which the autECLXfer object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time. Handle is a Long data type and is read-only. The following example shows this property.

```
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' Save the handle
Hand = Obj.Handle
```

ConnType

This is the connection type for which autECLXfer was set. This type may change over time. Conntype is a String data type and is read-only. The following example shows this property.

```
DIM  Type as String
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
```

```
Obj.SetConnectionByName("A")
' Save the type
Type = Obj.ConnType
```

Connection types for the ConnType property are:

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

CodePage

This is the code page of the connection for which autECLXfer was set. This code page may change over time. CodePage is a Long data type and is read-only. The following example shows this property.

```
DIM  CodePage as Long
DIM  Obj as Object
Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")
' Save the code page
CodePage = Obj.CodePage
```

Started

This indicates whether the emulator window is started. The value is True if the window is open; otherwise, it is False. Started is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If Obj.Started = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

CommStarted

This indicates the status of the connection to the host. The value is True if the host is connected; otherwise, it is False. CommStarted is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if communications are connected
' for A.  The results are sent to a text box called
' CommConn.
If Obj.CommStarted = False Then
    CommConn.Text = "No"
Else
    CommConn.Text = "Yes"
End If
```

APIEnabled

This indicates whether the emulator is API-enabled. A connection may be enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, choose **File -> API Settings**). The value is True if the emulator is enabled; otherwise, it is False. APIEnabled is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is API enabled.
' The results are sent to a text box called Result.
If Obj.APIEnabled = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

Ready

This indicates whether the emulator window is started, API enabled, and connected. This property checks for all three properties. The value is True if the emulator is ready; otherwise, it is False. Ready is a Boolean data type and is read-only. The following example shows this property.

```
DIM  Hand as Long
DIM  Obj as Object

Set Obj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
Obj.SetConnectionByName("A")

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If Obj.Ready = False Then
  Result.Text = "No"
Else
  Result.Text = "Yes"
End If
```

autECLXfer Methods

The following section describes the methods that are valid for the autECLXfer object.

void RegisterCommEvent()

void UnregisterCommEvent()

void SetConnectionByName(String Name)

void SetConnectionByHandle(Long Handle)

void SendFile(String PCFile, String HostFile, String Options)

void ReceiveFile(String PCFile, String HostFile, String Options)

void StartCommunication()

void StopCommunication()

RegisterCommEvent

This method registers an object to receive notification of all communication link connect/disconnect events.

Prototype

void RegisterCommEvent()

Parameters

None

Return Value

None

Example

SeeEvent Processing Example on page 1124 for an example.

UnregisterCommEvent

Ends Communications Link Event Processing.

Prototype

void UnregisterCommEvent()

Parameters

None

Return Value

None

SetConnectionByName

The SetConnectionByName method uses the connection name to set the connection for a newly created autECLXfer object. In Z and I Emulator for Windows this connection name is the short ID (character A-Z or a-z). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection "A" open at a time.

Note: Do not call this if using the autECLXfer object in autECLSession.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection (A-Z or a-z).

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLXfer object.

```
DIM  XferObj as Object

Set XferObj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
XferObj.SetConnectionByName("A")
```

SetConnectionByHandle

The SetConnectionByHandle method uses the connection handle to set the connection for a newly created autECLXfer object. In Z and I Emulator for Windows this connection handle is a Long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection “A” open at a time.

Note: Do not call this if using the autECLXfer object in autECLSession.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autECLXfer object.

```
DIM  XferObj as Object
DIM  autECLConnList as Object

Set XferObj = CreateObject("ZIEWin.autECLXfer")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection with the first connection in the list
autECLConnList.Refresh
XferObj.SetConnectionByHandle(autECLConnList(1).Handle)
```

SendFile

The SendFile method sends a file from the workstation to the host for the connection associated with the autECLXfer object.

Prototype

void SendFile(String PCFile, String HostFile, String Options)

Parameters

String PCFile

Name of the file on the workstation.

String HostFile

Name of the file on the host.

String Options

Host-dependent transfer options. See Usage Notes on page 1119 for more information.

Return Value

None

Usage Notes

File transfer options are host-dependent. The following is a list of some of the valid host options for a VM/CMS host.

ASCII
CRLF
APPEND
LRECL
RECFM
CLEAR/NOCLEAR
PROGRESS
QUIET

Refer to *Emulator Programming* for the list of supported hosts and associated file transfer options.

Example

The following example shows how to send a file from the workstation to the host for the connection associated with the autECLXfer object.

```
DIM  XferObj as Object
DIM  autECLConnList as Object
DIM  NumRows as Long
```

```
Set XferObj = CreateObject("ZIEWin.autECLXfer")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection with the first connection in the autECLConnList
autECLConnList.Refresh
XferObj.SetConnectionByHandle(autECLConnList(1).Handle)

' For example, send the file to VM
XferObj.SendFile "c:\windows\temp\thefile.txt",
                     "THEFILE TEXT A0",
                     "CRLF ASCII"
```

ReceiveFile

The ReceiveFile method receives a file from the host to the workstation for the connection associated with the autECLXfer object.

Prototype

void ReceiveFile(String PCFile, String HostFile, String Options)

Parameters

String PCFile

Name of the file on the workstation.

String HostFile

Name of the file on the host.

String Options

Host-dependent transfer options. See Usage Notes on page 1120 for more information.

Return Value

None

Usage Notes

File transfer options are host-dependent. The following is a list of some of the valid host options for a VM/CMS host:

ASCII
CRLF
APPEND
LRECL
RECFM
CLEAR/NOCLEAR

PROGRESS

QUIET

Refer to *Emulator Programming* manual for the list of supported hosts and associated file transfer options.

Example

The following example shows how to receive a file from the host and send it to the workstation for the connection associated with the autECLXfer object.

```
DIM  XferObj as Object
DIM  autECLConnList as Object
DIM  NumRows as Long

Set XferObj = CreateObject("ZIEWin.autECLXfer")
Set autECLConnList = CreateObject("ZIEWin.autECLConnList")

' Initialize the connection with the first connection in the list
autECLConnList.Refresh
XferObj.SetConnectionByHandle(autECLConnList(1).Handle)
' For example, send the file to VM
XferObj.ReceiveFile "c:\windows\temp\thefile.txt",
                      "THEFILE TEXT A0",
                      "CRLF ASCII"
```

StartCommunication

The StartCommunication collection element method connects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Connect**.

Prototype

void StartCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim XObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set XObj = CreateObject("ZIEWin.autECLXfer")
```

```
' Initialize the session
autECLConnList.Refresh
XObj.SetConnectionByHandle(autECLConnList(1).Handle)

XObj.StartCommunication()
```

StopCommunication

The StopCommunication collection element method disconnects the ZIEWin emulator to the host data stream. This has the same effect as going to the ZIEWin emulator **Communication** menu and choosing **Disconnect**.

Prototype

void StopCommunication()

Parameters

None

Return Value

None

Example

The following example shows how to connect a ZIEWin emulator session to the host.

```
Dim XObj as Object
Dim autECLConnList as Object

Set autECLConnList = CreateObject("ZIEWin.autECLConnList")
Set XObj = CreateObject("ZIEWin.autECLXfer")

' Initialize the session
autECLConnList.Refresh
XObj.SetConnectionByHandle(autECLConnList(1).Handle)

SessObj.StopCommunication()
```

autECLXfer Events

The following events are valid for autECLXfer:

void NotifyCommEvent(boolean bConnected)
NotifyCommError()
void NotifyCommStop(Long Reason)

NotifyCommEvent

A given communications link as been connected or disconnected.

Prototype

void NotifyCommEvent(boolean bConnected)

Parameters

boolean bConnected

True if Communications Link is currently Connected, False otherwise.

Example

See Event Processing Example on page 1124 for an example.

NotifyCommError

This event occurs when an error occurs in event processing.

Prototype

NotifyCommError()

Parameters

None

Example

See Event Processing Example on page 1124 for an example.

NotifyCommStop

This event occurs when event processing stops.

Prototype

void NotifyCommStop(Long Reason)

Parameters

Long Reason

Reason code for the stop. Currently this will always be 0.

Event Processing Example

The following is a short example of how to implement Xfer Events

```
Option Explicit
Private WithEvents mXfer As autECLXfer 'AutXfer added as reference

sub main()
'Create Objects
Set mXfer = New autECLXfer
mXfer.SetConnectionByName "A" 'Monitor Session A

mXfer.RegisterCommEvent ' register for Communications Link updates for session A

' Display your form or whatever here  (this should be a blocking call, otherwise sub just ends
call DisplayGUI()

mXfer.UnregisterCommEvent

set mXfer= Nothing
End Sub

'This sub will get called when the Communication Link Status of the registered
'connection changes
Private Sub mXfer _NotifyCommEvent()
' do your processing here
End Sub

'This event occurs if an error happens in Communications Link event processing
Private Sub mXfer _NotifyCommError()
'Do any error processing here
End Sub

'This event occurs when Communications Status Notification ends
Private Sub mXfer _NotifyCommStop()
'Do any stop processing here
End Sub
```

autSystem Class

The autSystem class is used to perform utility operations that are not present in some programming languages.

autSystem Methods

The following section describes the methods that are valid for the autSystem object.

Long Shell(VARIANT ExeName, VARIANT Parameters, VARIANT WindowStyle)

String Inputnd()

Shell

The shell function runs any executable file.

Prototype

Long Shell(VARIANT ExeName, VARIANT Parameters, VARIANT WindowStyle)

Parameters

VARIANT ExeName

Full path and file name of the executable file.

VARIANT Parameters

Any parameters to pass to the executable file. This parameter is optional.

VARIANT WindowStyle

The initial window style to show as executable. This parameter is optional and can have the following values:

1. Normal with focus (default)
2. Minimized with focus
3. Maximized
4. Normal without focus
5. Minimized without focus

Return Value

The method returns the Process ID if it is successful, or zero if it fails.

Example

```
Example autSystem - Shell()

'This example starts notepad with the file c:\test.txt loaded
dim ProcessID
dim SysObj as object

set SysObj = CreateObject("ZIEWin.autSystem")
ProcessID = SysObj.shell "Notepad.exe","C:\test.txt"
If ProcessID > 0 then
 Msgbox "Notepad Started, ProcessID = " + ProcessID
Else
 Msgbox "Notepad not started"
End if
```

Inputnd

The Inputnd method displays a popup input box to the user with a no-display text box so that when the user types in data only asterisks(*) are displayed.

Prototype

String Inputnd()

Parameters

None

Return Value

The characters typed into the input box, or "" if nothing was typed in.

Example

```
DIM strPassWord
dim SysObj as Object
dim PSObj as Object

set SysObj = CreateObject("ZIEWin.autSystem")
set PSObj = CreateObject("ZIEWin.autPS")

PSObj.SetConnectionByName("A")
'Prompt user for password
strPassWord = SysObj.Inputnd()
PSObj.SetText(strPasssWord)
DIM  XferObj as Object

Set XferObj = CreateObject("ZIEWin.autECLXfer")

' Initialize the connection
XferObj.SetConnectionByName("A")
```

autECLPageSettings Class

The autECLPageSettings object controls the page settings of a Z and I Emulator for Windows connection. Its name in the registry is ZIEWin.autECLPageSettings. This automation object can also be used in VB scripts.

The read-only property autECLPageSettings has been added to the autECLSession object. See autECLSession Class on page 1083 for information about how to use this property.

Note: The autECLPageSettings object in the autECLSession object is set by the autECLSession object.

The following example shows how to create and set the autECLPageSettings object in Visual Basic.

```
DIM PgSet as Object
Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
PgSet.SetConnectionByName("A")
```

Usage Notes

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection can be set only once. After the connection is set, any further calls to the set connection methods cause an exception. If you do not set the connection and try to access a property or method, an exception is raised.

The properties CPI, LPI, and FontSize are dependent on the property FaceName. Therefore, if CPI, LPI, or FontSize are set before the FaceName is set, and if they are not valid for the new FaceName, different CPI, LPI, or FontSize values might be reconfigured in the connection. You should set the FaceName before setting the CPI, LPI, or FontSize. Otherwise, every time you set FaceName, query CPI, LPI, and FontSize and make sure that they have the desired values.

Restrictions

The connection associated with each method must be in a particular state for the method to succeed. If the restrictions are not met, an appropriate exception is raised.

The following restrictions must be satisfied while any property or method of the autECLPageSettings object is invoked.

- The host session should not be printing when this API is invoked.
- The **File → Page Setup** and **File → Printer Setup** dialogs must not be in use.
- The associated connection must not be in PDT mode.

Additional restrictions might apply for each specific property or method.

Connection types

The following connection types are valid for the methods in the autECLPageSettings class:

- 3270 display
- 3270 printer
- 5250 display
- VT (ASCII)

If a property or method is accessed or called on an unsupported connection, an exception is raised. Use the ConnType property to determine the connection type.

Properties

This section describes the properties for the autECLPageSettings object.

Type	Name	Attributes
Long	CPI	Read/Write
Boolean	FontCPI	Read-only
Long	LPI	Read/Write
Boolean	FontLPI	Read-only
String	FaceName	Read/Write
Long	FontSize	Read/Write
Long	MaxLinesPerPage	Read/Write
Long	MaxCharsPerLine	Read/Write
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Long	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

CPI

This property determines the number of characters printed per inch. This is a Long data type and is read/write enabled.

Set this property to the predefined constant pcFontCPI to select the Font CPI in Page Settings or set it to some specific CPI value. If this property is queried when FontCPI is configured in the connection, the actual CPI value is returned and the constant pcFontCPI is not returned.

To determine whether FontCPI is set in the connection, use the property FontCPI.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim CPI as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

CPI = PgSet.CPI ' get the CPI value
' or...
PgSet.CPI = pcFontCPI 'set the connection to use Font CPI.
```

FontCPI

This determines whether Font CPI is set in the connection. FontCPI is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

'check if Font CPI is set
If PgSet.FontCPI Then
 ...
```

LPI

This property determines the number of lines printed per inch. This is a Long data type and is read/write enabled. Set it to the predefined constant pcFontLPI to select the Font LPI in Page Settings or set it to some specific LPI value. If this property is queried when FontLPI is configured in the connection, the actual LPI value is returned and the constant pcFontLPI is not returned. To determine whether FontLPI is set in the connection, use the property FontLPI.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim LPI as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

LPI = PgSet.LPI ' get the LPI value
' or...
PgSet.LPI = pcFontLPI 'set the connection to use Font LPI.
```

FontLPI

This property determines whether Font LPI is set in the connection. FontLPI is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object
```

```
Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

'check if Font LPI is set
If PgSet.FontLPI Then
 ...
```

FaceName

This is the Font Face Name of the Page Settings of the connection. FaceName is a String data type and is read/write enabled.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim FaceName as String

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)
FaceName = PgSet.FaceName ' get the FaceName
' or...
PgSet.FaceName = "Courier New" 'set the FaceName
```

MaxLinesPerPage

This property is the maximum number of lines that can be printed per page. This is also called *maximum print lines* or MPL. Valid values are in the range 1–255. This is a Long data type and is read/write enabled.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim MPL as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

MPL = PgSet.MaxLinesPerPage ' get the MaxLinesPerPage
' or...
PgSet.MaxLinesPerPage = 20 'set the MaxLinesPerPage
```

MaxCharsPerLine

This property is the maximum number of characters that can be printed per line. This is also called *maximum print position* or MPP. Valid values are in the range 1–255. This is a Long data type and is read/write enabled.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim MPP as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

MPP = PgSet.MaxCharsPerLine ' get the MaxCharsPerLine
' or...
PgSet.MaxCharsPerLine = 80 'set the MaxCharsPerLine
```

Name

This property is the connection name string of the connection for which autECLPageSettings was set. Z and I Emulator for Windows returns only the short character ID (a single alphabetical character from **A** to **Z**) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection **A** open at a time. Name is a String data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object
DIM Name as String

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

Name = PgSet.Name 'Save the name
```

Handle

This property is the handle of the connection for which the autECLPageSettings object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection **A** open at a time. Handle is a Long data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim Hand as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

Hand = PgSet.Handle ' save the handle
```

ConnType

This property is the connection type for which autECLPageSettings was set. This type might change over time. ConnType is a String data type and is read-only.

String Value	Connection Type
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim Type as String

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

Type = PgSet.ConnType ' save the type
```

CodePage

This property is the connection type for which autECLPageSettings was set. This type might change over time. ConnType is a String data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Dim CodePage as Long

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

CodePage = PgSet.CodePage ' save the codepage
```

Started

This property indicates whether the emulator window is started. The value is TRUE if the window is open; otherwise, it is FALSE. Started is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If PgSet.Started = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

CommStarted

This property indicates the status of the connection to the host. The value is TRUE if the host is connected; otherwise, it is FALSE. CommStarted is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
```

```
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if communications are connected
' for A. The results are sent to a text box called
' CommConn.
If PgSet.CommStarted = False Then
 CommConn.Text = "No"
Else
 CommConn.Text = "Yes"
End If
```

APIEnabled

This property indicates whether the emulator is API-enabled. A connection can be API-enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, click **Settings → API**). The value is TRUE if the emulator is API-enabled; otherwise, it is FALSE. APIEnabled is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if A is API-enabled.
' The results are sent to a text box called Result.
If PgSet.APIEnabled = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

Ready

This property indicates whether the emulator window is started, API-enabled, and connected. This property checks for all three properties. The value is TRUE if the emulator is ready; otherwise, it is FALSE. Ready is a Boolean data type and is read-only.

Example

```
Dim PgSet as Object
Dim ConnList as Object
Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
```

```
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)
' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If PgSet.Ready = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

autECLPageSettings Methods

The following section describes the methods that are valid for the autECLPageSettings object.

void RestoreTextDefaults()

void SetConnectionByName (String Name)

void SetConnectionByHandle (Long Handle)

RestoreTextDefaults

The RestoreTextDefaults method restores the system default values of the **Text** property page in the Page Setup dialog of the connection. This is equivalent to pressing the **Default** button on the **Text** property page of the Page Setup Dialog of the connection.

Prototype

void RestoreTextDefaults()

Parameters

None

Return Value

None

Example

The following example shows the RestoreTextDefaults method.

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)
PgSet.RestoreTextDefaults 'Restores Text Default Settings
```

SetConnectionByName

The SetConnectionByName method uses the connection name to set the connection for a newly created autECLPageSettings object. This connection name is the short connection ID (a single alphabetical character from **A** to **Z**). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection **A** open at a time.

Note: Do not call this method if you are using the autECLPageSettings object contained in the autECLSession object.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection. Valid values are A–Z.

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLPageSettings object.

```
Dim PgSet as Object
Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
' Initialize the connection
PgSet.SetConnectionByName("A")
' For example, see if Font CPI is set
If PgSet.FontCPI Then
'your logic here...
End If
```

SetConnectionByHandle

The SetConnectionByHandle method uses the connection handle to set the connection for a newly created autECLPageSettings object. In Z and I Emulator for Windows, this connection handle is a Long integer. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection **A** open at a time.

Note: Do not call this method if you are using the autECLPageSettings object contained in the autECLSession object.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

Long integer value of the connection to be set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autECLPageSettings object.

```
Dim PgSet as Object
Dim ConnList as Object

Set PgSet = CreateObject("ZIEWin.autECLPageSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PgSet.SetConnectionByHandle(ConnList(1).Handle)

' For example, see if Font CPI is set
If PgSet.FontCPI Then
'your logic here...
End If
```

autECLPrinterSettings Class

The autECLPrinterSettings object controls the Printer Settings of a Z and I Emulator for Windows connection. Its name in the registry is `ZIEWin.autECLPrinterSettings`. This automation object can also be used in VB scripts.

The read-only property **autECLPrinterSettings** has been added to the autECLSession object. See autECLSession Class on page 1083 for information about how to use this property.

Note: The autECLPrinterSettings object in the autECLSession object is set by the autECLSession object.

The following example shows how to create and set the autECLPrinterSettings object in Visual Basic.

```
DIM PrSet as Object
Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
PrSet.SetConnectionByName("A")
```

Usage Notes

You must initially set the connection for the object you create. Use SetConnectionByName or SetConnectionByHandle to initialize your object. The connection can be set only once. After the connection is set, any further calls to the set connection methods cause an exception. If you do not set the connection and try to access a property or method, an exception is raised.

The properties CPI, LPI, and FontSize are dependent on the property FaceName. Therefore, if CPI, LPI, or FontSize are set before the FaceName is set, and if they are not valid for the new FaceName, different CPI, LPI, or FontSize values might be reconfigured in the connection. You should set the FaceName before setting the CPI, LPI, or FontSize. Otherwise, every time you set FaceName, query CPI, LPI, and FontSize and make sure that they have the desired values.

Restrictions

The connection associated with each method must be in a particular state for the method to succeed. If the restrictions are not met, an appropriate exception is raised.

The following restrictions must be satisfied while any property or method of the autECLPageSettings object is invoked.

- The host session should not be printing when this API is invoked.
- The **File → Page Setup** and **File → Printer Setup** dialogs should not be in use.

Additional restrictions might apply for each specific property or method.

Properties

This section describes the properties for the autECLPrinterSettings object.

Type	Name	Attributes
Boolean	PDTMode	Read-only
String	PDTFile	Read-only
Long	PrintMode	Read-only
String	Printer	Read-only
String	PrtToDskAppendFile	Read-only
String	PrtToDskSeparateFile	Read-only

Type	Name	Attributes
Boolean	PrompDialogOption	Read/Write
String	Name	Read-only
Long	Handle	Read-only
String	ConnType	Read-only
Boolean	CodePage	Read-only
Boolean	Started	Read-only
Boolean	CommStarted	Read-only
Boolean	APIEnabled	Read-only
Boolean	Ready	Read-only

PDTMode

This property determines whether the connection is in PDT mode or not. PDTMode is a Boolean data type and is read/write enabled.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

'check if in PDT mode.
If PrSet.PDTMode Then
 ...
```

PDTFile

This property is the PDT file configured in the connection. This property gives a null string if the PDT file is not configured in the connection. Otherwise, this property gives the fully qualified path name of the PDT file. PDTFile is a String data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
```

```
PrSet.SetConnectionByHandle(ConnList(1).Handle)

If PrSet. PDTFile = vbNullString Then ' get the
  ...
Else
  ...
```

PrintMode

This property indicates the print mode of the connection. PrintMode is a Long data type and is read-only. This property returns one of the following four enumerated values:

Value	Name of the enum constant	Description
1	pcPrtToDskAppend	**Print to Disk-Append** mode. This means the **Print to Disk → Append** option is selected in the **Printer** listbox in the Printer Setup dialog of the connection.
2	pcPrtToDskSeparate	**Print to Disk-Separate** mode. This means the **Print to Disk → Separate** option is selected in the **Printer** listbox in the Printer Setup dialog of the connection.
3	pcSpecificPrinter	**Specific Printer** mode. This means one of the printers is selected in the **Printer** listbox in the Printer Setup dialog of the connection and the **Use Windows Default Printer** checkbox is clear.
4	pcWinDefaultPrinter	**Windows® Default Printer** mode. This means that the **Use Windows Default Printer** checkbox is selected.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

If PrSet.PrintMode = pcPrtToDskAppend Then
  ...
ElseIf PrSet.PrintMode = pcPrtToDskSeparate Then
  ...
ElseIf PrSet.PrintMode = pcSpecificPrinter Then
  ...
ElseIf PrSet.PrintMode = pcWinDefaultPrinter Then
  ...
```

Printer

This property is the name of the printer. It contains one of the following:

- The name of the specific printer if the PrintMode of the connection is pcSpecificPrinter.
- The name of the Windows default printer if the PrintMode of the connection is pcWinDefaultPrinter.
- A null string if a printer is not configured in the connection or if the PrintMode of the connection is pcPrtToDskAppend or pcPrtToDskSeparate.

Printer is a String data type and is read-only.

The value must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `HP LaserJet 4050 Series PCL 6 on LPT1`

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim Printer as String

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

Printer = PrSet.Printer ' get the Printer Name
```

PrtToDskAppendFile

This property is the name of the file set for the **Print to Disk-Append** mode. This file is called the *Print to Disk-Append* file. This property contains one of the following:

- The fully qualified path name of the **Print to Disk-Append** file of the connection.
- A null string if the **Print to Disk-Append** file is not configured in the connection.

PrtToDskAppendFile is a String data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim DskAppFile as String

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

DskAppFile = PrSet. PrtToDskAppendFile ' get the Disk append file.
```

PrtToDskSeparateFile

This property is the name of the file set for the **Print to Disk-Separate** mode. This file is called the ***Print to Disk-Separate*** file. This property contains one of the following:

- The fully qualified path name of the **Print to Disk-Separate** file of the connection.
- A null string if the **Print to Disk-Separate** file is not configured in the connection.

PrtToDskSeparateFile is a String data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim DskSepFile as String

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

DskSepFile = PrSet. PrtToDskSeparateFile ' get the Disk separate file.
```

PromptDialogOption

This property indicates whether the option to show the Printer Setup dialog before printing is set or not. PromptDialogOption is a Boolean data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim PromptDialog as Boolean

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

PromptDialog = PrSet.PromptDialogOption ' get the Prompt Dialog option
```

```
' or...
PrSet.PromptDialogOption = True 'set the Prompt Dialog option
```

Name

This property is the connection name string of the connection for which autECLPrinterSettings was set. Z and I Emulator for Windows returns only the short character ID (a single alphabetical character from **A** to **Z**) in the string. There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection **A** open at a time. Name is a String data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
DIM Name as String

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

Name = PrSet.Name 'Save the name
```

Handle

This property is the handle of the connection for which the autECLPrinterSettings object was set. There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection **A** open at a time. Handle is a Long data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim Hand as Long

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

Hand = PrSet.Handle ' save the handle
```

ConnType

This property is the connection type for which autECLPrinterSettings was set. This type might change over time. ConnType is a String data type and is read-only.

String Value	Connection Type
DISP3270	3270 display
DISP5250	5250 display
PRNT3270	3270 printer
PRNT5250	5250 printer
ASCII	VT emulation

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim Type as String

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

Type = PrSet.ConnType ' save the type
```

CodePage

This property is the code page of the connection for which autECLPrinterSettings was set. This code page might change over time. CodePage is a Long data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object
Dim CodePage as Long

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

CodePage = PrSet.CodePage ' save the codepage
```

Started

This property indicates whether the emulator window is started. The value is TRUE if the window is open; otherwise, it is FALSE. Started is a Boolean data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject(".autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if A is started.
' The results are sent to a text box called Result.
If PrSet.Started = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

CommStarted

This property indicates the status of the connection to the host. The value is TRUE if the host is connected; otherwise, it is FALSE. CommStarted is a Boolean data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if communications are connected
' for A. The results are sent to a text box called
' CommConn.
If PrSet.CommStarted = False Then
 CommConn.Text = "No"
Else
 CommConn.Text = "Yes"
End If
```

APIEnabled

This property indicates whether the emulator is API-enabled. A connection is API-enabled or disabled depending on the state of its API settings (in a Z and I Emulator for Windows window, click **Settings → API**).

The value is TRUE if the emulator is API-enabled; otherwise, it is FALSE. APIEnabled is a Boolean data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if A is API-enabled.
' The results are sent to a text box called Result.
If PrSet.APIEnabled = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

Ready

This property indicates whether the emulator window is started, API-enabled, and connected. This property checks for all three properties. The value is TRUE if the emulator is ready; otherwise, it is FALSE. Ready is a Boolean data type and is read-only.

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

' This code segment checks to see if A is ready.
' The results are sent to a text box called Result.
If PrSet.Ready = False Then
 Result.Text = "No"
Else
 Result.Text = "Yes"
End If
```

autECLPrinterSettings Methods

The following sections describe the methods that are valid for the autECLPrinterSettings object.

void SetPDTMode(Boolean bPDTMode, [optional] String PDTFile)

void SetPrtToDskAppend([optional] String FileName)

```
void SetPrtToDskSeparate([optional] String FileName)
void SetSpecificPrinter(String Printer)
void SetWinDefaultPrinter()
void SetConnectionByName (String Name)
void SetConnectionByHandle (Long Handle)
```

SetPDTMode

The SetPDTMode method sets the connection in PDT mode with the given PDT file or sets the connection in non-PDT mode (also called *GDI mode*).

Restriction

If this method is called with bPDTMode set to FALSE, PrintMode of the associated connection must already be set to SpecificPrinter or WinDefaultPrinter.

Prototype

void SetPDTMode(Boolean bPDTMode, [optional] String PDTFile)

Parameters

Boolean bPDTMode

Possible values are as follows:

- **TRUE** to set the connection to PDT mode.
- **FALSE** to set the connection to non-PDT mode (GDI mode).

String PDTFile

This optional parameter contains the PDT file name.

This parameter is used only if bPDTMode is TRUE. If this parameter is not specified and bPDTMode is set to TRUE, the PDT file configured in the connection is used. If there is no PDT file already configured in the connection, this method fails with an exception.

This parameter ignored if bPDTMode is FALSE.

Possible values are as follows:

- File name without path

  PDTFile in the PDFPDT subfolder in the Z and I Emulator for Windows installation path is used.
- Fully qualified path name of the file

  If PDTFile does not exist, this method fails with an exception.

Return Value

None

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

PrSet.SetPDTMode(True, "epson.pdt") 'Set PDT mode
PrSet.SetPDTMode(False) 'Set non-PDT mode (also called GDI mode)
```

SetPrtToDskAppend

This method sets the PrintMode of the connection to **Print to Disk-Append** mode. It also sets the appropriate file for this mode.

Note:

1. The folder where this file is to be set must have write access. Otherwise, this method fails with an exception.
2. The associated connection must be in PDT mode.

Prototype

void SetPrtToDskAppend([optional] String FileName)

Parameters

String FileName

This optional parameter contains the name of the **Print to Disk-Append** file.

If the file exists, it is used. Otherwise, it is created when printing is complete.

Possible values are as follows:

- File name, without the path

  The user-class application data directory path will be used to locate the file.

- Fully qualified path name of the file

  The directory must exist in the path, or the method will fail with an exception. It is not necessary that the file exist in the path.

If this parameter is not specified, the file configured for this PrintMode in the connection is used. If there is no file already configured in the connection, this method fails with an exception.

Return Value

None

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

'If PDTMode, set PrintMode to pcPrtToDskAppend
If PrSet.PDTMode Then
 PrSet.SetPrtToDskAppend("dskapp.txt")
```

SetPrtToDskSeparate

This method sets the PrintMode of the connection to **Print to Disk-Separate** mode. It also sets the appropriate file for this mode.

Note:

1. The folder where this file is to be set must have write access. Otherwise, this method fails with an exception.
2. The associated connection must be in PDT mode.

Prototype

void SetPrtToDskSeparate([optional] String FileName)

Parameters

String FileName

This optional parameter contains the name of the **Print to Disk-Separate** file.

If this parameter is not specified, the file configured for this PrintMode in the connection is used.

Possible values are:

- **NULL** (default)

  The file that is currently configured for this PrintMode in the connection is used. If there is no file already configured in the connection, the method fails with an exception.
- File name, without the path

  The user-class application data directory path will be used to locate the file.
- Fully qualified path name of the file

  The directory must exist in the path, or the method will fail with an exception. It is not necessary that the file exist in the path.

Note: The file name must not contain an extension. If it contains an extension, the method fails with an exception.

Return Value

None

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

'If PDTMode, set PrintMode to pcPrtToDskSeparate
If PrSet.PDTMode Then
 PrSet.SetPrtToDskSeparate("dsksep")
```

SetSpecificPrinter

This method sets the PrintMode of the connection to Specific Printer mode with the printer specified by the Printer parameter.

Prototype

void SetSpecificPrinter(String Printer)

Parameters

String Printer

Contains the name of the printer. If the printer does not exist, this method fails with an exception.

The value must have the following format:

```
<Printer name> on <Port Name>
```

For example:

- `HP LaserJet 4050 Series PCL 6 on LPT1`

Return Value

None

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

'Set PrintMode to pcSpecificPrinter
PrSet. SetSpecificPrinter("HCL InfoPrint 40 PS on Network Port")
```

SetWinDefaultPrinter

This method sets the PrintMode of the connection to Windows Default Printer mode (the connection will use the Windows default printer). If no Windows default printer is configured, this method fails with an exception.

Prototype

void SetWinDefaultPrinter()

Parameters

None

Return Value

None

Example

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

'Set PrintMode to pcWinDefaultPrinter
PrSet. SetWinDefaultPrinter
```

SetConnectionByName

The SetConnectionByName method uses the connection name to set the connection for a newly created autECLPrinterSettings object. In Z and I Emulator for Windows, this connection name is the short connection ID (a single alphabetical character from **A** to **Z**). There can be only one Z and I Emulator for Windows connection open with a given name. For example, there can be only one connection **A** open at a time.

Note: Do not call this if you are using the autECLPrinterSettings object contained in the autECLSession object.

Prototype

void SetConnectionByName(String Name)

Parameters

String Name

One-character string short name of the connection. Valid values are A–Z.

Return Value

None

Example

The following example shows how to use the connection name to set the connection for a newly created autECLPrinterSettings object.

```
Dim PrSet as Object
Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
' Initialize the connection
PrSet.SetConnectionByName("A")
' For example, see if PDTMode
If PrSet.PDTMode Then
```

```
'your logic here...
End If
```

SetConnectionByHandle

The SetConnectionByHandle method uses the connection handle to set the connection for a newly created autECLPrinterSettings object. In Z and I Emulator for Windows, this connection handle is a Long integer.

There can be only one Z and I Emulator for Windows connection open with a given handle. For example, there can be only one connection **A** open at a time.

Note: Do not call this method if you are using the autECLPrinterSettings object contained in the autECLSession object.

Prototype

void SetConnectionByHandle(Long Handle)

Parameters

Long Handle

The Long integer value of the connection to set for the object.

Return Value

None

Example

The following example shows how to use the connection handle to set the connection for a newly created autECLPrinterSettings object.

```
Dim PrSet as Object
Dim ConnList as Object

Set PrSet = CreateObject("ZIEWin.autECLPrinterSettings")
Set ConnList = CreateObject("ZIEWin.autECLConnList")
' Initialize the connection
ConnList.Refresh
PrSet.SetConnectionByHandle(ConnList(1).Handle)

' For example, see if PDTMode
If PrSet.PDTMode Then
'your logic here...
End If
```

Support For Primary Interop Assemblies for Automation Objects

Automation objects exposed by HCL Z and I Emulator for Windows can be used by applications written in any language that targets the .NET framework. Managed .NET applications can program Z and I Emulator for Windows by using the Primary Interop Assemblies (PIA) that wrap the automation objects. Interop Assemblies are the mechanism with which managed (.NET) applications use COM-compliant objects. Interop Assemblies contain binding and metadata information, which enables the .NET framework (CLR) to load or marshall COM objects and wrap them for .NET applications. The PIA contains the official description of the COM types as defined by the publisher of those COM types. The PIA is always digitally signed by the publisher of the original COM type.

There are two ways a .NET application can reference an assembly.

- If it is a simple application or the only application that uses the assembly, Microsoft recommends that the assembly be copied in the same directory as the application.
- If multiple applications are referencing the assembly, you can install them in the Global Assembly Cache (GAC) and have all the solutions reference the assembly in the GAC.

The model for programming the types exposed by Interop Assemblies is very similar to COM. The methods, properties, and events exposed by the COM object can be accessed by any .NET language, using the syntax of the language. A sample application (ECLSamps.net) written in C# is provided in the \samples directory in the Z and I Emulator for Windows installation image. The sample demonstrates the simple usage of various Interop Assembly types.

For Visual Basic 6.0, projects that use Z and I Emulator for Windows automation objects and have been migrated to Visual Basic .NET using the conversion assistant wizard, you only need to replace the references that the conversion assistant wizard implicitly generates with the corresponding Z and I Emulator for Windows Interop references (from the \Interops directory) and recompile. The way to replace the references is to delete all the references generated by the conversion assistant and use Visual Studio .NET to add the .NET interop references. If you have registered the assemblies in the GAC and want to use them, add the references and set the Copy Local property for the Z and I Emulator for Windows Interop references to **False**.

The PIAs for the Z and I Emulator for Windows emulator automation objects are installed in the \Interops directory in the Z and I Emulator for Windows installation image. If the Z and I Emulator for Windows product installer detects that the .NET framework is present, it gives you the additional option to register the types in the GAC. While installing the assemblies in the GAC, the PIAs will also be placed in the registry, under the registry key of the corresponding type library.

Table 93: Primary Interop Assemblies for Z and I Emulator for Windows Automation Objects on page 1154 lists the PIAs supplied for the Z and I Emulator for Windows automation objects

Table 93. Primary Interop Assemblies for Z and I Emulator for Windows Automation Objects

Automation Object	Interop Assembly Dependency
autECLConnList	Interop.AutConnListTypeLibrary.dll
autECLConnMgr	Interop.AutConnMgrTypeLibrary.dll

Table 93. Primary Interop Assemblies for Z and I Emulator for Windows Automation Objects (continued)

Automation Object	Interop Assembly Dependency
autECLConnList	Interop.AutPSTypeLibrary.dll
autECLOIA	Interop.AutOIATypeLibrary.dll
autECLPS	Interop.AutPSTypeLibrary.dll
autECLScreenDesc	Interop.AutScreenDescTypeLibrary-.dll
autECLScreenReco	Interop.AutScreenRecoTypeLibrary-.dll
autECLSession	Interop.AutSessTypeLibrary.dll
autECLPageSettings	Interop.AutSettingsTypeLibrary.dll
autECLPrinterSettings	Interop.AutSettingsTypeLibrary.dll
autECLWinMetrics	Interop.AutWinMetricsTypeLibrary.dll
autECLXfer	Interop.AutXferTypeLibrary.dll
autSystem	Interop.AutSystemTypeLibrary.dll

Host Access Class Library for Java

The Host Access Class Library (HACL) Java classes expose the Z and I Emulator for Windows HACL functions to the Java programming environment. This allows the creation of Java applets and applications that utilize the functions provided in the HACL classes.

The HACL Java HTML files can be found in the Docs_Admin_Aids zip folder delivered along with Z and I Emulator for Windows product documentation in the following path : *ZIEWin_3.0_Docs_Admin_Aids.zip\publications\en_US\doc \hacl* directory.

Troubleshooting

You can use the following self-help information resources and tools to help you troubleshoot problems:

- Refer to the release information for your product for known issues, workaround, and troubleshooting information.
- Check if a download or fix is available to resolve your problem.
- Search the available knowledge bases to see if the resolution to your problem is already documented.
- If you still need help, contact HCL Software Support and report your problem.

HCL Z and I Emulator for Windows .NET Interop assemblies fail to trigger session OIA notifications

Problem

.NET applications which register for OIA event notifications are not notified of these events. Also several methods of corresponding COM type library are not shown by Visual studio's intellisense feature.

Cause

.NET Interop assemblies are derived from corresponding COM type library using the tool TlbImp.exe shipped with Microsoft SDK. The Type Library Importer converts the type definitions found within a COM type library into equivalent definitions in a common language runtime assembly. The runtime marshaler however cannot marshal all the data types. Therefore some COM type library definitions are not found in the resulting common language runtime assembly.

Resolution

This is a limitation of TlbImp.exe.

Sendkeys Mnemonic Keywords

Table 94: Mnemonic Keywords for the Sendkeys Method on page 1156 contains the mnemonic keywords for the Sendkeys method.

Table 94. Mnemonic Keywords for the Sendkeys Method

Keyword	Description
[backtab]	Back tab
[clear]	Clear screen
[delete]	Delete
[enter]	Enter
[eraseeof]	Erase end of field
[help]	Help
[insert]	Insert
[jump]	Jump
[left]	Left
[newline]	New line
[space]	Space
[print]	Print
[reset]	Reset
[tab]	Tab
[up]	Up
[Down]	Down
[capslock]	CapsLock

Table 94. Mnemonic Keywords for the Sendkeys Method (continued)

Keyword	Description
[right]	Right
[home]	Home
[pf1]	PF2
[pf2]	PF2
[pf3]	PF3
[pf4]	PF4
[pf5]	PF5
[pf6]	PF6
[pf7]	PF7
[pf8]	PF8
[pf9]	PF9
[pf10]	PF10
[pf11]	PF11
[pf12]	PF12
[pf13]	PF13
[pf14]	PF14
[pf15]	PF15
[pf16]	PF16
[pf17]	PF17
[pf18]	PF18
[pf19]	PF19
[pf20]	PF20
[pf21]	PF21
[pf22]	PF22
[pf23]	PF23
[pf24]	PF24
[eof]	End of file
[scrlock]	Scroll Lock
[numlock]	Num Lock
[pageup]	Page Up
[pagedn]	Page Down
[pa1]	PA 1
[pa2]	PA 2
[pa3]	PA 3
[test]	Test
[worddel]	Word Delete
[fldext]	Field Exit

Table 94. Mnemonic Keywords for the Sendkeys Method (continued)

Keyword	Description
[erinp]	Erase Input
[sysreq]	System Request
[instog]	Insert Toggle
[crsel]	Cursor Select
[fastleft]	Cursor Left Fast
[attn]	Attention
[devcance]	Device Cancel
[printps]	Print Presentation Space
[fastup]	Cursor Up Fast
[fastdown]	Cursor Down Fast
[hex]	Hex
[fastright]	Cursor Right Fast
[revvideo]	Reverse Video
[underscr]	Underscore
[rstvideo]	Reset Reverse Video
[red]	Red
[pink]	Pink
[green]	Green
[yellow]	Yellow
[blue]	Blue
[turq]	Turquoise
[white]	White
[rstcolor]	Reset Host Color
[printpc]	Print (PC)
[wordright]	Forward Word Tab
[wordleft]	Backward Word Tab
[field-]	Field -
[field+]	Field +
[rcdbacksp]	Record Backspace
[printhost]	Print Presentation Space on Host
[dup]	Dup
[fieldmark]	Field Mark
[dispsosi]	Display SO/SI
[gensosi]	Generate SO/SI
[dispattr]	Display Attribute
[fwdchar]	Forward Character
[splitbar]	Split Vertical Bar

Table 94. Mnemonic Keywords for the Sendkeys Method (continued)

Keyword	Description
[altcsr]	Alternate Cursor
[backspace]	Backspace
[null]	Null

ECL Planes — Format and Content

This appendix describes the format and contents of the different data planes in the ECL presentation space model. Each plane represents a distinct aspect of the host presentation space, such as its character contents, color specifications, field attributes, and so on. The ECL::GetScreen methods and others return data from the different presentation space planes.

Each plane contains one byte per host presentation space character position. Each plane is described in the following sections in terms of its logical contents and data format. The plane types are enumerated in the ECLPS.HPP header file.

TextPlane

The text plane represents the visible characters of the presentation space. Non-display fields are shown in the text plane. The byte value of each element of the text plane corresponds to the ASCII value of the displayed character. The text plane does not contain any binary zero (null) character values. Any null characters in the presentation space (such as null-padded input fields) are represented as ASCII blank (0x20) characters.

FieldPlane

The field plane represents the field positions and their attributes in the presentation space. This plane is meaningful only for field-formatted presentation spaces. (For example, VT connections are not formatted).

This plane is a sparse-array of field attribute values. All values in this plane are binary zero except for where field attribute characters are present in the presentation space. At those positions, the values are the attributes of the field which starts at that location. The length of a field is the linear distance between the field attribute position and the next field attribute in the presentation space, not including the attribute position itself.

The value of the field attribute positions are as shown in the following tables.

Note: Attribute values are different for different types of connections.

Table 95. 3270 Field Attributes

Bit Position (0 is least significant bit)	Meaning
7	Always "1"
6	Always "1"
5	**0** Unprotected **1** Protected
4	**0** Alphanumeric data **1** Numeric data only
3, 2	**0, 0** Normal intensity, not pen detectable **0, 1** Normal intensity, pen detectable **1, 0** High intensity, pen detectable **1, 1** Nondisplay, not pen detectable
1	Reserved
0	**0** Field has not been modified **1** Unprotected field has been modified

Table 96. 5250 Field Attributes

Bit Position (0 is least significant bit)	Meaning
7	Always "1"
6	**0** Nondisplay

Table 96. 5250 Field Attributes (continued)

Bit Position (0 is least significant bit)	Meaning
	1 Display
5	**0** Unprotected **1** Protected
4	**0** Normal intensity **1** High intensity
3, 2, 1	**0, 0, 0** Alphanumeric data **0, 0, 1** Alpha only **0, 1, 0** Numeric shift **0, 1, 1** Numeric data plus numeric specials **1, 0, 1** Numeric only **1, 1, 0** Magnetic stripe reading device data only **1, 1, 1** Signed numeric only
0	**0** Field has not been modified **1** Unprotected field has been modified

Table 97: Mask Values on page 1162 defines the various mask values:

Table 97. Mask Values

Mnemonic	Mask	Description
FATTR_MDT	0x01	Modified field
FATTR_PEN_MASK	0x0C	Pen detectable field
FATTR_BRIGHT	0x08	Intensified field
FATTR_DISPLAY	0x0C	Visible field
FATTR_ALPHA	0x10	Alphanumeric field
FATTR_NUMERIC	0x10	Numeric only field
FATTR_PROTECTED	0x20	Protected field
FATTR_PRESENT	0x80	Field attribute present
FATTR_52_BRIGHT	0x10	5250 intensified field
FATTR_52_DISP	0x40	5250 visible field

ColorPlane

The color plane contains color information for each character of the presentation space. The foreground and background color of each character is represented as it is specified in the host data stream. The colors in the color plane are not modified by any color display mapping of the emulator window. Each byte of the color plane contains the following color information.

Table 98. Color Plane Information

Bit Position (0 is least significant bit)	Meaning
7 - 4	**Background character color** **0x0** Blank **0x1** Blue **0x2** Green **0x3** Cyan **0x4** Red **0x5** Magenta

Table 98. Color Plane Information (continued)

Bit Position (0 is least significant bit)	Meaning
	0x6 Brown (3270), Yellow (5250) **0x7** White
3-0	**Foreground character color** **0x0** Blank **0x1** Blue **0x2** Green **0x3** Cyan **0x4** Red **0x5** Magenta **0x6** Brown (3270), Yellow (5250) **0x7** White (normal intensity) **0x8** Gray **0x9** Light blue **0xA** Light green **0xB** Light cyan

Table 98. Color Plane Information (continued)

Bit Position (0 is least significant bit)	Meaning
	0xC Light red **0xD** Light magenta **0xE** Yellow **0xF** White (high intensity)

ExfieldPlane

This plane contains extended character attribute data.

This plane is a sparse-array of extended character attribute values. All values in the array are binary zero except for character in the presentation space for which the host has specified extended character attributes. The meaning of the extended character attribute values are as follows.

Table 99. 3270 Extended Character Attributes

Bit Position (0 is least significant bit)	Meaning
7, 6	**Character highlighting** **0, 0** Normal **0, 1** Blink **1, 0** Reverse video **1, 1** Underline
5, 4, 3	**Character color** **0, 0, 0** Default

Table 99. 3270 Extended Character Attributes (continued)

Bit Position (0 is least significant bit)	Meaning
	0, 0, 1 Blue **0, 1, 0** Red **0, 1, 1** Pink **1, 0, 0** Green **1, 0, 1** Turquoise **1, 1, 0** Yellow **1, 1, 1** White
2, 1	**Character attribute** **00** Default **11** Double byte character
0	Reserved

Table 100. 5250 Extended Character Attributes

Bit Position (0 is least significant bit)	Meaning
7	**0** Normal image **1** Reverse image
6	**0** No underline

Table 100. 5250 Extended Character Attributes (continued)

Bit Position (0 is least significant bit)	Meaning
	1 Underline
5	**0** No blink **1** Blink
4	**0** No column separator **1** Column separator
3, 2, 1, 0	Reserved

Notices

This information was developed for products and services offered in the United States. HCL may not offer the products, services, or features discussed in this information in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you..

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. HCL may make improvements and/or changes in the product(s) and/or program(s) described in this information at any time without notice.

Any references in this information to non-HCL documentation or non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this HCL product and use of those documents or Web sites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM® or other companies.

Reference Materials

Keyboard Layout and Mapping Reference

Contents

[Top of Page on page 1168 | Previous Page | Next Page on page 1169 | Table of Contents on page 1168]

Keyboard Layout and Mapping Reference

Figures

[Top of Page on page 1169 | Previous Page on page 1168 | Next Page on page 1169 | Table of Contents on page 1168]

Keyboard Layout and Mapping Reference

Tables

[Top of Page on page 1169 | Previous Page on page 1169 | Next Page on page 1169 | Table of Contents on page 1168]

Keyboard Layout and Mapping Reference

Keyboard Layouts

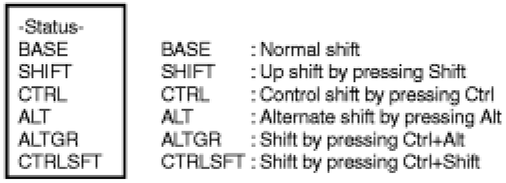
This chapter describes U.S. keyboard layouts supported by the host system. The keyboards are selected during installation or customization procedures. You can use keyboard layouts to locate character positions on the keyboard.

Different characters can appear in different key positions, depending on the country language selected. The function keys located across the top of the keyboard are not shown. Shaded areas show keys that are not available on some U.S. keyboards.

Refer to *Emulator User's Reference* and the online Personal Communications help for more information on keyboard layouts and mapping. Refer to your Windows (R) documentation for more information about the country and code page options.

The function keys have the following shift statuses.

Figure 1. Shift Statuses for Function Keys on page 1169



-Status-
BASE
SHIFT
CTRL
ALT
ALTGR
CTRLSFT

BASE : Normal shift
SHIFT : Up shift by pressing Shift
CTRL : Control shift by pressing Ctrl
ALT : Alternate shift by pressing Alt
ALTGR : Shift by pressing Ctrl+Alt
CTRLSFT : Shift by pressing Ctrl+Shift

Following are some abbreviations for the keys.

AltCur

Alternate Cursor

AltView

Alternate View

BackSp

Back Space

BackTb

Backtab

BgnOLn

Begin of Line

BotOPg

Bottom of Page

BtbWord

Backtab Word

CarRtn

Carrier Return

ChgFmt

Change Format

ChgScrn

Change Screen

ChrAdv

Character Advance

Ctrl

Control

CurBlink

Cursor Blink

CurSel

Cursor Select

DelWord

Delete Word

DevCan

Device Cancel

DspTxC

Display TextCode

Dup

Duplicate

EditCop

Edit Copy

EditPst

Edit Paste

EditUnd

Edit Undo

EndFld

End of Field

EndOLn

End of Line

ErEOF

Erase EOF

ErFld

Erase Field

ErInp

Erase Input

Fld-

Field-

Fld+

Field+

FldColor

Field Color

FldExt

Field Exit

FldHigh

Field Highlight

FldMark

Field Mark

FldTrns

Transparency Field Inherit

FastDn

Fast Down

FastDow

Fast Down

FastUp

Fast Up

FieldCol

Field Color

GrpCsr

Graphic Cursor

HostPr

Host Print

JmpNext

Jump Next

JumpNxt

Jump Next

MarkLef

Mark Left

MkDown

Mark Down

MkLeft

Mark Left

MkRight

Mark Right

MrkUp

Mark Up

MarkDow

Mark Down

MarkRig

Mark Right

MoveDow

Move Down

MoveLef

Move Left

MoveRig

Move Right

MvDown

Move Down

MvLeft

Move Left

MvRight

Move Right

MvUp

Move Up

NewLin

New Line

ReqBSp

Required Backspace

ReqTab

Required Tab

RolDwn

Roll Down

RollDow

Roll Down

RolUp

Roll Up

Rst/Ctrl

Reset/Ctrl

RPause

Record/Play Pause

SOSI

SOSI Display

SOSI/G

SOSI Generate

SysAttn

System Attention

SysReq

System Request

TabFld

Tab Field

TopOPg

Top of Page

TransOp

Transparency Opaque

TrnfldI

Transparency Field Inherit

TrnOp

Transparency Opaque

TstReq

Test Request

Turquois

Turquoise

UnderSc

Underscore

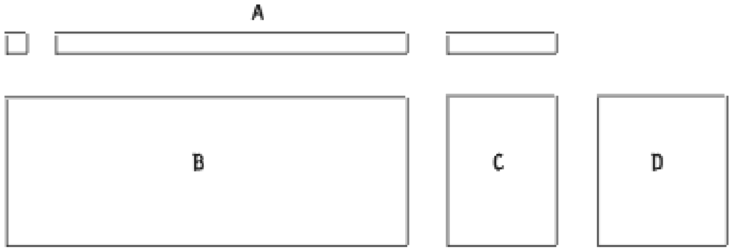
UndScr

Underscore

WdWrap

Word Wrap

Enhanced Keyboard, Microsoft Natural Keyboard (3270 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	-----	-----	Record	Play	RPause	-----	-----	-----	RTM	-----
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
-----	Red	Pink	Green	Yellow	Blue	Turq	White	FldColor	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	Break
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

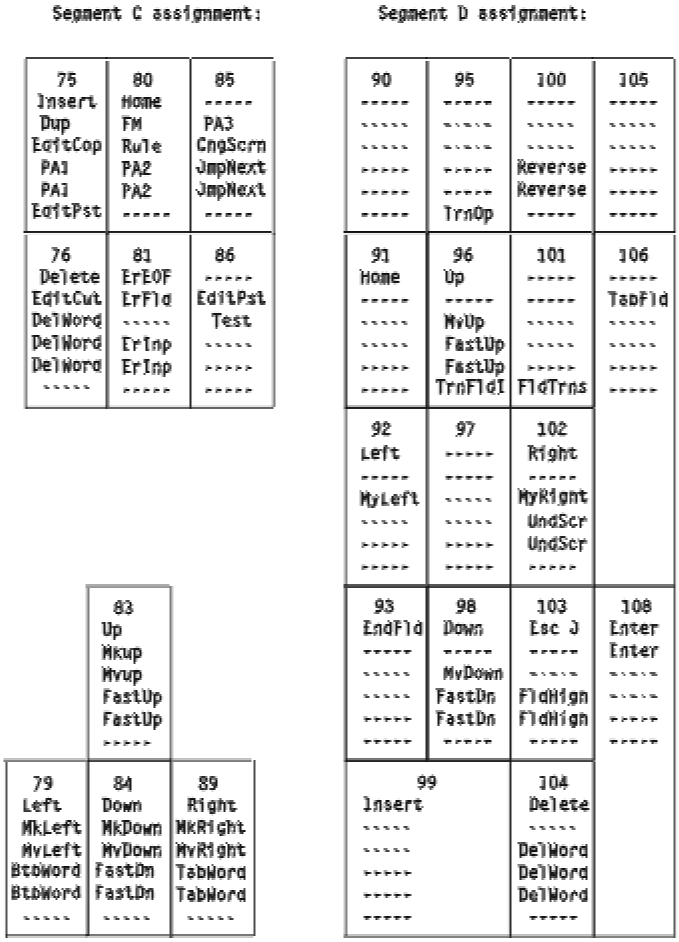
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	NUL	-----	-----	-----	RS	-----	-----	-----	-----	US	RS	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	EditUnd
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

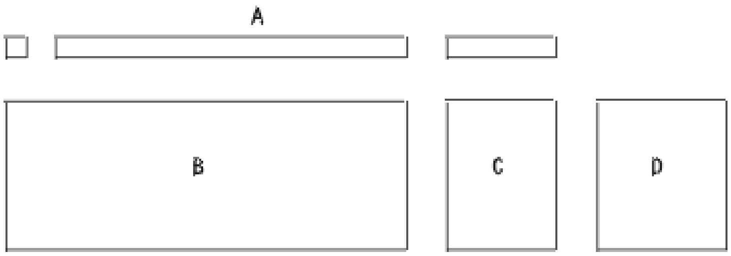
44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	SUB	CAN	ETX	SYN	STX	SO	CR	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----



```
    Segment C assignment:              Segment D assignment:

 ---------------------------     ------------------------------------
|   75    |   80    |   85    |   |   90    |   95    |  100    |  105    |
| Insert  | Home    | -----   |   | -----   | -----   | -----   | -----   |
| Dup     | FM      | PA3     |   | -----   | -----   | -----   | -----   |
| EditCop | Rule    | CngScrn |   | -----   | -----   | -----   | -----   |
| PA1     | PA2     | JmpNext |   | -----   | -----   | Reverse | -----   |
| PA1     | PA2     | JmpNext |   | -----   | -----   | Reverse | -----   |
| EditPst | -----   | -----   |   | -----   | TrnOp   | -----   | -----   |
|---------|---------|---------|   |---------|---------|---------|---------|
|   76    |   81    |   86    |   |   91    |   96    |  101    |  106    |
| Delete  | ErEOF   | -----   |   | Home    | Up      | -----   | -----   |
| EditCut | ErFld   | EditPst |   | -----   | -----   | -----   | TabFld  |
| DelWord | -----   | Test    |   | -----   | MvUp    | -----   | -----   |
| DelWord | ErInp   | -----   |   | -----   | FastUp  | -----   | -----   |
| DelWord | ErInp   | -----   |   | -----   | FastUp  | -----   | -----   |
| -----   | -----   | -----   |   | -----   | TrnFldI | FldTrns | -----   |
 ---------------------------     |---------|---------|---------|         |
                                  |   92    |   97    |  102    |         |
                                  | Left    | -----   | Right   |         |
                                  | -----   | -----   | -----   |         |
                                  | MyLeft  | -----   | MyRight |         |
                                  | -----   | -----   | UndScr  |         |
                                  | -----   | -----   | UndScr  |         |
                                  | -----   | -----   | -----   |         |
           ---------              |---------|---------|---------|---------|
          |   83    |             |   93    |   98    |  103    |  108    |
          | Up      |             | EndFld  | Down    | Esc J   | Enter   |
          | Mkup    |             | -----   | -----   | -----   | Enter   |
          | Mvup    |             | -----   | MvDown  | -----   | -----   |
          | FastUp  |             | -----   | FastDn  | FldHign | -----   |
          | FastUp  |             | -----   | FastDn  | FldHign | -----   |
          | -----   |             | -----   | -----   | -----   | -----   |
 ---------|---------|---------    |-------------------|---------|         |
|   79    |   84    |   89    |   |        99         |  104    |         |
| Left    | Down    | Right   |   | Insert            | Delete  |         |
| MkLeft  | MkDown  | MkRight |   | -----             | -----   |         |
| MvLeft  | MvDown  | MvRight |   | -----             | DelWord |         |
| BtbWord | FastDn  | TabWord |   | -----             | DelWord |         |
| BtbWord | FastDn  | TabWord |   | -----             | DelWord |         |
| -----   | -----   | -----   |   | -----             | -----   |         |
 ---------------------------      ------------------------------------
```

Enhanced Keyboard, Microsoft Natural Keyboard (5250 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	-----	-----	Record	Play	RPause	SOSI/G	-----	-----	-----	-----
-----	Help	-----	-----	-----	-----	-----	-----	-----	-----	CurBlink	AltCur	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	SOSI/G	-----	CurBlink	AltCur	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	FieldCol	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	HostPr
-----	-----	TstReq
-----	-----	-----
-----	-----	-----

Segment B assignment:

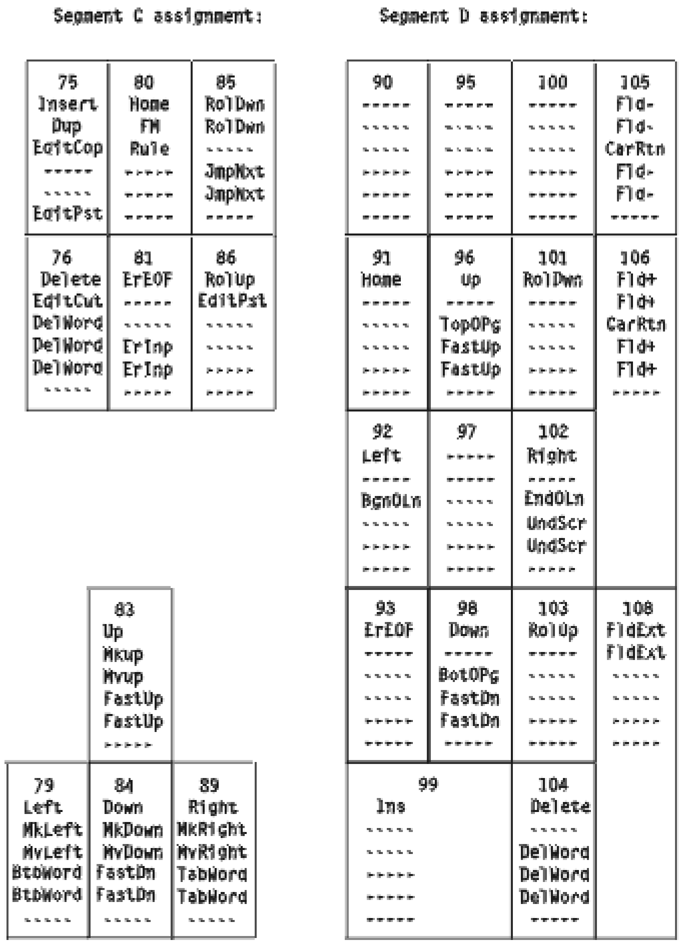
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	ReqBSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTb	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
ReqTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	FldExt
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLin
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	CarRtn
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----



Segment C assignment:

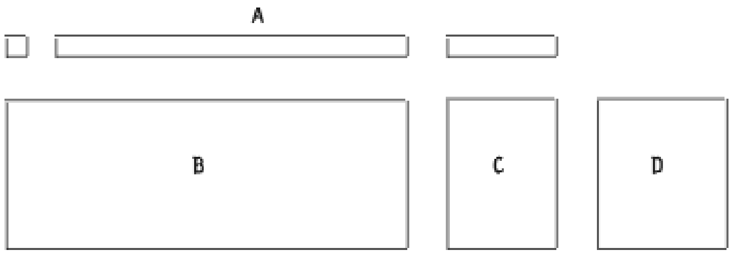
75 Insert Dup EditCop ----- ----- EditPst	80 Home FM Rule ----- ----- -----	85 RolDwn RolDwn ----- JmpNxt JmpNxt -----
76 Delete EditCut DelWord DelWord DelWord -----	81 ErEOF ----- ----- ErInp ErInp -----	86 RolUp EditPst ----- ----- ----- -----

	83 Up Mkup Mvup FastUp FastUp -----	
79 Left MkLeft MvLeft BtbWord BtbWord -----	84 Down MkDown MvDown FastDn FastDn -----	89 Right MkRight MvRight TabWord TabWord -----

Segment D assignment:

90 ----- ----- ----- ----- ----- -----	95 ----- ----- ----- ----- ----- -----	100 ----- ----- ----- ----- ----- -----	105 Fld- Fld- CarRtn Fld- Fld- -----
91 Home ----- ----- ----- ----- -----	96 Up ----- TopOPg FastUp FastUp -----	101 RolDwn ----- ----- ----- ----- -----	106 Fld+ Fld+ CarRtn Fld+ Fld+ -----
92 Left ----- BgnOLn ----- ----- -----	97 ----- ----- ----- ----- ----- -----	102 Right ----- EndOLn UndScr UndScr -----	
93 ErEOF ----- ----- ----- ----- -----	98 Down ----- BotOPg FastDn FastDn -----	103 RolUp ----- ----- ----- ----- -----	108 FldExt FldExt ----- ----- ----- -----
99 Ins ----- ----- ----- ----- -----		104 Delete ----- DelWord DelWord DelWord -----	

Enhanced Keyboard, Microsoft Natural Keyboard (Combined 3270 and 5250)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	~~~~~	~~~~~	~~~~~	Record	Play	RPause	~~~~~	~~~~~	~~~~~	RTM	~~~~~
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
~~	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
~~	Red	Pink	Green	Yellow	Blue	Turq	White	FldColor	-----	-----	-----	-----

124	125	126
~~	~~~~~	Clear
~~	~~~~~	~~~~~
-----	-----	Break
-----	-----	-----
-----	-----	-----
~~	~~~~~	~~~~~

Segment B assignment:

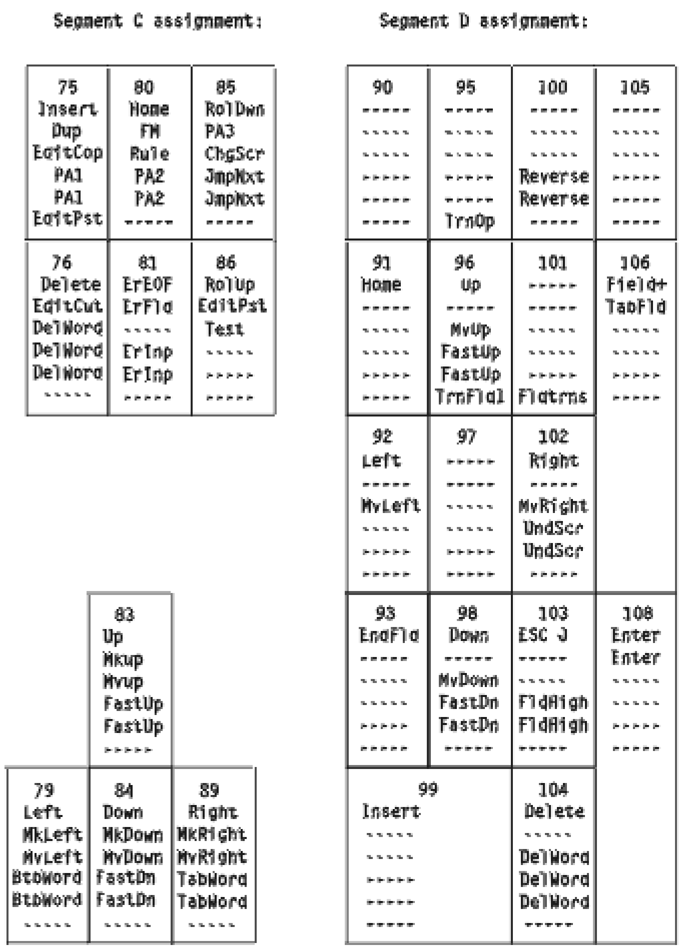
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	ChrAdv
~~	~~~~~	NUL	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	US	RS	~~~~~	~~~~~
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
~~	SUB	CAN	ETX	SYN	STX	SO	CR	~~~~~	~~~~~	~~~~~	~~~~~
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctrl
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctrl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~



Segment C assignment:

75 Insert Dup EditCop PA1 PA1 EditPst	80 Home FM Rule PA2 PA2 -----	85 RolDwn PA3 ChgScr JmpNxt JmpNxt -----
76 Delete EditCut DelWord DelWord DelWord -----	81 ErEOF ErFld ----- ErInp ErInp -----	86 RolUp EditPst Test ----- ----- -----
	83 Up Mkup Mvup FastUp FastUp -----	
79 Left MkLeft MvLeft BtbWord BtbWord -----	84 Down MkDown MvDown FastDn FastDn -----	89 Right MkRight MvRight TabWord TabWord -----

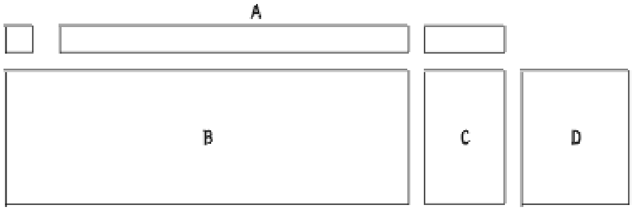
Segment D assignment:

90 ----- ----- ----- ----- ----- -----	95 ----- ----- ----- ----- ----- TrnOp	100 ----- ----- ----- Reverse Reverse -----	105 ----- ----- ----- ----- ----- -----
91 Home ----- ----- ----- ----- -----	96 Up ----- MvUp FastUp FastUp TrnFldl	101 ----- ----- ----- ----- ----- Fldtrns	106 Field+ TabFld ----- ----- ----- -----
92 Left ----- MvLeft ----- ----- -----	97 ----- ----- ----- ----- ----- -----	102 Right ----- MvRight UndScr UndScr -----	
93 EndFld ----- ----- ----- ----- -----	98 Down ----- MvDown FastDn FastDn -----	103 ESC J ----- ----- FldHigh FldHigh -----	108 Enter Enter ----- ----- ----- -----
99 Insert ----- ----- ----- ----- -----		104 Delete ----- DelWord DelWord DelWord -----	

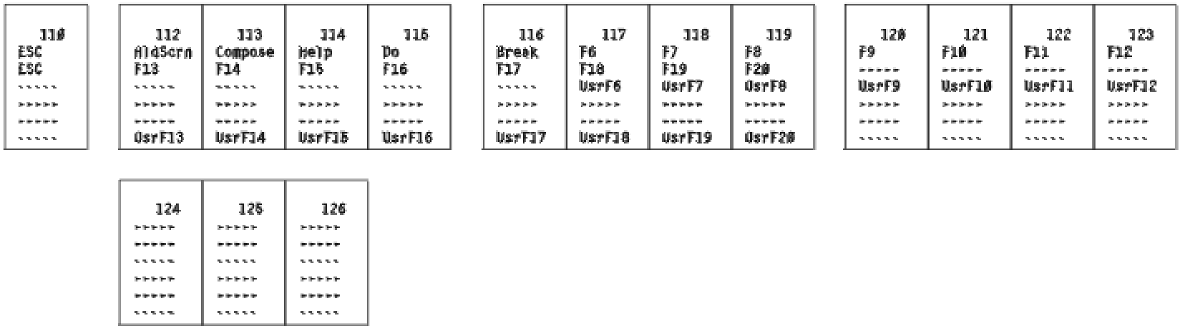
Enhanced Keyboard, Microsoft Natural Keyboard (VT Keyboard Layout)

Note:

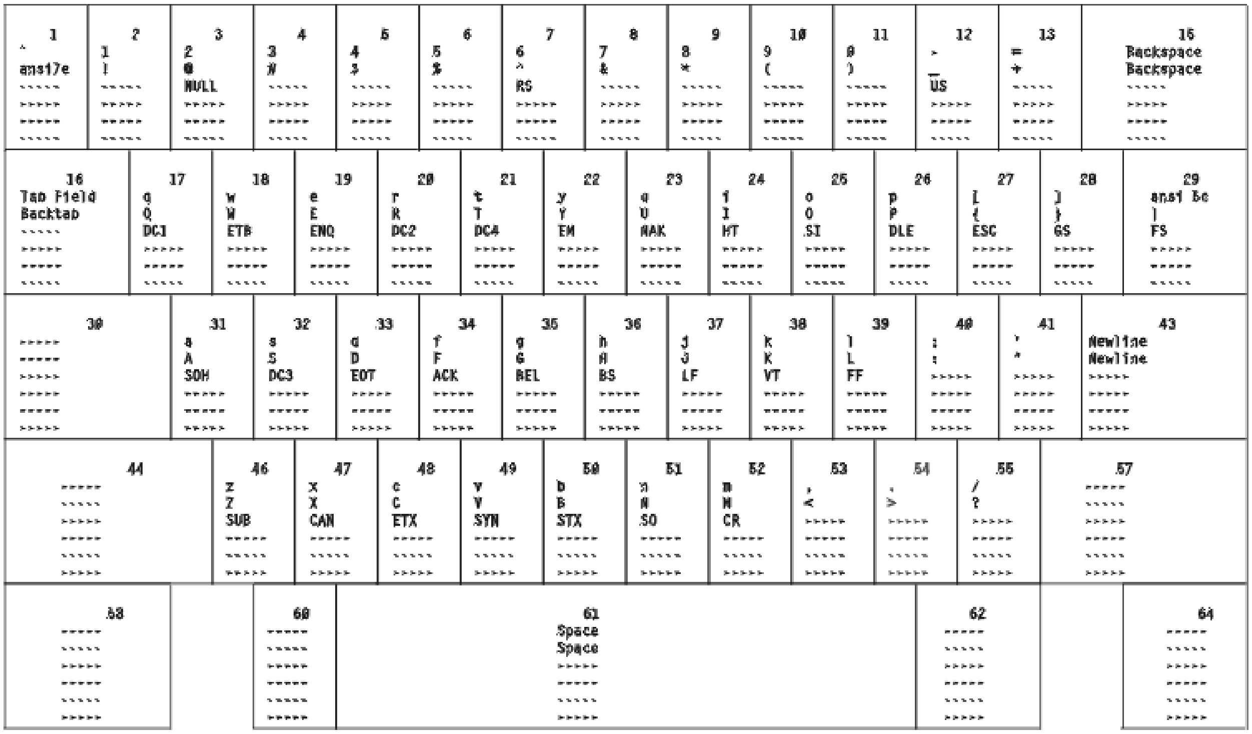
This keyboard layout does not include Personal Communications local function (for example, edit copy, edit paste, etc.).

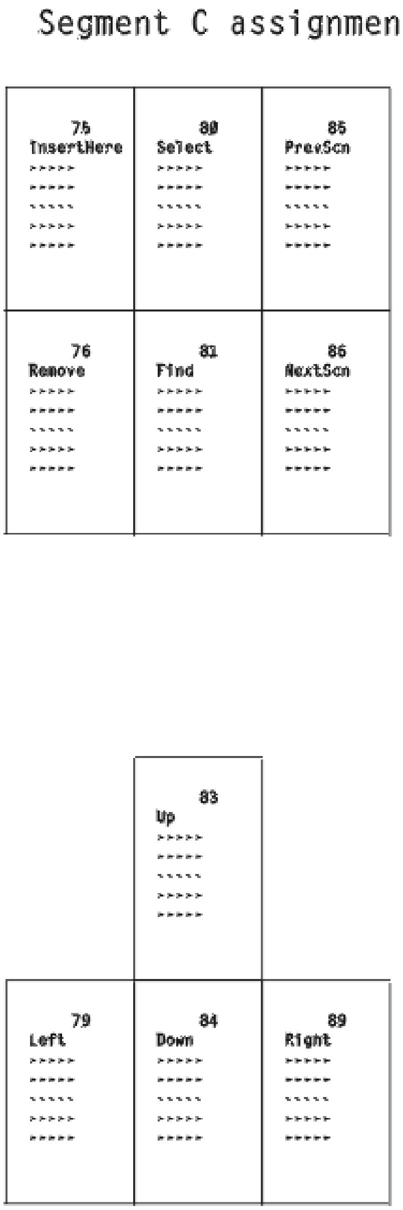


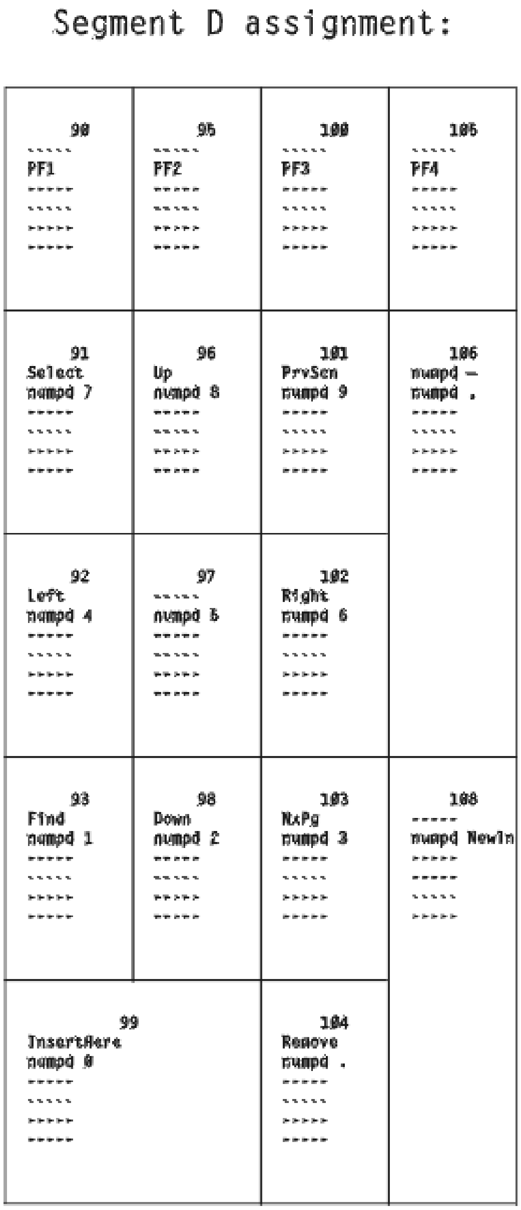
Segment A assignment:



Segment B assignment:



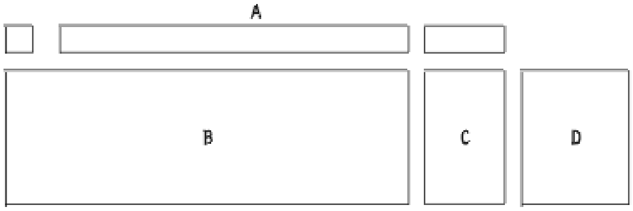




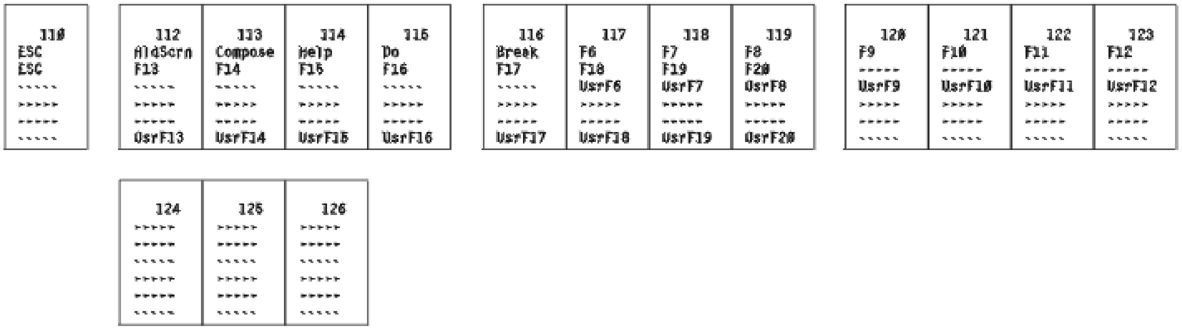
Enhanced Keyboard, Microsoft Natural Keyboard (VT Local Edit Mode Only)

Note:

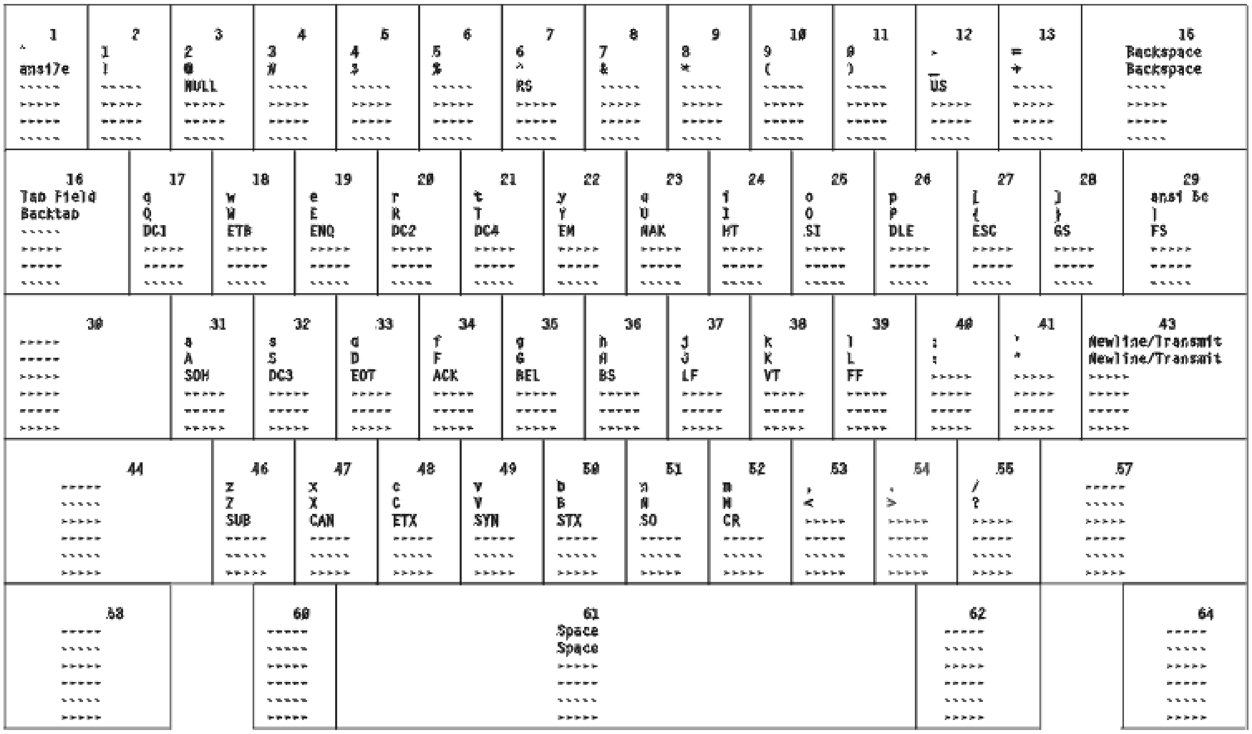
This keyboard layout does not include Personal Communications local function (for example, edit copy, edit paste, etc.).

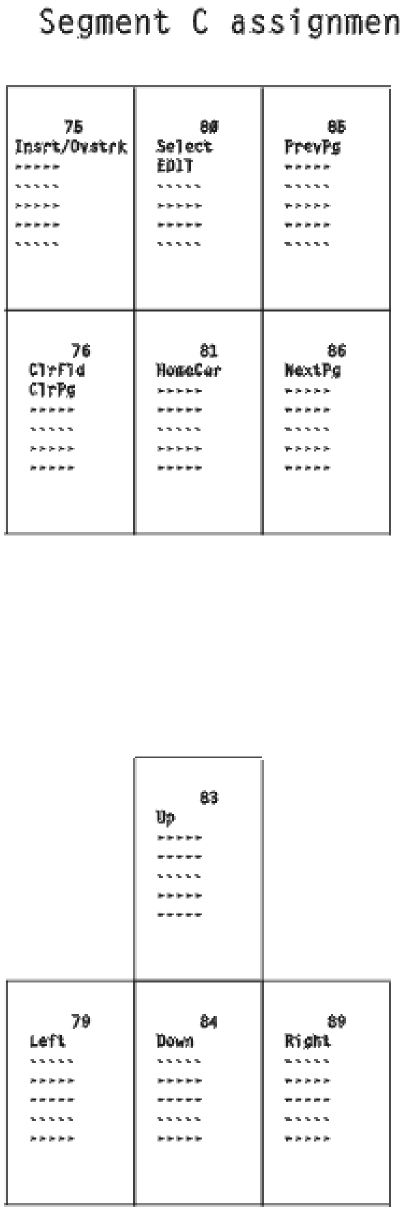


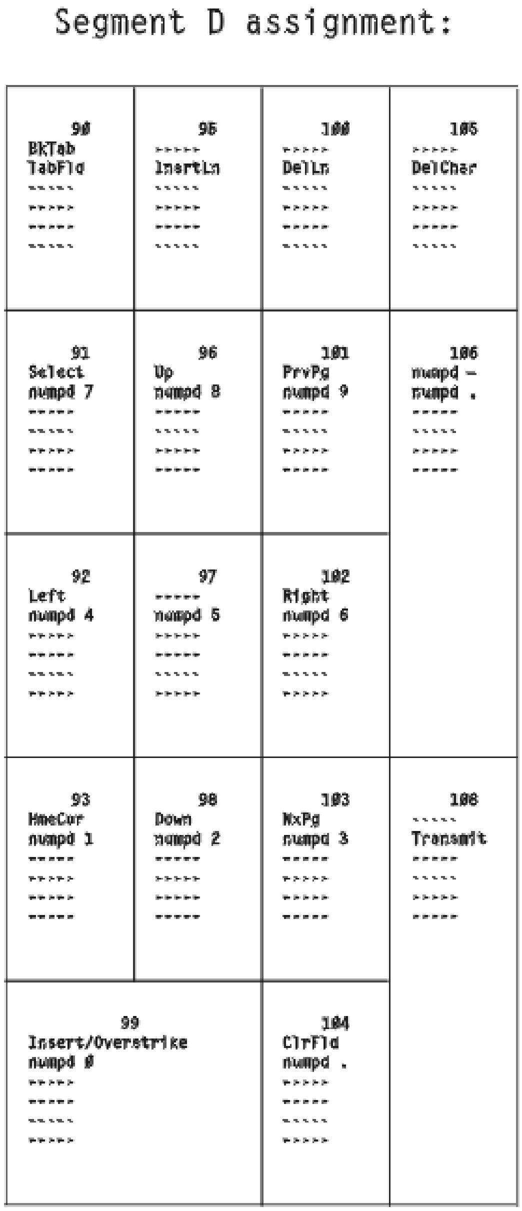
Segment A assignment:



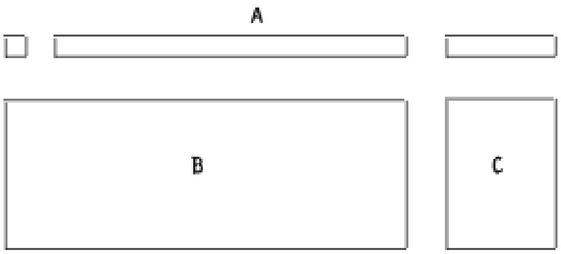
Segment B assignment:







Space-Saving Keyboard (3270 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	-----	-----	-----	-----	Record	Play	RPause	-----	-----	-----	RTM	-----
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
-----	Red	Pink	Green	Yellow	Blue	Turq	White	FldColor	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	Break
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

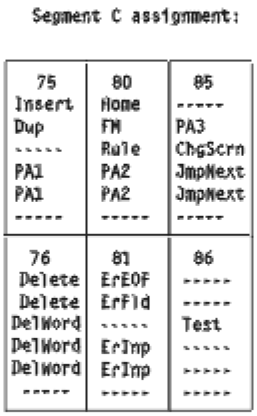
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

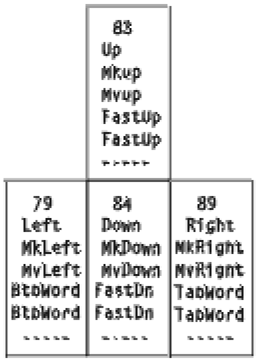
44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----



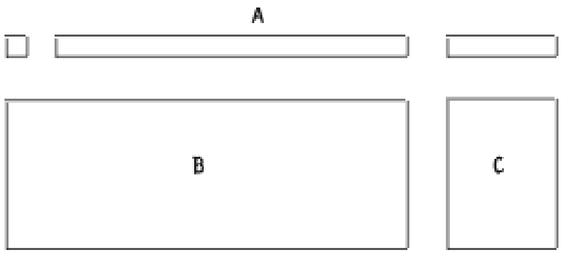
Segment C assignment:

75 Insert Dup ----- PA1 PA1 -----	80 Home FM Rule PA2 PA2 -----	85 ----- PA3 ChgScrn JmpNext JmpNext -----
76 Delete Delete DelWord DelWord DelWord -----	81 ErEOF ErFld ----- ErInp ErInp -----	86 ----- ----- Test ----- ----- -----



	83 Up Mkup Mvup FastUp FastUp -----	
79 Left MkLeft MvLeft BtbWord BtbWord -----	84 Down MkDown MvDown FastDn FastDn -----	89 Right MkRight MvRight TabWord TabWord -----

Space-Saving Keyboard (5250 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn SysReq ----- ----- ----- -----	PF1 PF13 ----- Help ----- -----	PF2 PF14 ----- ----- ----- -----	PF3 PF15 ----- ----- ----- -----	PF4 PF16 ----- ----- ----- -----	PF5 PF17 Record ----- ----- -----	PF6 PF18 Play ----- ----- -----	PF7 PF19 RPause ----- ----- -----	PF8 PF20 ----- ----- ----- -----	PF9 PF21 ----- ----- ----- -----	PF10 PF22 ----- CurBlink CurBlink -----	PF11 PF23 ----- AltCur AltCur -----	PF12 PF24 ----- ----- ----- -----

124	125	126
----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	Clear ----- HostPr TstReq ----- -----

Segment B assignment:

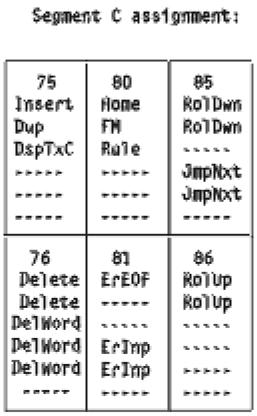
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	BackSp BackSp ReqBSp ----- ----- -----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld BackTb ReqTab ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----

30	31	32	33	34	35	36	37	38	39	40	41	43
----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	FldExt NewLin CarRtn ----- ----- -----

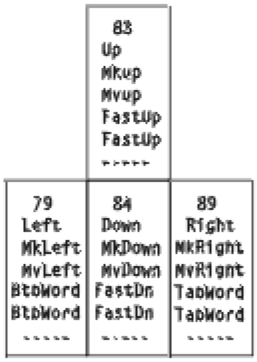
44	46	47	48	49	50	51	52	53	54	55	57
----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----

58	60	61	62	64
Reset/Ctrl Reset/Ctrl ----- Quit Quit -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	----- ----- ----- ----- ----- -----	Enter/Ctrl Enter/Ctrl ----- ----- ----- -----



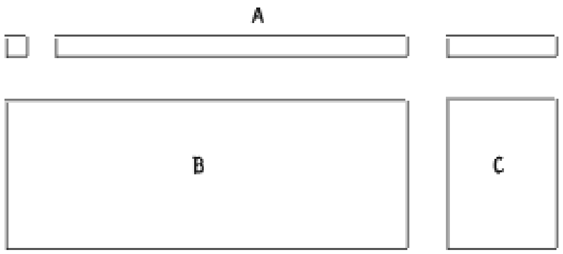
Segment C assignment:

75 Insert Dup DspTxC ----- ----- -----	80 Home FM Rule ----- ----- -----	85 RolDwn RolDwn ----- JmpNxt JmpNxt -----
76 Delete Delete DelWord DelWord DelWord -----	81 ErEOF ----- ----- ErInp ErInp -----	86 RolUp RolUp ----- ----- ----- -----



	83 Up Mkup Mvup FastUp FastUp -----	
79 Left MkLeft MvLeft BtbWord BtbWord -----	84 Down MkDown MvDown FastDn FastDn -----	89 Right MkRight MvRight TabWord TabWord -----

Space-Saving Keyboard (Combined 3270 and 5250)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	~~~~~	~~~~~	~~~~~	~~~~~	Record	Play	RPause	~~~~~	~~~~~	~~~~~	RTM	~~~~~
-----	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
~~	Doc	WdWrap	CfgFmt	-----	-----	-----	-----	APL	CurSel	CurBlink	AltCur	+Cr
~~	Red	Pink	Green	Yellow	Blue	Turq	White	FldColor	-----	-----	-----	-----

124	125	126
~~	~~~~~	Clear
~~	~~~~~	~~~~~
>>>>>	>>>>>	Break
-----	-----	-----
-----	-----	-----
~~	~~~~~	~~~~~

Segment B assignment:

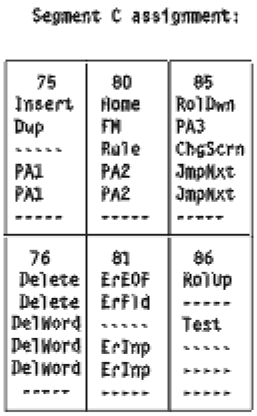
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	ChrAdv
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
TabFld	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

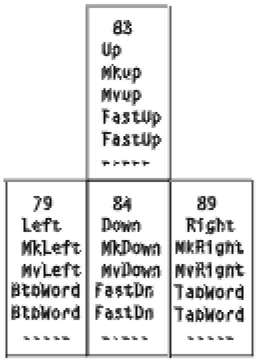
30	31	32	33	34	35	36	37	38	39	40	41	43
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	NewLine
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctrl
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctrl
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~

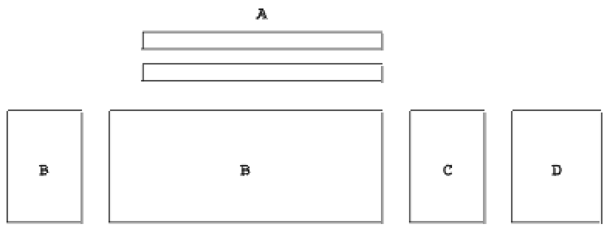


```
Segment C assignment:

+--------+--------+--------+
|   75   |   80   |   85   |
| Insert | Home   | RolDwn |
| Dup    | FM     | PA3    |
| -----  | Rule   | ChgScrn|
| PA1    | PA2    | JmpNxt |
| PA1    | PA2    | JmpNxt |
| -----  | -----  | -----  |
+--------+--------+--------+
|   76   |   81   |   86   |
| Delete | ErEOF  | RolUp  |
| Delete | ErFld  | -----  |
| DelWord| -----  | Test   |
| DelWord| ErInp  | -----  |
| DelWord| ErInp  | -----  |
|  ----- | -----  | -----  |
+--------+--------+--------+
```

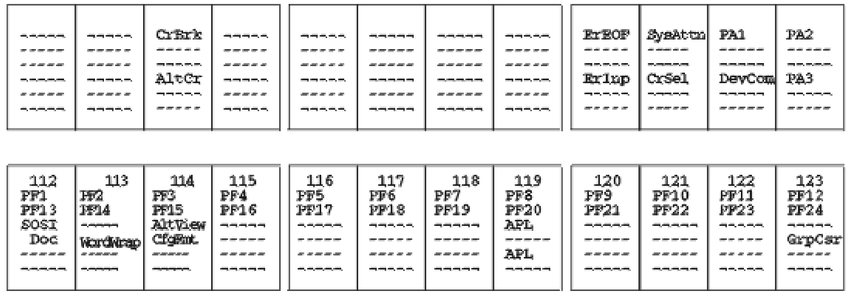


```
         +--------+
         |   83   |
         | Up     |
         | Mkup   |
         | Mvup   |
         | FastUp |
         | FastUp |
         | -----  |
+--------+--------+--------+
|   79   |   84   |   89   |
| Left   | Down   | Right  |
| MkLeft | MkDown | MkRight|
| MvLeft | MvDown | MvRignt|
| BtbWord| FastDn | TabWord|
| BtbWord| FastDn | TabWord|
|  ----- | -----  | -----  |
+--------+--------+--------+
```

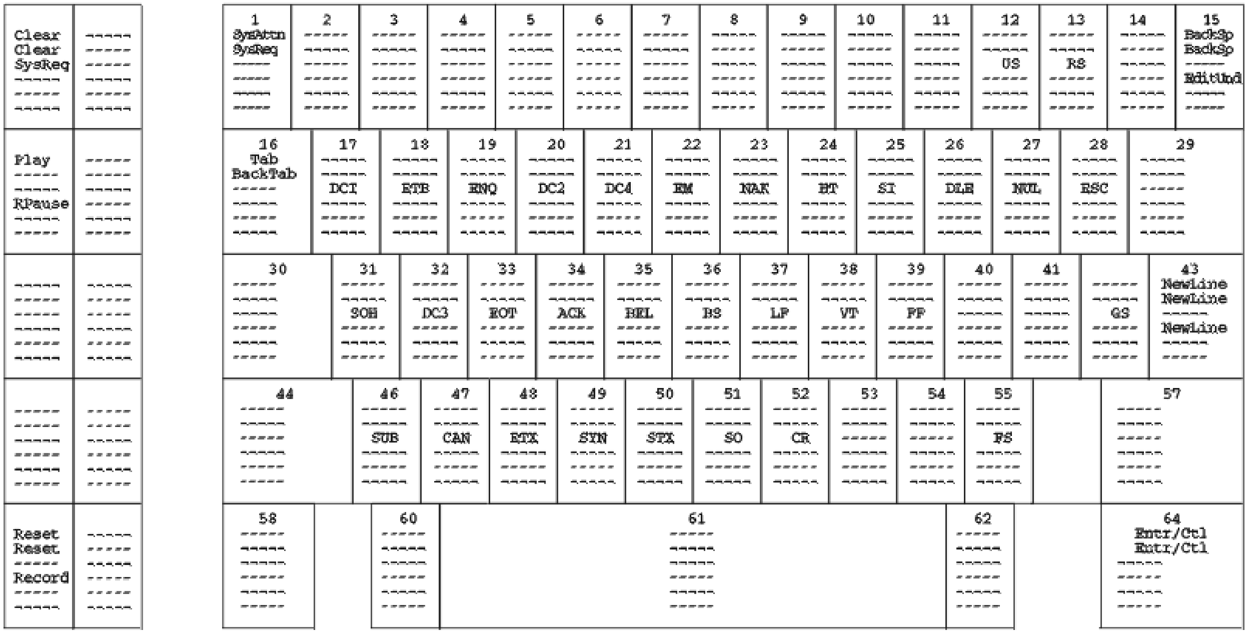
5576-001 (3270 Only)

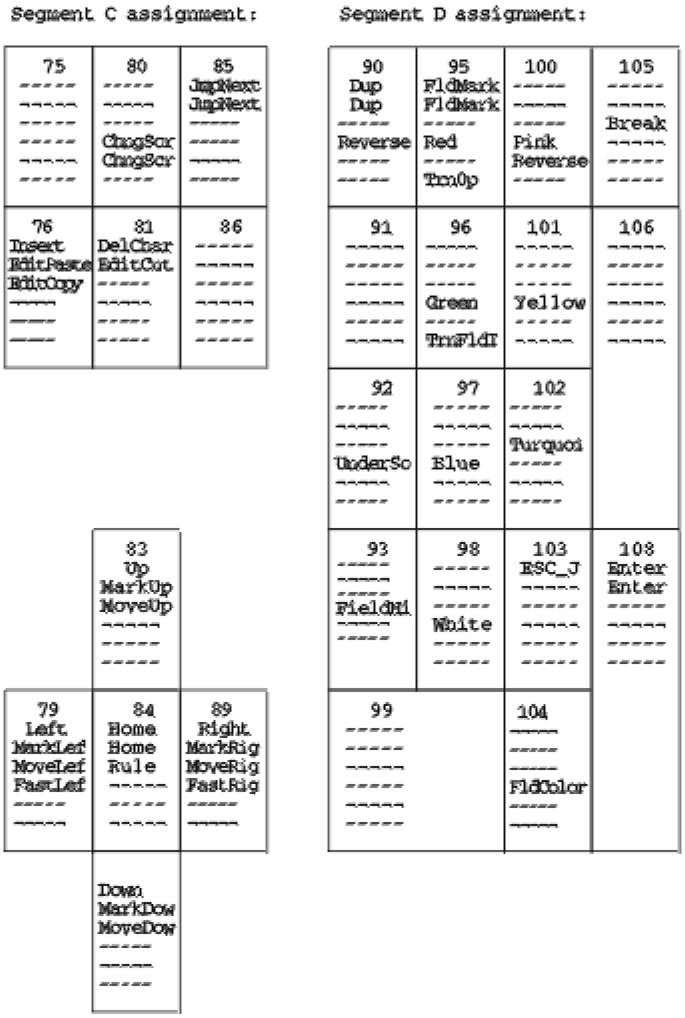


Segment A assignment:



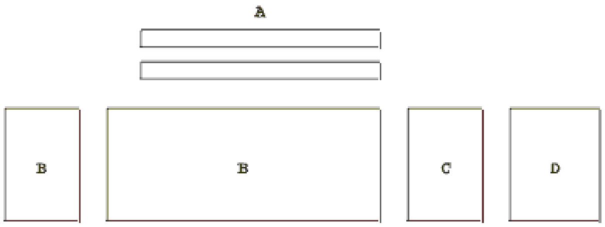
Segment B assignment:



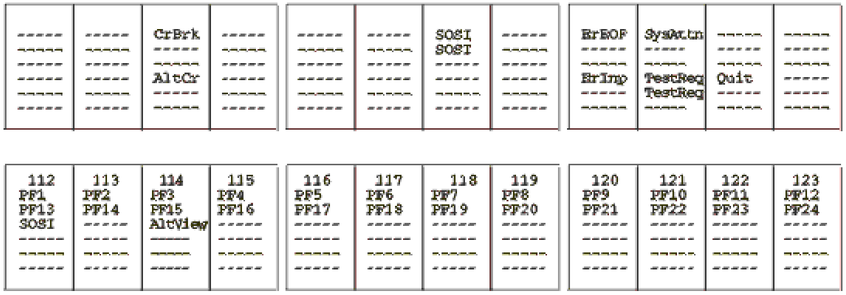


```
Segment C assignment:                Segment D assignment:

 75        80        85               90        95        100       105
-----     -----     JmpNext           Dup      FldMark   -----     -----
-----     -----     JmpNext           Dup      FldMark   -----     -----
-----     -----     -----            -----     -----     -----     Break
-----     ChngScr   -----            Reverse   Red       Pink      -----
-----     ChngScr   -----            -----     -----     Reverse   -----
-----     -----     -----            -----     TrnOp     -----     -----

 76        81        86               91        96        101       106
Insert    DelChar   -----            -----     -----     -----     -----
EditPaste EditCut   -----            -----     -----     -----     -----
EditCopy  -----     -----            -----     -----     -----     -----
-----     -----     -----            -----     Green     Yellow    -----
-----     -----     -----            -----     -----     -----     -----
-----     -----     -----            -----     TrnFldI   -----     -----

                                      92        97        102
                                     -----     -----     -----
                                     -----     -----     -----
                                     -----     -----     Turquoi
                                     UnderSo   Blue      -----
                                     -----     -----     -----
                                     -----     -----     -----

           83                         93        98        103       108
           Up                        -----     -----     ESC_J     Enter
          MarkUp                     -----     -----     -----     Enter
          MoveUp                     -----     -----     -----     -----
          -----                      FieldMi   White     -----     -----
          -----                      -----     -----     -----     -----
          -----                      -----     -----     -----     -----

 79        84        89               99                  104
Left      Home      Right            -----               -----
MarkLef   Home      MarkRig          -----               -----
MoveLef   Rule      MoveRig          -----               -----
FastLef   -----     FastRig          -----               FldColor
-----     -----     -----            -----               -----
-----     -----     -----            -----               -----

          Down
          MarkDow
          MoveDow
          -----
          -----
          -----
```

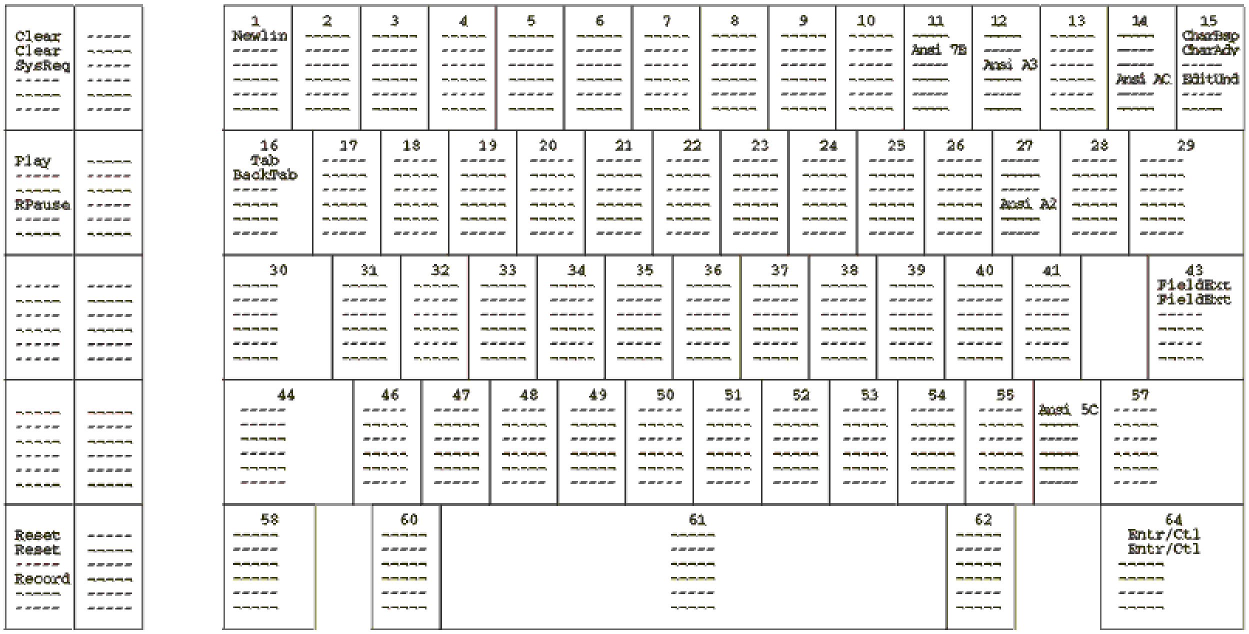
5576-001 (5250 Only)

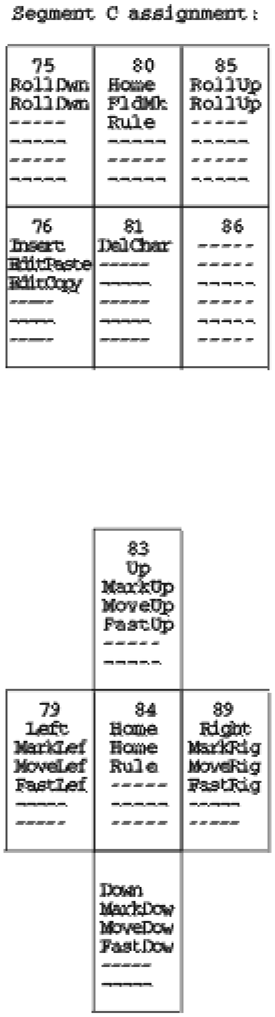


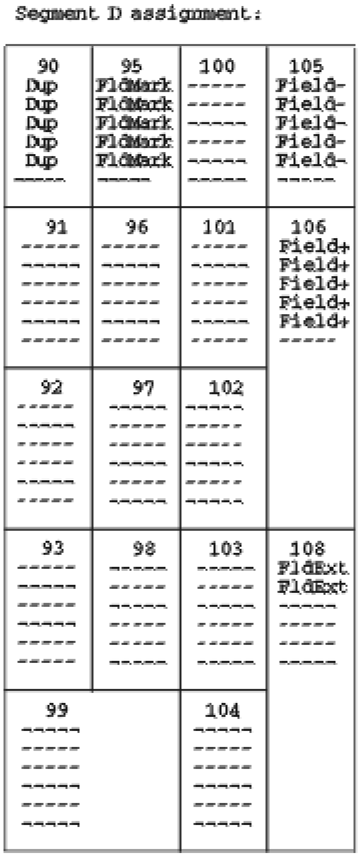
Segment A assignment:



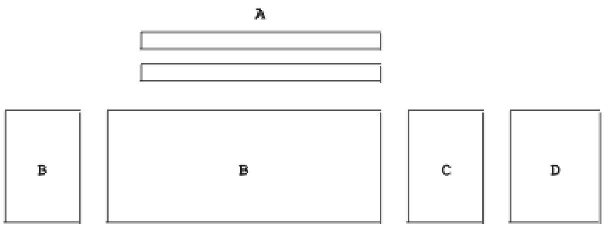
Segment B assignment:



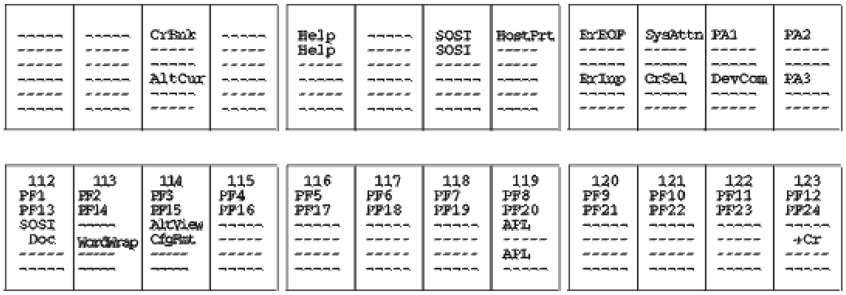

Segment C assignment:
75 RollDwn RollDwn ----- ----- ----- -----
80 Home FldMk Rule ----- ----- -----
85 RollUp RollUp ----- ----- ----- -----
76 Insert EditPaste EditCopy ----- ----- -----
81 DelChar ----- ----- ----- ----- -----
86 ----- ----- ----- ----- ----- -----
83 Up MarkUp MoveUp FastUp ----- -----
79 Left MarkLef MoveLef FastLef ----- -----
84 Home Home Rule ----- ----- -----
89 Right MarkRig MoveRig FastRig ----- -----
Down MarkDow MoveDow FastDow ----- -----



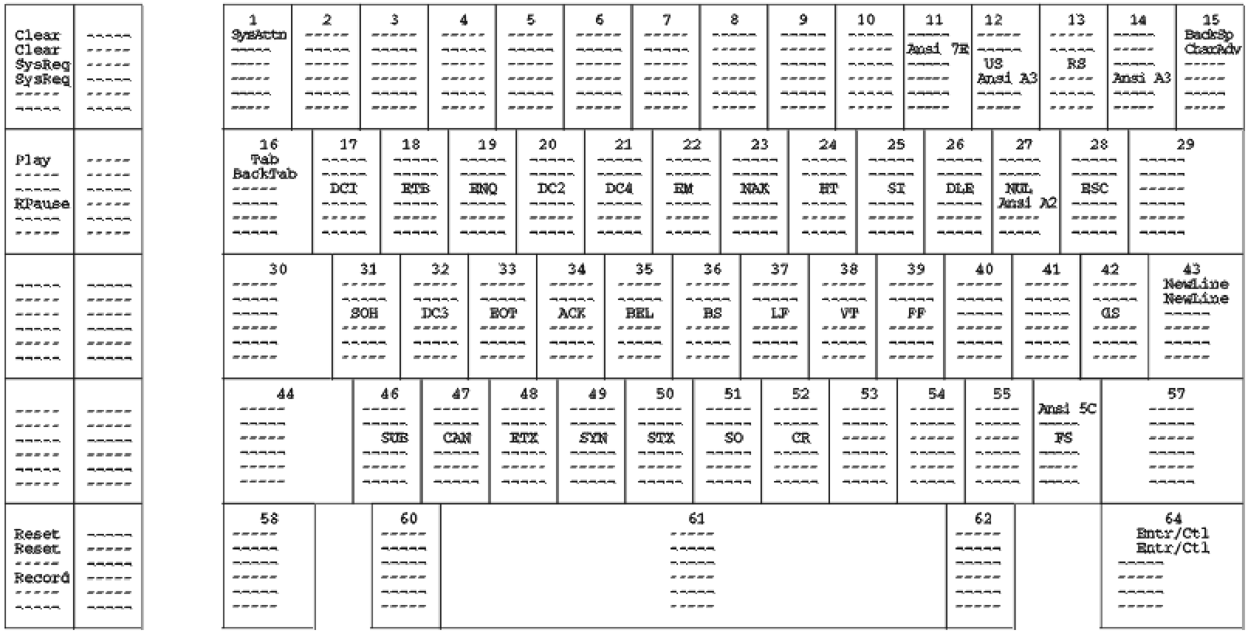
5576-001 (Combined 3270 and 5250)

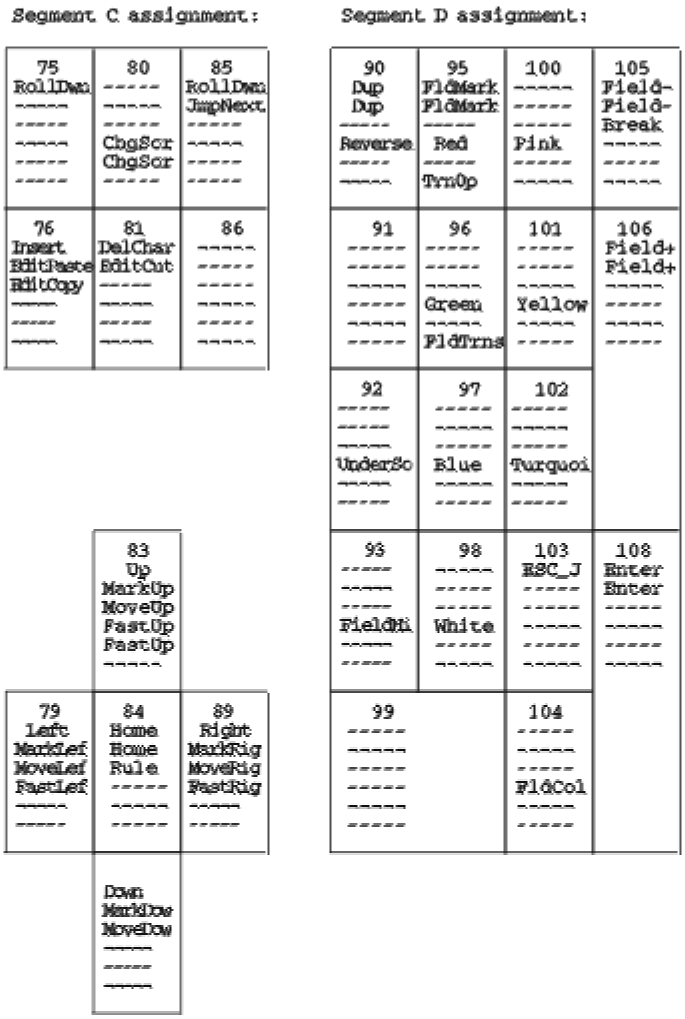


Segment A assignment:



Segment B assignment:

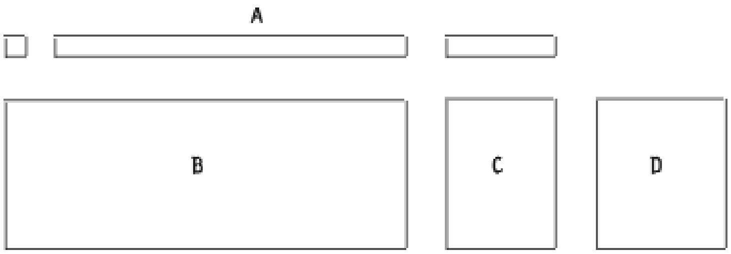




```
Segment C assignment:

75        80        85
RollDwn   -----     RollDwn
-----     -----     JmpNext
-----     -----     -----
-----     ChgScr    -----
-----     ChgScr    -----
-----     -----     -----

76        81        86
Insert    DelChar   -----
EditPaste EditCut   -----
EditCopy  -----     -----
-----     -----     -----
-----     -----     -----
-----     -----     -----

          83
          Up
          MarkUp
          MoveUp
          FastUp
          FastUp
          -----

79        84        89
Left      Home      Right
MarkLef   Home      MarkRig
MoveLef   Rule      MoveRig
FastLef   -----     FastRig
-----     -----     -----
-----     -----     -----

          Down
          MarkDow
          MoveDow
          -----
          -----
          -----
```

```
Segment D assignment:

90        95        100       105
Dup       FldMark   -----     Field-
Dup       FldMark   -----     Field-
-----     -----     -----     Break
Reverse   Red       Pink      -----
-----     -----     -----     -----
-----     TrnOp     -----     -----

91        96        101       106
-----     -----     -----     Field+
-----     -----     -----     Field+
-----     -----     -----     -----
-----     Green     Yellow    -----
-----     -----     -----     -----
-----     FldTrns   -----     -----

92        97        102
-----     -----     -----
-----     -----     -----
-----     -----     -----
UnderSo   Blue      Turquoi
-----     -----     -----
-----     -----     -----

93        98        103       108
-----     -----     ESC_J     Enter
-----     -----     -----     Enter
-----     -----     -----     -----
FieldMi   White     -----     -----
-----     -----     -----     -----
-----     -----     -----     -----

99                  104
-----               -----
-----               -----
-----               -----
-----               FldCol
-----               -----
-----               -----
```

5576-A01 (3270 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	AltView	-----	Record	Play	RPause	APL	CrSel	CrBnk	RTM	-----
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	-----	-----	-----	AltCr	GrpCsr
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	APL	CrSel	CrBnk	AltCr	GrpCsr
-----	Red	Pink	Green	Yellow	Blue	Turq	White	FldColor	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	Break
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

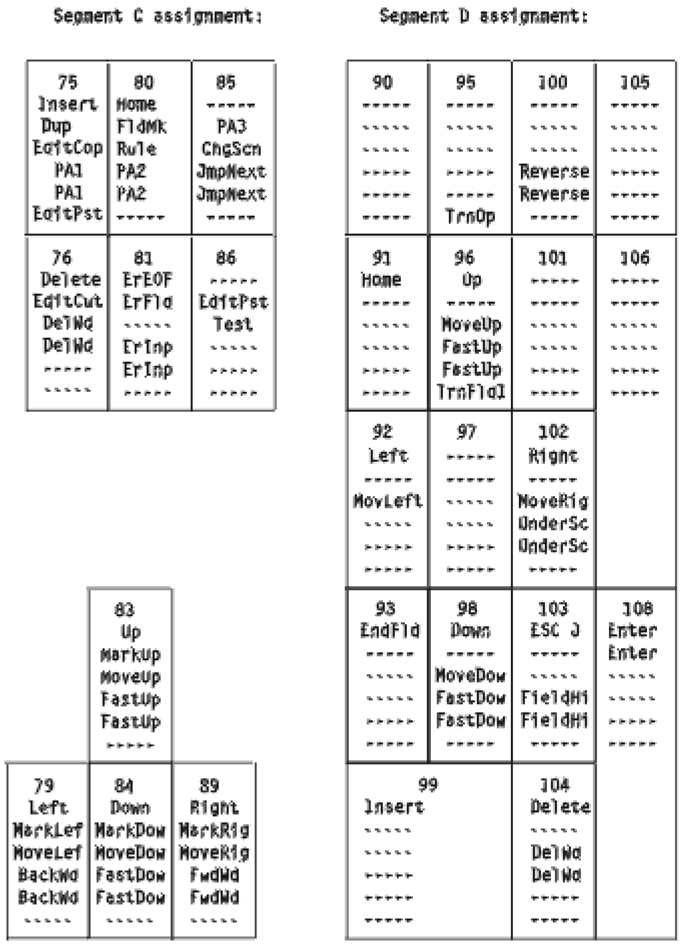
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	NUL	-----	-----	-----	RS	-----	-----	-----	-----	US	RS	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	EditUnd
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	SUB	CAN	ETX	SYN	STX	SO	CR	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Enter/Ctl
Reset/Ctrl	-----	-----	-----	Enter/Ctl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----



Segment C assignment:

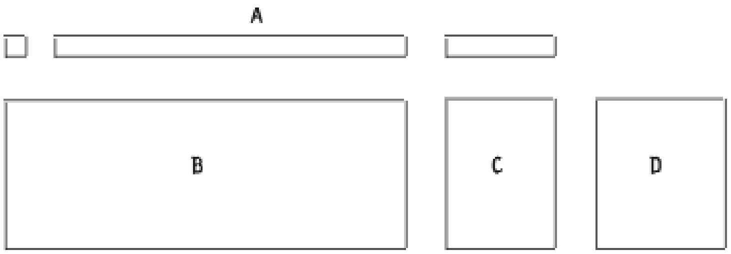
75 Insert Dup EditCop PA1 PA1 EditPst	80 Home FldMk Rule PA2 PA2 -----	85 ----- PA3 ChgScn JmpNext JmpNext -----
76 Delete EditCut DelWd DelWd ----- ~~~~~	81 ErEOF ErFld ~~~~~ ErInp ErInp -----	86 ----- EditPst Test ~~~~~ ----- -----

	83 Up MarkUp MoveUp FastUp FastUp -----	
79 Left MarkLef MoveLef BackWd BackWd ~~~~~	84 Down MarkDow MoveDow FastDow FastDow ~~~~~	89 Right MarkRig MoveRig FwdWd FwdWd ~~~~~

Segment D assignment:

90 ----- ~~~~~ ~~~~~ ----- ----- -----	95 ----- ~~~~~ ~~~~~ ----- ----- TrnOp	100 ----- ~~~~~ ~~~~~ Reverse Reverse -----	105 ----- ~~~~~ ~~~~~ ----- ----- -----
91 Home ----- ~~~~~ ~~~~~ ----- -----	96 Up ----- MoveUp FastUp FastUp TrnFldI	101 ----- ----- ~~~~~ ~~~~~ ----- -----	106 ----- ----- ~~~~~ ~~~~~ ----- -----
92 Left ----- MovLeft ~~~~~ ----- -----	97 ----- ----- ~~~~~ ~~~~~ ----- -----	102 Right ----- MoveRig UnderSc UnderSc -----	
93 EndFld ----- ~~~~~ ~~~~~ ----- -----	98 Down ----- MoveDow FastDow FastDow -----	103 ESC J ----- ~~~~~ FieldHi FieldHi -----	108 Enter Enter ~~~~~ ~~~~~ ----- -----
99 Insert ~~~~~ ~~~~~ ----- ----- -----		104 Delete ~~~~~ DelWd DelWd ----- -----	

5576-A01 Keyboard (5250 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	AltView	-----	Record	Play	RPause	SOSI/G	-----	-----	AltCur	-----
-----	Help	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	SOSI/G	-----	-----	AltCur	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	FieldCol	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	HostPr
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

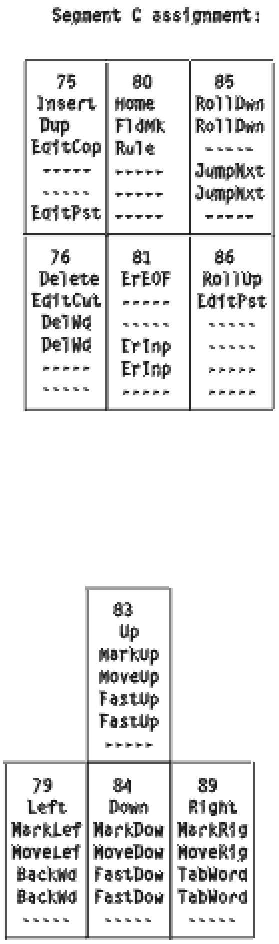
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	CharBsp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi 7E	-----	-----	-----	CharAdv
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi A3	-----	Ansi AC	EditUnd
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi A2	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

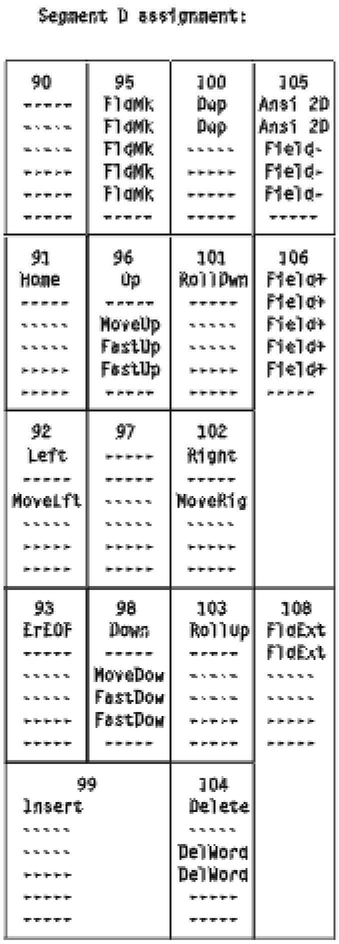
30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	FieldExt
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	FieldExt
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
Reset/Ctrl	-----	-----	-----	Enter/Ctrl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----
-----	-----	-----	-----	-----

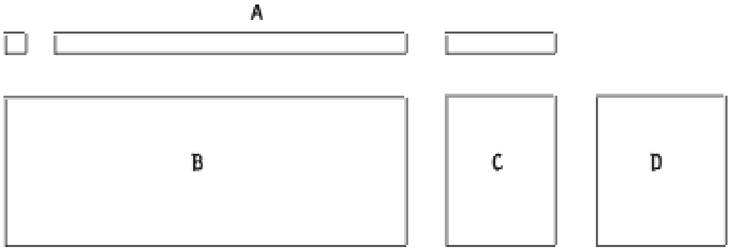


```
Segment C assignment:

 75        80        85
 Insert    Home      RollDwn
 Dup       FldMk     RollDwn
 EditCop   Rule      ~~~~~
 -----     -----     JumpNxt
 ~~~~~     -----     JumpNxt
 EditPst   -----     -----

 76        81        86
 Delete    ErEOF     RollUp
 EditCut   -----     EditPst
 DelWd     ~~~~~     ~~~~~
 DelWd     ErInp     ~~~~~
 -----     ErInp     -----
 ~~~~~     -----     -----

           83
           Up
           MarkUp
           MoveUp
           FastUp
           FastUp
           -----

 79        84        89
 Left      Down      Right
 MarkLef   MarkDow   MarkRig
 MoveLef   MoveDow   MoveRig
 BackWd    FastDow   TabWord
 BackWd    FastDow   TabWord
 ~~~~~     ~~~~~     ~~~~~
```

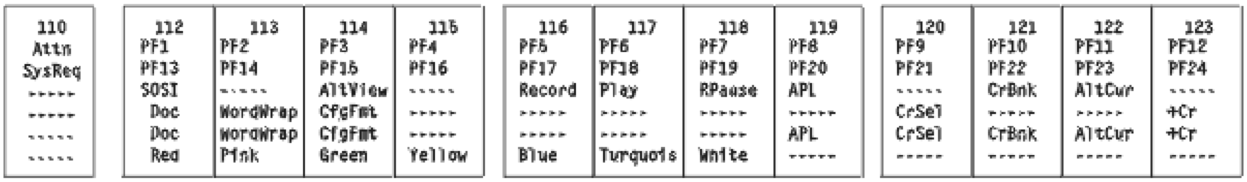


```
Segment D assignment:

 90        95        100       105
 -----     FldMk     Dup       Ansi 2D
 ~.~.~     FldMk     Dup       Ansi 2D
 ~.~.~     FldMk     ~~~~~     Field-
 -----     FldMk     -----     Field-
 -----     FldMk     -----     Field-
 -----     -----     -----     -----

 91        96        101       106
 Home      Up        RollDwn   Field+
 -----     -----     -----     Field+
 ~~~~~     MoveUp    ~~~~~     Field+
 ~~~~~     FastUp    ~~~~~     Field+
 -----     FastUp    -----     Field+
 -----     -----     -----     -----

 92        97        102
 Left      -----     Right
 -----     -----     -----
 MoveLft   ~~~~~     MoveRig
 ~~~~~     ~~~~~     ~~~~~
 -----     -----     -----
 -----     -----     -----

 93        98        103       108
 ErEOF     Down      Rollup    FldExt
 -----     -----     -----     FldExt
 ~~~~~     MoveDow   ~.~.~     ~~~~~
 ~~~~~     FastDow   ~.~.~     ~~~~~
 -----     FastDow   -----     -----
 -----     -----     -----     -----

           99          104
 Insert                Delete
 ~~~~~                 ~~~~~
 ~~~~~                 DelWord
 -----                 DelWord
 -----                 -----
 -----                 -----
```

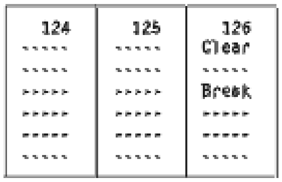
5576-A01 Keyboard (Combined 3270 and 5250)



Segment A assignment:

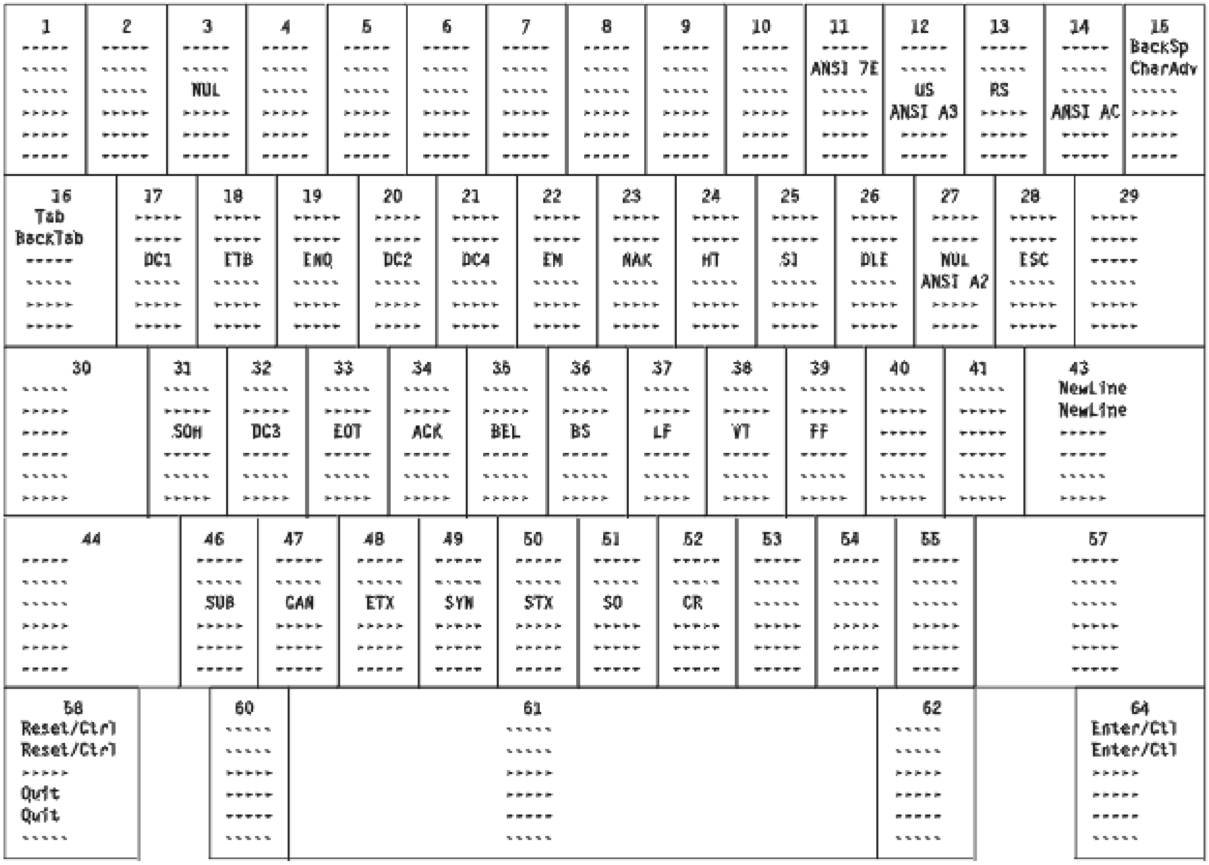


110	112	113	114	115	116	117	118	119	120	121	122	123
Attn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
>>>>>	SOSI	~~~~~	AltView	~~~~~	Record	Play	RPause	APL	~~~~~	CrBnk	AltCur	~~~~~
-----	Doc	WordWrap	CfgFmt	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	CrSel	>>>>>	>>>>>	+Cr
~~	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	APL	CrSel	CrBnk	AltCur	+Cr
~~	Red	Pink	Green	Yellow	Blue	Turquois	White	-----	-----	-----	-----	-----



124	125	126
~~	~~~~~	Clear
~~	~~~~~	~~~~~
>>>>>	>>>>>	Break
>>>>>	>>>>>	>>>>>
-----	-----	-----
~~	~~~~~	~~~~~

Segment B assignment:



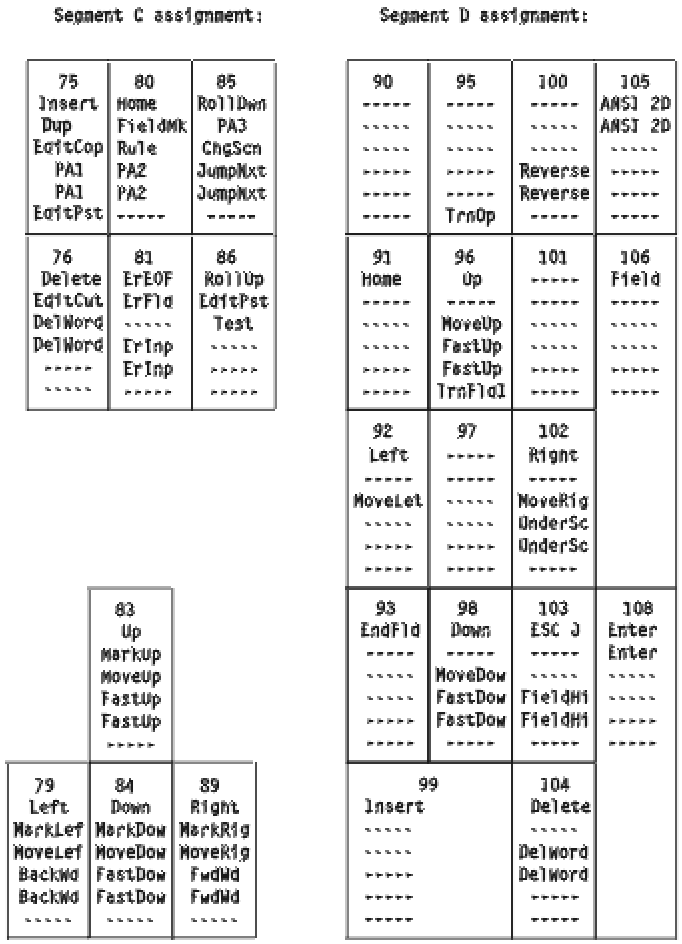
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	ANSI 7E	~~~~~	~~~~~	~~~~~	CharAdv
~~	~~~~~	NUL	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	US	RS	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	ANSI A3	>>>>>	ANSI AC	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	ANSI A2	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	NewLine
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~	~~~~~
~~	SUB	CAN	ETX	SYN	STX	SO	CR	~~~~~	~~~~~	~~~~~	~~~~~
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctl
Reset/Ctrl	~~~~~	~~~~~	~~~~~	Enter/Ctl
>>>>>	>>>>>	>>>>>	>>>>>	>>>>>
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
~~	~~~~~	~~~~~	~~~~~	~~~~~



Segment C assignment:

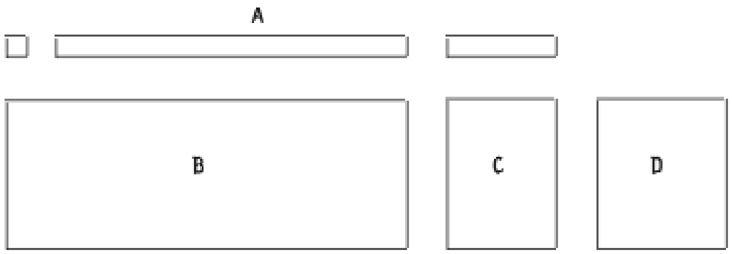
75 Insert Dup EditCop PA1 PA1 EditPst	80 Home FieldMk Rule PA2 PA2 -----	85 RollDwn PA3 ChgScn JumpNxt JumpNxt -----
76 Delete EditCut DelWord DelWord ----- -----	81 ErEOF ErFld ----- ErInp ErInp -----	86 RollUp EditPst Test ----- ----- -----

	83 Up MarkUp MoveUp FastUp FastUp -----	
79 Left MarkLef MoveLef BackWd BackWd -----	84 Down MarkDow MoveDow FastDow FastDow -----	89 Right MarkRig MoveRig FwdWd FwdWd -----

Segment D assignment:

90 ----- ----- ----- ----- ----- -----	95 ----- ----- ----- ----- ----- TrnOp	100 ----- ----- ----- Reverse Reverse -----	105 ANSI 2D ANSI 2D ----- ----- ----- -----
91 Home ----- ----- ----- ----- -----	96 Up ----- MoveUp FastUp FastUp TrnFldI	101 ----- ----- ----- ----- ----- -----	106 Field ----- ----- ----- ----- -----
92 Left ----- MoveLet ----- ----- -----	97 ----- ----- ----- ----- ----- -----	102 Right ----- MoveRig UnderSc UnderSc -----	
93 EndFld ----- ----- ----- ----- -----	98 Down ----- MoveDow FastDow FastDow -----	103 ESC J ----- ----- FieldHi FieldHi -----	108 Enter Enter ----- ----- ----- -----
99 Insert ----- ----- ----- ----- -----		104 Delete ----- DelWord DelWord ----- -----	

5576-002/003 (3270 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
Sysattn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysRq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	AltView	-----	Record	Play	RPause	APL	CrSel	CrBnk	RTM	-----
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	-----	-----	-----	AltCr	GrpCsr
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	APL	CrSel	CrBnk	AltCr	GrpCsr
-----	Red	Pink	Green	Yellow	Blue	Turquois	White	FieldCol	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	Break
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

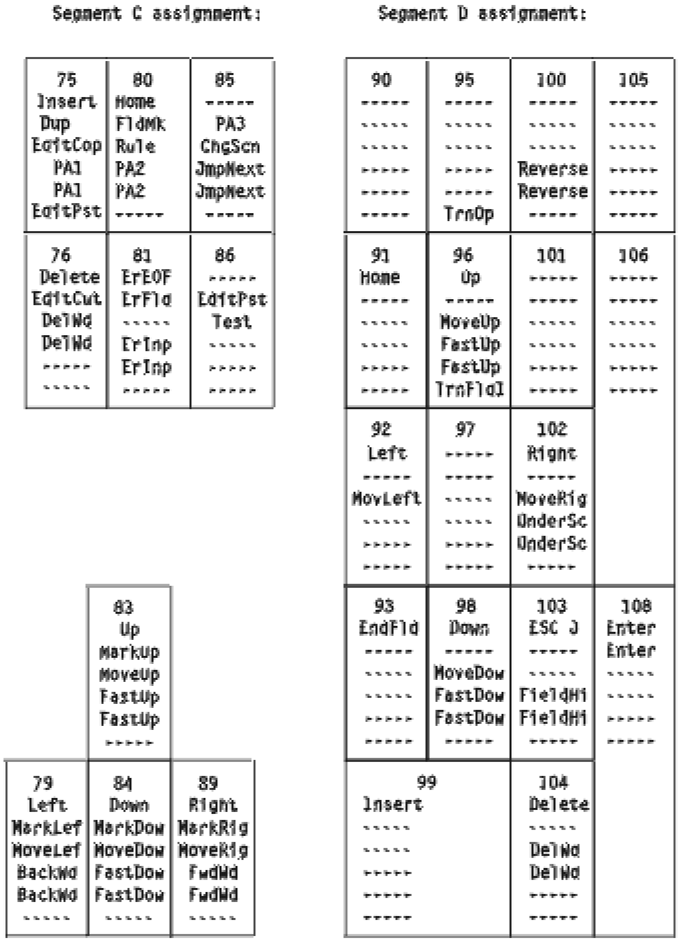
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	NUL	-----	-----	-----	RS	-----	-----	-----	-----	US	RS	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	EditUnd
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	SUB	CAN	ETX	SYN	STX	SO	CR	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctrl	-----	-----	-----	Entr/Ctl
Reset/Ctrl	-----	-----	-----	Entr/Ctl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----



Segment C assignment:

75	80	85
Insert	Home	-----
Dup	FldMk	PA3
EditCop	Rule	ChgScn
PA1	PA2	JmpNext
PA1	PA2	JmpNext
EditPst	-----	-----

76	81	86
Delete	ErEOF	-----
EditCut	ErFld	EditPst
DelWd	-----	Test
DelWd	ErInp	-----
-----	ErInp	-----
-----	-----	-----

83
Up
MarkUp
MoveUp
FastUp
FastUp

79	84	89
Left	Down	Right
MarkLef	MarkDow	MarkRig
MoveLef	MoveDow	MoveRig
BackWd	FastDow	FwdWd
BackWd	FastDow	FwdWd
-----	-----	-----

Segment D assignment:

90	95	100	105
-----	-----	-----	-----
-----	-----	-----	-----
-----	-----	-----	-----
-----	-----	Reverse	-----
-----	-----	Reverse	-----
-----	TrnOp	-----	-----

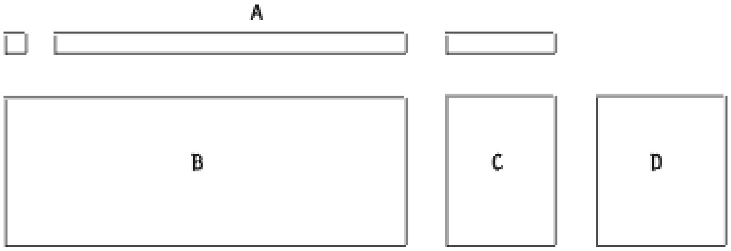
91	96	101	106
Home	Up	-----	-----
-----	-----	-----	-----
-----	MoveUp	-----	-----
-----	FastUp	-----	-----
-----	FastUp	-----	-----
-----	TrnFldI	-----	-----

92	97	102
Left	-----	Right
-----	-----	-----
MovLeft	-----	MoveRig
-----	-----	UnderSc
-----	-----	UnderSc
-----	-----	-----

93	98	103	108
EndFld	Down	ESC J	Enter
-----	-----	-----	Enter
-----	MoveDow	-----	-----
-----	FastDow	FieldHi	-----
-----	FastDow	FieldHi	-----
-----	-----	-----	-----

99	104
Insert	Delete
-----	-----
-----	DelWd
-----	DelWd
-----	-----
-----	-----

5576-002/003 Keyboard (5250 Only)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
SysAttn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	AltView	-----	Record	Play	Pause	SOSI/6	-----	-----	AltCur	-----
-----	Help	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	SOSI/6	-----	-----	AltCur	TestReq
-----	-----	-----	-----	-----	-----	-----	-----	FieldCol	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	HostPr
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

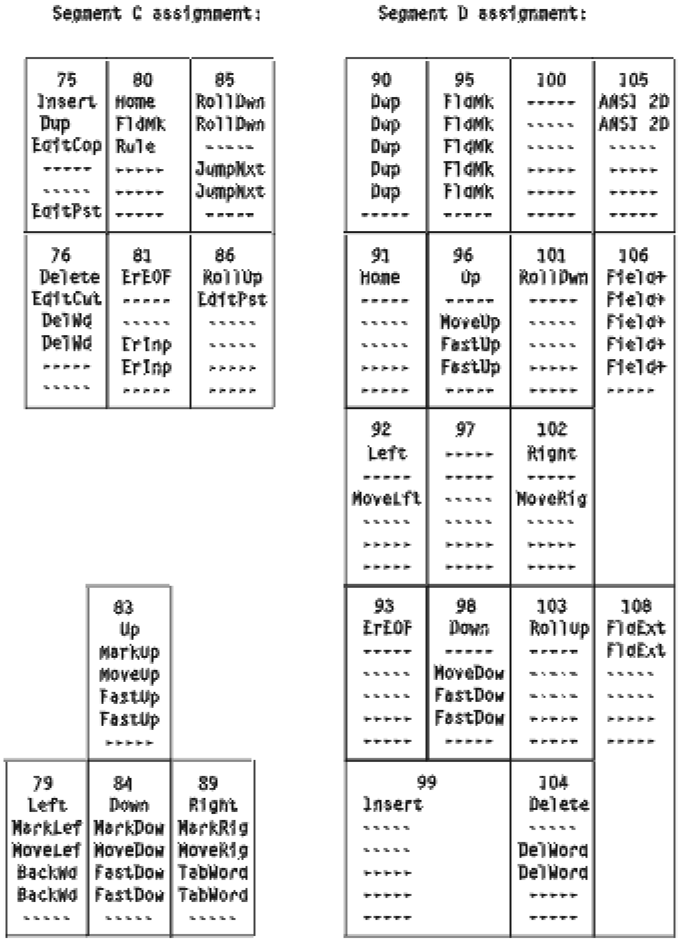
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	CharBsp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi 7E	-----	-----	-----	CharAdv
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	RS	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi A3	-----	ANSI AC	EditUnd
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	Ansi A2	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	FieldExt
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	FieldExt
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

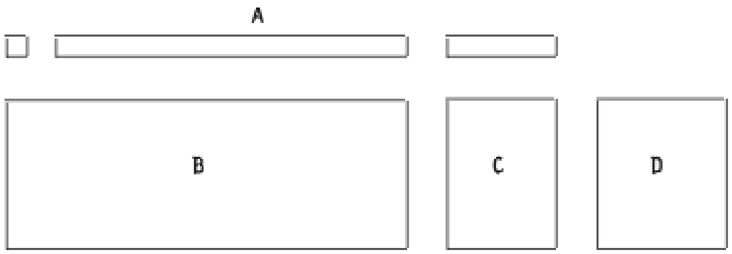
58	60	61	62	64
Reset/Ctl	-----	-----	-----	Entr/Ctl
Reset/Ctl	-----	-----	-----	Entr/Ctl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----
-----	-----	-----	-----	-----



```
Segment C assignment:

 75        80        85
 Insert    Home      RollDwn
 Dup       FldMk     RollDwn
 EditCop   Rule      ~~~~~
 -----     -----     JumpNxt
 ~~~~~     -----     JumpNxt
 EditPst   -----     -----

 76        81        86
 Delete    ErEOF     RollUp
 EditCut   -----     EditPst
 DelWd     ~~~~~     ~~~~~
 DelWd     ErInp     ~~~~~
 -----     ErInp     -----
 ~~~~~     -----     -----

           83
           Up
           MarkUp
           MoveUp
           FastUp
           FastUp
           -----

 79        84        89
 Left      Down      Right
 MarkLef   MarkDow   MarkRig
 MoveLef   MoveDow   MoveRig
 BackWd    FastDow   TabWord
 BackWd    FastDow   TabWord
 ~~~~~     ~~~~~     ~~~~~
```

```
Segment D assignment:

 90        95        100       105
 Dup       FldMk     -----     ANSI 2D
 Dup       FldMk     ~~~~~     ANSI 2D
 Dup       FldMk     ~~~~~     ~~~~~
 Dup       FldMk     -----     -----
 Dup       FldMk     -----     -----
 -----     -----     -----     -----

 91        96        101       106
 Home      Up        RollDwn   Field+
 -----     -----     -----     Field+
 ~~~~~     MoveUp    ~~~~~     Field+
 ~~~~~     FastUp    ~~~~~     Field+
 -----     FastUp    -----     Field+
 -----     -----     -----     -----

 92        97        102
 Left      -----     Right
 -----     -----     -----
 MoveLft   ~~~~~     MoveRig
 ~~~~~     ~~~~~     ~~~~~
 -----     -----     -----
 -----     -----     -----

 93        98        103       108
 ErEOF     Down      RollUp    FldExt
 -----     -----     -----     FldExt
 ~~~~~     MoveDow   ~~~~~     ~~~~~
 ~~~~~     FastDow   ~~~~~     ~~~~~
 -----     FastDow   -----     -----
 -----     -----     -----     -----

       99            104
 Insert              Delete
 ~~~~~               ~~~~~
 ~~~~~               DelWord
 -----               DelWord
 -----               -----
 -----               -----
```

5576-002/003 Keyboard (Combined 3270 and 5250)



Segment A assignment:

110	112	113	114	115	116	117	118	119	120	121	122	123
Attn	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10	PF11	PF12
SysReq	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20	PF21	PF22	PF23	PF24
-----	SOSI	-----	AltView	-----	Record	Play	Pause	APL	-----	CrBnk	AltCur	-----
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	-----	CrSel	-----	-----	+Cr
-----	Doc	WordWrap	CfgFmt	-----	-----	-----	-----	APL	CrSel	CrBnk	AltCur	+Cr
-----	Red	Pink	Green	Yellow	Blue	Turquois	White	-----	-----	-----	-----	-----

124	125	126
-----	-----	Clear
-----	-----	-----
-----	-----	Break
-----	-----	-----
-----	-----	-----
-----	-----	-----

Segment B assignment:

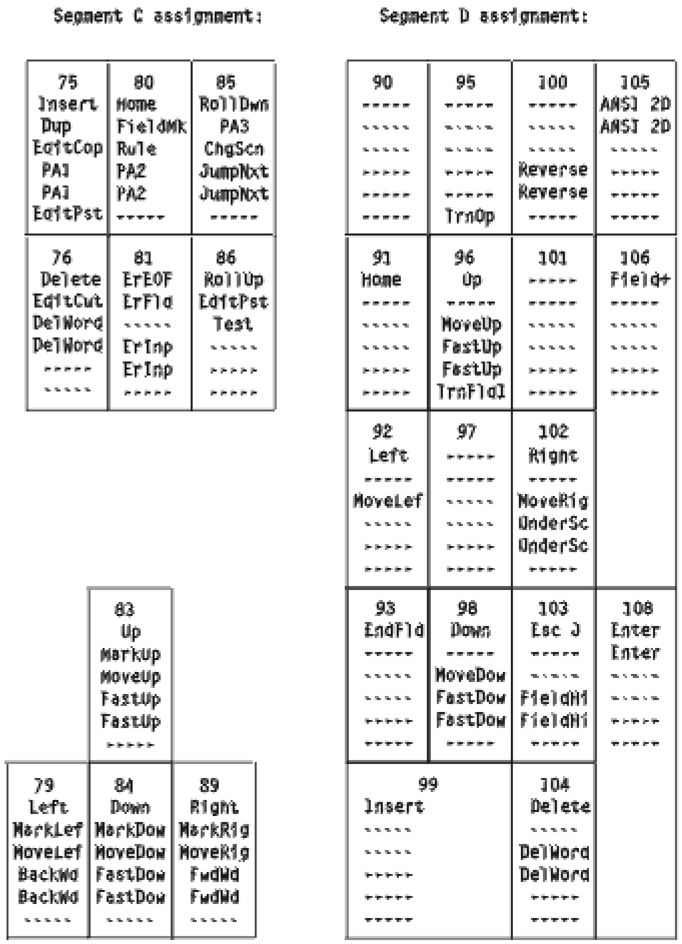
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	BackSp
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	ANSI 7E	-----	-----	-----	CharAdv
-----	-----	NUL	-----	-----	-----	-----	-----	-----	-----	-----	US	RS	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	ANSI A3	-----	ANSI AC	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

16	17	18	19	20	21	22	23	24	25	26	27	28	29
Tab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
BackTab	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	DC1	ETB	ENQ	DC2	DC4	EM	NAK	HT	SI	DLE	NUL	ESC	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	ANSI A2	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

30	31	32	33	34	35	36	37	38	39	40	41	43
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	NewLine
-----	SOH	DC3	EOT	ACK	BEL	BS	LF	VT	FF	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

44	46	47	48	49	50	51	52	53	54	55	57
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	SUB	CAN	ETX	SYN	STX	SO	CR	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

58	60	61	62	64
Reset/Ctl	-----	-----	-----	Entr/Ctl
Reset/Ctl	-----	-----	-----	Entr/Ctl
-----	-----	-----	-----	-----
Quit	-----	-----	-----	-----
Quit	-----	-----	-----	-----
-----	-----	-----	-----	-----


Segment C assignment:
75 Insert Dup EditCop PA1 PA1 EditPst
80 Home FieldMk Rule PA2 PA2
85 RollDwn PA3 ChgScn JumpNxt JumpNxt
76 Delete EditCut DelWord DelWord
81 ErEOF ErFld ErInp ErInp
86 RollUp EditPst Test
83 Up MarkUp MoveUp FastUp FastUp
79 Left MarkLef MoveLef BackWd BackWd
84 Down MarkDow MoveDow FastDow FastDow
89 Right MarkRig MoveRig FwdWd FwdWd
Segment D assignment:
90
95 TrnOp
100 Reverse Reverse
105 ANSI 2D ANSI 2D
91 Home
96 Up MoveUp FastUp FastUp TrnFldI
101
106 Field+
92 Left MoveLef
97
102 Right MoveRig UnderSc UnderSc
93 EndFld
98 Down MoveDow FastDow FastDow
103 Esc J FieldHi FieldHi
108 Enter Enter
99 Insert
104 Delete DelWord DelWord

APL Keyboard Layouts (3270 only)

This section shows the APL keyboard layouts for each country. Only APL characters are shown in each shift position (up/down/ALT).

Language	Type
Austrian/German	1
Belgian	2
Canadian-Bilingual	1
Denmark	1
Finnish/Swedish	1
French AZERTY	2
Hangeul	4
Italian	1
Japanese	4

Language	Type
Latin American Spanish	1
Norwegian	1
Portuguese	1
Simplified Chinese	4
Spanish	1
Swiss-French	1
Swiss-German	1
Traditional Chinese	4
U.K. English	1
U.S. English	3

Figure 2. Type-1 APL Keyboard on page 1169

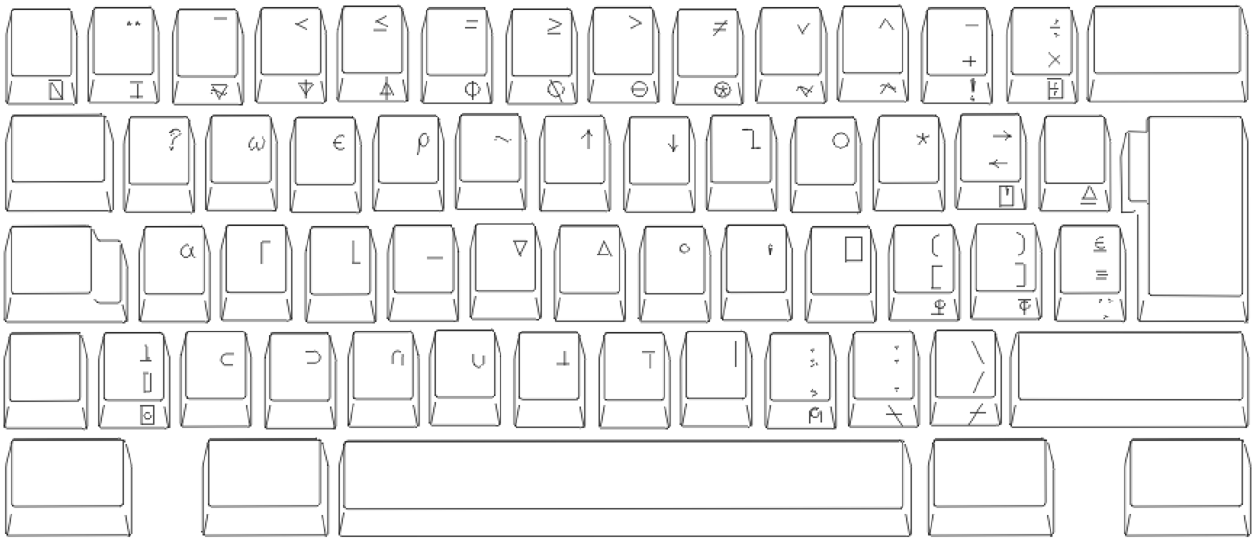


Figure 3. Type-2 APL Keyboard on page 1169

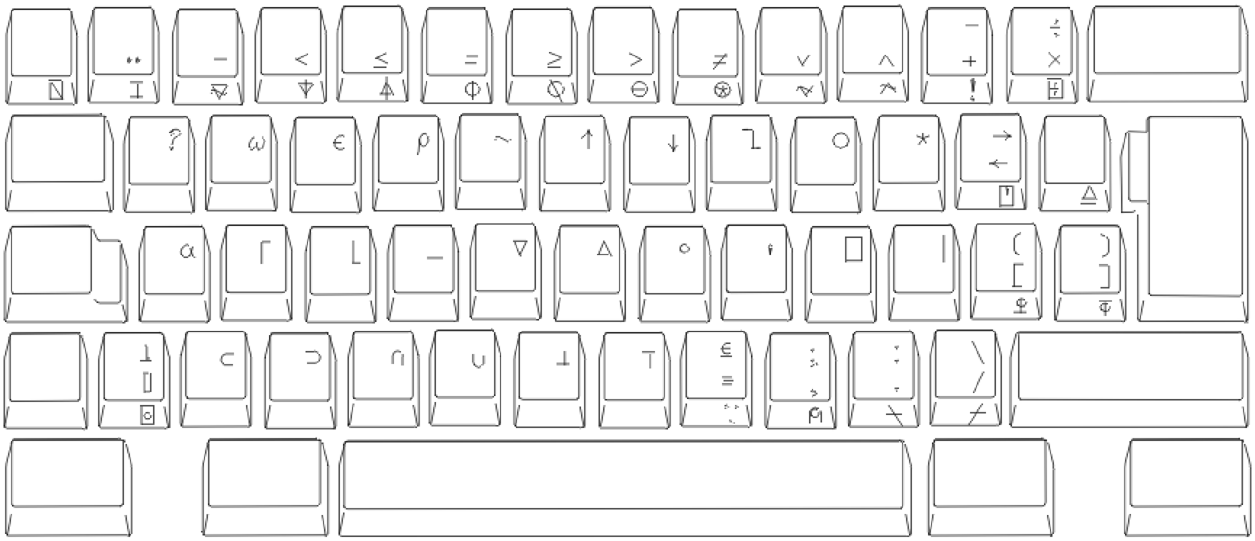


Figure 4. Type-3 APL Keyboard on page 1169

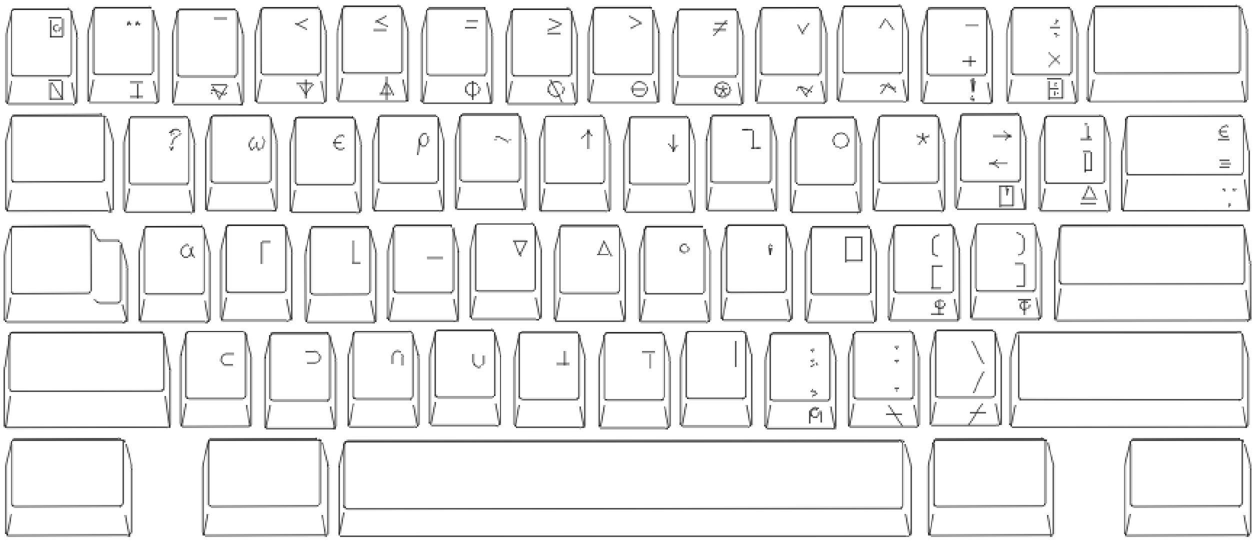
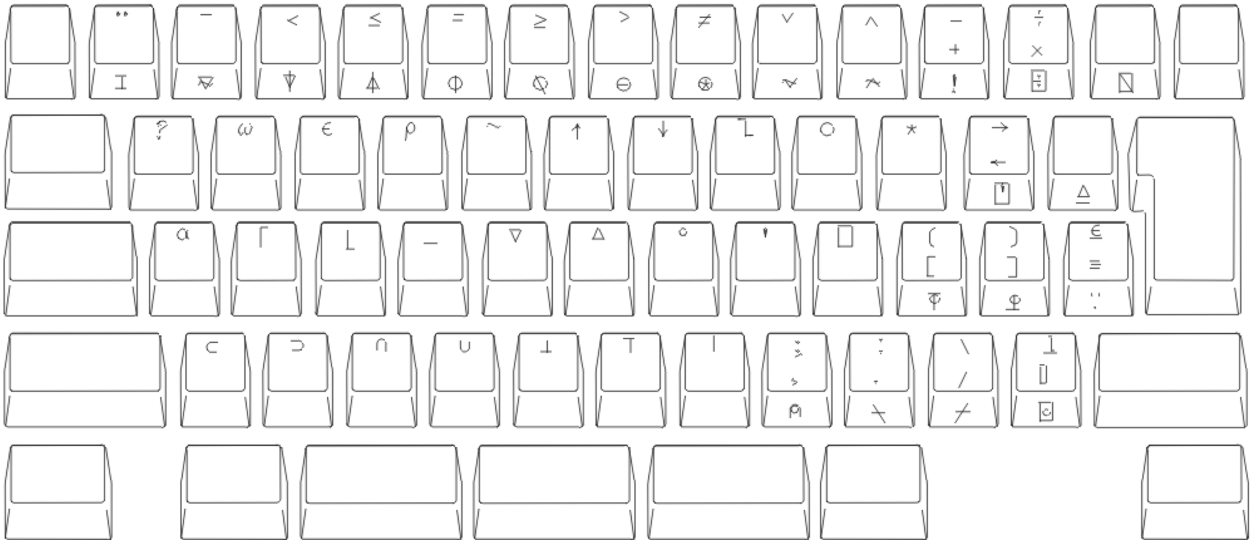


Figure 5. Type-4 APL Keyboard on page 1169



[Top of Page on page 1169 | Previous Page on page 1169 | Next Page on page 1212 | Table of Contents on page 1168]

Keyboard Layout and Mapping Reference

Keyboard Mapping

This chapter describes keyboard mapping supported by the host system. The keyboards are selected during installation or customization procedures. You can use keyboard layouts to locate character positions on the keyboard. Refer to *Emulator User's Reference* for more information.

For keys with three or four characters shown, use the key combinations in Table 1 on page 1212 to produce the desired upper-right and lower-right characters. Lower-left characters require no additional keys. Use the shift key for upper-left characters.

Table 1. Key Combinations for Upper-Right and Lower-Right Characters on page 1169

Country	Lower-Right Character Enhanced Template [1]	Upper-Right Character All Templates
Denmark	AltGr	Ctrl
Finland	AltGr	Ctrl
Norway	AltGr	Ctrl
Sweden	AltGr	Ctrl
All others	AltGr	None

Note:

[1] Alt and AltGr keys produce the same character on the Enhanced keyboard in DOS mode.

Local Edit Keys

Table 2 on page 1213 shows the keys used by local editing. These keys behave differently between the local editing mode and the interactive editing (ECHO) mode.

Table 2. Local Edit Keys on page 1169

VT Key Mnemonics	Local Edit Function	Action in Local Edit Mode
VT Find	Home Cursor	Moves the cursor to the top left position on a page in memory.
VT Insert	Insert/ Override	Selects whether new characters typed displace existing characters to the right, or replace existing characters.
VT Remove	Clear	Clears an unprotected field of all characters. Press in conjunction with the SHIFT key to clear all unprotected fields in the scrolling region.
VT Select	Edit	Press in conjunction with the SHIFT key to enter or exit local edit mode. The status line shows the current state.
VT Prev	PrevPage	Moves the cursor to the beginning of the previous page.
VT Next	NextPage	Moves the cursor to the beginning of the next page.
VT PF1	Tab	Advances the cursor to the first occurrence of: • A tab stop at the beginning of an unprotected field • An unprotected field • The end of the scrolling region Pressing SHIFT with this key moves the cursor back to the first occurrence of: • The previous tab stop • The beginning of the current unprotected field • The beginning of the previous unprotected field • The beginning of the scrolling region
VT PF2	Insert Line	Adds a blank line on the screen and moves the following lines down. This key cannot be selected on a line containing a protected field.
VT PF3	DeleteLine	Deletes a line from the screen and moves the following lines up. This key cannot be selected on a line containing a protected field.
VT PF4	DeleteChar	Deletes an unprotected character at the cursor.
VT Numpad Enter Newline	Transmit	Sends a block of edited text to the host. Works like the Transmit key, if you select Line Transmit.

Key Map for Home3270

This section shows the position of the following control codes: NUL (X'00') ESC (X'1B') FS (X'1C') GS (X'1D') RS (X'1E') US (X'1F')

The positions of these control codes are fixed and common for all languages (including U.S. English), regardless of the characters assigned to BASE and UP SHIFT positions of the keys. These control codes are positioned to the CONTROL positions of the keys shown in Figure 6 on page 1214 through Figure 7 on page 1214.

Figure 6. Common Control Code of Keyboard Core Segment for Non-Japanese Keyboard on page 1169

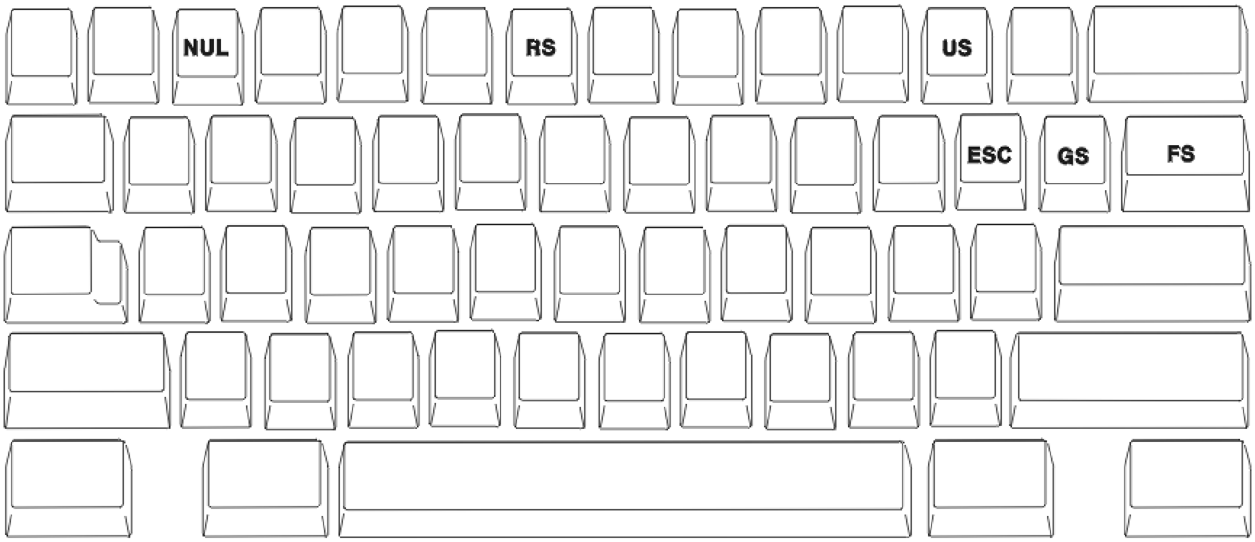
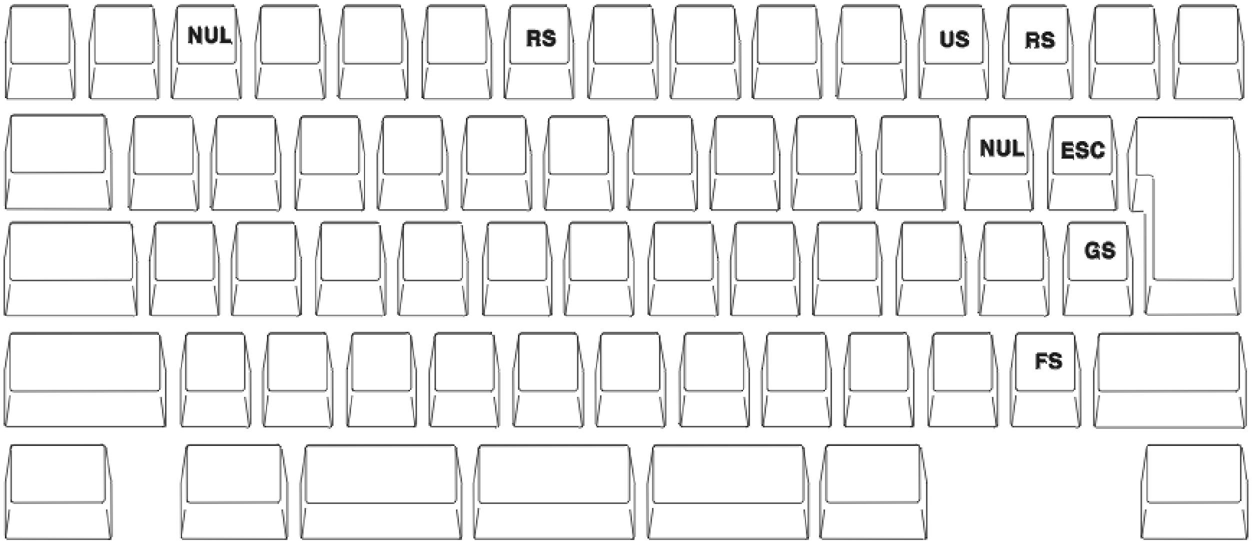


Figure 7. Common Control Code of Keyboard Core Segment for Japanese Keyboard on page 1169



The positions of other control codes vary for each language. These control codes are positioned to the CONTROL position of the associated alphabetic key. Those control codes and the associated alphabetic characters are as follows.

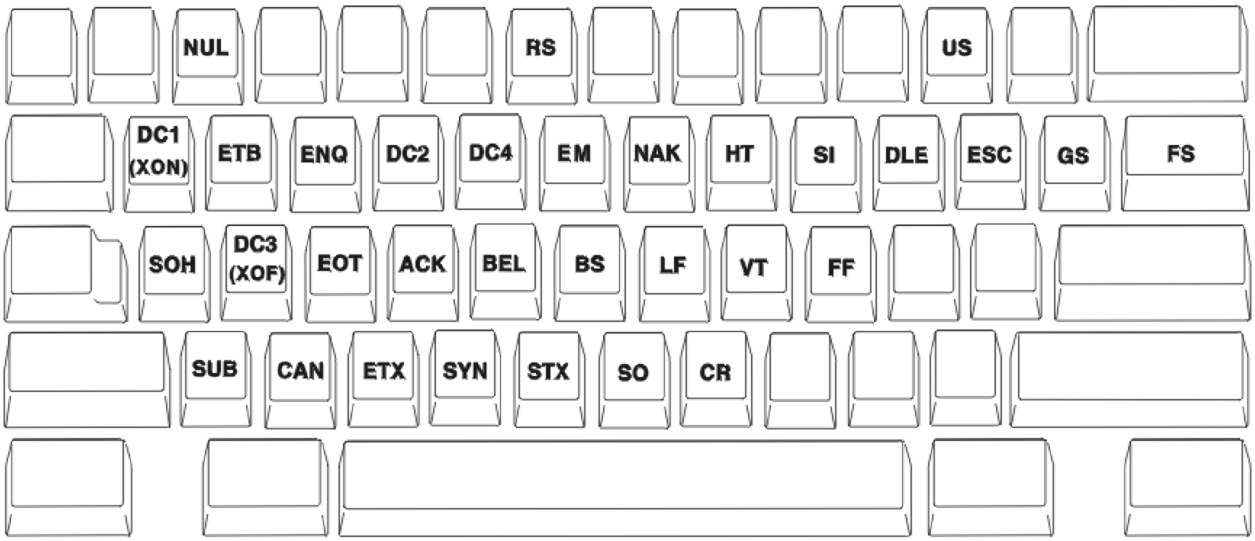
Table 3. Other Control Code Map of Key Segment B on page 1169

Control Code	Associated Letter
SOH (X'01')	a

Control Code	Associated Letter
STX (X'02')	b
ETX (X'03')	c
EOT (X'04')	d
ENQ (X'05')	e
ACK (X'06')	f
BEL (X'07')	g
BS (X'08')	h
HT (X'09')	i
LF (X'0A')	j
VT (X'0B')	k
FF (X'0C')	l
CR (X'0D')	m
SO (X'0E')	n
SI (X'0F')	o
DLE (X'10')	p
DC1(XON) (X'11')	q
DC2 (X'12')	r
DC3(XOF) (X'13')	s
DC4 (X'14')	t
NAK (X'15')	u
SYN (X'16')	v
ETB (X'17')	w
CAN (X'18')	x
EM (X'19')	y
SUB (X'1A')	z

Following is an example of control-code mapping for the U.S. Enhanced Keyboard.

Figure 8. Control-Code Map of Keyboard Core Segment for U.S. Enhanced Keyboard on page 1169



[Top of Page on page 1212 | Previous Page on page 1169 | Next Page | Table of Contents on page 1168]

Host Code Page Reference

Contents

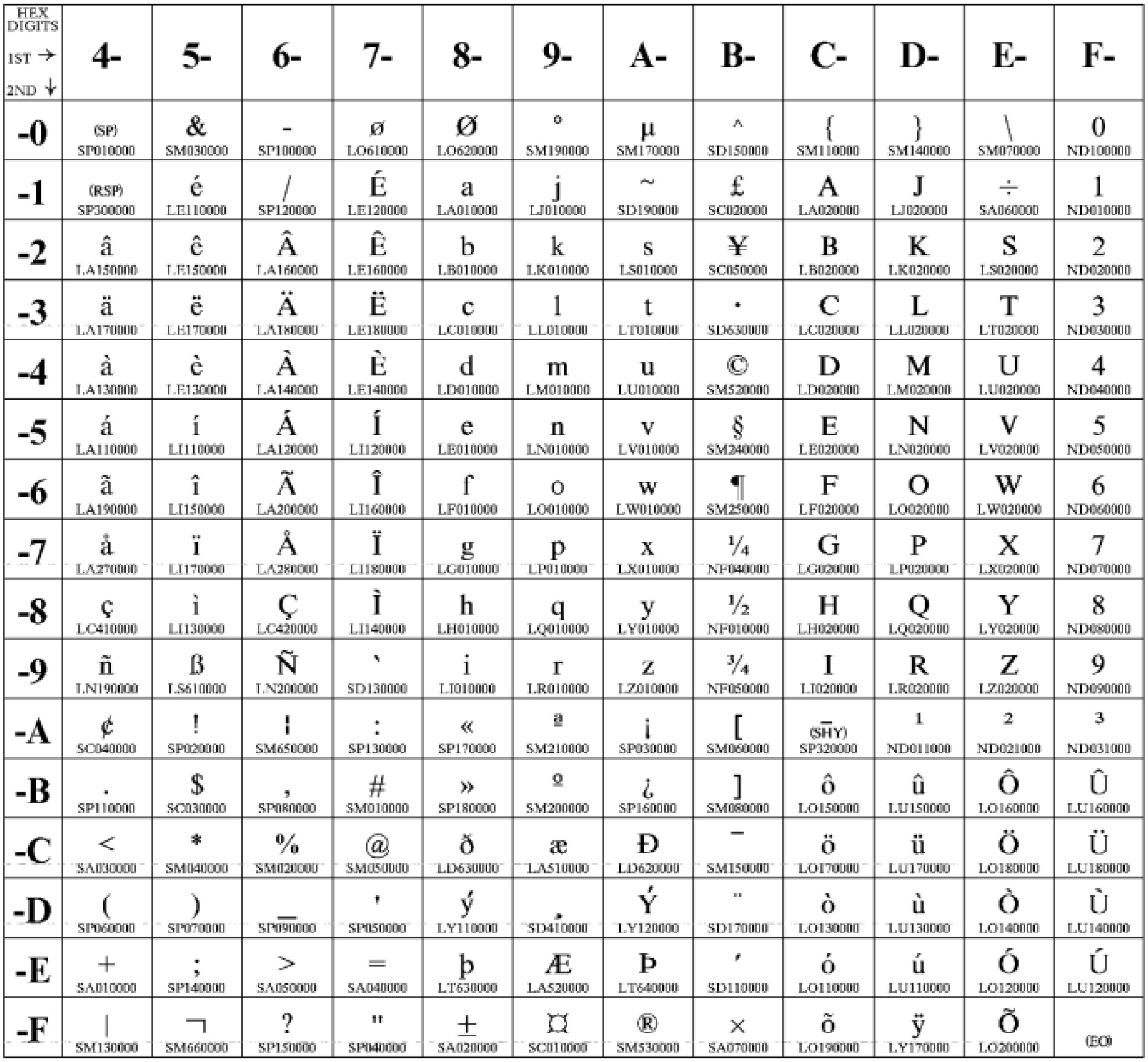
Host Code Page Tables

[Top of Page on page 1216 | Previous Page | Next Page | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 037-1/697-1 Brazil, Canada, Netherlands, Portugal, U.S., and 037/1175 Traditional Chinese

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	[SM060000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	] SM080000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

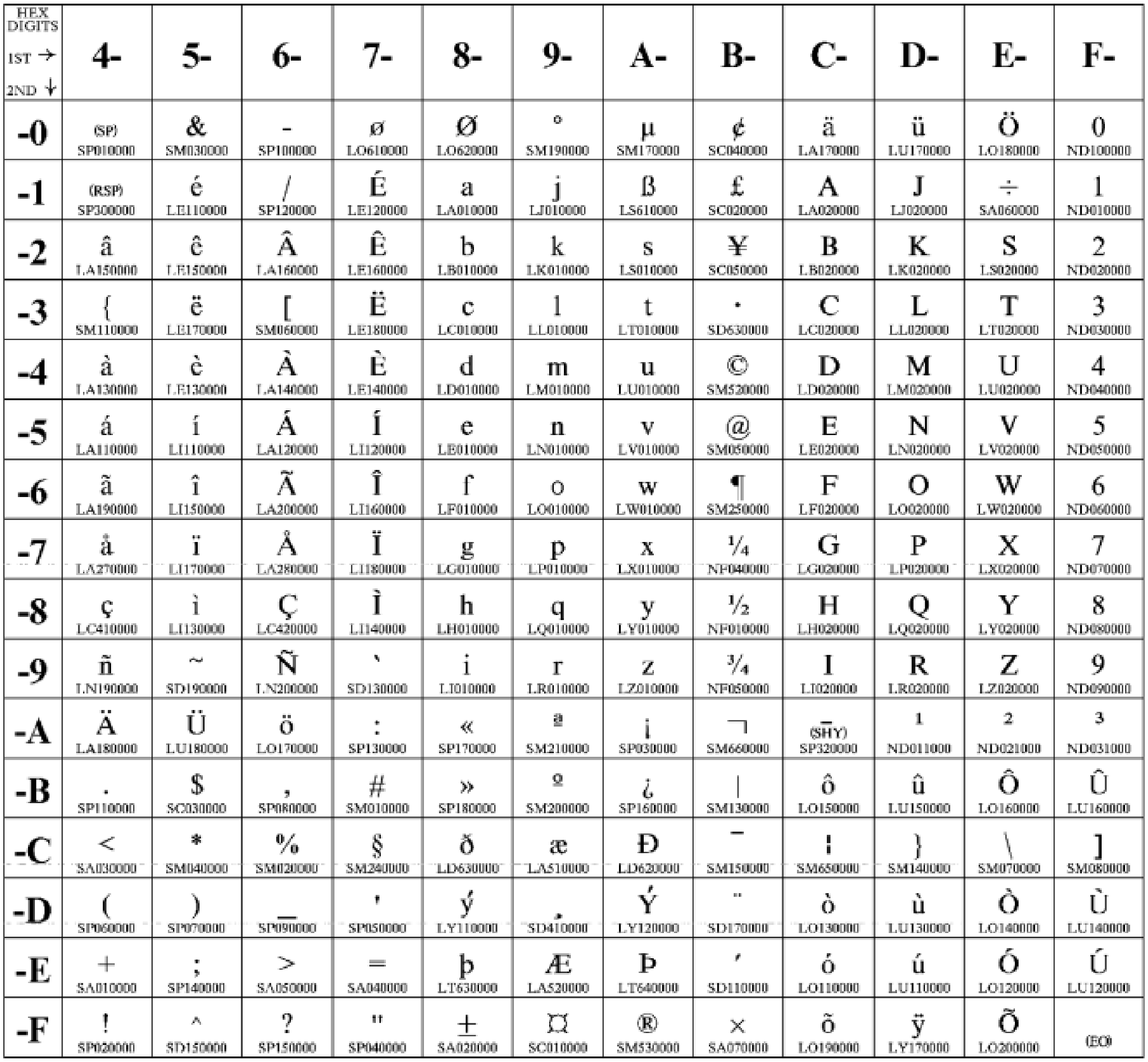
Code Page 00037

[Top of Page on page 1217 | **Previous Page** | Next Page on page 1218 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 273-1/697-1 Austria, Germany

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	ü LU170000	Ö LO180000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ß LS610000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	[SM060000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	@ SM050000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	~ SD190000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ä LA180000	Ü LU180000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	§ SM240000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	¦ SM650000	} SM140000	\\ SM070000	] SM080000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

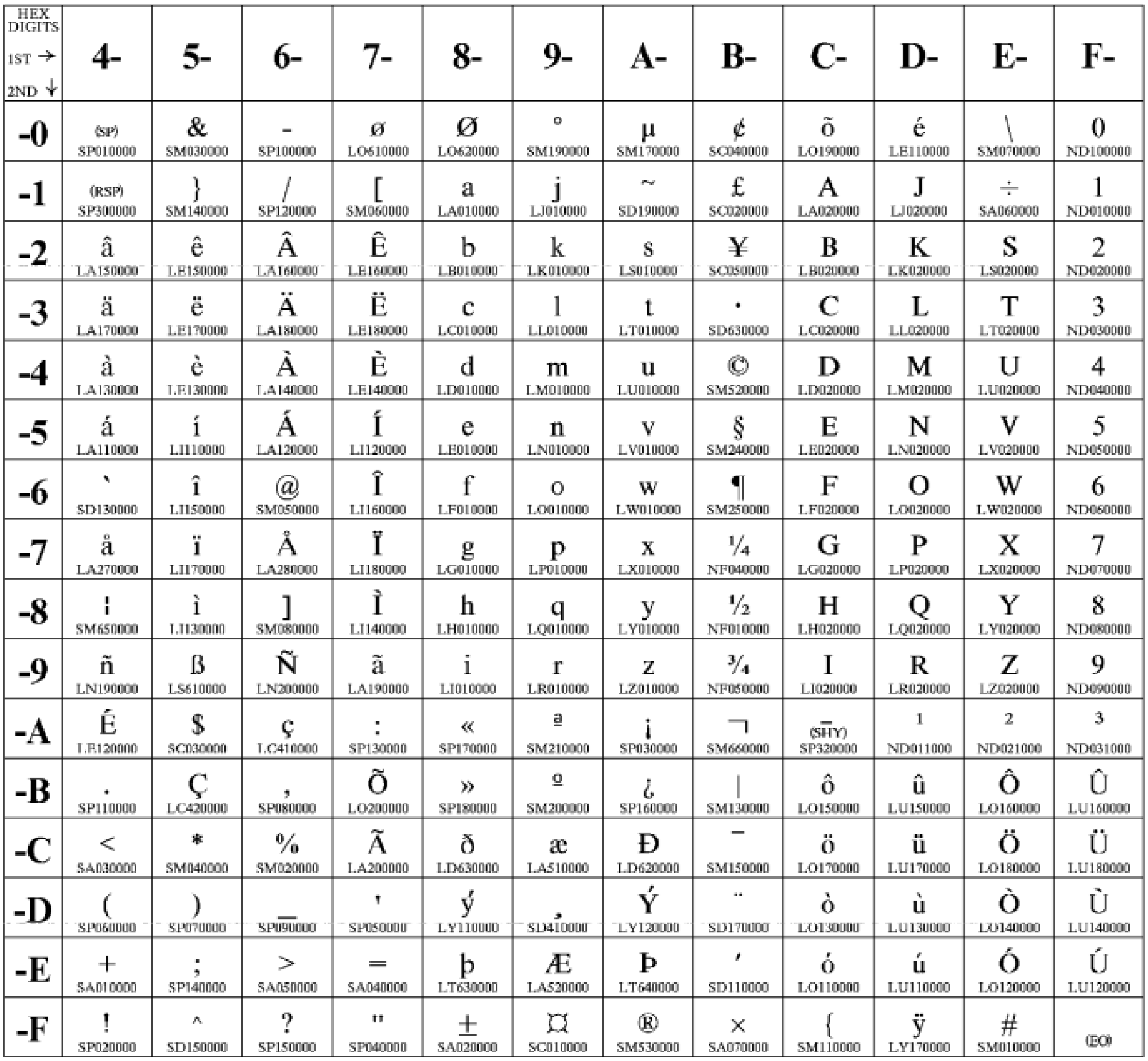
Code Page 00273

[Top of Page on page 1218 | Previous Page on page 1217 | Next Page on page 1219 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 275-1/697-1 Brazil

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	õ LO190000	é LE110000	\ SM070000	0 ND100000
-1	(RSP) SP300000	} SM140000	/ SP120000	[SM060000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	` SD130000	î LI150000	@ SM050000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	¦ SM650000	ì LI130000	] SM080000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ã LA190000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	É LE120000	$ SC030000	ç LC410000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Ç LC420000	, SP080000	Õ LO200000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ã LA200000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	{ SM110000	ÿ LY170000	# SM010000	(EO)

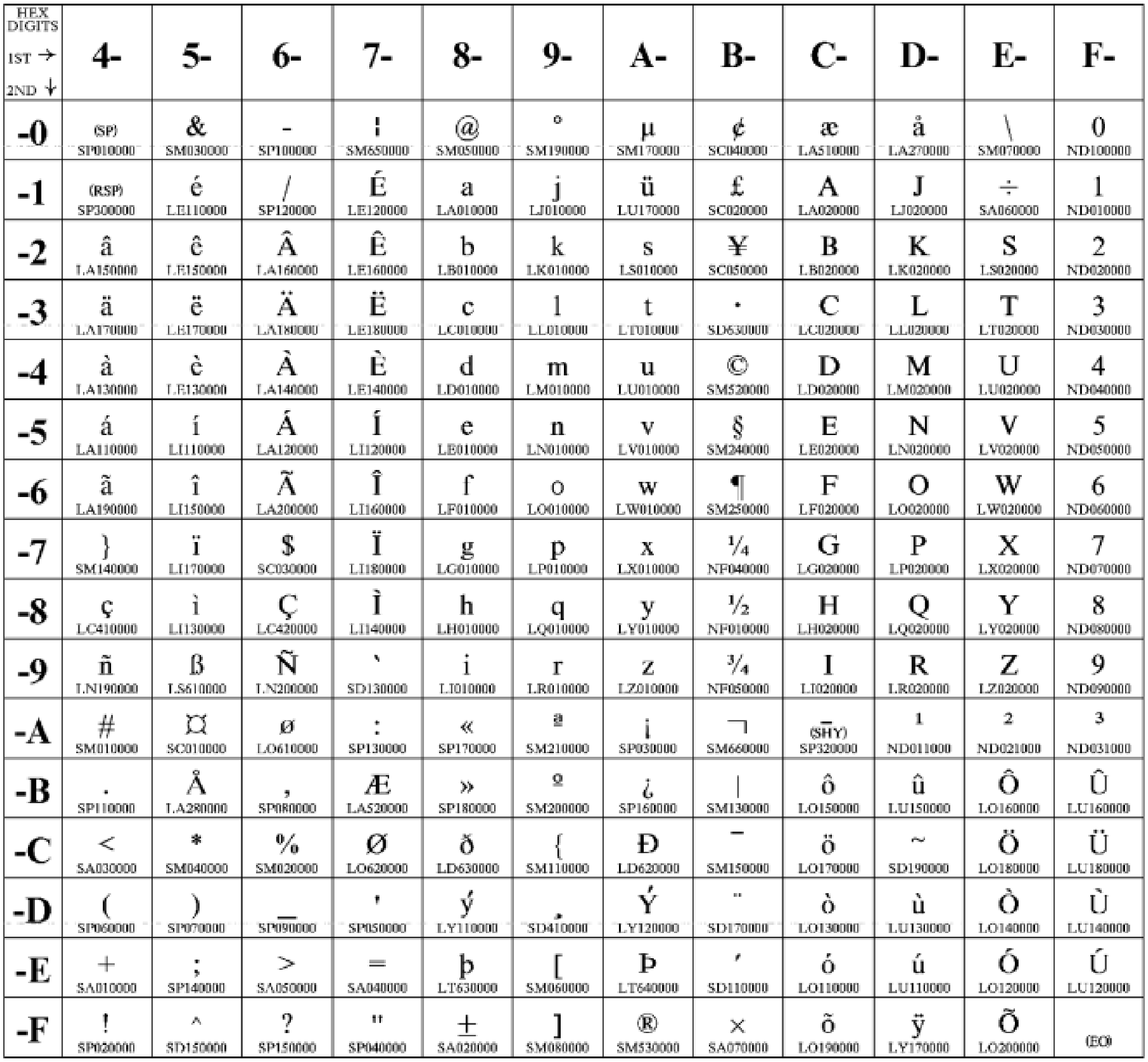
Code Page 00275

[Top of Page on page 1219 | Previous Page on page 1218 | Next Page on page 1220 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 277-1/697-1 Denmark, Norway

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	¦ SM650000	@ SM050000	° SM190000	µ SM170000	¢ SC040000	æ LA510000	å LA270000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	ƒ LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	# SM010000	¤ SC010000	ø LO610000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Æ LA520000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ø LO620000	ð LD630000	{ SM110000	Ð LD620000	¯ SM150000	ö LO170000	~ SD190000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	[SM060000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

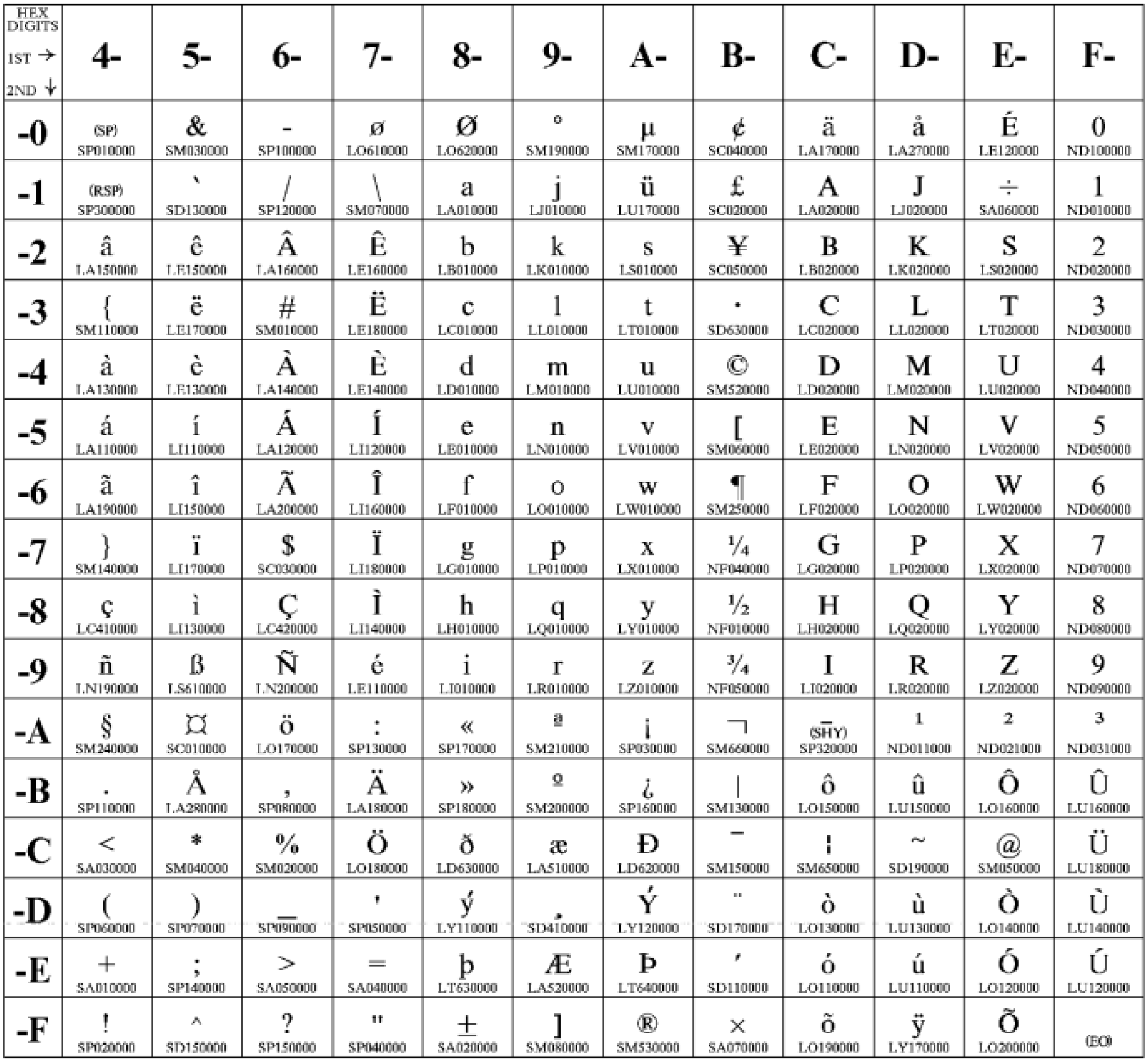
Code Page 00277

[Top of Page on page 1220 | Previous Page on page 1219 | Next Page on page 1221 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 278-1/697-1 Finland, Sweden

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	å LA270000	É LE120000	0 ND100000
-1	(RSP) SP300000	` SD130000	/ SP120000	\ SM070000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	# SM010000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	[SM060000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	é LE110000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	§ SM240000	¤ SC010000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Ä LA180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ö LO180000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	¦ SM650000	~ SD190000	@ SM050000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

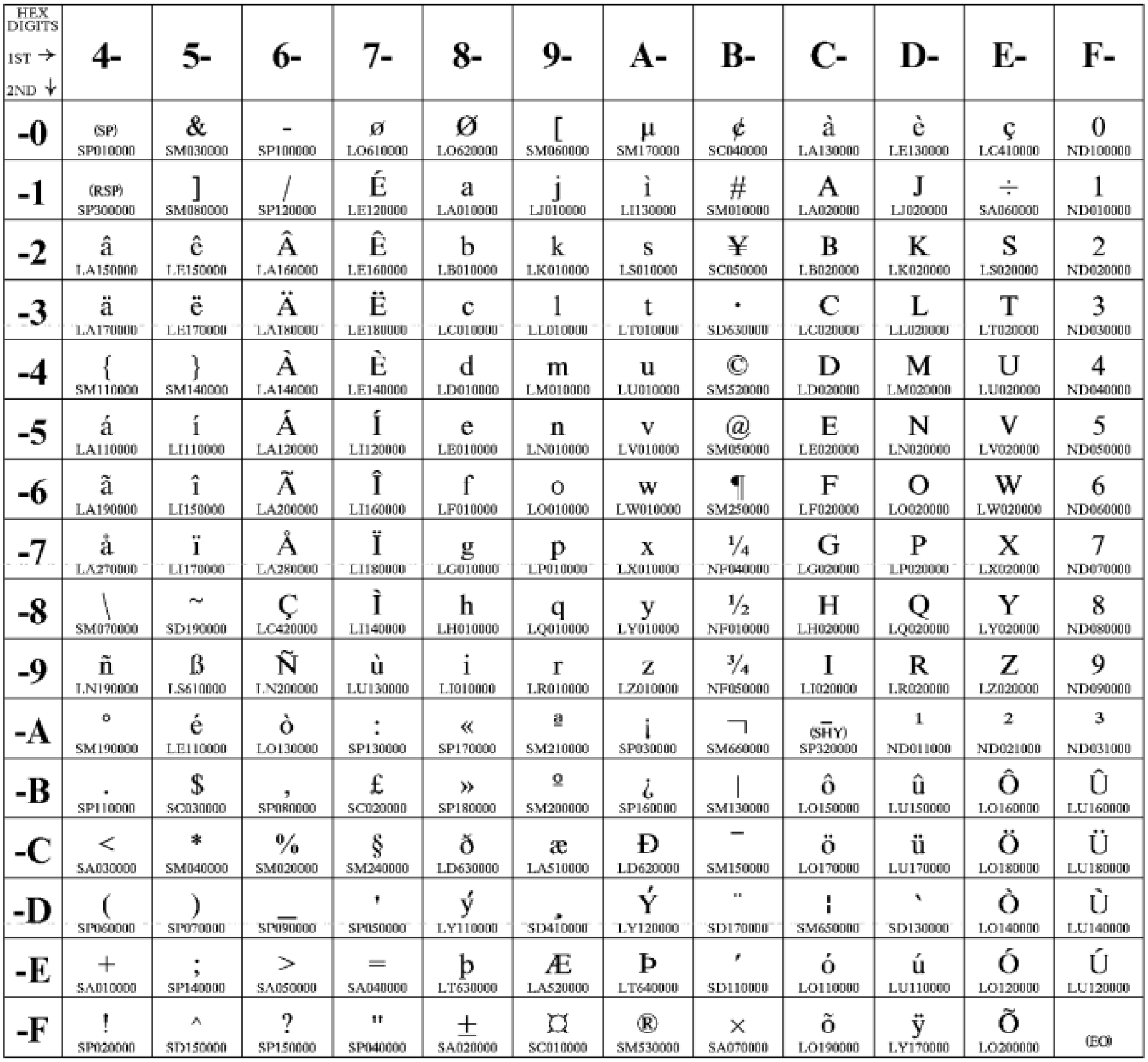
Code Page 00278

[Top of Page on page 1221 | Previous Page on page 1220 | Next Page on page 1222 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 280-1/697-1 Italy

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	[SM060000	µ SM170000	¢ SC040000	à LA130000	è LE130000	ç LC410000	0 ND100000
-1	(RSP) SP300000	] SM080000	/ SP120000	É LE120000	a LA010000	j LJ010000	ì LI130000	# SM010000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	{ SM110000	} SM140000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	@ SM050000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	\\ SM070000	~ SD190000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ù LU130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	° SM190000	é LE110000	ò LO130000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	£ SC020000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	§ SM240000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	¦ SM650000	` SD130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

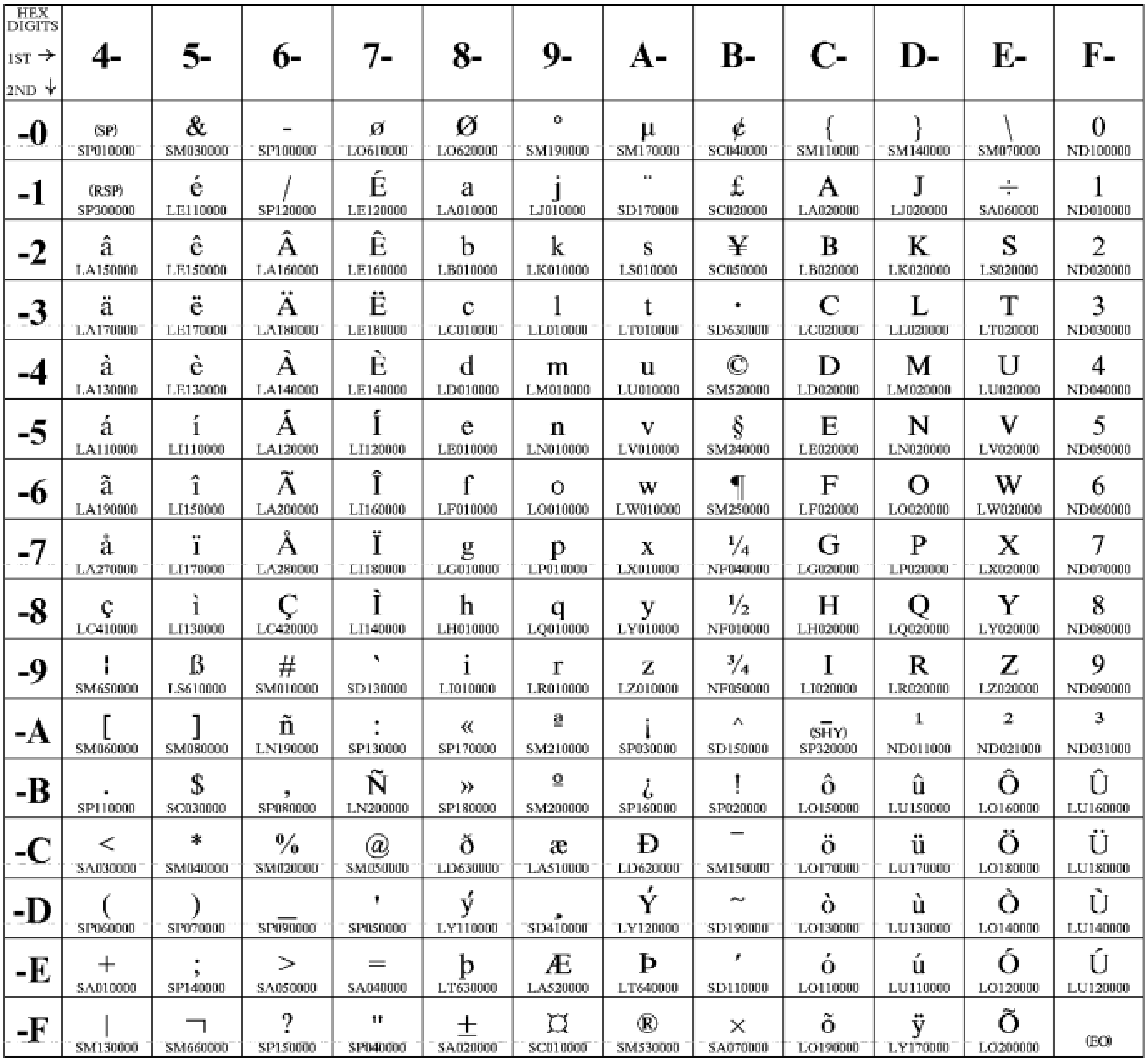
Code Page 00280

[Top of Page on page 1222 | Previous Page on page 1221 | Next Page on page 1223 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 284-1/697-1 Latin America, Spain

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¨ SD170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	¦ SM650000	ß LS610000	# SM010000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	ñ LN190000	: SP130000	« SP170000	ª SM210000	¡ SP030000	^ SD150000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	Ñ LN200000	» SP180000	º SM200000	¿ SP160000	! SP020000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	~ SD190000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

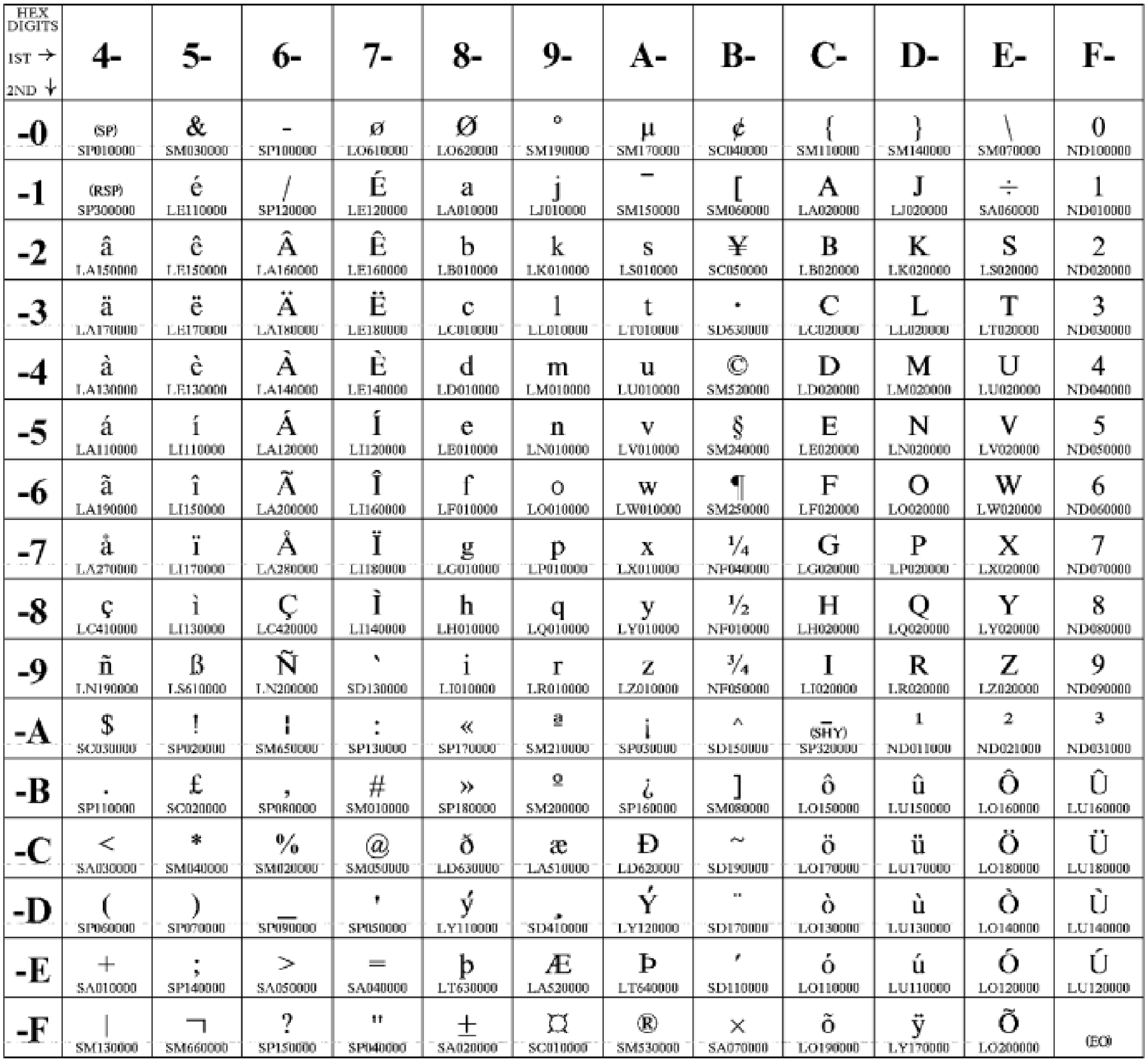
Code Page 00284

[Top of Page on page 1223 | Previous Page on page 1222 | Next Page on page 1224 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 285-1/697-1 United Kingdom

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¯ SM150000	[SM060000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	$ SC030000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	^ SD150000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	£ SC020000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	] SM080000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	~ SD190000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

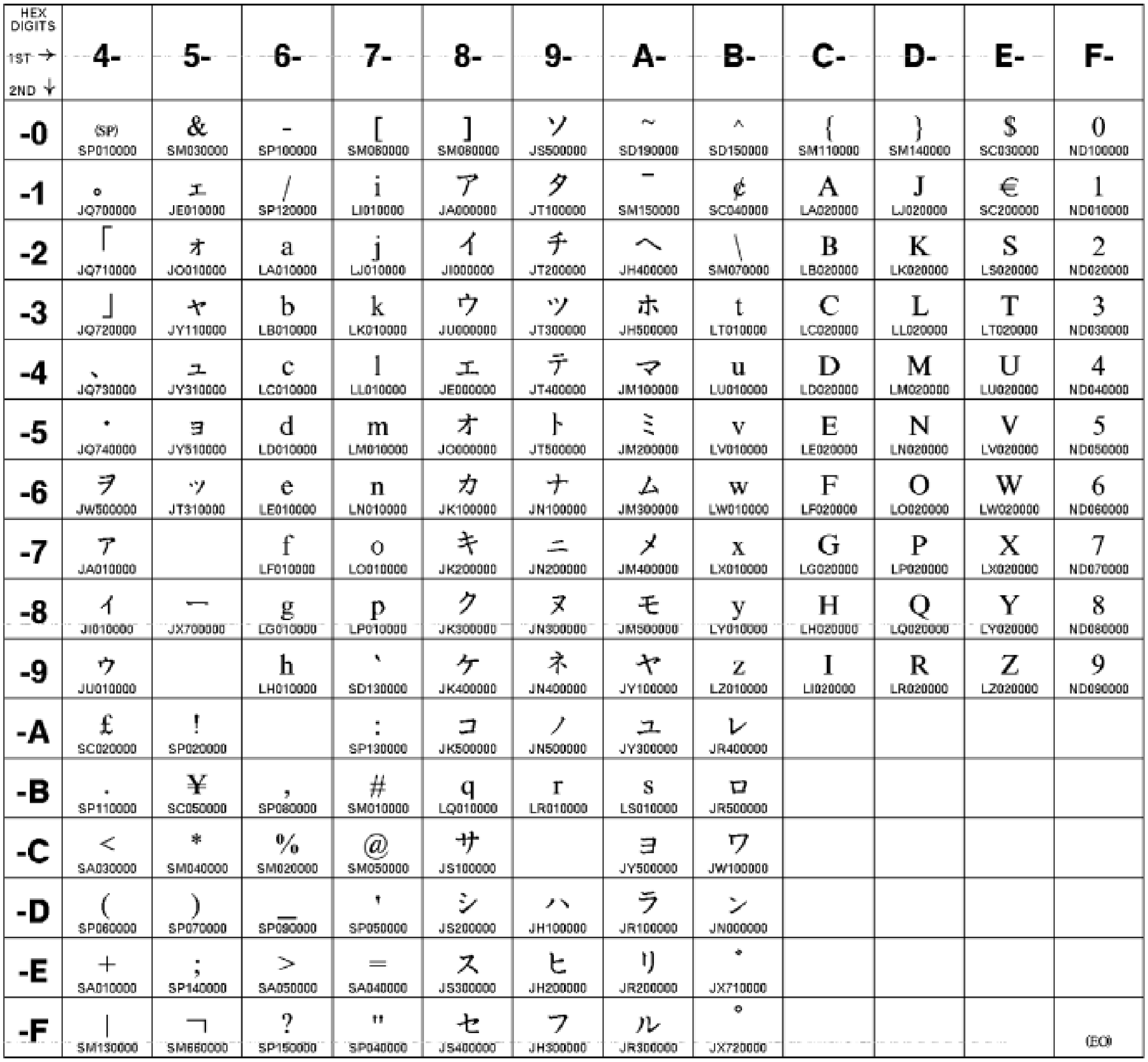
Code Page 00285

[Top of Page on page 1224 | Previous Page on page 1223 | Next Page on page 1225 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 290/930 Japan (Katakana) Extended

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	[SM060000	] SM080000	ｿ JS500000	~ SD190000	^ SD150000	{ SM110000	} SM140000	$ SC030000	0 ND100000
-1	｡ JQ700000	ｪ JE010000	/ SP120000	i LI010000	ｱ JA000000	ﾀ JT100000	‾ SM150000	¢ SC040000	A LA020000	J LJ020000	€ SC200000	1 ND010000
-2	｢ JQ710000	ｫ JO010000	a LA010000	j LJ010000	ｲ JI000000	ﾁ JT200000	ﾍ JH400000	\ SM070000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	｣ JQ720000	ｬ JY110000	b LB010000	k LK010000	ｳ JU000000	ﾂ JT300000	ﾎ JH500000	t LT010000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	､ JQ730000	ｭ JY310000	c LC010000	l LL010000	ｴ JE000000	ﾃ JT400000	ﾏ JM100000	u LU010000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	･ JQ740000	ｮ JY510000	d LD010000	m LM010000	ｵ JO000000	ﾄ JT500000	ﾐ JM200000	v LV010000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ｦ JW500000	ｯ JT310000	e LE010000	n LN010000	ｶ JK100000	ﾅ JN100000	ﾑ JM300000	w LW010000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	ｧ JA010000		f LF010000	o LO010000	ｷ JK200000	ﾆ JN200000	ﾒ JM400000	x LX010000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ｨ JI010000	ｰ JX700000	g LG010000	p LP010000	ｸ JK300000	ﾇ JN300000	ﾓ JM500000	y LY010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ｩ JU010000		h LH010000	` SD130000	ｹ JK400000	ﾈ JN400000	ﾔ JY100000	z LZ010000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	£ SC020000	! SP020000		: SP130000	ｺ JK500000	ﾉ JN500000	ﾕ JY300000	ﾚ JR400000				
-B	. SP110000	¥ SC050000	, SP080000	# SM010000	q LQ010000	r LR010000	s LS010000	ﾛ JR500000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ｻ JS100000		ﾖ JY500000	ﾜ JW100000				
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ｼ JS200000	ﾊ JH100000	ﾗ JR100000	ﾝ JN000000				
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ｽ JS300000	ﾋ JH200000	ﾘ JR200000	ﾞ JX710000				
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ｾ JS400000	ﾌ JH300000	ﾙ JR300000	ﾟ JX720000				(EO)

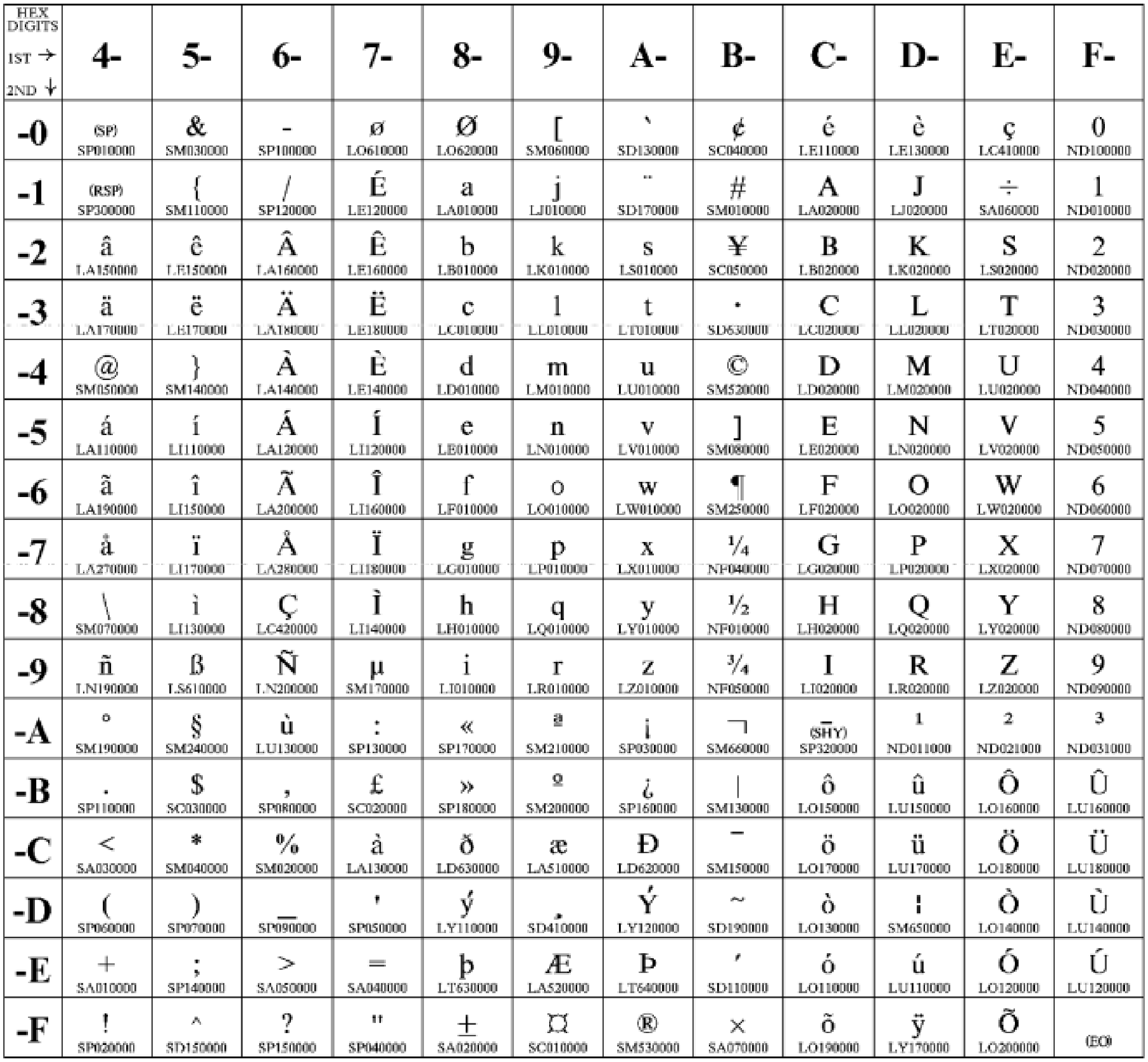
Code Page 00290

[Top of Page on page 1225 | Previous Page on page 1224 | Next Page on page 1226 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 297-1/697-1 France

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	[SM060000	` SD130000	¢ SC040000	é LE110000	è LE130000	ç LC410000	0 ND100000
-1	(RSP) SP300000	{ SM110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¨ SD170000	# SM010000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	@ SM050000	} SM140000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	] SM080000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	\ SM070000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	µ SM170000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	° SM190000	§ SM240000	ù LU130000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	£ SC020000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	à LA130000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	~ SD190000	ò LO130000	¦ SM650000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

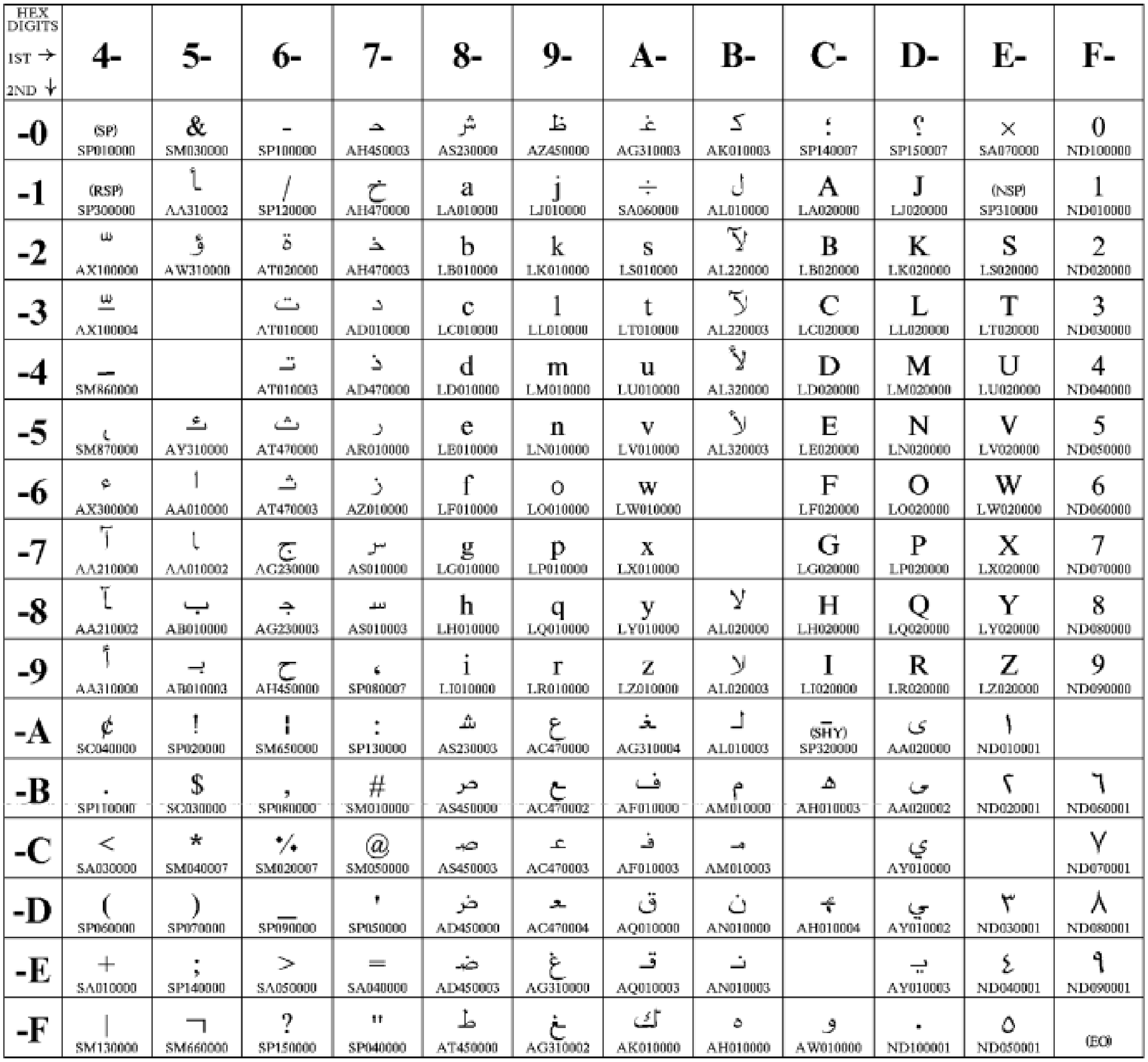
Code Page 00297

[Top of Page on page 1226 | Previous Page on page 1225 | Next Page on page 1227 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 420 Arabic Bilingual

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ﺣ AH450003	ش AS230000	ظ AZ450000	ﻏ AG310003	ﻛ AK010003	؛ SP140007	؟ SP150007	× SA070000	0 ND100000
-1	(RSP) SP300000	ﺄ AA310002	/ SP120000	خ AH470000	a LA010000	j LJ010000	÷ SA060000	ل AL010000	A LA020000	J LJ020000	(NSP) SP310000	1 ND010000
-2	ّ AX100000	ؤ AW310000	ة AT020000	ﺧ AH470003	b LB010000	k LK010000	s LS010000	ﻵ AL220000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ـّ AX100004		ت AT010000	د AD010000	c LC010000	l LL010000	t LT010000	ﻶ AL220003	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ـ SM860000		ﺗ AT010003	ذ AD470000	d LD010000	m LM010000	u LU010000	ﻷ AL320000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	ٕ SM870000	ﺋ AY310000	ث AT470000	ر AR010000	e LE010000	n LN010000	v LV010000	ﻸ AL320003	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ء AX300000	ا AA010000	ﺛ AT470003	ز AZ010000	f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7	آ AA210000	ﺎ AA010002	ج AG230000	س AS010000	g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8	ﺂ AA210002	ب AB010000	ﺟ AG230003	ﺳ AS010003	h LH010000	q LQ010000	y LY010000	ﻻ AL020000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	أ AA310000	ﺑ AB010003	ح AH450000	، SP080007	i LI010000	r LR010000	z LZ010000	ﻼ AL020003	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	ﺷ AS230003	ع AC470000	ﻐ AG310004	ﻟ AL010003	(SHY) SP320000	ى AA020000	١ ND010001	
-B	. SP110000	$ SC030000	, SP080000	# SM010000	ص AS450000	ﻊ AC470002	ف AF010000	م AM010000	ﻫ AH010003	ﻰ AA020002	٢ ND020001	٦ ND060001
-C	< SA030000	* SM040007	% SM020007	@ SM050000	ﺻ AS450003	ﻋ AC470003	ﻓ AF010003	ﻣ AM010003		ي AY010000		٧ ND070001
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ض AD450000	ﻌ AC470004	ق AQ010000	ن AN010000	ﻬ AH010004	ﻲ AY010002	٣ ND030001	٨ ND080001
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ﺿ AD450003	غ AG310000	ﻗ AQ010003	ﻧ AN010003		ﻳ AY010003	٤ ND040001	٩ ND090001
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ط AT450000	ﻎ AG310002	ك AK010000	ه AH010000	و AW010000	٠ ND100001	٥ ND050001	(EO)

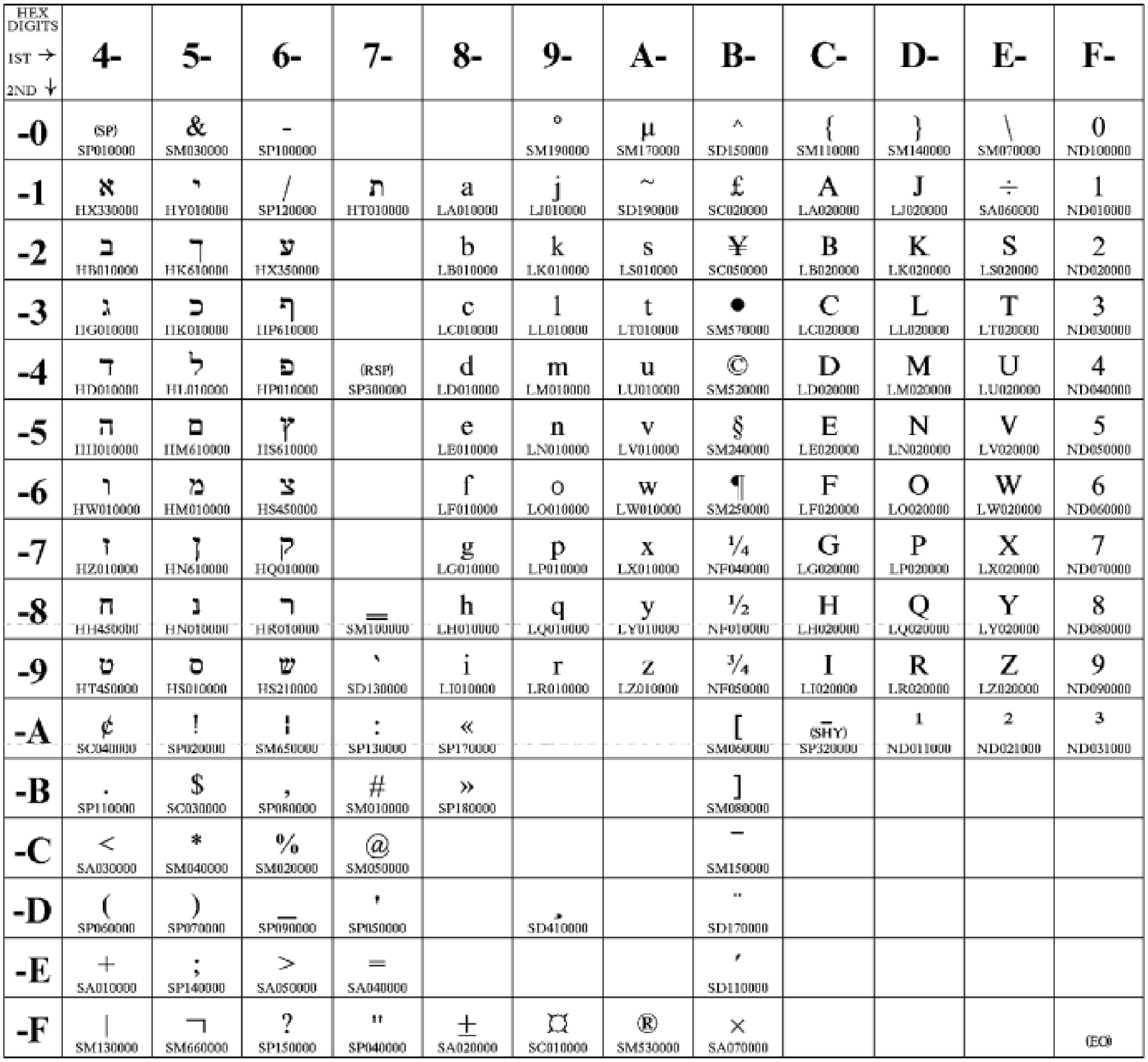
Code Page 00420

[Top of Page on page 1227 | Previous Page on page 1226 | Next Page on page 1228 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 424/941 Israel (Hebrew - Bulletin Code)

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000			° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	א HX330000	י HY010000	/ SP120000	ת HT010000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	ב HB010000	ך HK610000	ע HX350000		b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ג HG010000	כ HK010000	ף HP610000		c LC010000	l LL010000	t LT010000	● SM570000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ד HD010000	ל HL010000	פ HP010000	(RSP) SP300000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	ה HH010000	ם HM610000	ץ HS610000		e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ו HW010000	מ HM010000	צ HS450000		f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	ז HZ010000	ן HN610000	ק HQ010000		g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ח HH450000	נ HN010000	ר HR010000	‗ SM100000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ט HT450000	ס HS010000	ש HS210000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000			[SM060000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000			] SM080000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000				¯ SM150000				
-D	(SP060000	) SP070000	_ SP090000	' SP050000		¸ SD410000		¨ SD170000				
-E	+ SA010000	; SP140000	> SA050000	= SA040000				´ SD110000				
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000				(EO)

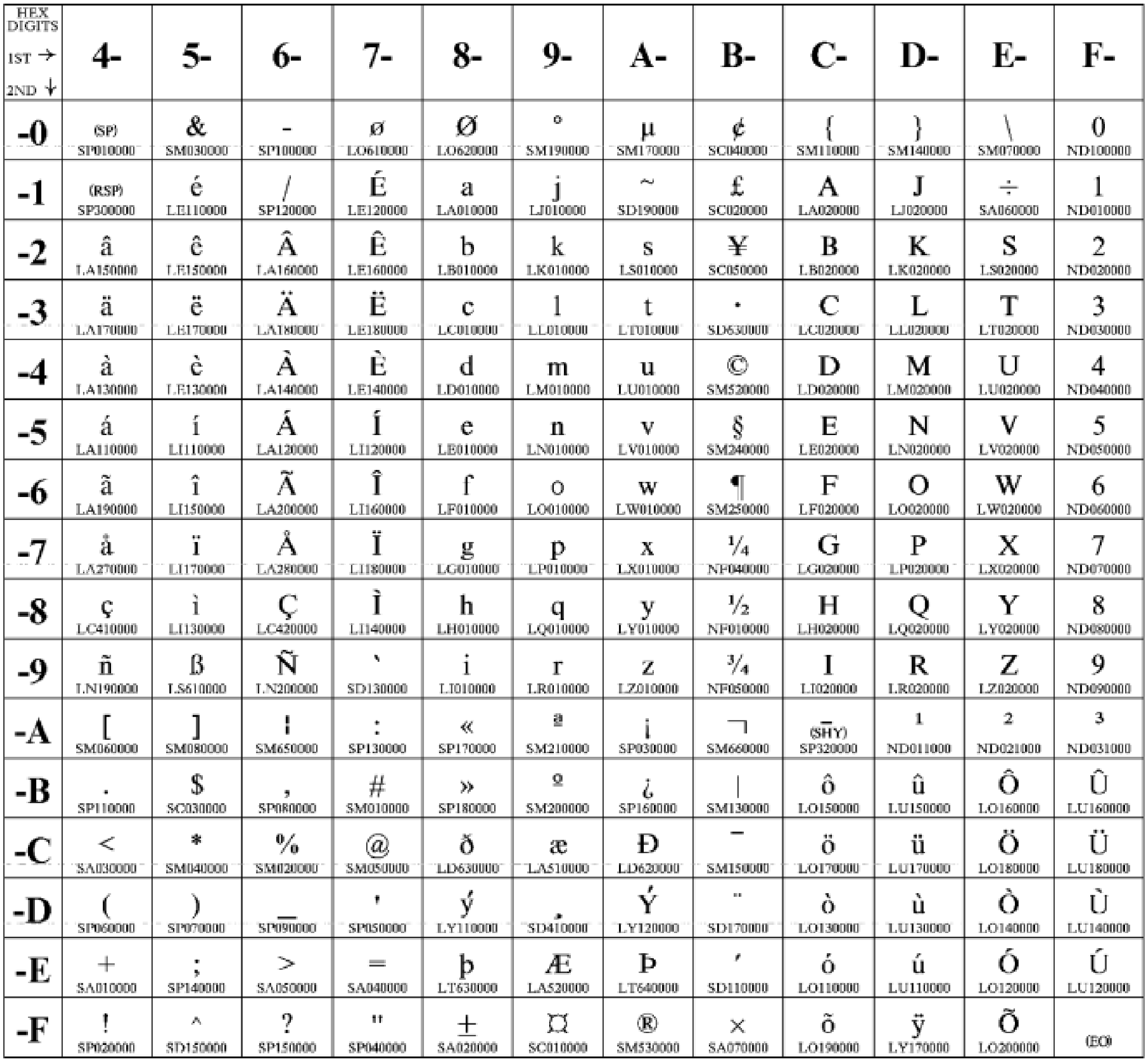
Code Page 00424

[Top of Page on page 1228 | Previous Page on page 1227 | Next Page on page 1229 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 500-1/697-1 International

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

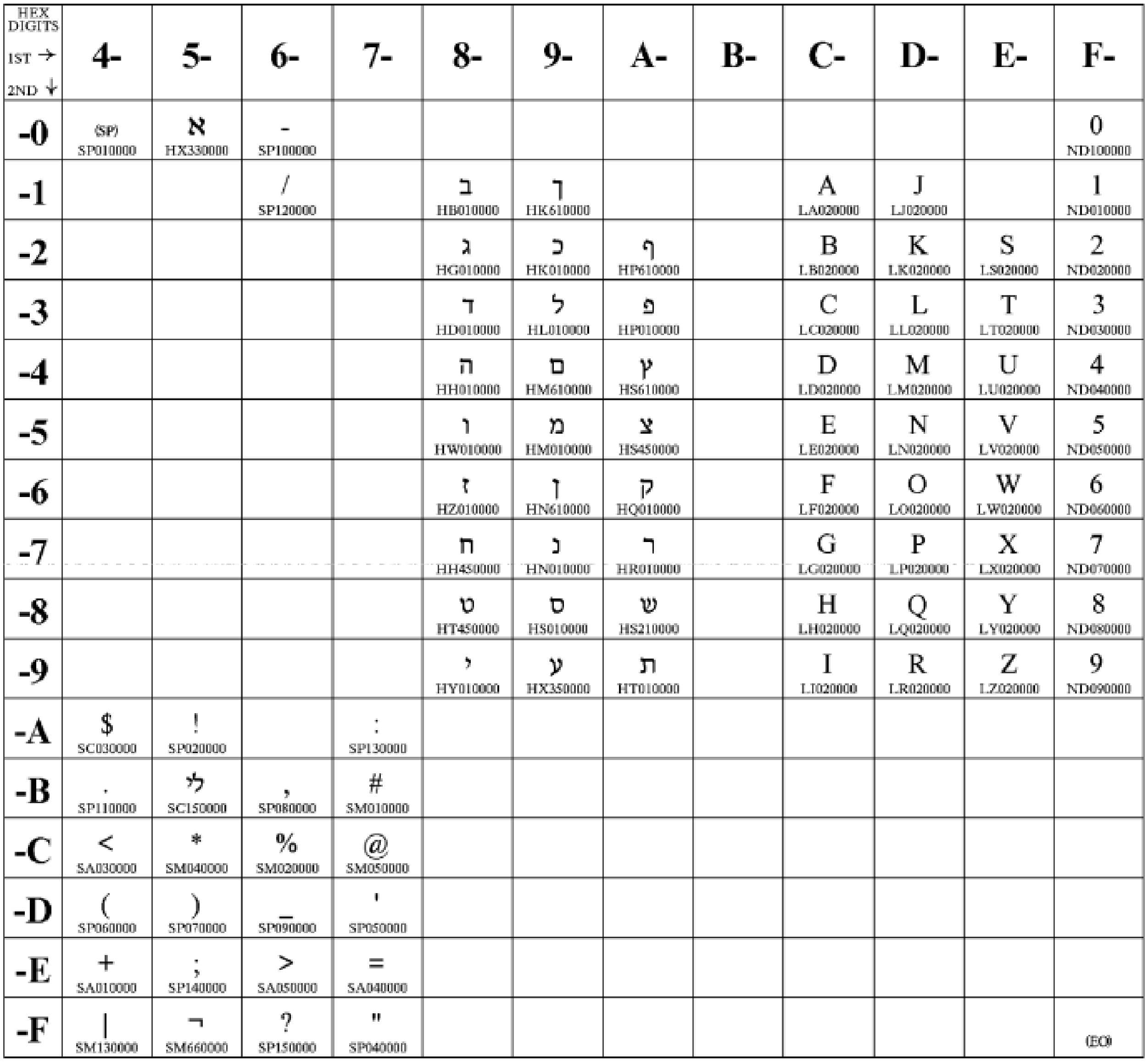
Code Page 00500

[Top of Page on page 1229 | Previous Page on page 1228 | Next Page on page 1230 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 803 Israel (Hebrew - Old Code)

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	א HX330000	- SP100000									0 ND100000
-1			/ SP120000		ב HB010000	ך HK610000			A LA020000	J LJ020000		1 ND010000
-2					ג HG010000	כ HK010000	ף HP610000		B LB020000	K LK020000	S LS020000	2 ND020000
-3					ד HD010000	ל HL010000	פ HP010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4					ה HH010000	ם HM610000	ץ HS610000		D LD020000	M LM020000	U LU020000	4 ND040000
-5					ו HW010000	מ HM010000	צ HS450000		E LE020000	N LN020000	V LV020000	5 ND050000
-6					ז HZ010000	ן HN610000	ק HQ010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7					ח HH450000	נ HN010000	ר HR010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8					ט HT450000	ס HS010000	ש HS210000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9					י HY010000	ע HX350000	ת HT010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	$ SC030000	! SP020000		: SP130000								
-B	. SP110000	לי SC150000	, SP080000	# SM010000								
-C	< SA030000	* SM040000	% SM020000	@ SM050000								
-D	(SP060000	) SP070000	_ SP090000	' SP050000								
-E	+ SA010000	; SP140000	> SA050000	= SA040000								
-F	| SM130000	¬ SM660000	? SP150000	" SP040000								(EO)

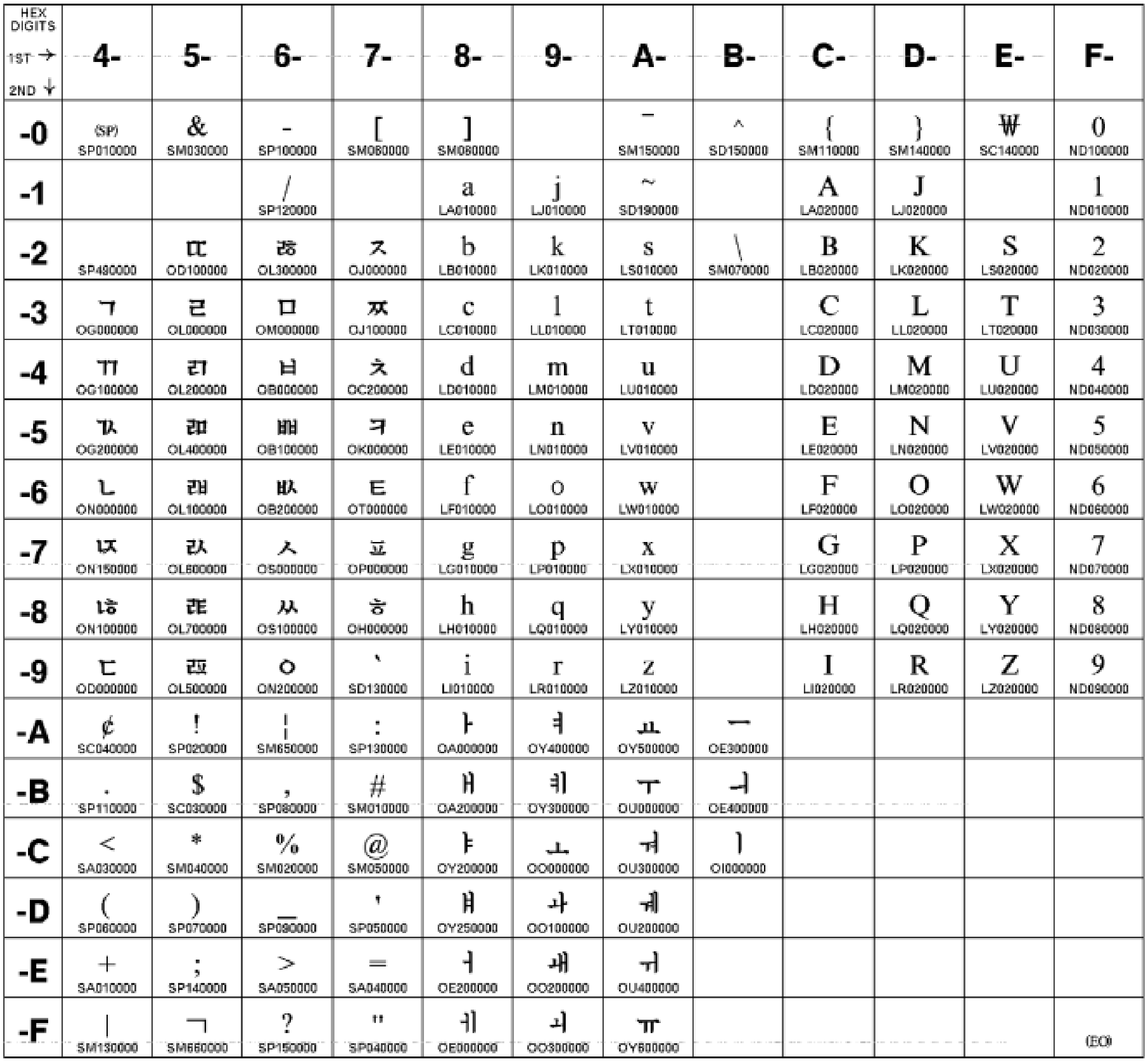
Code Page 00803

[Top of Page on page 1230 | Previous Page on page 1229 | Next Page on page 1231 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 833/1173 Hangeul

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	[SM060000	] SM080000		‾ SM150000	^ SD150000	{ SM110000	} SM140000	₩ SC140000	0 ND100000
-1			/ SP120000		a LA010000	j LJ010000	~ SD190000		A LA020000	J LJ020000		1 ND010000
-2	SP490000	ㄸ OD100000	ㅀ OL300000	ㅈ OJ000000	b LB010000	k LK010000	s LS010000	\ SM070000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ㄱ OG000000	ㄹ OL000000	ㅁ OM000000	ㅉ OJ100000	c LC010000	l LL010000	t LT010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4	ㄲ OG100000	ㄺ OL200000	ㅂ OB000000	ㅊ OC200000	d LD010000	m LM010000	u LU010000		D LD020000	M LM020000	U LU020000	4 ND040000
-5	ㄳ OG200000	ㄻ OL400000	ㅃ OB100000	ㅋ OK000000	e LE010000	n LN010000	v LV010000		E LE020000	N LN020000	V LV020000	5 ND050000
-6	ㄴ ON000000	ㄼ OL100000	ㅄ OB200000	ㅌ OT000000	f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7	ㄵ ON150000	ㄽ OL600000	ㅅ OS000000	ㅍ OP000000	g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8	ㄶ ON100000	ㄾ OL700000	ㅆ OS100000	ㅎ OH000000	h LH010000	q LQ010000	y LY010000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ㄷ OD000000	ㄿ OL500000	ㅇ ON200000	` SD130000	i LI010000	r LR010000	z LZ010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	ㅏ OA000000	ㅕ OY400000	ㅛ OY500000	ㅡ OE300000				
-B	. SP110000	$ SC030000	, SP080000	# SM010000	ㅐ OA200000	ㅖ OY300000	ㅜ OU000000	ㅢ OE400000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ㅑ OY200000	ㅗ OO000000	ㅝ OU300000	ㅣ OI000000				
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ㅒ OY250000	ㅘ OO100000	ㅞ OU200000					
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ㅓ OE200000	ㅙ OO200000	ㅟ OU400000					
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ㅔ OE000000	ㅚ OO300000	ㅠ OY600000					(EO)

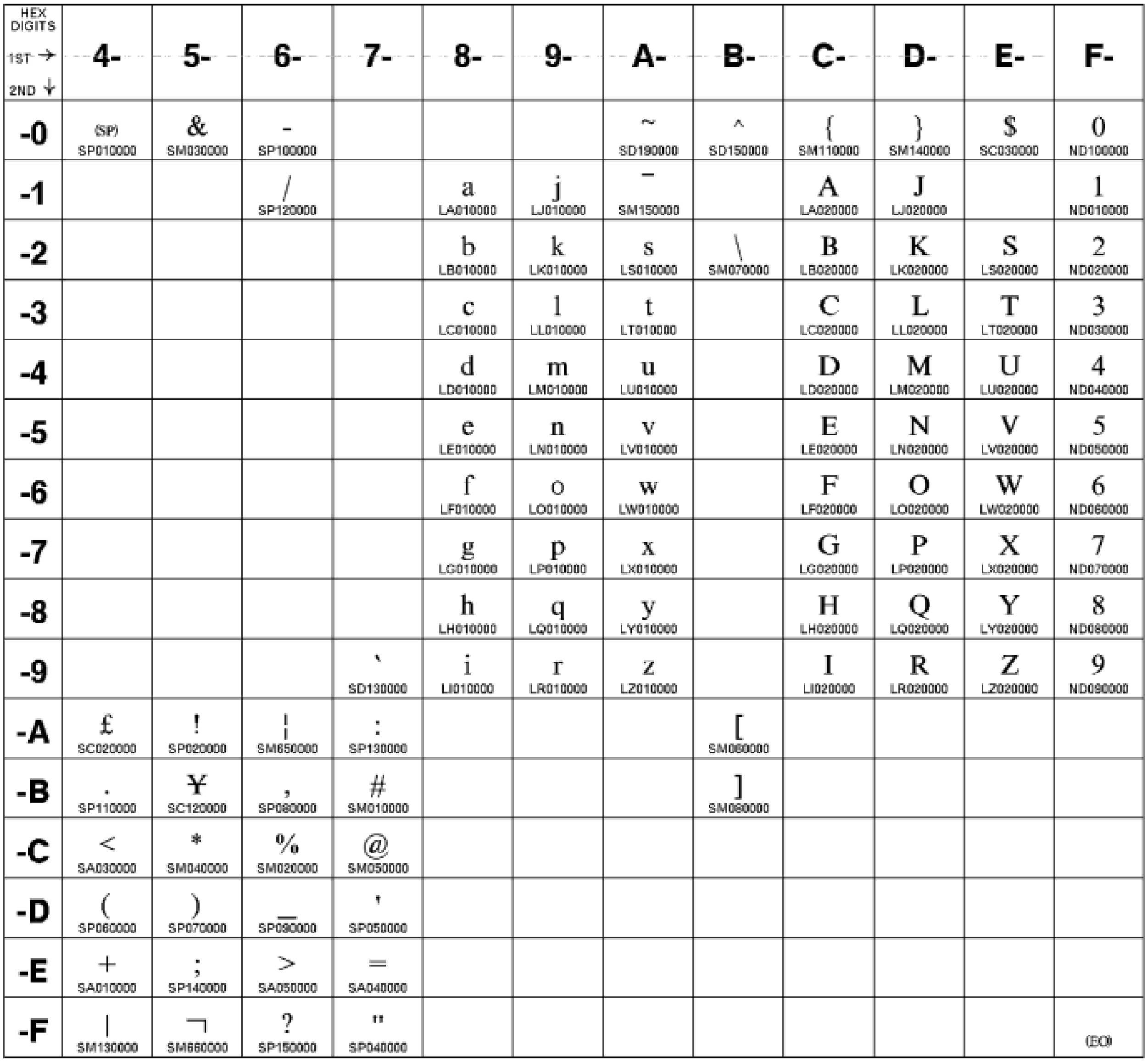
Code Page 00833

[Top of Page on page 1231 | Previous Page on page 1230 | Next Page on page 1232 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 836/1174 Simplified Chinese

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000				~ SD190000	^ SD150000	{ SM110000	} SM140000	$ SC030000	0 ND100000
-1			/ SP120000		a LA010000	j LJ010000	‾ SM150000		A LA020000	J LJ020000		1 ND010000
-2					b LB010000	k LK010000	s LS010000	\ SM070000	B LB020000	K LK020000	S LS020000	2 ND020000
-3					c LC010000	l LL010000	t LT010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4					d LD010000	m LM010000	u LU010000		D LD020000	M LM020000	U LU020000	4 ND040000
-5					e LE010000	n LN010000	v LV010000		E LE020000	N LN020000	V LV020000	5 ND050000
-6					f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7					g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8					h LH010000	q LQ010000	y LY010000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9				` SD130000	i LI010000	r LR010000	z LZ010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	£ SC020000	! SP020000	¦ SM650000	: SP130000				[SM060000				
-B	. SP110000	¥ SC120000	, SP080000	# SM010000				] SM080000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000								
-D	(SP060000	) SP070000	_ SP090000	' SP050000								
-E	+ SA010000	; SP140000	> SA050000	= SA040000								
-F	| SM130000	¬ SM660000	? SP150000	" SP040000								(EO)

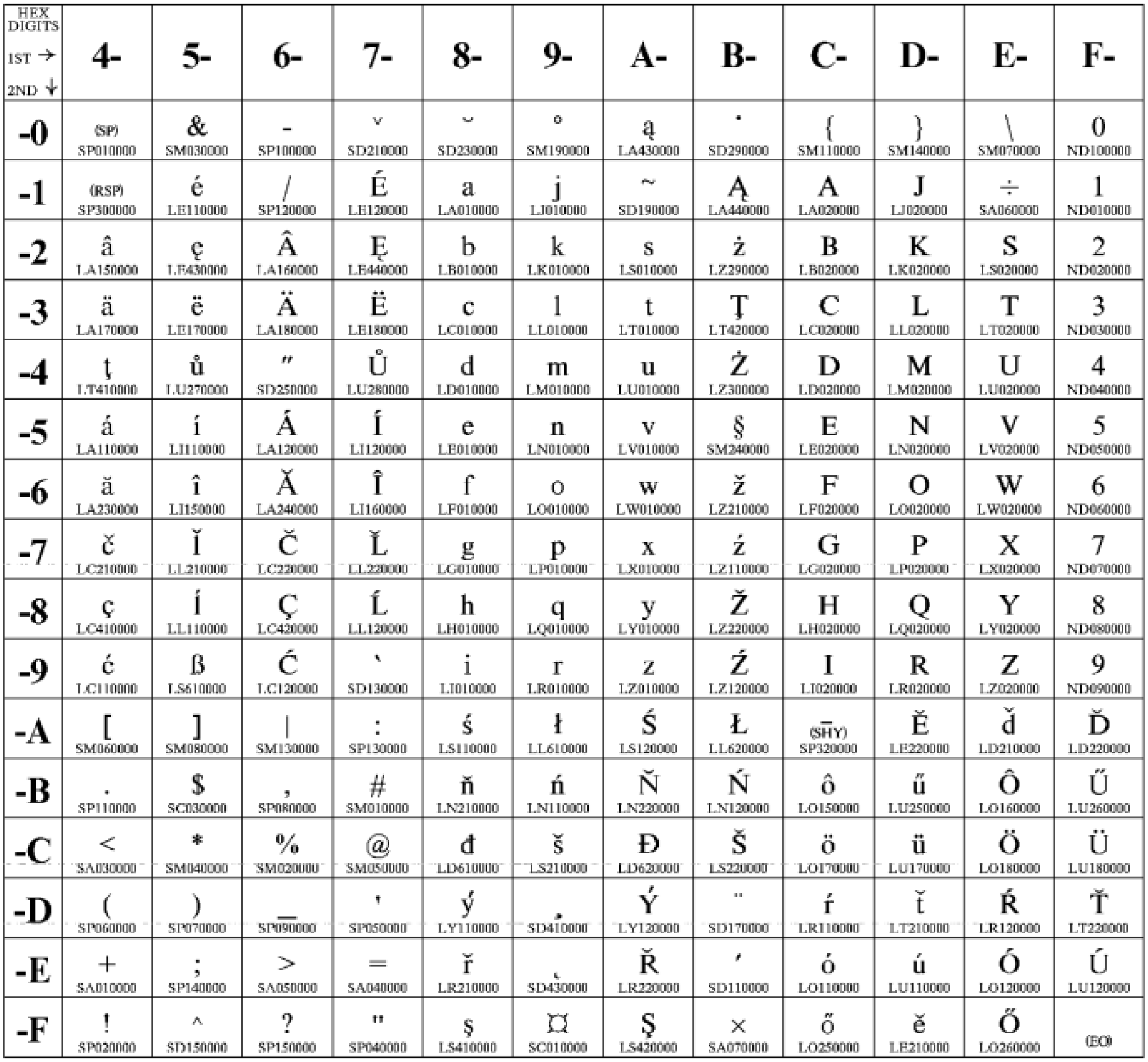
Code Page 00836

[Top of Page on page 1232 | Previous Page on page 1231 | Next Page on page 1233 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 870/959 Latin 2 - EBCDIC Multilingual

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ˇ SD210000	˘ SD230000	° SM190000	ą LA430000	˙ SD290000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	Ą LA440000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ę LE430000	Â LA160000	Ę LE440000	b LB010000	k LK010000	s LS010000	ż LZ290000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	Ţ LT420000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ţ LT410000	ů LU270000	˝ SD250000	Ů LU280000	d LD010000	m LM010000	u LU010000	Ż LZ300000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ă LA230000	î LI150000	Ă LA240000	Î LI160000	f LF010000	o LO010000	w LW010000	ž LZ210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	č LC210000	ľ LL210000	Č LC220000	Ľ LL220000	g LG010000	p LP010000	x LX010000	ź LZ110000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ĺ LL110000	Ç LC420000	Ĺ LL120000	h LH010000	q LQ010000	y LY010000	Ž LZ220000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ć LC110000	ß LS610000	Ć LC120000	` SD130000	i LI010000	r LR010000	z LZ010000	Ź LZ120000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	ś LS110000	ł LL610000	Ś LS120000	Ł LL620000	(SHY) SP320000	Ě LE220000	ď LD210000	Ď LD220000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	ň LN210000	ń LN110000	Ň LN220000	Ń LN120000	ô LO150000	ű LU250000	Ô LO160000	Ű LU260000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	đ LD610000	š LS210000	Đ LD620000	Š LS220000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	˛ SD410000	Ý LY120000	¨ SD170000	ŕ LR110000	ť LT210000	Ŕ LR120000	Ť LT220000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ř LR210000	¸ SD430000	Ř LR220000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	ş LS410000	¤ SC010000	Ş LS420000	× SA070000	ő LO250000	ě LE210000	Ő LO260000	(EO)

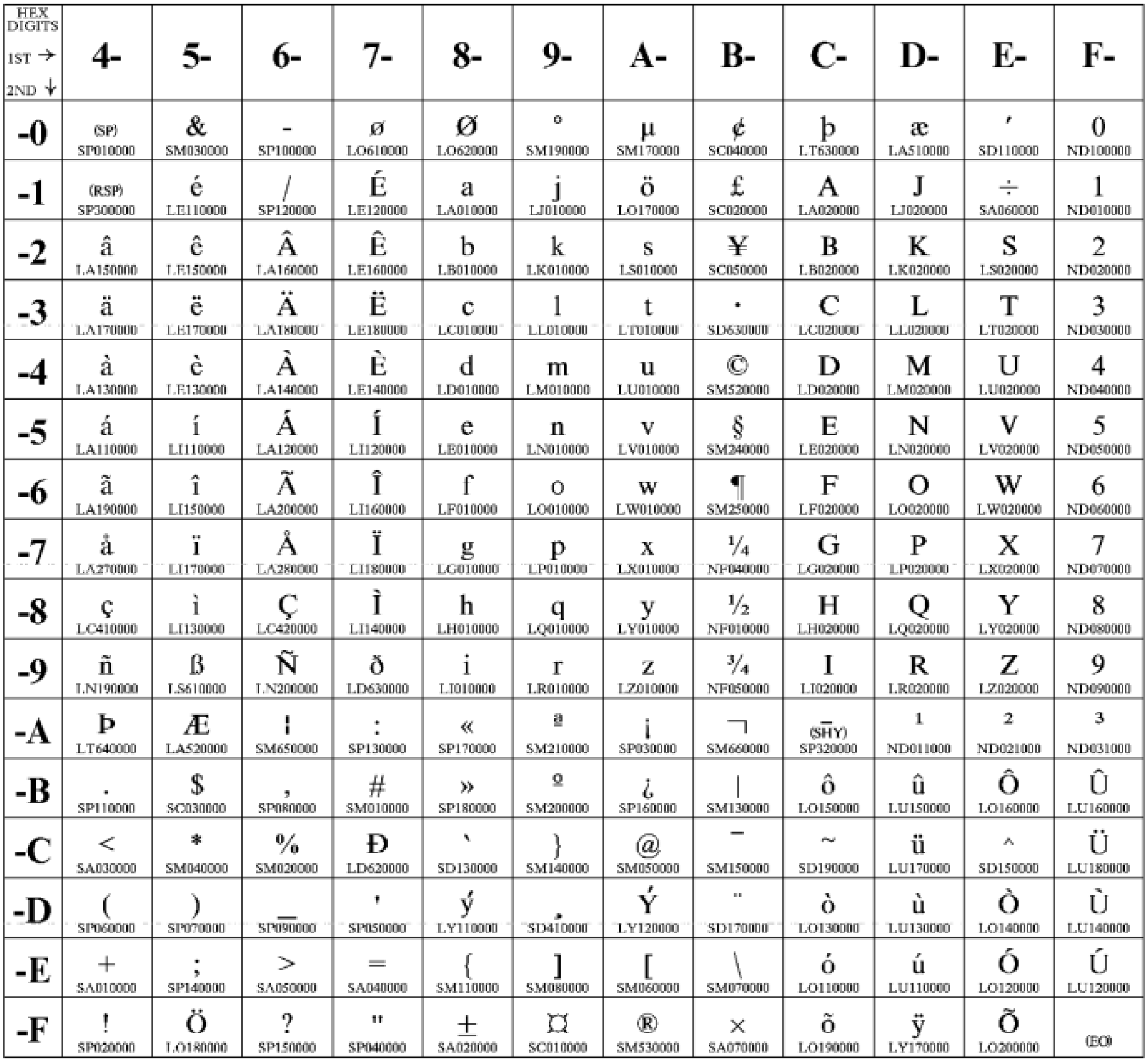
Code Page 00870

[Top of Page on page 1233 | Previous Page on page 1232 | Next Page on page 1234 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 871-1/697-1 Iceland

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	þ LT630000	æ LA510000	´ SD110000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ö LO170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ð LD630000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Þ LT640000	Æ LA520000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Đ LD620000	` SD130000	} SM140000	@ SM050000	¯ SM150000	~ SD190000	ü LU170000	^ SD150000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	{ SM110000	] SM080000	[SM060000	\ SM070000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	Ö LO180000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

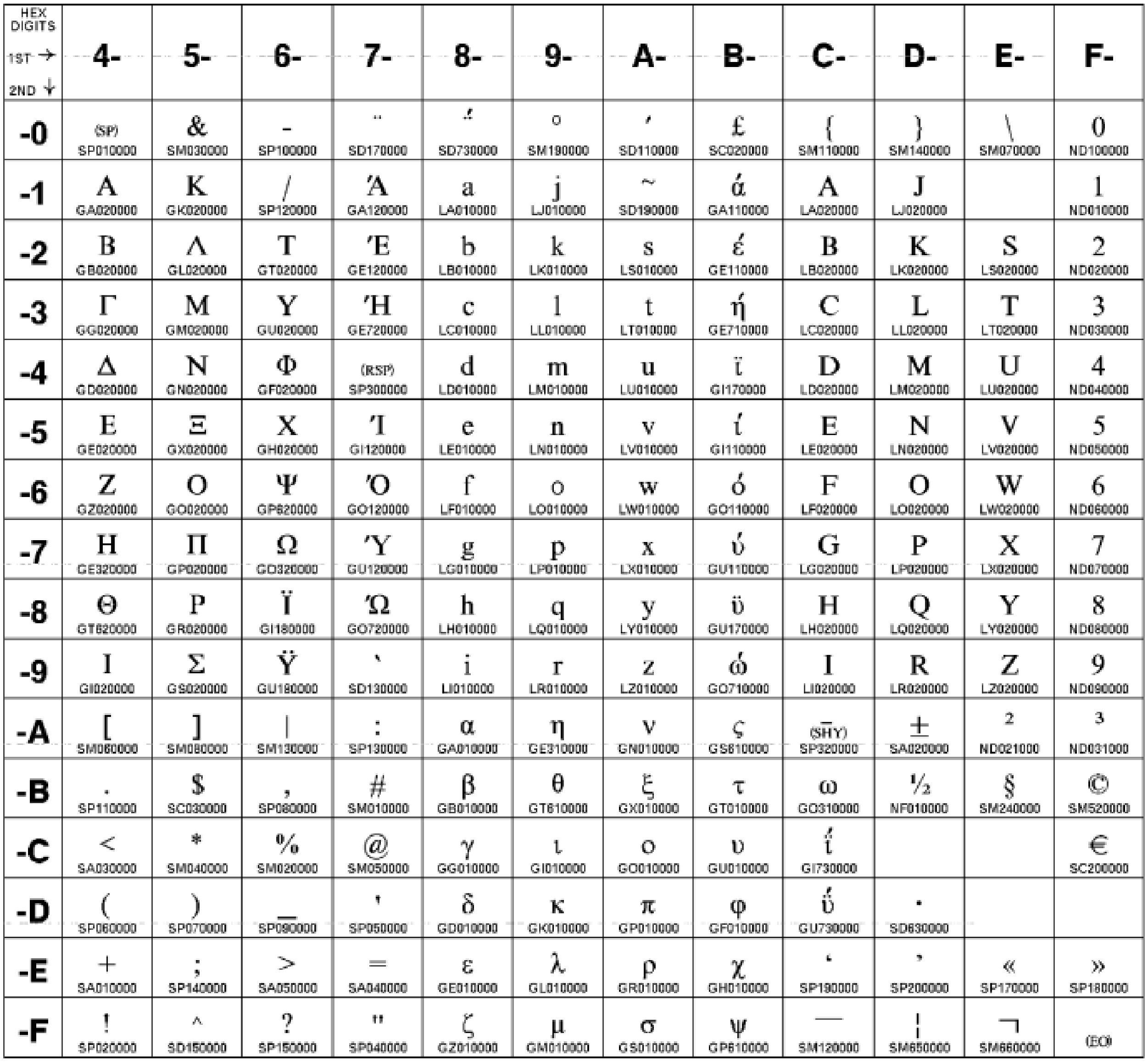
Code Page 00871

[Top of Page on page 1234 | Previous Page on page 1233 | Next Page on page 1235 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 875 Greece

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	¨ SD170000	΅ SD730000	° SM190000	΄ SD110000	£ SC020000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	Α GA020000	Κ GK020000	/ SP120000	Ά GA120000	a LA010000	j LJ010000	~ SD190000	ά GA110000	A LA020000	J LJ020000		1 ND010000
-2	Β GB020000	Λ GL020000	Τ GT020000	Έ GE120000	b LB010000	k LK010000	s LS010000	έ GE110000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	Γ GG020000	Μ GM020000	Υ GU020000	Ή GE720000	c LC010000	l LL010000	t LT010000	ή GE710000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	Δ GD020000	Ν GN020000	Φ GF020000	(RSP) SP300000	d LD010000	m LM010000	u LU010000	ϊ GI170000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	Ε GE020000	Ξ GX020000	Χ GH020000	Ί GI120000	e LE010000	n LN010000	v LV010000	ί GI110000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	Ζ GZ020000	Ο GO020000	Ψ GP620000	Ό GO120000	f LF010000	o LO010000	w LW010000	ό GO110000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	Η GE320000	Π GP020000	Ω GO320000	Ύ GU120000	g LG010000	p LP010000	x LX010000	ύ GU110000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	Θ GT620000	Ρ GR020000	Ϊ GI180000	Ώ GO720000	h LH010000	q LQ010000	y LY010000	ϋ GU170000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	Ι GI020000	Σ GS020000	Ϋ GU180000	` SD130000	i LI010000	r LR010000	z LZ010000	ώ GO710000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	α GA010000	η GE310000	ν GN010000	ς GS610000	(SHY) SP320000	± SA020000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	β GB010000	θ GT610000	ξ GX010000	τ GT010000	ω GO310000	½ NF010000	§ SM240000	© SM520000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	γ GG010000	ι GI010000	ο GO010000	υ GU010000	ΐ GI730000			€ SC200000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	δ GD010000	κ GK010000	π GP010000	φ GF010000	ΰ GU730000	· SD630000		
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ε GE010000	λ GL010000	ρ GR010000	χ GH010000	‘ SP190000	’ SP200000	« SP170000	» SP180000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	ζ GZ010000	μ GM010000	σ GS010000	ψ GP610000	― SM120000	¦ SM650000	¬ SM660000	(EO)

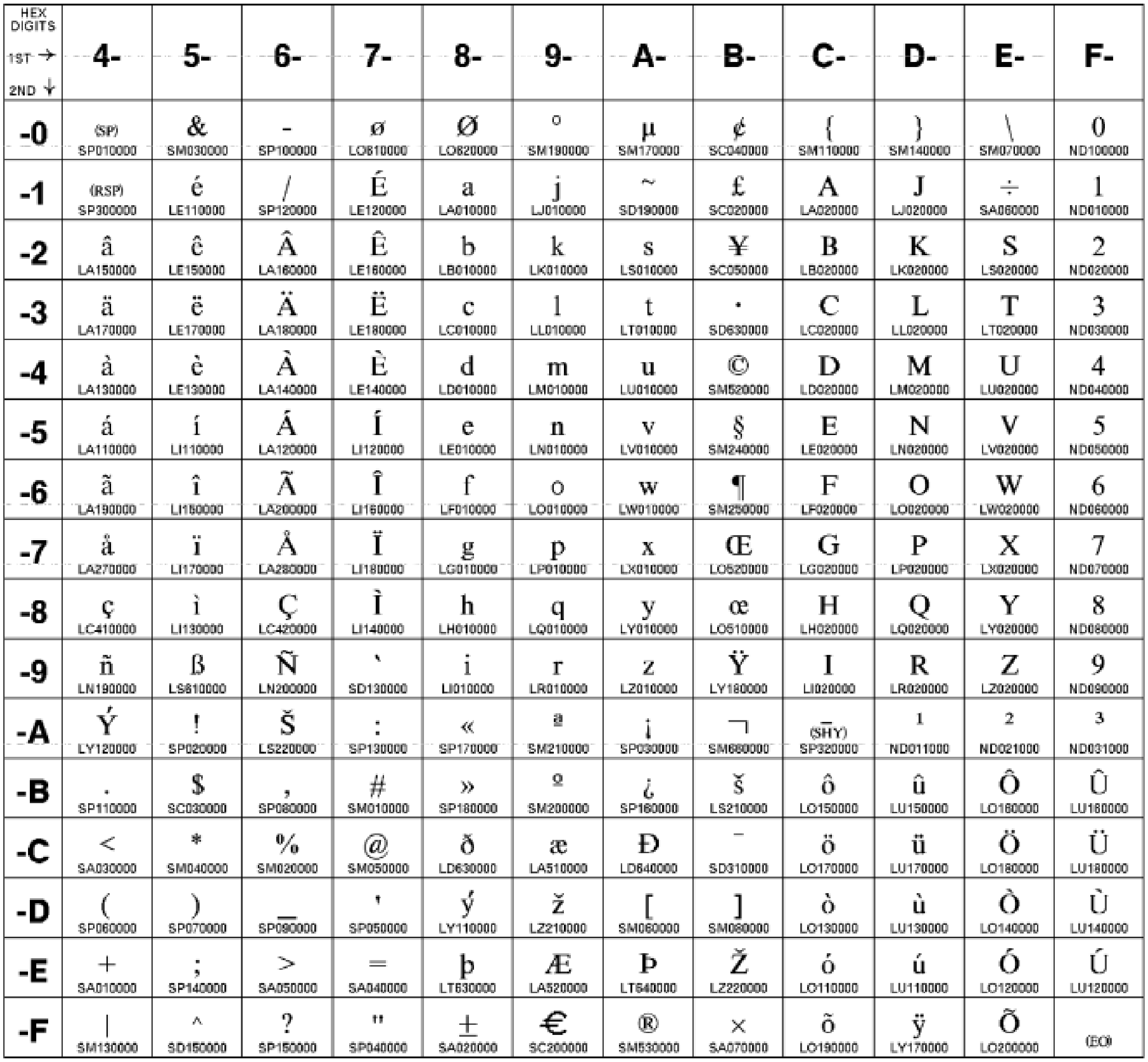
Code Page 00875

[Top of Page on page 1235 | Previous Page on page 1234 | Next Page on page 1236 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 924-1/1353-1 International

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	Œ LO520000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	œ LO510000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	Ÿ LY180000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ý LY120000	! SP020000	Š LS220000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	š LS210000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Đ LD640000	‾ SD310000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	ž LZ210000	[SM060000	] SM080000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	Ž LZ220000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

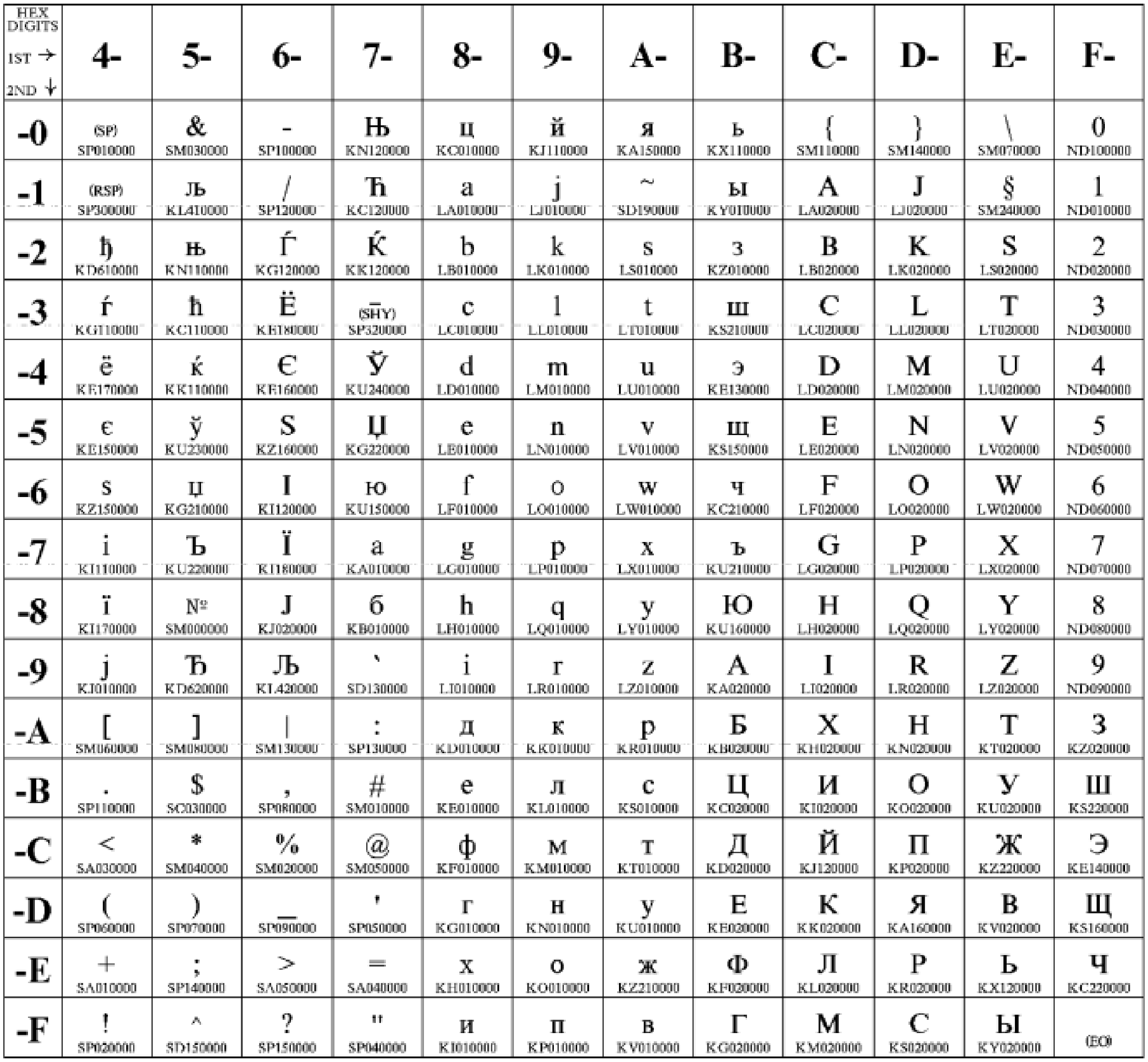
Code Page 00924

[Top of Page on page 1236 | Previous Page on page 1235 | Next Page on page 1237 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1025/1150 Cyrillic

The column indicates the first digit, and the row indicates the second digit. If this code page is not used with a Cyrillic host machine, certain characters might not display properly.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	Њ KN120000	ц KC010000	й KJ110000	я KA150000	ь KX110000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	љ KL410000	/ SP120000	Ћ KC120000	a LA010000	j LJ010000	~ SD190000	ы KY010000	A LA020000	J LJ020000	§ SM240000	1 ND010000
-2	ђ KD610000	њ KN110000	Ѓ KG120000	Ќ KK120000	b LB010000	k LK010000	s LS010000	з KZ010000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ѓ KG110000	ћ KC110000	Ё KE180000	(SHY) SP320000	c LC010000	l LL010000	t LT010000	ш KS210000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ё KE170000	ќ KK110000	Є KE160000	Ў KU240000	d LD010000	m LM010000	u LU010000	э KE130000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	є KE150000	ў KU230000	Ѕ KZ160000	Џ KG220000	e LE010000	n LN010000	v LV010000	щ KS150000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ѕ KZ150000	џ KG210000	І KI120000	ю KU150000	f LF010000	o LO010000	w LW010000	ч KC210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	і KI110000	Ъ KU220000	Ї KI180000	а KA010000	g LG010000	p LP010000	x LX010000	ъ KU210000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ї KI170000	№ SM000000	Ј KJ020000	б KB010000	h LH010000	q LQ010000	y LY010000	Ю KU160000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ј KJ010000	Ђ KD620000	Љ KL420000	` SD130000	i LI010000	r LR010000	z LZ010000	А KA020000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	д KD010000	к KK010000	р KR010000	Б KB020000	Х KH020000	Н KN020000	Т KT020000	З KZ020000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	е KE010000	л KL010000	с KS010000	Ц KC020000	И KI020000	О KO020000	У KU020000	Ш KS220000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ф KF010000	м KM010000	т KT010000	Д KD020000	Й KJ120000	П KP020000	Ж KZ220000	Э KE140000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	г KG010000	н KN010000	у KU010000	Е KE020000	К KK020000	Я KA160000	В KV020000	Щ KS160000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	х KH010000	о KO010000	ж KZ210000	Ф KF020000	Л KL020000	Р KR020000	Ь KX120000	Ч KC220000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	и KI010000	п KP010000	в KV010000	Г KG020000	М KM020000	С KS020000	Ы KY020000	(EO)

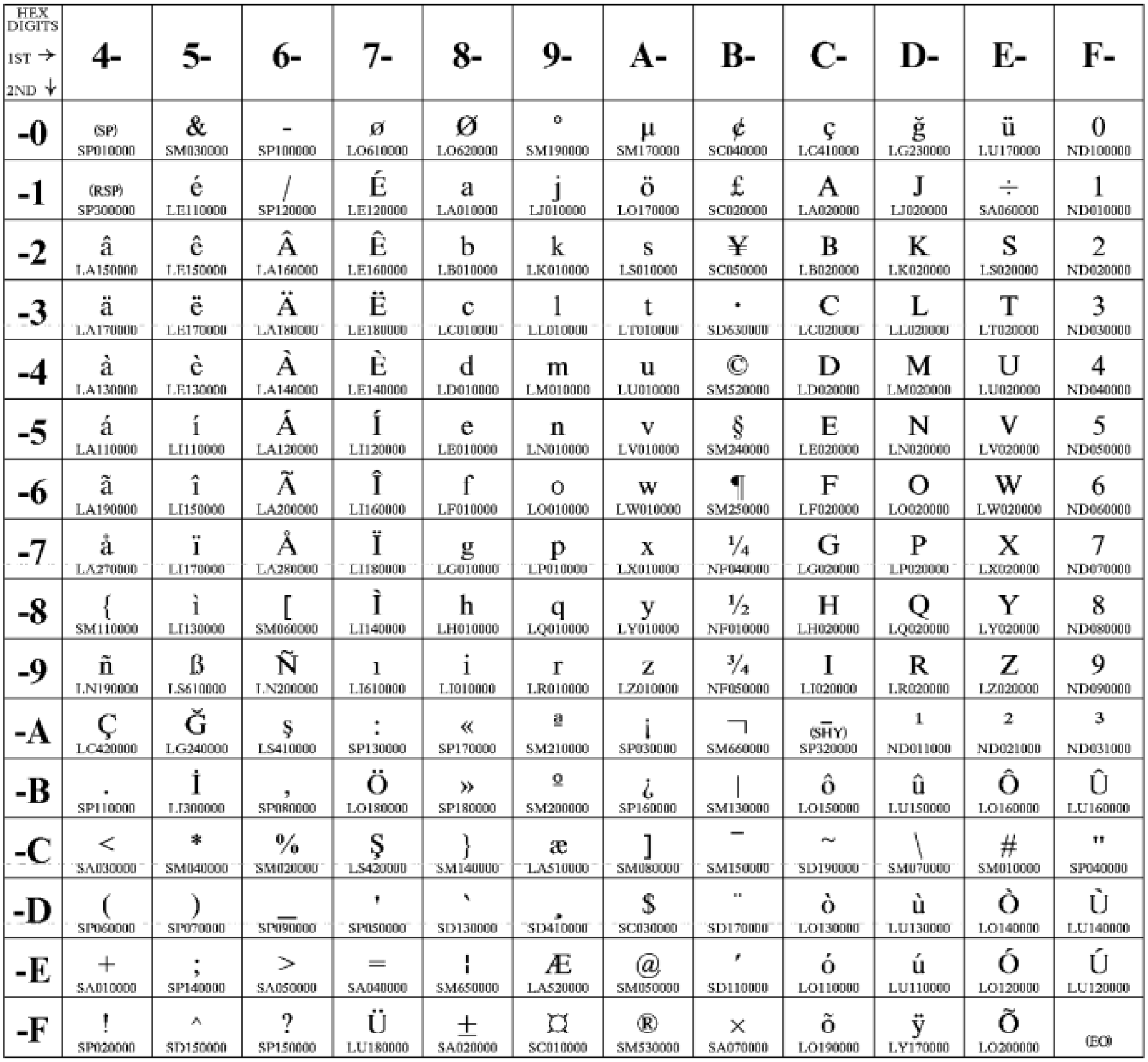
Code Page 01025

[Top of Page on page 1237 | Previous Page on page 1236 | Next Page on page 1238 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1026/1152 Latin 5 - Turkey

The column indicates the first digit, and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ç LC410000	ğ LG230000	ü LU170000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ö LO170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	{ SM110000	ì LI130000	[SM060000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ı LI610000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ç LC420000	Ğ LG240000	ş LS410000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	İ LI300000	, SP080000	Ö LO180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ş LS420000	} SM140000	æ LA510000	] SM080000	¯ SM150000	~ SD190000	\ SM070000	# SM010000	" SP040000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	` SD130000	¸ SD410000	$ SC030000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	¦ SM650000	Æ LA520000	@ SM050000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	Ü LU180000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

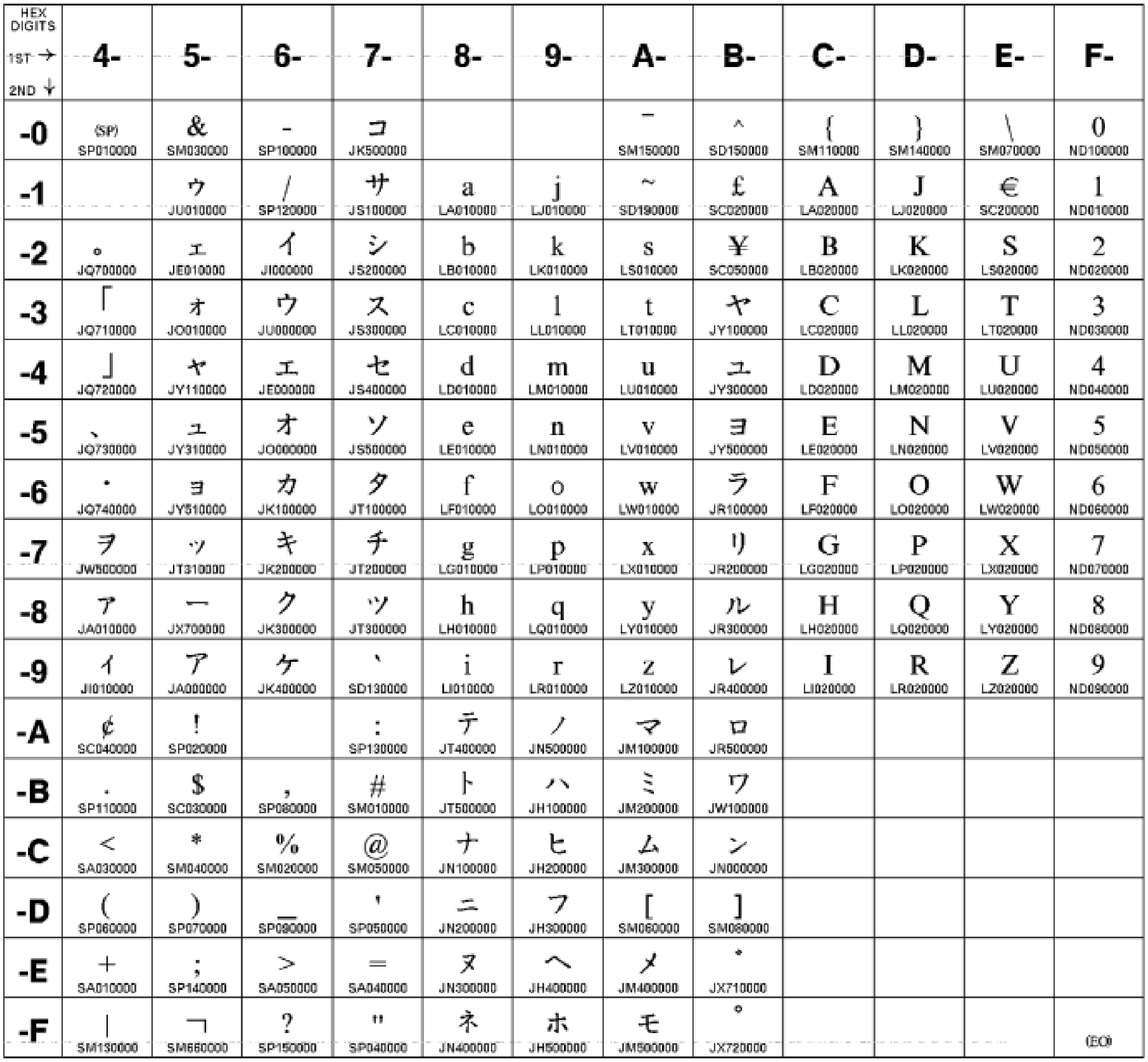
Code Page 01026

[Top of Page on page 1238 | Previous Page on page 1237 | Next Page on page 1239 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1027/939 Japan (Latin) Extended

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ｺ JK500000			‾ SM150000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1		ｩ JU010000	/ SP120000	ｻ JS100000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	€ SC200000	1 ND010000
-2	｡ JQ700000	ｪ JE010000	ｲ JI000000	ｼ JS200000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	｢ JQ710000	ｫ JO010000	ｳ JU000000	ｽ JS300000	c LC010000	l LL010000	t LT010000	ﾔ JY100000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	｣ JQ720000	ｬ JY110000	ｴ JE000000	ｾ JS400000	d LD010000	m LM010000	u LU010000	ﾕ JY300000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	､ JQ730000	ｭ JY310000	ｵ JO000000	ｿ JS500000	e LE010000	n LN010000	v LV010000	ﾖ JY500000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	･ JQ740000	ｮ JY510000	ｶ JK100000	ﾀ JT100000	f LF010000	o LO010000	w LW010000	ﾗ JR100000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	ｦ JW500000	ｯ JT310000	ｷ JK200000	ﾁ JT200000	g LG010000	p LP010000	x LX010000	ﾘ JR200000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ｧ JA010000	ｰ JX700000	ｸ JK300000	ﾂ JT300000	h LH010000	q LQ010000	y LY010000	ﾙ JR300000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ｨ JI010000	ｱ JA000000	ｹ JK400000	` SD130000	i LI010000	r LR010000	z LZ010000	ﾚ JR400000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000		: SP130000	ﾃ JT400000	ﾉ JN500000	ﾏ JM100000	ﾛ JR500000				
-B	. SP110000	$ SC030000	, SP080000	# SM010000	ﾄ JT500000	ﾊ JH100000	ﾐ JM200000	ﾜ JW100000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ﾅ JN100000	ﾋ JH200000	ﾑ JM300000	ﾝ JN000000				
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ﾆ JN200000	ﾌ JH300000	[SM060000	] SM080000				
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ﾇ JN300000	ﾍ JH400000	ﾒ JM400000	ﾞ JX710000				
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ﾈ JN400000	ﾎ JH500000	ﾓ JM500000	ﾟ JX720000				(EO)

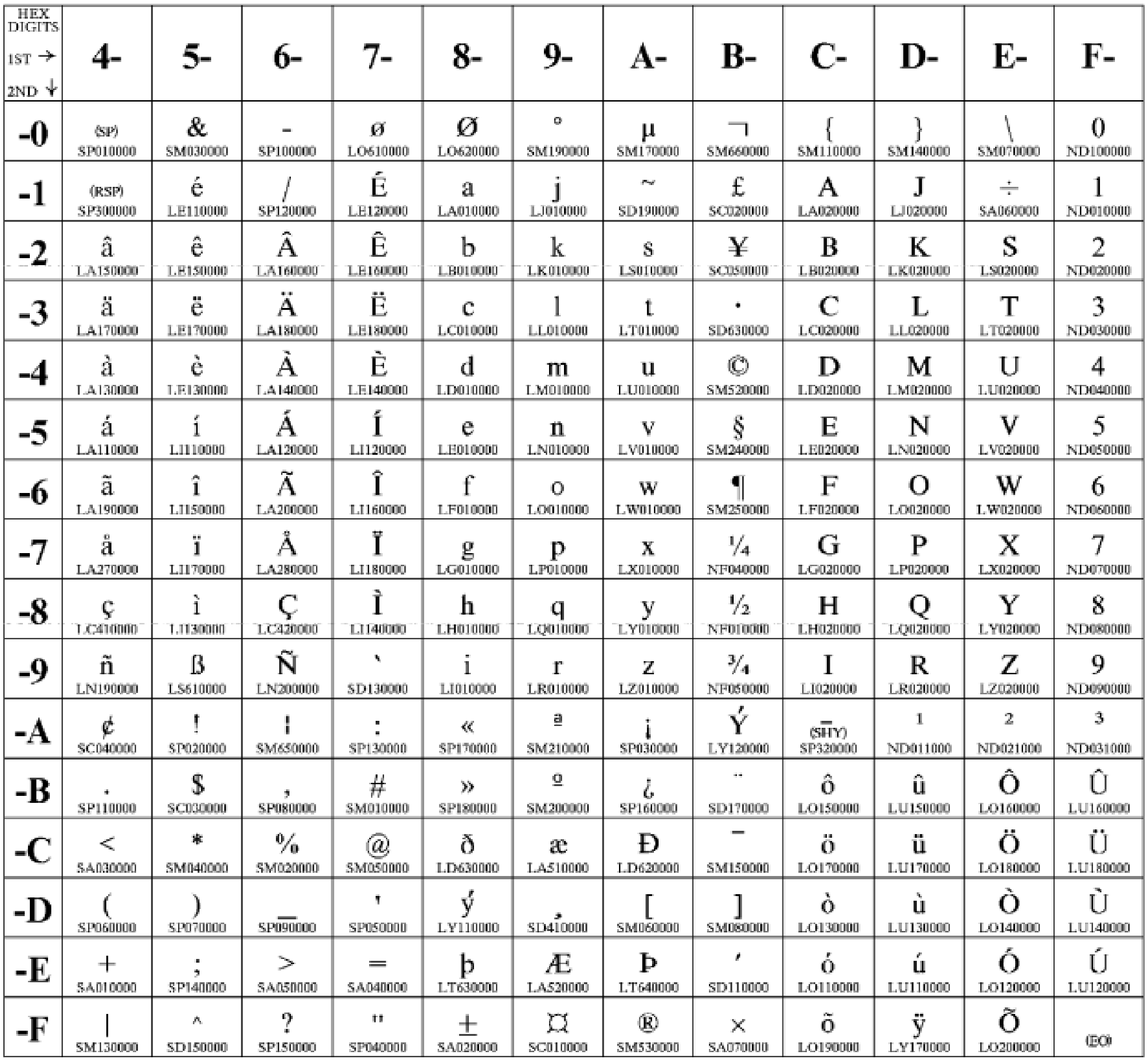
Code Page 01027

[Top of Page on page 1239 | Previous Page on page 1238 | Next Page on page 1240 | Table of Contents on page 1216 **]**

Host Code Page Reference

Host Code Page 1047/103 Latin 1 (Open Systems)

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¬ SM660000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	Ý LY120000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	¨ SD170000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	[SM060000	] SM080000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

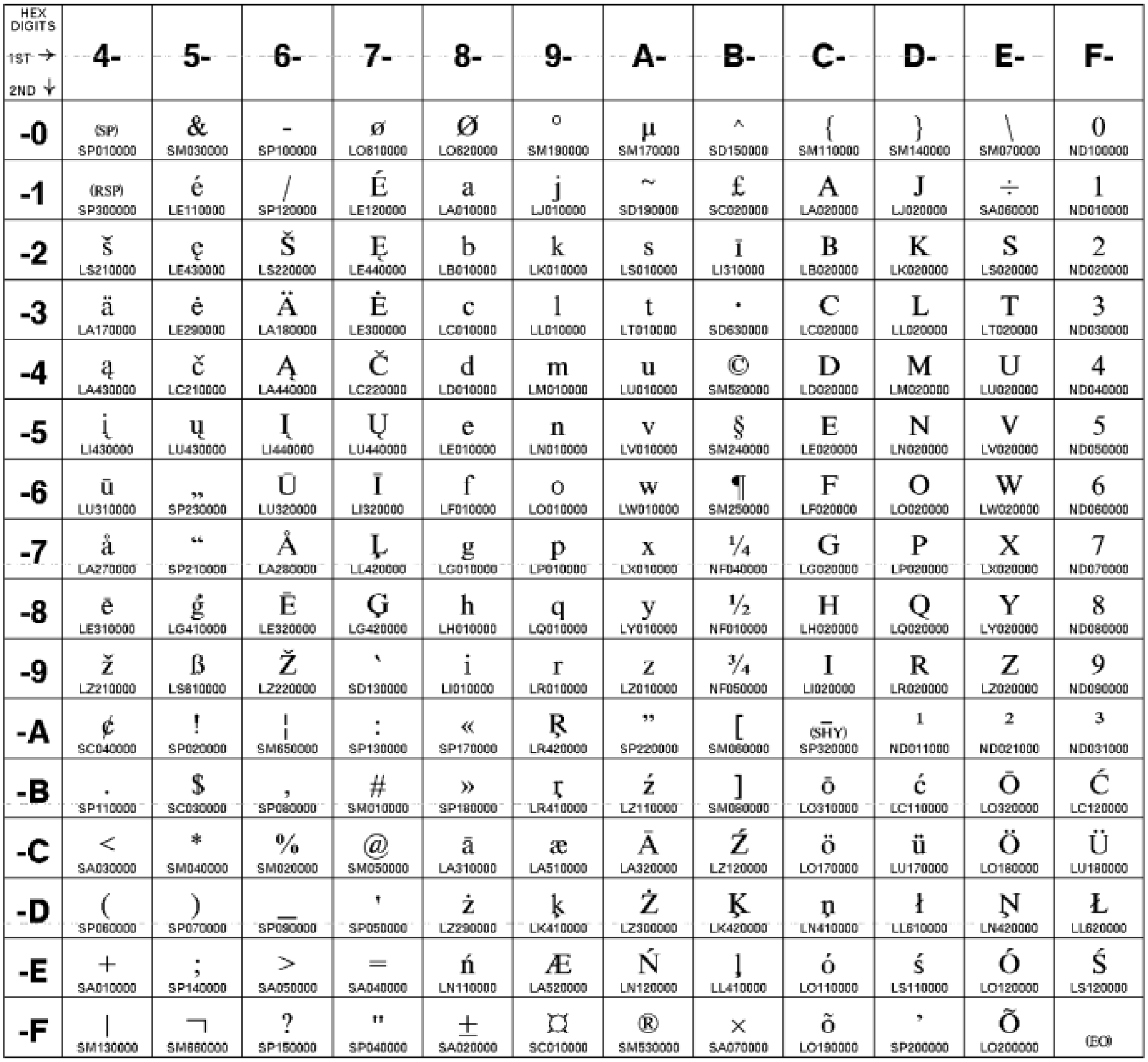
Code Page 01047

[Top of Page on page 1240 | Previous Page on page 1239 | Next Page on page 1241 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1112/1035 Latvia, Lithuania

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	š LS210000	ç LE430000	Š LS220000	Ę LE440000	b LB010000	k LK010000	s LS010000	ī LI310000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ė LE290000	Ä LA180000	Ė LE300000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ą LA430000	č LC210000	Ą LA440000	Č LC220000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	į LI430000	ų LU430000	Į LI440000	Ų LU440000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ū LU310000	„ SP230000	Ū LU320000	Ī LI320000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	“ SP210000	Å LA280000	Ļ LL420000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ē LE310000	ģ LG410000	Ē LE320000	Ģ LG420000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ž LZ210000	ß LS610000	Ž LZ220000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	Ŗ LR420000	” SP220000	[SM060000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	ŗ LR410000	ź LZ110000	] SM080000	ō LO310000	ć LC110000	Ō LO320000	Ć LC120000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ā LA310000	æ LA510000	Ā LA320000	Ź LZ120000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ż LZ290000	ķ LK410000	Ż LZ300000	Ķ LK420000	ņ LN410000	ł LL610000	Ņ LN420000	Ł LL620000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ń LN110000	Æ LA520000	Ń LN120000	ļ LL410000	ó LO110000	ś LS110000	Ó LO120000	Ś LS120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	’ SP200000	Õ LO200000	(EO)

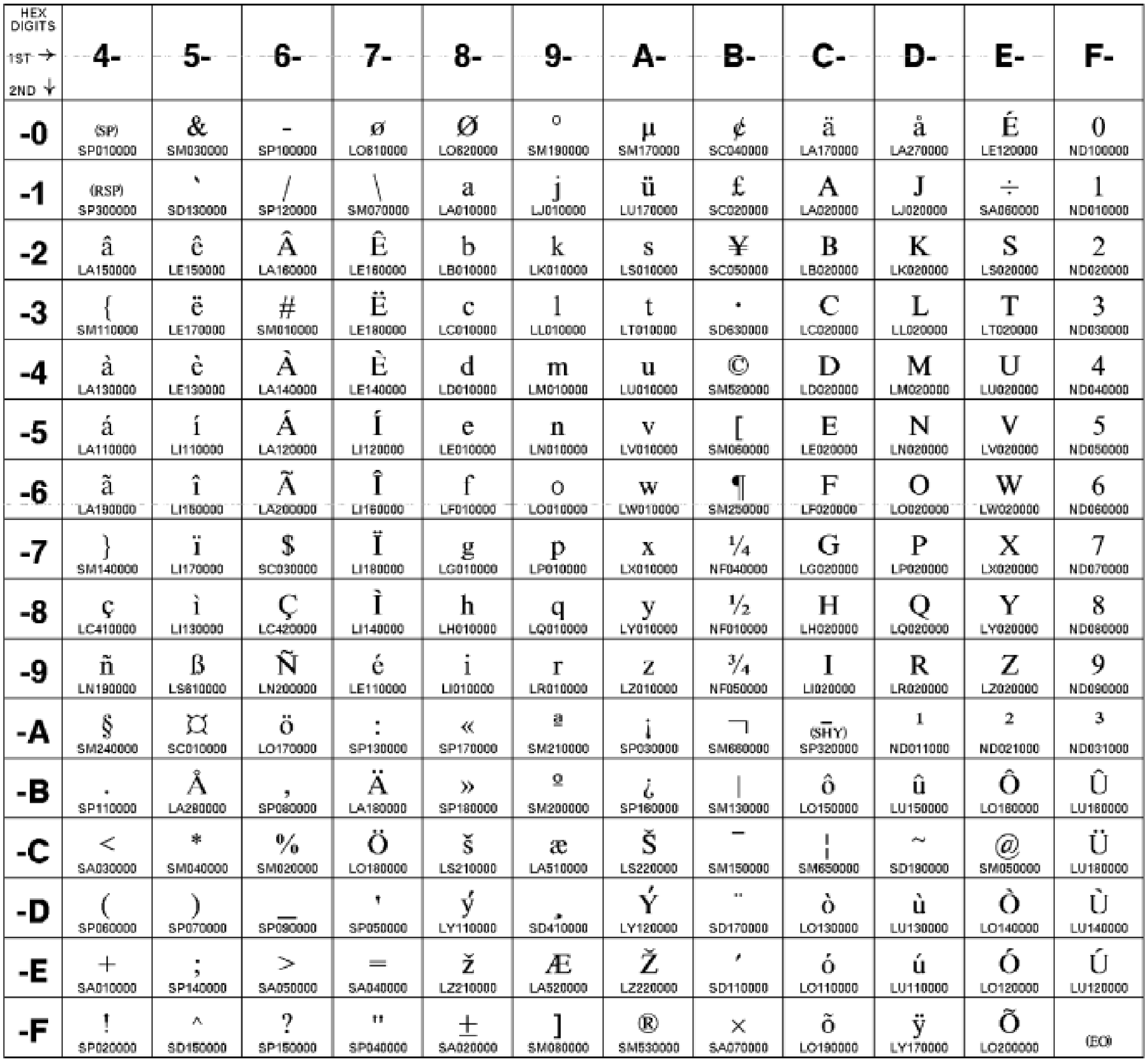
Code Page 01112

[Top of Page on page 1241 | Previous Page on page 1240 | Next Page on page 1242 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1122/1037 Estonia

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	å LA270000	É LE120000	0 ND100000
-1	(RSP) SP300000	` SD130000	/ SP120000	\\ SM070000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	# SM010000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	[SM060000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI160000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	é LE110000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	§ SM240000	¤ SC010000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Ä LA180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ö LO180000	š LS210000	æ LA510000	Š LS220000	¯ SM150000	¦ SM650000	~ SD190000	@ SM050000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ž LZ210000	Æ LA520000	Ž LZ220000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

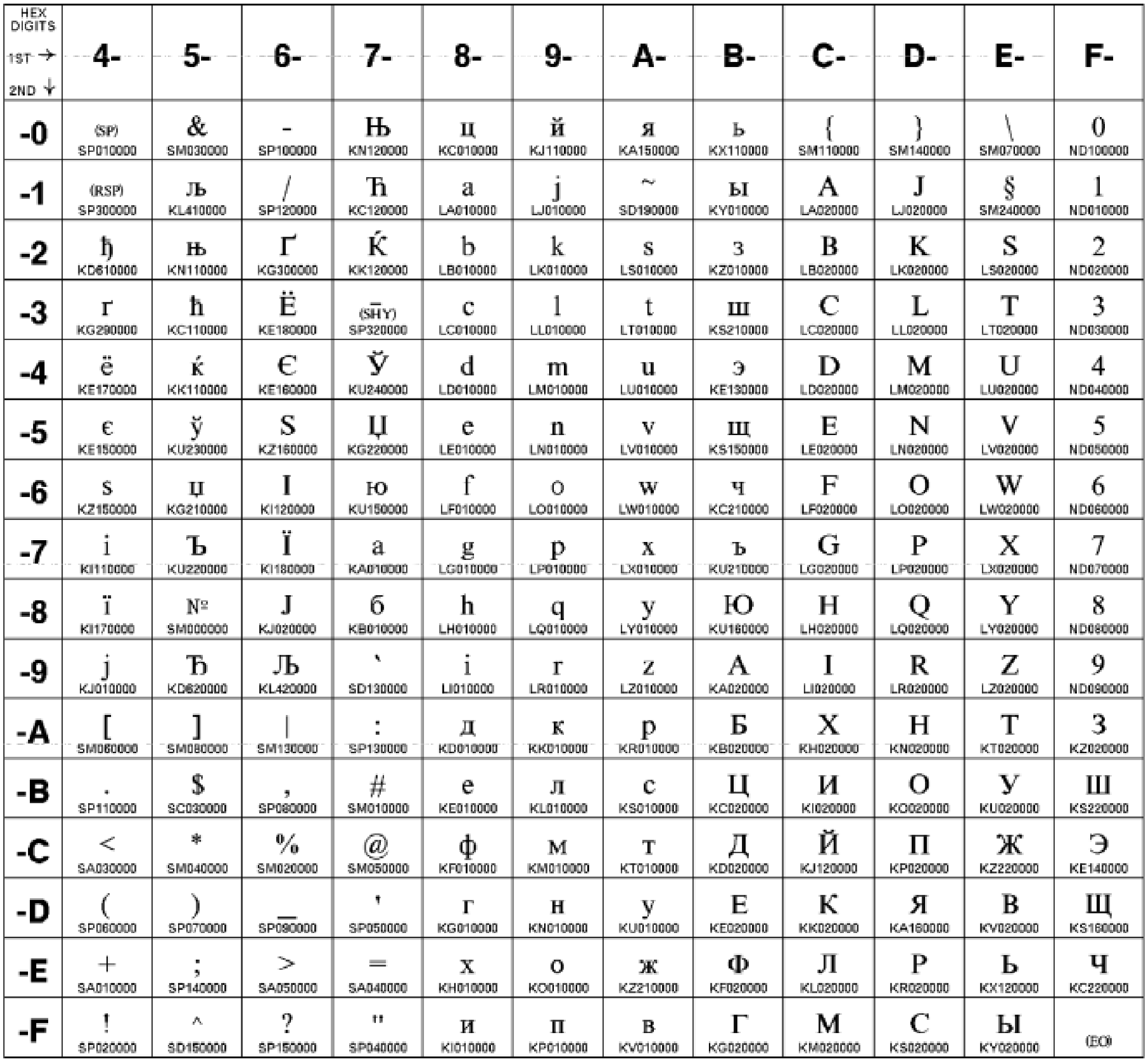
Code Page 01122

[Top of Page on page 1242 | Previous Page on page 1241 | Next Page on page 1243 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1123 Ukraine

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	Њ KN120000	ц KC010000	й KJ110000	я KA150000	ь KX110000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	љ KL410000	/ SP120000	Ћ KC120000	a LA010000	j LJ010000	~ SD190000	ы KY010000	A LA020000	J LJ020000	§ SM240000	1 ND010000
-2	ђ KD610000	њ KN110000	Ґ KG300000	Ќ KK120000	b LB010000	k LK010000	s LS010000	з KZ010000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	г KG290000	ћ KC110000	Ё KE180000	(SHY) SP320000	c LC010000	l LL010000	t LT010000	ш KS210000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ё KE170000	ќ KK110000	Є KE160000	Ў KU240000	d LD010000	m LM010000	u LU010000	э KE130000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	є KE150000	ў KU230000	Ѕ KZ160000	Џ KG220000	e LE010000	n LN010000	v LV010000	щ KS150000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ѕ KZ150000	џ KG210000	І KI120000	ю KU150000	f LF010000	o LO010000	w LW010000	ч KC210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	і KI110000	Ъ KU220000	Ї KI180000	а KA010000	g LG010000	p LP010000	x LX010000	ъ KU210000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ї KI170000	№ SM000000	Ј KJ020000	б KB010000	h LH010000	q LQ010000	y LY010000	Ю KU160000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ј KJ010000	Ђ KD620000	Љ KL420000	` SD130000	i LI010000	r LR010000	z LZ010000	А KA020000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	д KD010000	к KK010000	р KR010000	Б KB020000	Х KH020000	Н KN020000	Т KT020000	З KZ020000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	е KE010000	л KL010000	с KS010000	Ц KC020000	И KI020000	О KO020000	У KU020000	Ш KS220000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ф KF010000	м KM010000	т KT010000	Д KD020000	Й KJ120000	П KP020000	Ж KZ220000	Э KE140000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	г KG010000	н KN010000	у KU010000	Е KE020000	К KK020000	Я KA160000	В KV020000	Щ KS160000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	х KH010000	о KO010000	ж KZ210000	Ф KF020000	Л KL020000	Р KR020000	Ь KX120000	Ч KC220000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	и KI010000	п KP010000	в KV010000	Г KG020000	М KM020000	С KS020000	Ы KY020000	(EO)

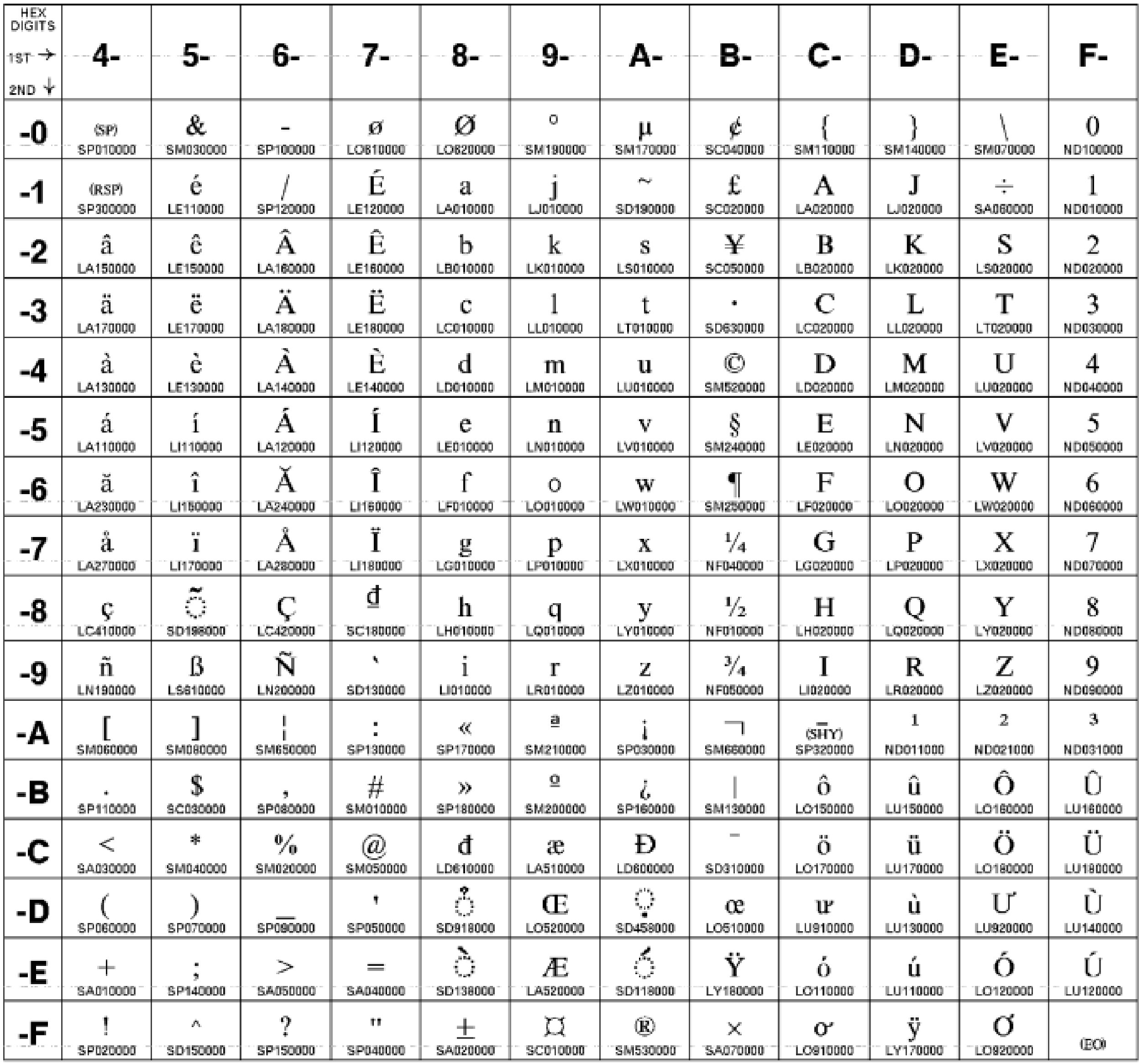
Code Page 01123

[Top of Page on page 1243 | Previous Page on page 1242 | Next Page on page 1244 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1130 Vietnam

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ă LA230000	î LI150000	Ă LA240000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	◌̃ SD198000	Ç LC420000	₫ SC180000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	đ LD610000	æ LA510000	Đ LD600000	¯ SD310000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	◌̉ SD918000	Œ LO520000	◌̣ SD458000	œ LO510000	ư LU910000	ù LU130000	Ư LU920000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	◌̀ SD138000	Æ LA520000	◌́ SD118000	Ÿ LY180000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	ơ LO910000	ÿ LY170000	Ơ LO920000	(EO)

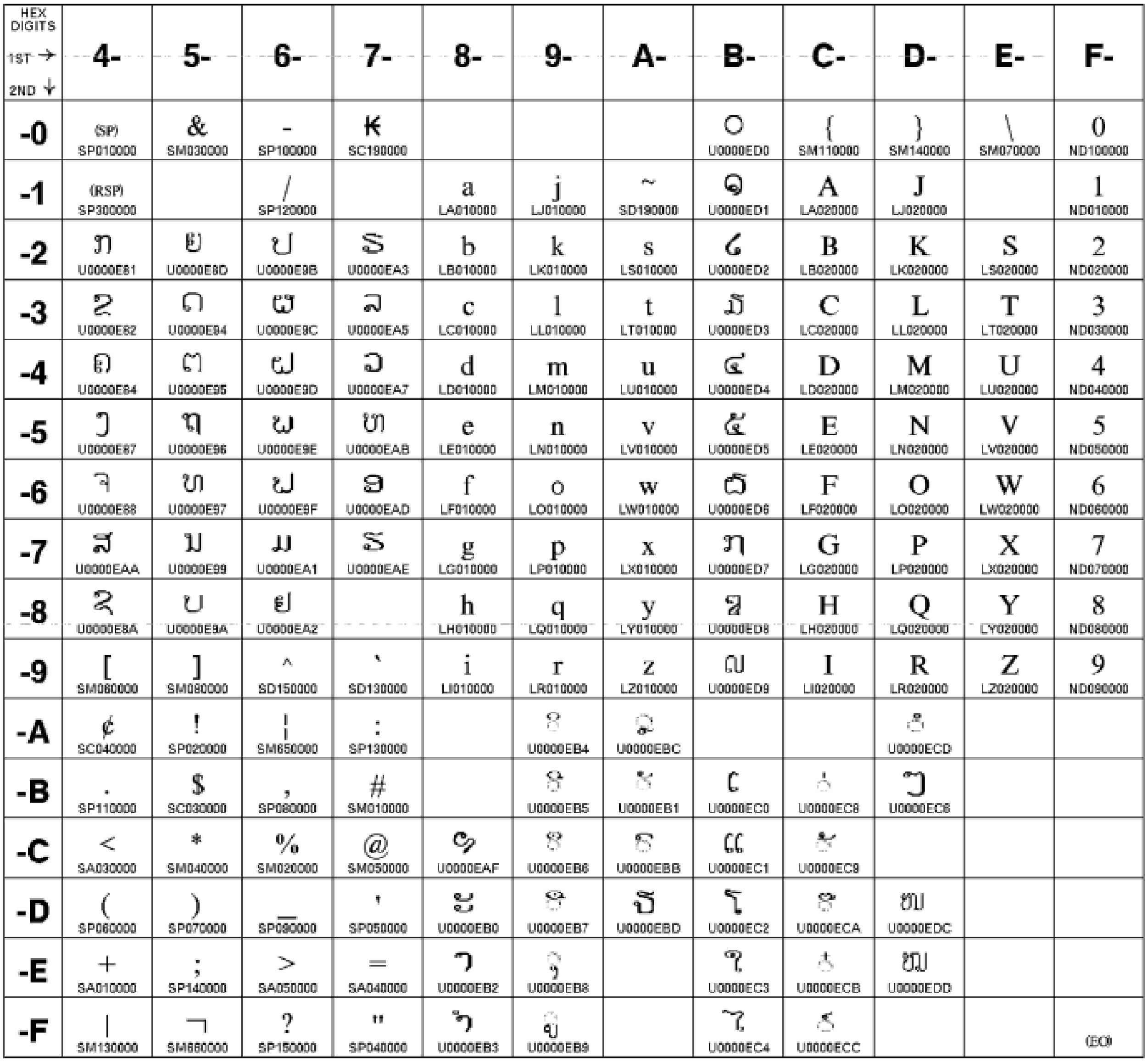
Code Page 01130

[Top of Page on page 1244 | Previous Page on page 1243 | Next Page on page 1245 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1132 Laos

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	₭ SC190000				໐ U0000ED0	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000		/ SP120000		a LA010000	j LJ010000	~ SD190000	໑ U0000ED1	A LA020000	J LJ020000		1 ND010000
-2	ກ U0000E81	ຍ U0000E8D	ປ U0000E9B	ຣ U0000EA3	b LB010000	k LK010000	s LS010000	໒ U0000ED2	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ຂ U0000E82	ດ U0000E94	ຜ U0000E9C	ລ U0000EA5	c LC010000	l LL010000	t LT010000	໓ U0000ED3	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ຄ U0000E84	ຕ U0000E95	ຝ U0000E9D	ວ U0000EA7	d LD010000	m LM010000	u LU010000	໔ U0000ED4	D LD020000	M LM020000	U LU020000	4 ND040000
-5	ງ U0000E87	ຖ U0000E96	ພ U0000E9E	ຫ U0000EAB	e LE010000	n LN010000	v LV010000	໕ U0000ED5	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ຈ U0000E88	ທ U0000E97	ຟ U0000E9F	ອ U0000EAD	f LF010000	o LO010000	w LW010000	໖ U0000ED6	F LF020000	O LO020000	W LW020000	6 ND060000
-7	ສ U0000EAA	ນ U0000E99	ມ U0000EA1	ຮ U0000EAE	g LG010000	p LP010000	x LX010000	໗ U0000ED7	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ຊ U0000E8A	ບ U0000E9A	ຢ U0000EA2		h LH010000	q LQ010000	y LY010000	໘ U0000ED8	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	[SM060000	] SM080000	^ SD150000	` SD130000	i LI010000	r LR010000	z LZ010000	໙ U0000ED9	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000		◌ິ U0000EB4	◌ຼ U0000EBC			◌ໍ U0000ECD		
-B	. SP110000	$ SC030000	, SP080000	# SM010000		◌ີ U0000EB5	◌ັ U0000EB1	ເ U0000EC0	◌່ U0000EC8	ໆ U0000EC6		
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ຯ U0000EAF	◌ຶ U0000EB6	◌ົ U0000EBB	ແ U0000EC1	◌້ U0000EC9			
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ະ U0000EB0	◌ື U0000EB7	ຽ U0000EBD	ໂ U0000EC2	◌໊ U0000ECA	ໜ U0000EDC		
-E	+ SA010000	; SP140000	> SA050000	= SA040000	າ U0000EB2	◌ຸ U0000EB8		ໃ U0000EC3	◌໋ U0000ECB	ໝ U0000EDD		
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ຳ U0000EB3	◌ູ U0000EB9		ໄ U0000EC4	◌໌ U0000ECC			(EO)

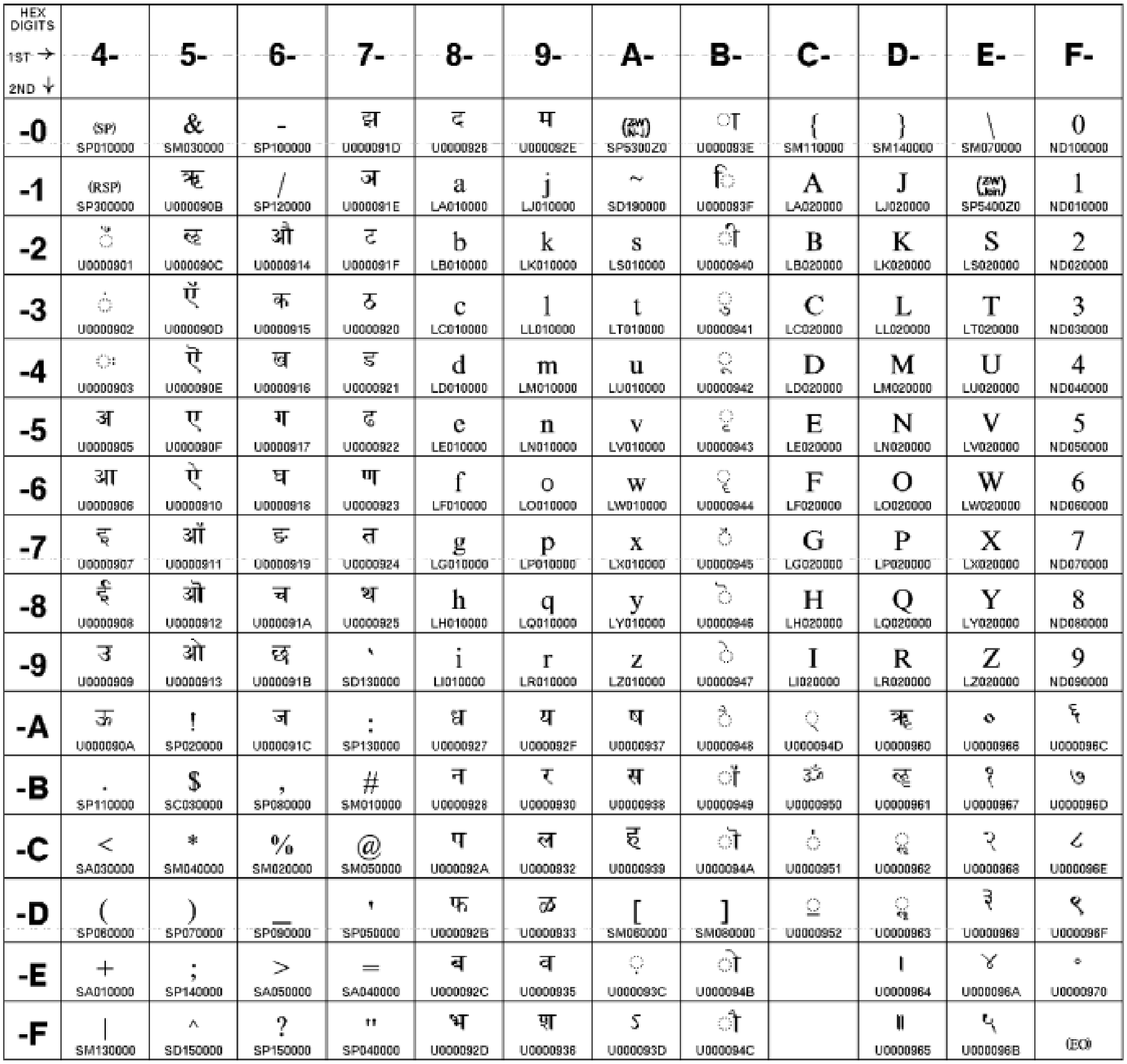
Code Page 01132

[Top of Page on page 1245 | Previous Page on page 1244 | Next Page on page 1246 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1137 India

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	झ U000091D	द U0000926	म U000092E	(ZW NJ) SP5300Z0	◌ा U000093E	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	ऋ U000090B	/ SP120000	ञ U000091E	a LA010000	j LJ010000	~ SD190000	ि◌ U000093F	A LA020000	J LJ020000	(ZW Join) SP5400Z0	1 ND010000
-2	◌ँ U0000901	ऌ U000090C	औ U0000914	ट U000091F	b LB010000	k LK010000	s LS010000	◌ी U0000940	B LB020000	K LK020000	S LS020000	2 ND020000
-3	◌ं U0000902	ऍ U000090D	क U0000915	ठ U0000920	c LC010000	l LL010000	t LT010000	◌ु U0000941	C LC020000	L LL020000	T LT020000	3 ND030000
-4	◌ः U0000903	ऎ U000090E	ख U0000916	ड U0000921	d LD010000	m LM010000	u LU010000	◌ू U0000942	D LD020000	M LM020000	U LU020000	4 ND040000
-5	अ U0000905	ए U000090F	ग U0000917	ढ U0000922	e LE010000	n LN010000	v LV010000	◌ृ U0000943	E LE020000	N LN020000	V LV020000	5 ND050000
-6	आ U0000906	ऐ U0000910	घ U0000918	ण U0000923	f LF010000	o LO010000	w LW010000	◌ॄ U0000944	F LF020000	O LO020000	W LW020000	6 ND060000
-7	इ U0000907	ऑ U0000911	ङ U0000919	त U0000924	g LG010000	p LP010000	x LX010000	◌ॅ U0000945	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ई U0000908	ऒ U0000912	च U000091A	थ U0000925	h LH010000	q LQ010000	y LY010000	◌ॆ U0000946	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	उ U0000909	ओ U0000913	छ U000091B	` SD130000	i LI010000	r LR010000	z LZ010000	◌े U0000947	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	ऊ U000090A	! SP020000	ज U000091C	: SP130000	ध U0000927	य U000092F	ष U0000937	◌ै U0000948	◌् U000094D	ॠ U0000960	० U0000966	६ U000096C
-B	. SP110000	$ SC030000	, SP080000	# SM010000	न U0000928	र U0000930	स U0000938	◌ॉ U0000949	ॐ U0000950	ॡ U0000961	१ U0000967	७ U000096D
-C	< SA030000	* SM040000	% SM020000	@ SM050000	प U000092A	ल U0000932	ह U0000939	◌ॊ U000094A	◌॑ U0000951	◌ॢ U0000962	२ U0000968	८ U000096E
-D	(SP060000	) SP070000	_ SP090000	' SP050000	फ U000092B	ळ U0000933	[SM060000	] SM080000	◌॒ U0000952	◌ॣ U0000963	३ U0000969	९ U000096F
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ब U000092C	व U0000935	◌़ U000093C	◌ो U000094B		। U0000964	४ U000096A	॰ U0000970
-F	| SM130000	^ SD150000	? SP150000	" SP040000	भ U000092D	श U0000936	ऽ U000093D	◌ौ U000094C		॥ U0000965	५ U000096B	(EO)

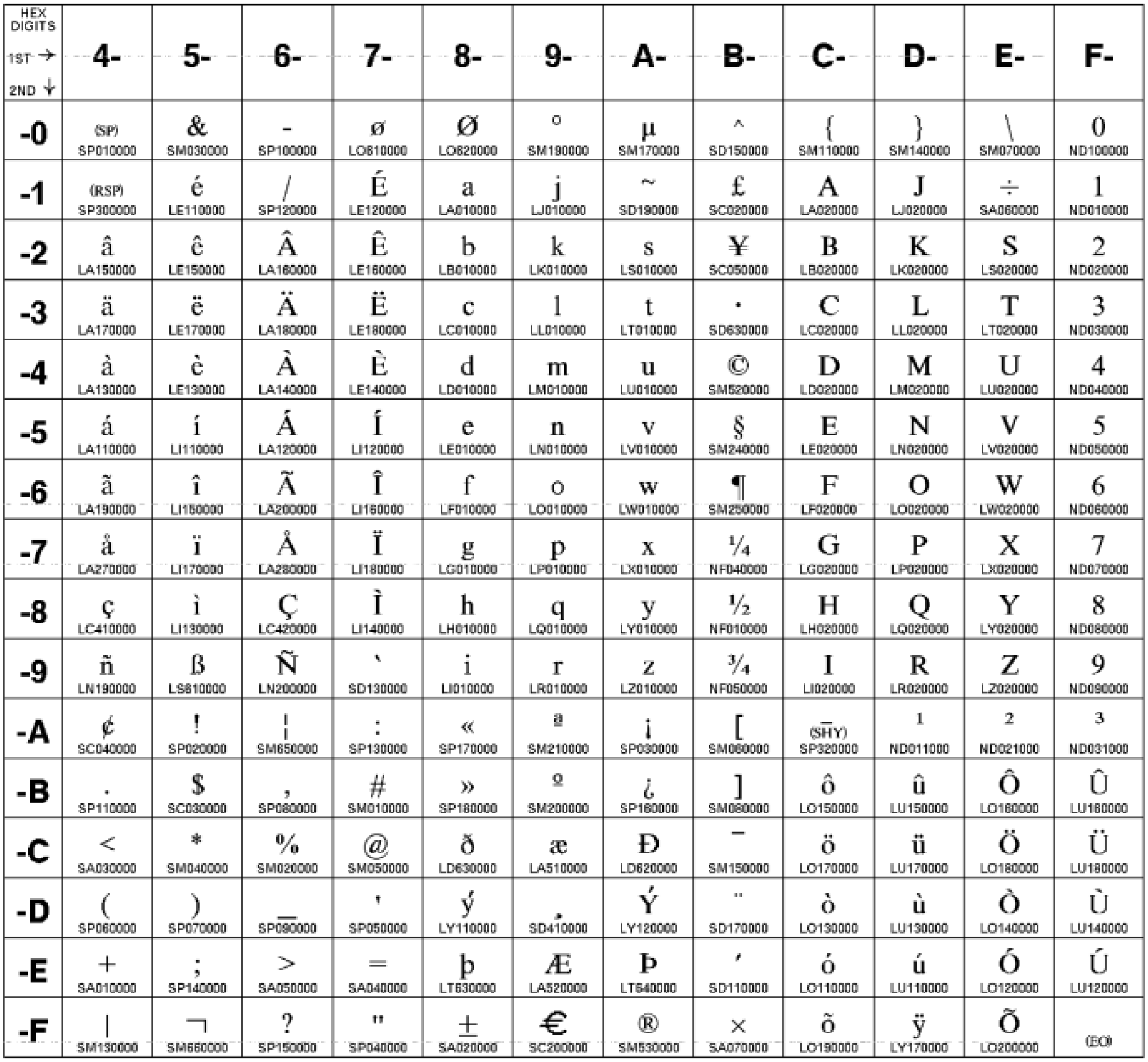
Code Page 01137

[Top of Page on page 1246 | Previous Page on page 1245 | Next Page on page 1247 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1140-1/695-1 Brazil, Canada, Netherlands, Portugal, U.S., and 1140/1175 Traditional Chinese

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	[SM060000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	] SM080000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

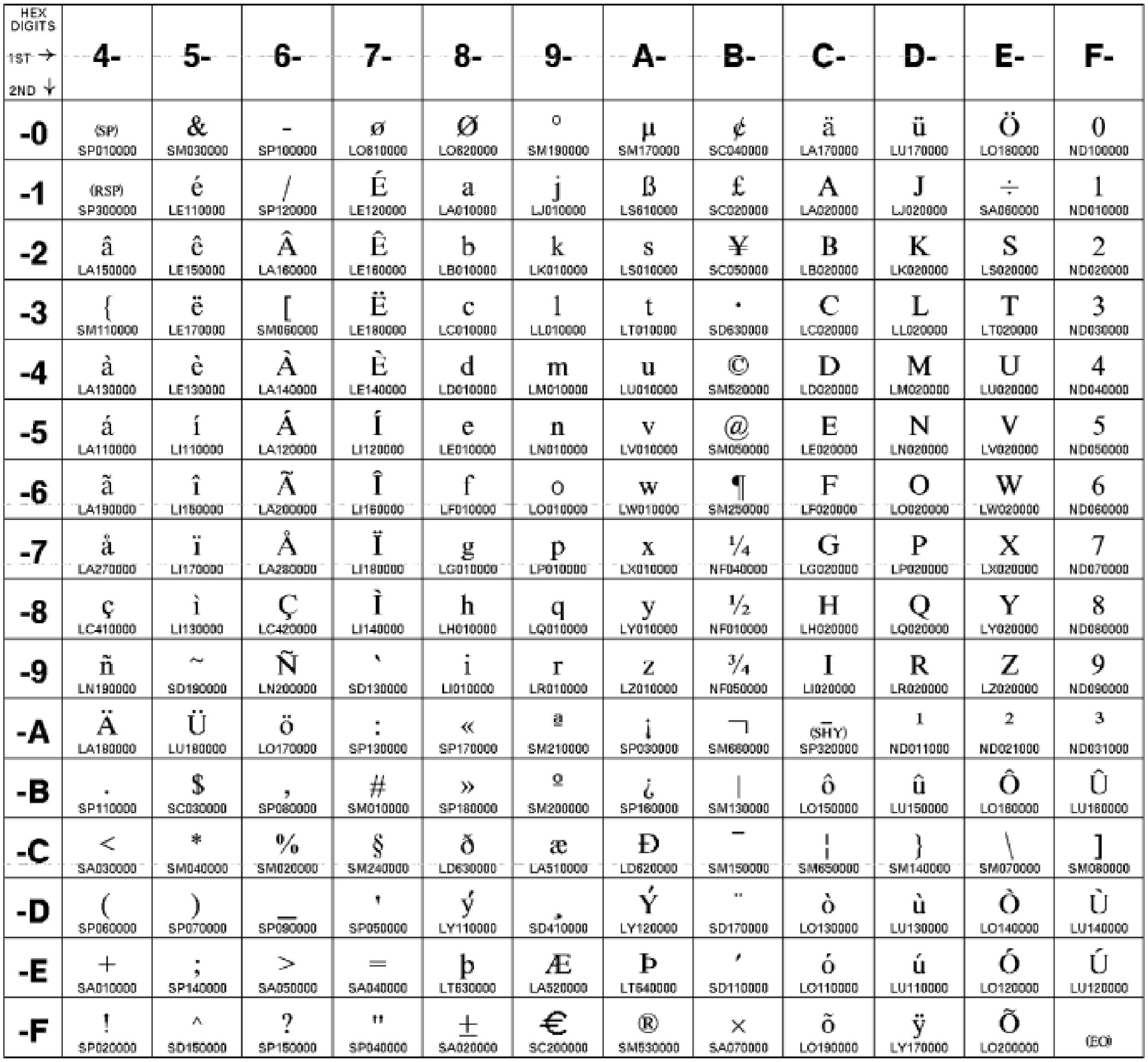
Code Page 01140

[Top of Page on page 1247 | Previous Page on page 1246 | Next Page on page 1248 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1141-1/695-1 Austria, Germany

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	ü LU170000	Ö LO180000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ß LS610000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	[SM060000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	@ SM050000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	~ SD190000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ä LA180000	Ü LU180000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM680000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	§ SM240000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	¦ SM650000	} SM140000	\ SM070000	] SM080000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

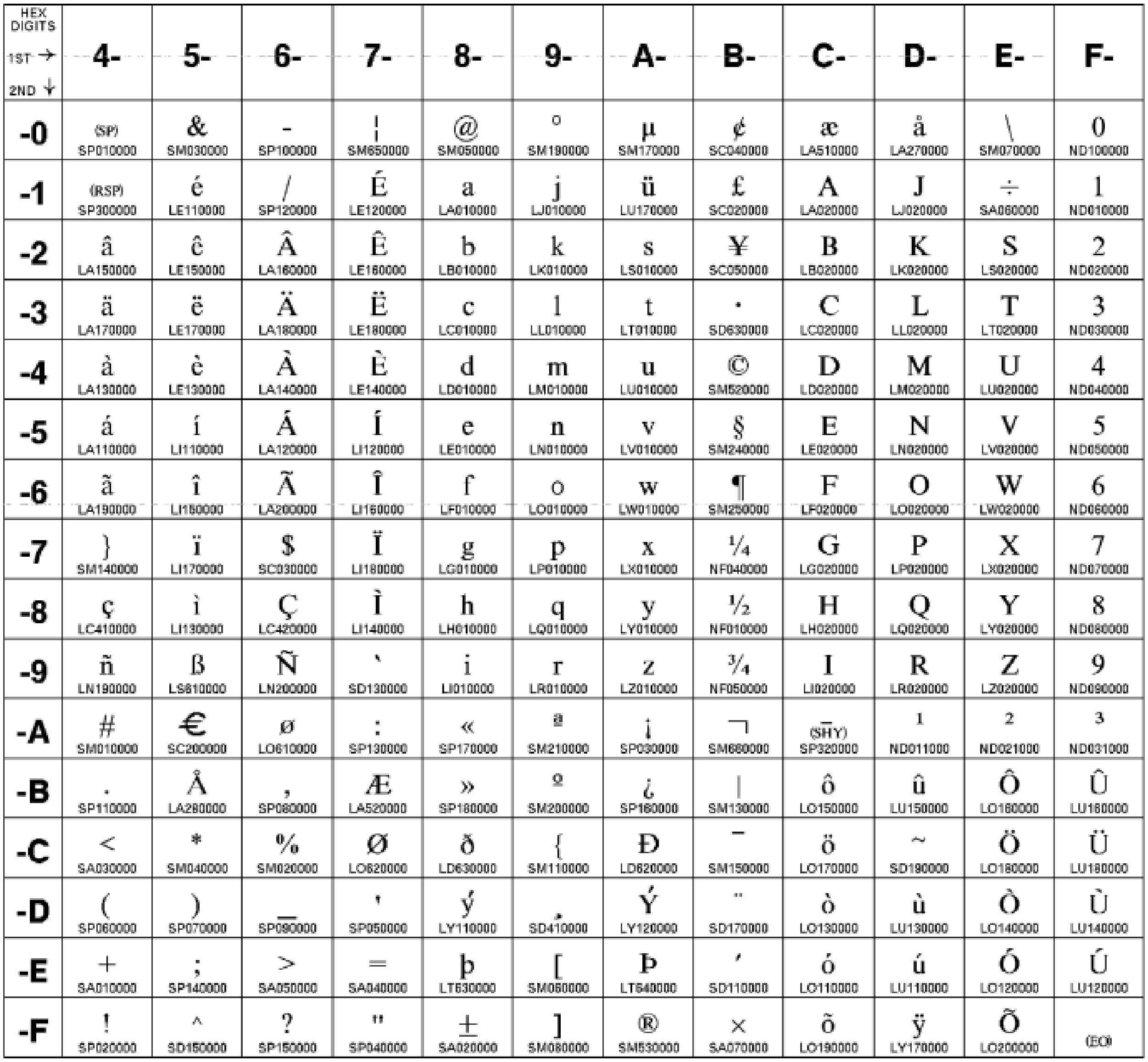
Code Page 01141

[Top of Page on page 1248 | Previous Page on page 1247 | Next Page on page 1249 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1142-1/695-1 Denmark, Norway

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	¦ SM650000	@ SM050000	° SM190000	µ SM170000	¢ SC040000	æ LA510000	å LA270000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	# SM010000	€ SC200000	ø LO610000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Æ LA520000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ø LO620000	ð LD630000	{ SM110000	Ð LD620000	¯ SM150000	ö LO170000	~ SD190000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	[SM060000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

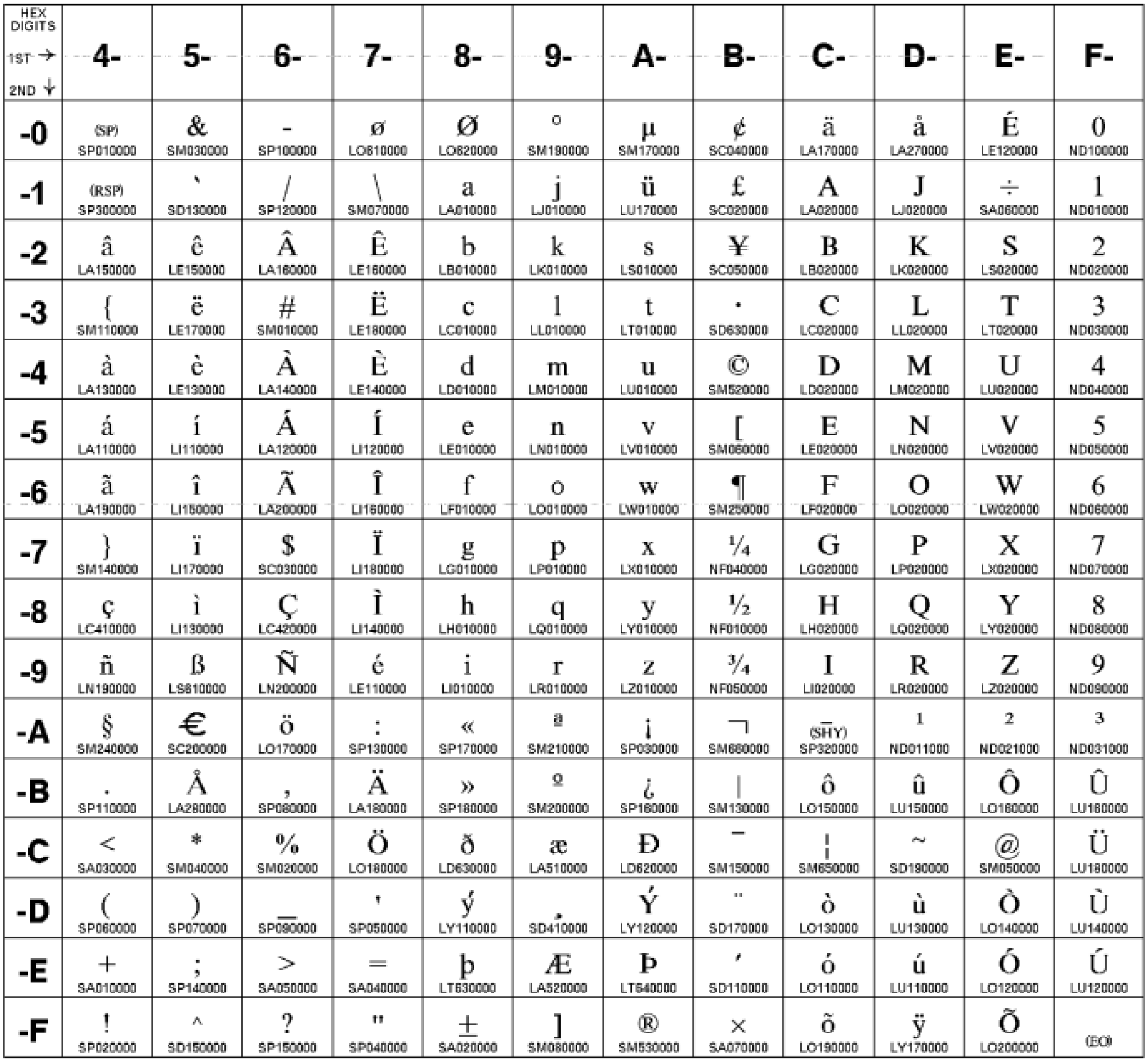
Code Page 01142

[Top of Page on page 1249 | Previous Page on page 1248 | Next Page on page 1250 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1143-1/695-1 Finland, Sweden

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	å LA270000	É LE120000	0 ND100000
-1	(RSP) SP300000	` SD130000	/ SP120000	\ SM070000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	# SM010000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	[SM060000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	é LE110000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	§ SM240000	€ SC200000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Ä LA180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ö LO180000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	¦ SM650000	~ SD190000	@ SM050000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

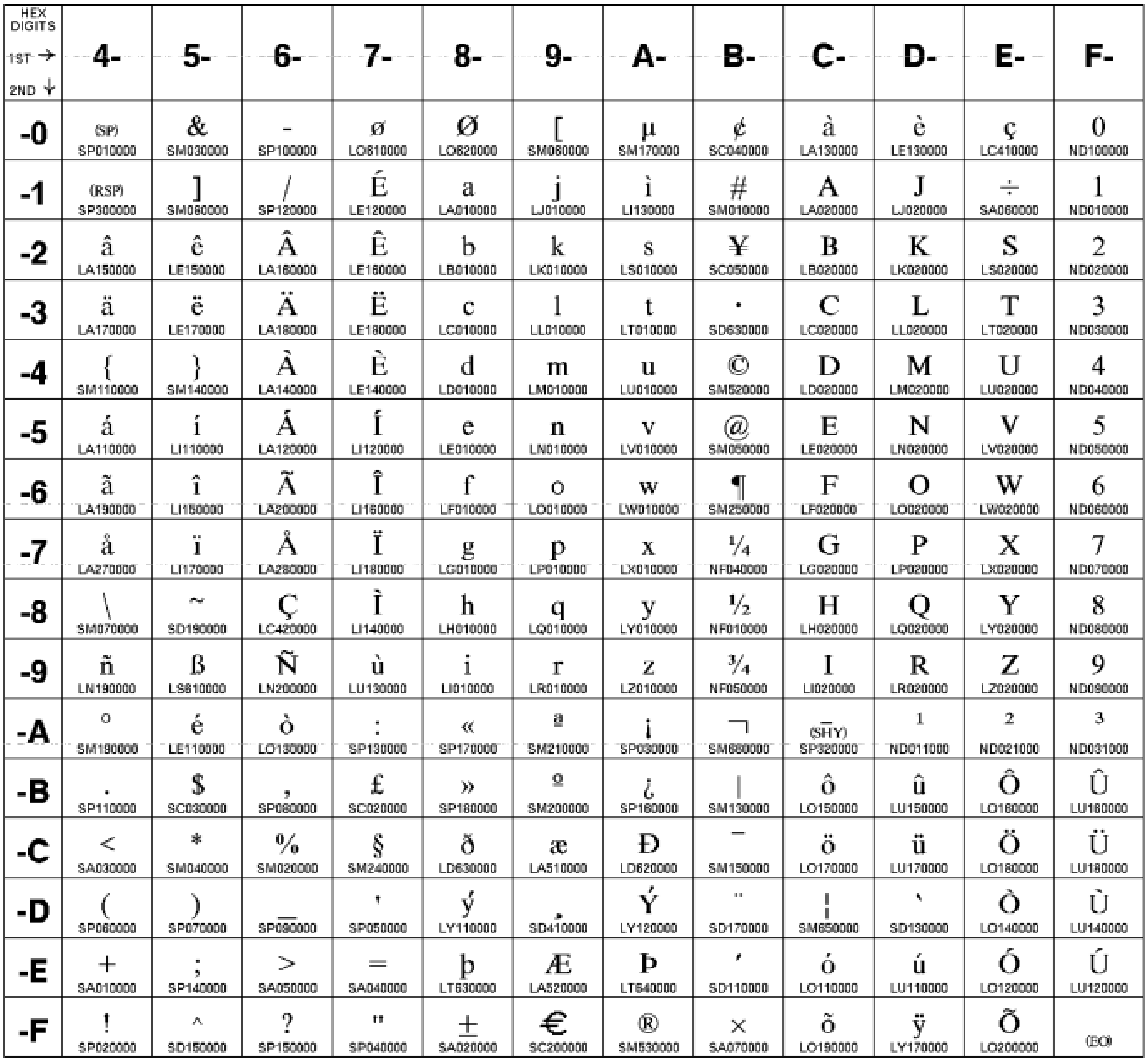
Code Page 01143

[Top of Page on page 1250 | Previous Page on page 1249 | Next Page on page 1251 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1144-1/695-1 Italy

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	[SM060000	µ SM170000	¢ SC040000	à LA130000	è LE130000	ç LC410000	0 ND100000
-1	(RSP) SP300000	] SM080000	/ SP120000	É LE120000	a LA010000	j LJ010000	ì LI130000	# SM010000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	{ SM110000	} SM140000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	@ SM050000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	\ SM070000	~ SD190000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ù LU130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	° SM190000	é LE110000	ò LO130000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	£ SC020000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	§ SM240000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	¦ SM650000	` SD130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

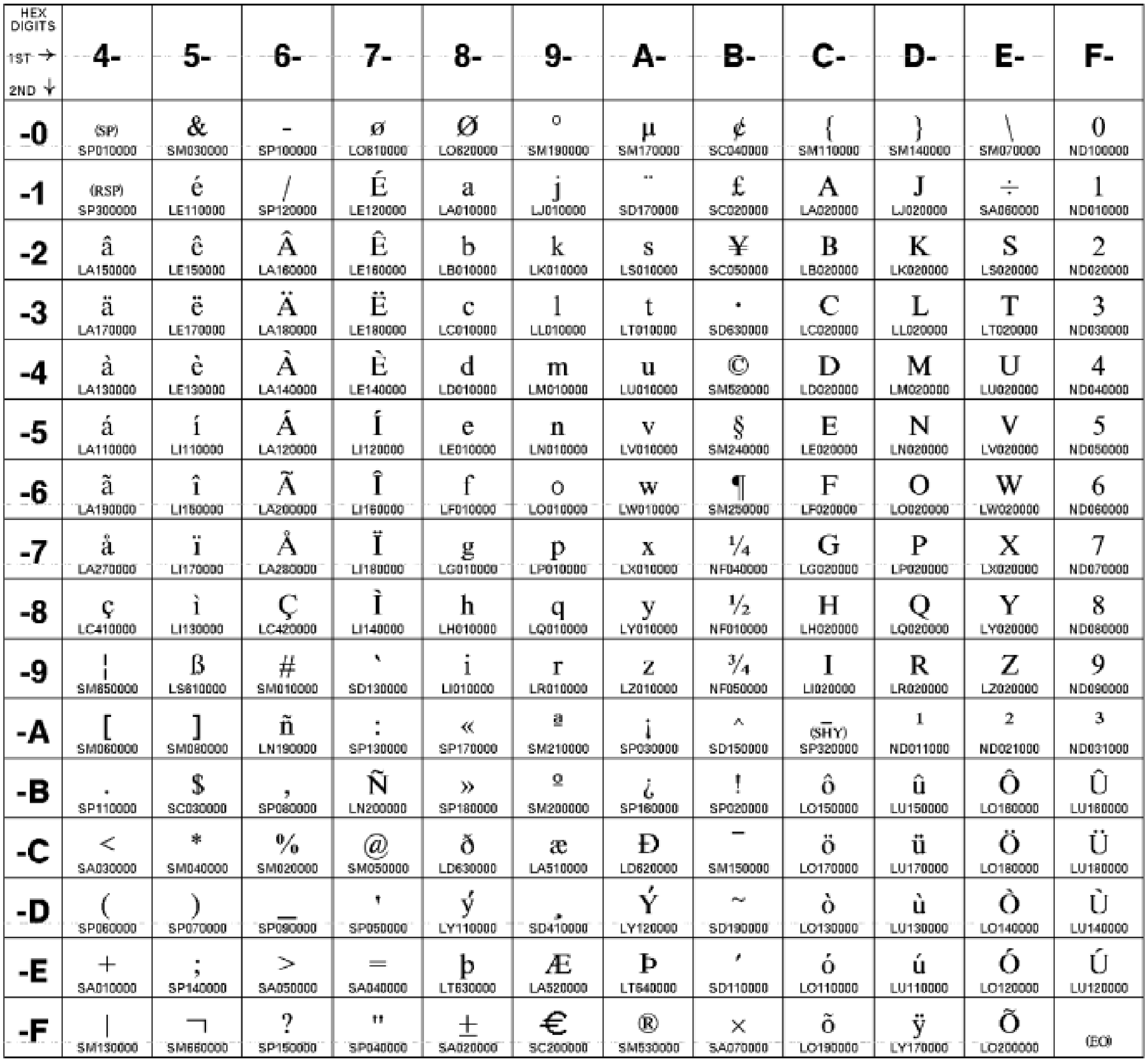
Code Page 01144

[Top of Page on page 1251 | Previous Page on page 1250 | Next Page on page 1252 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1145-1/695-1 Latin America, Spain

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¨ SD170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	¦ SM650000	ß LS610000	# SM010000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	ñ LN190000	: SP130000	« SP170000	ª SM210000	¡ SP030000	^ SD150000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	Ñ LN200000	» SP180000	º SM200000	¿ SP160000	! SP020000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	~ SD190000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

Code Page 01145

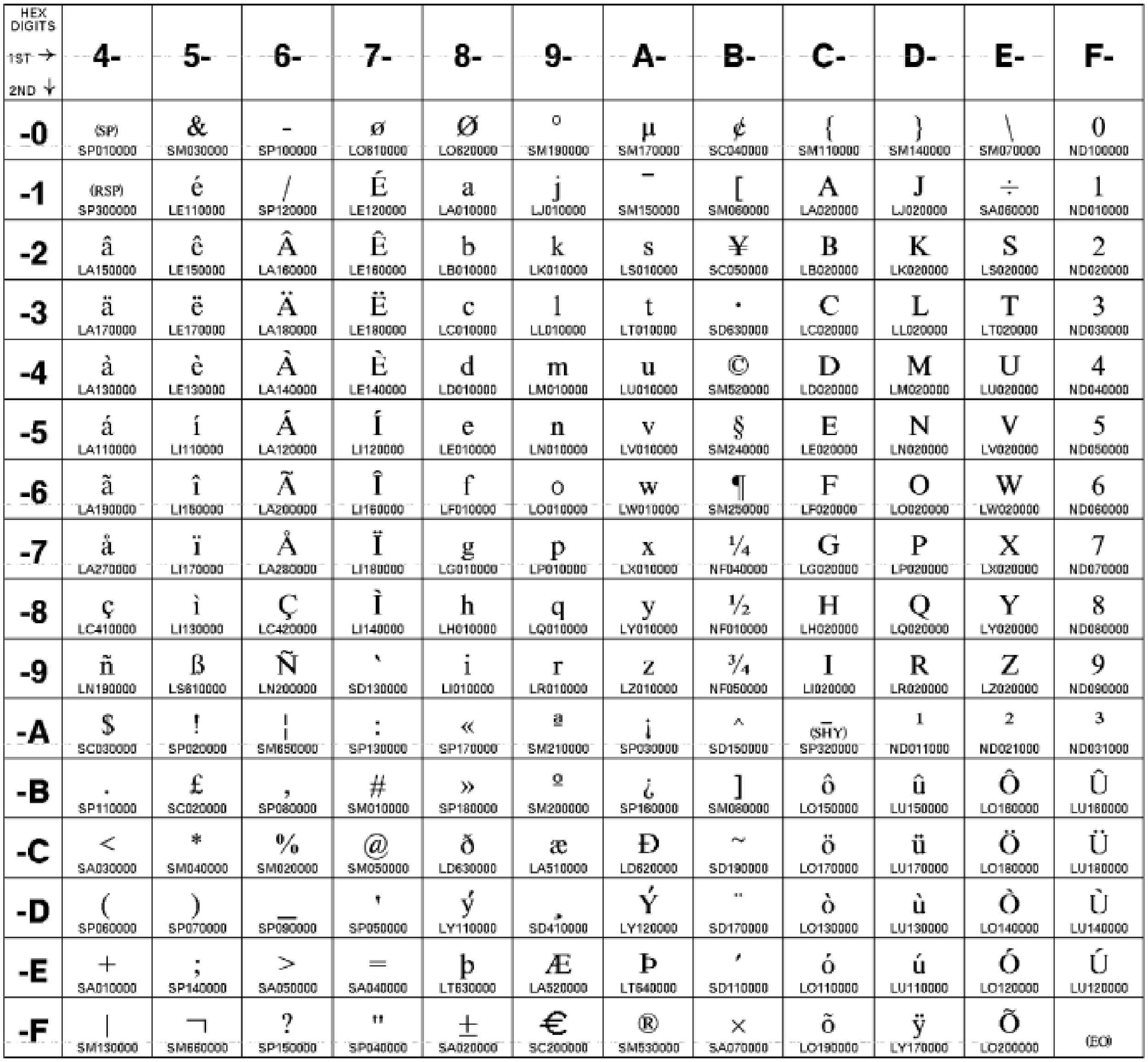
[Top of Page on page 1252 | Previous Page on page 1251 | Next Page on page 1253 | Table of Contents on page 1216]

Host Code Page Reference

hcp_

Host Code Page 1146-1/695-1 United Kingdom

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¯ SM150000	[SM060000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	$ SC030000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	^ SD150000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	£ SC020000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	] SM080000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	~ SD190000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

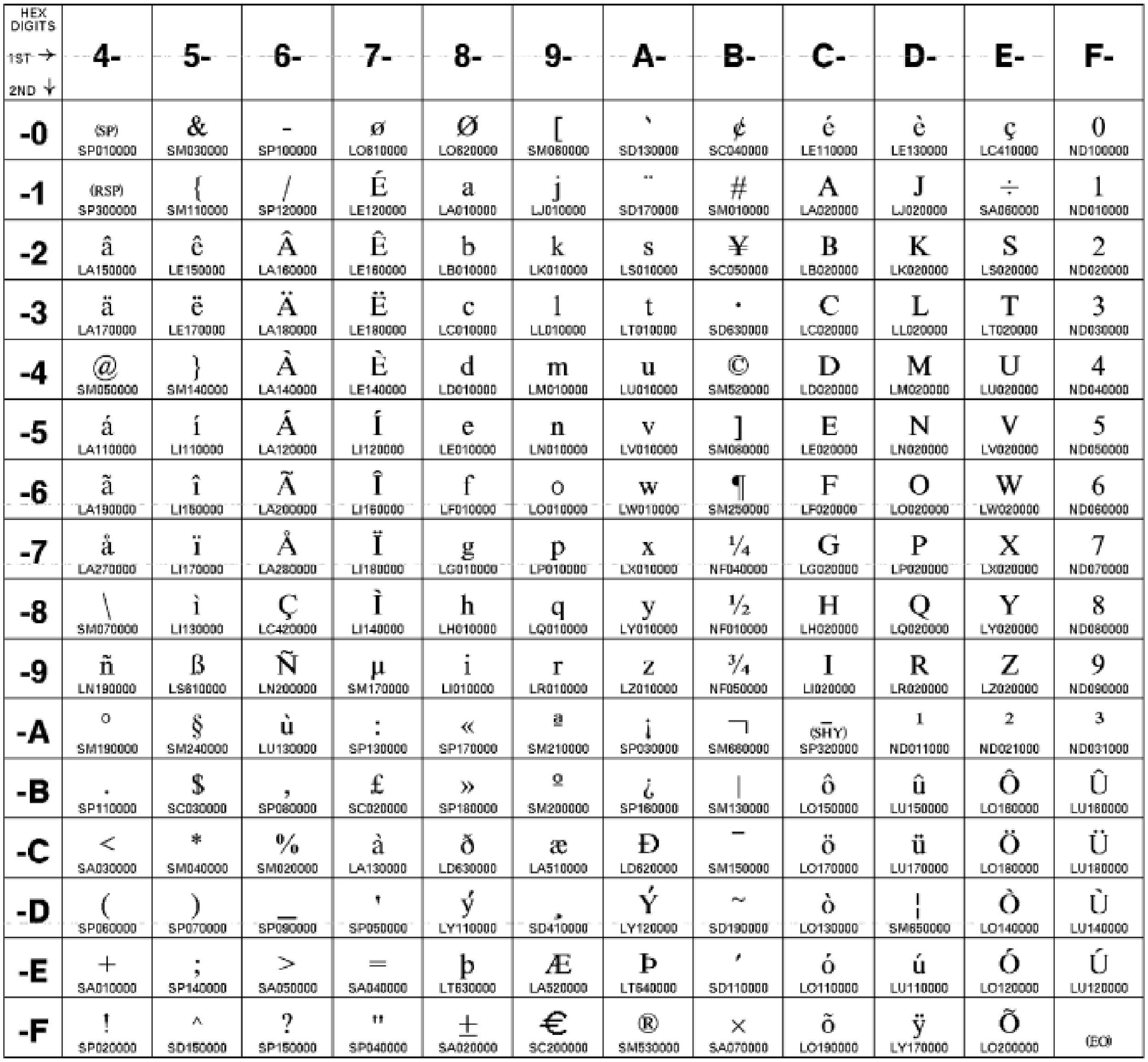
Code Page 01146

[Top of Page on page 1254 | Previous Page on page 1252 | Next Page on page 1255 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1147-1/695-1 France

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	[SM060000	` SD130000	¢ SC040000	é LE110000	è LE130000	ç LC410000	0 ND100000
-1	(RSP) SP300000	{ SM110000	/ SP120000	É LE120000	a LA010000	j LJ010000	¨ SD170000	# SM010000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	@ SM050000	} SM140000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	] SM080000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	\\ SM070000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	µ SM170000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	° SM190000	§ SM240000	ù LU130000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	£ SC020000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	à LA130000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	~ SD190000	ò LO130000	¦ SM650000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

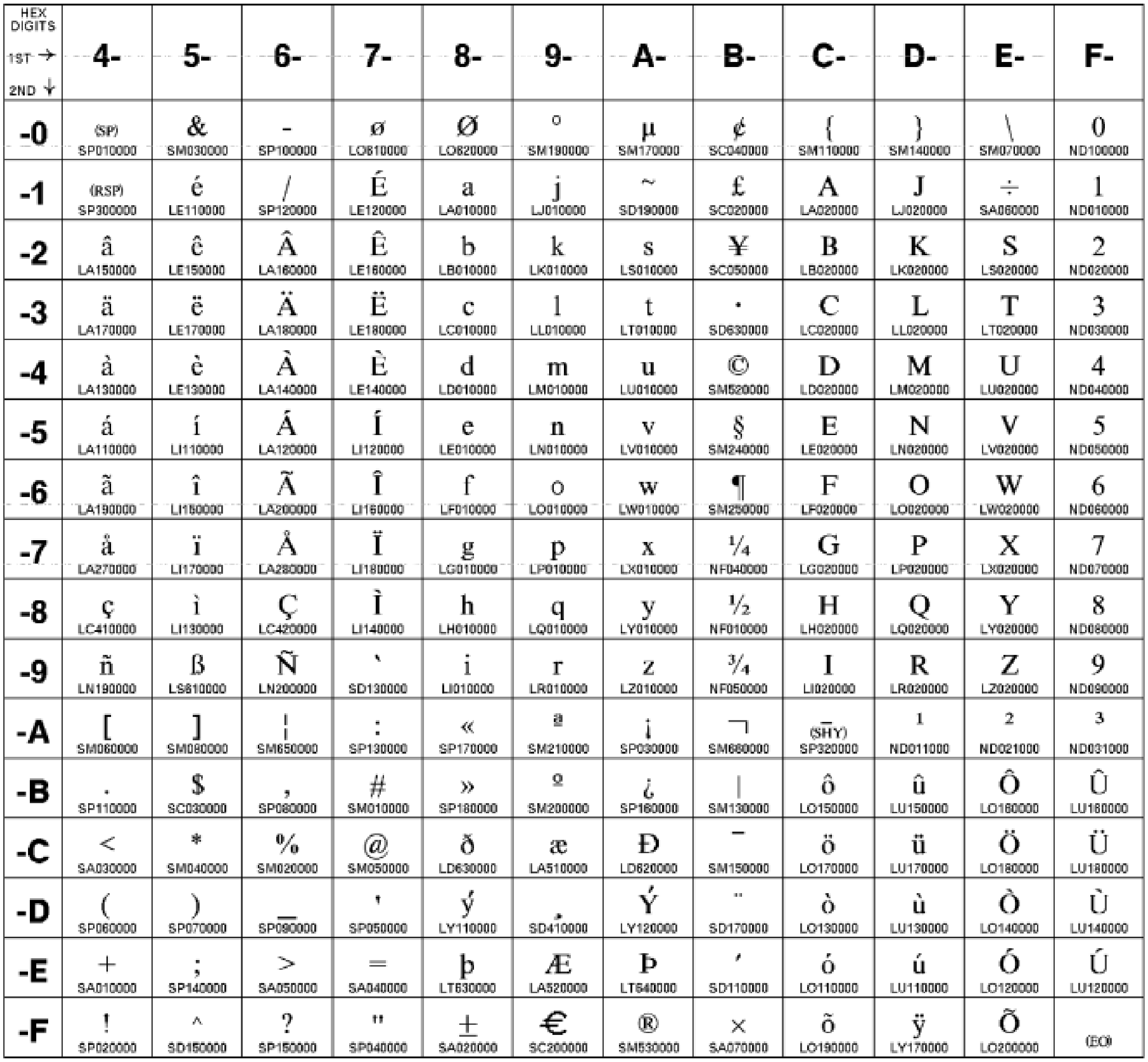
Code Page 01147

[Top of Page on page 1255 | Previous Page on page 1253 | Next Page on page 1256 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1148-1/695-1 International

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Đ LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

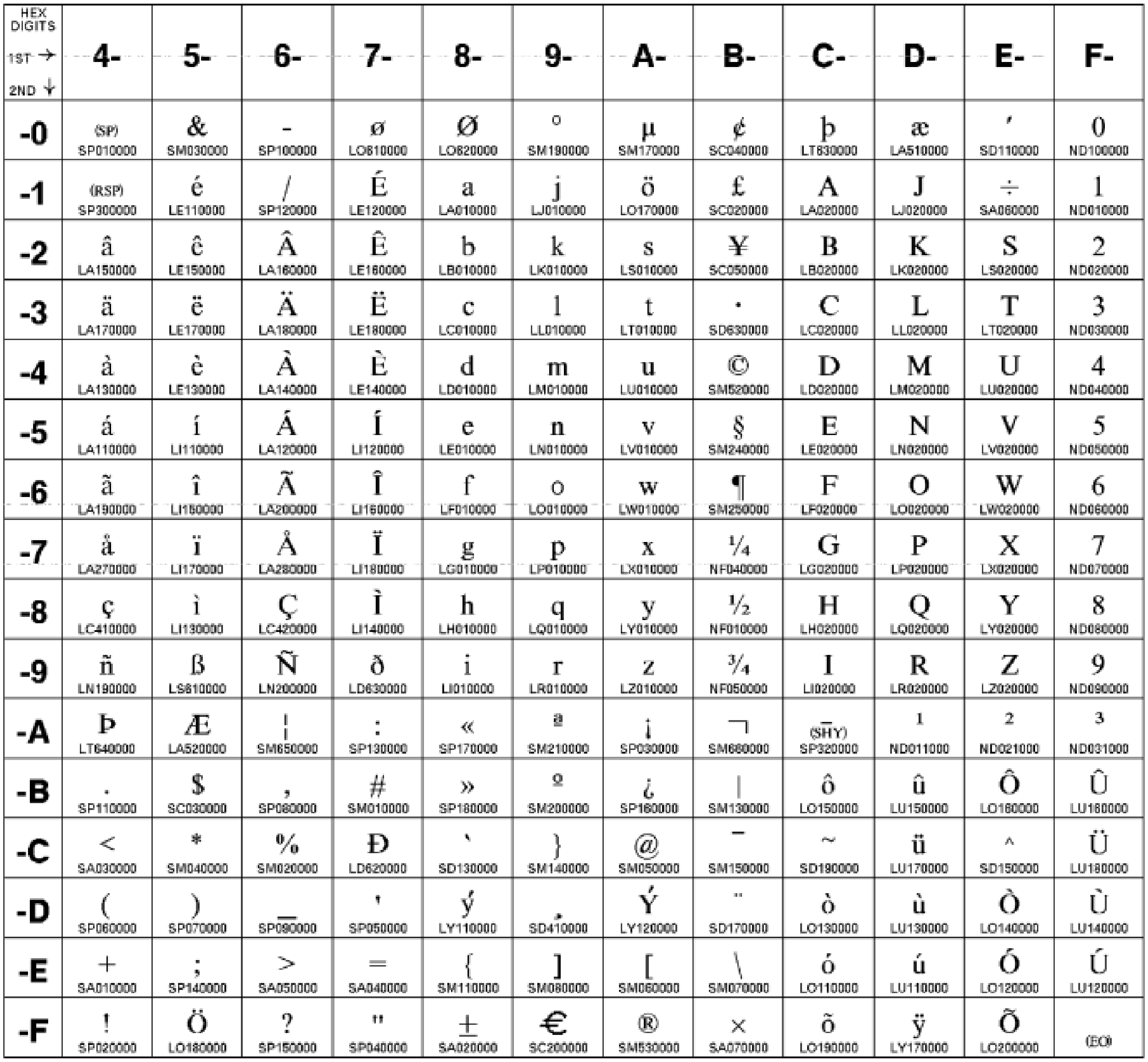
Code Page 01148

[Top of Page on page 1256 | Previous Page on page 1255 | Next Page on page 1257 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1149-1/695-1 Iceland

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	þ LT630000	æ LA510000	´ SD110000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ö LO170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ð LD630000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Þ LT640000	Æ LA520000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ð LD620000	` SD130000	} SM140000	@ SM050000	¯ SM150000	~ SD190000	ü LU170000	^ SD150000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	{ SM110000	] SM080000	[SM060000	\ SM070000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	Ö LO180000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

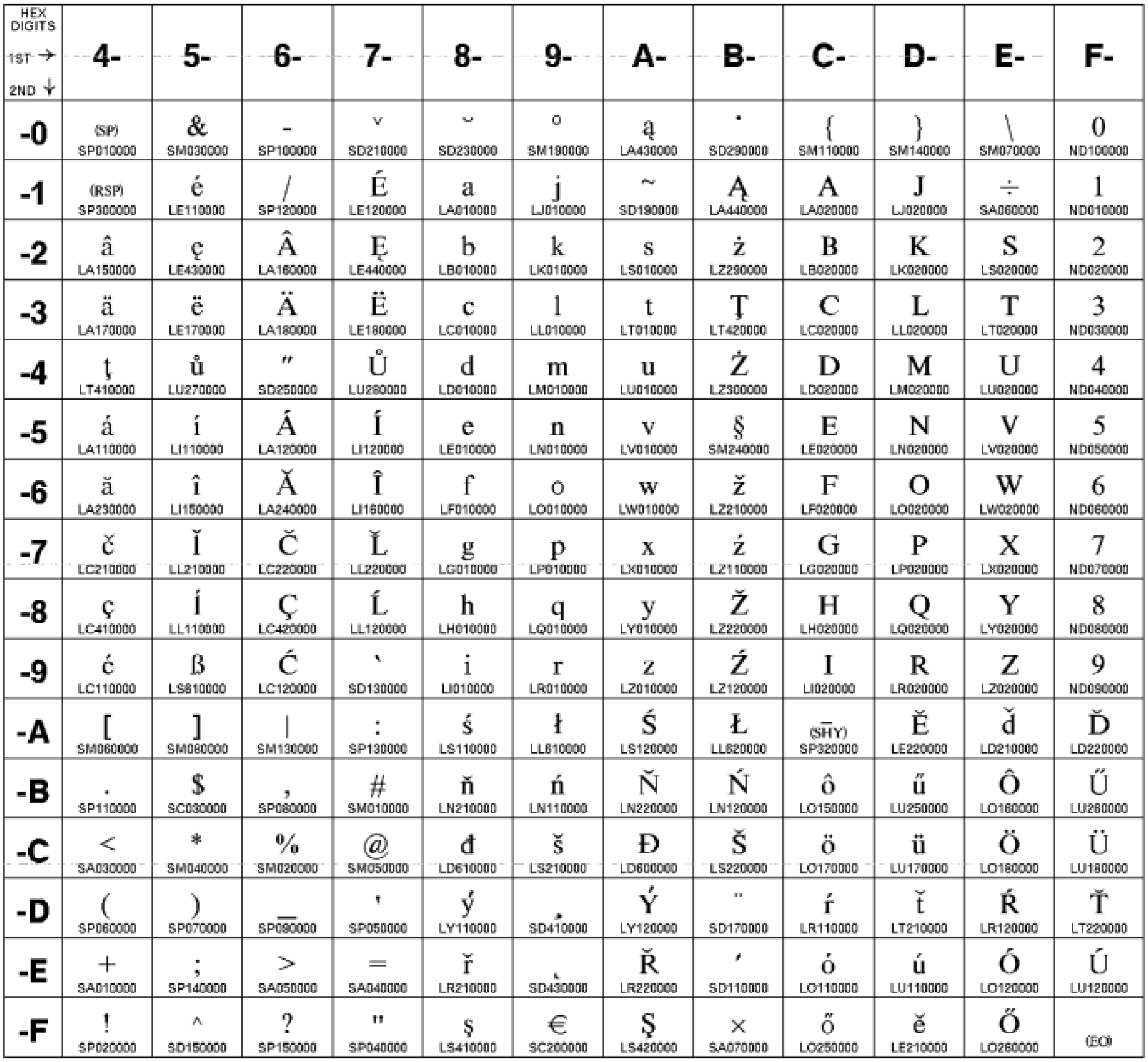
Code Page 01149

[Top of Page on page 1257 | Previous Page on page 1256 | Next Page on page 1258 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1153/1375 Latin 2 - EBCDIC Multilingual

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ˇ SD210000	˘ SD230000	° SM190000	ą LA430000	˙ SD290000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	Ą LA440000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ę LE430000	Â LA160000	Ę LE440000	b LB010000	k LK010000	s LS010000	ż LZ290000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	Ţ LT420000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ţ LT410000	ů LU270000	˝ SD250000	Ů LU280000	d LD010000	m LM010000	u LU010000	Ż LZ300000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ă LA230000	î LI150000	Ă LA240000	Î LI160000	f LF010000	o LO010000	w LW010000	ž LZ210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	č LC210000	ľ LL210000	Č LC220000	Ľ LL220000	g LG010000	p LP010000	x LX010000	ź LZ110000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ĺ LL110000	Ç LC420000	Ĺ LL120000	h LH010000	q LQ010000	y LY010000	Ž LZ220000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ć LC110000	ß LS610000	Ć LC120000	` SD130000	i LI010000	r LR010000	z LZ010000	Ź LZ120000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	ś LS110000	ł LL610000	Ś LS120000	Ł LL620000	(SHY) SP320000	Ě LE220000	ď LD210000	Ď LD220000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	ň LN210000	ń LN110000	Ň LN220000	Ń LN120000	ô LO150000	ű LU250000	Ô LO160000	Ű LU260000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	đ LD610000	š LS210000	Đ LD600000	Š LS220000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	˛ SD410000	Ý LY120000	¨ SD170000	ŕ LR110000	ť LT210000	Ŕ LR120000	Ť LT220000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ř LR210000	¸ SD430000	Ř LR220000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	ş LS410000	€ SC200000	Ş LS420000	× SA070000	ő LO250000	ě LE210000	Ő LO260000	(EO)

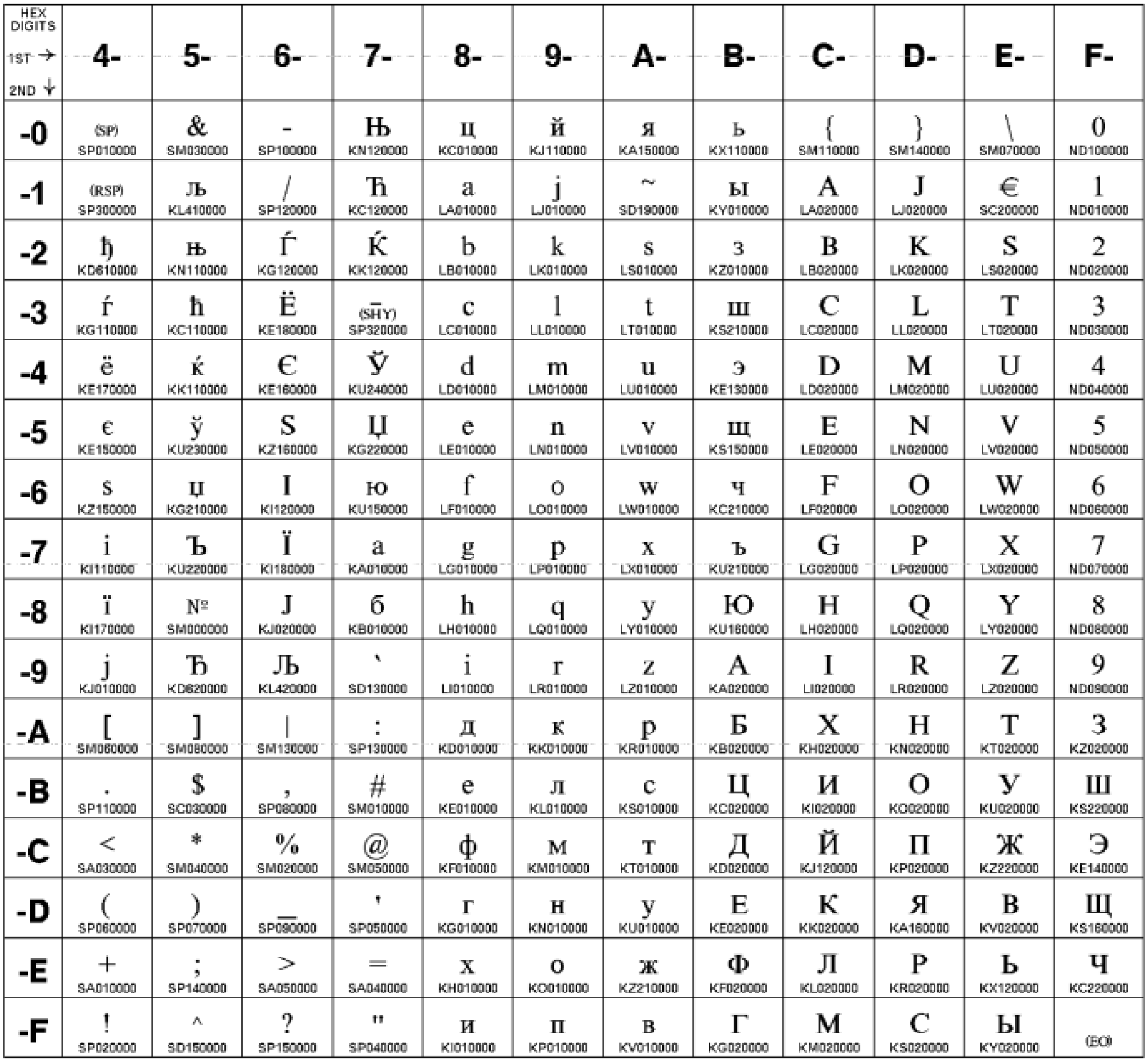
Code Page 01153

[Top of Page on page 1258 | Previous Page on page 1257 | Next Page on page 1259 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1154/1381 Cyrillic

The column indicates the first digit, and the row indicates the second digit. If this code page is not used with a Cyrillic host machine, certain characters might not display properly.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	Њ KN120000	ц KC010000	й KJ110000	я KA150000	ь KX110000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	љ KL410000	/ SP120000	Ћ KC120000	a LA010000	j LJ010000	~ SD190000	ы KY010000	A LA020000	J LJ020000	€ SC200000	1 ND010000
-2	ђ KD610000	њ KN110000	Ѓ KG120000	Ќ KK120000	b LB010000	k LK010000	s LS010000	з KZ010000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ѓ KG110000	ћ KC110000	Ё KE180000	(SHY) SP320000	c LC010000	l LL010000	t LT010000	ш KS210000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ё KE170000	ќ KK110000	Є KE160000	Ў KU240000	d LD010000	m LM010000	u LU010000	э KE130000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	є KE150000	ў KU230000	Ѕ KZ160000	Џ KG220000	e LE010000	n LN010000	v LV010000	щ KS150000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ѕ KZ150000	џ KG210000	І KI120000	ю KU150000	f LF010000	o LO010000	w LW010000	ч KC210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	і KI110000	Ъ KU220000	Ї KI180000	а KA010000	g LG010000	p LP010000	x LX010000	ъ KU210000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ї KI170000	№ SM000000	Ј KJ020000	б KB010000	h LH010000	q LQ010000	y LY010000	Ю KU160000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ј KJ010000	Ђ KD620000	Љ KL420000	` SD130000	i LI010000	r LR010000	z LZ010000	А KA020000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	д KD010000	к KK010000	р KR010000	Б KB020000	Х KH020000	Н KN020000	Т KT020000	З KZ020000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	е KE010000	л KL010000	с KS010000	Ц KC020000	И KI020000	О KO020000	У KU020000	Ш KS220000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ф KF010000	м KM010000	т KT010000	Д KD020000	Й KJ120000	П KP020000	Ж KZ220000	Э KE140000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	г KG010000	н KN010000	у KU010000	Е KE020000	К KK020000	Я KA160000	В KV020000	Щ KS160000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	х KH010000	о KO010000	ж KZ210000	Ф KF020000	Л KL020000	Р KR020000	Ь KX120000	Ч KC220000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	и KI010000	п KP010000	в KV010000	Г KG020000	М KM020000	С KS020000	Ы KY020000	(EO)

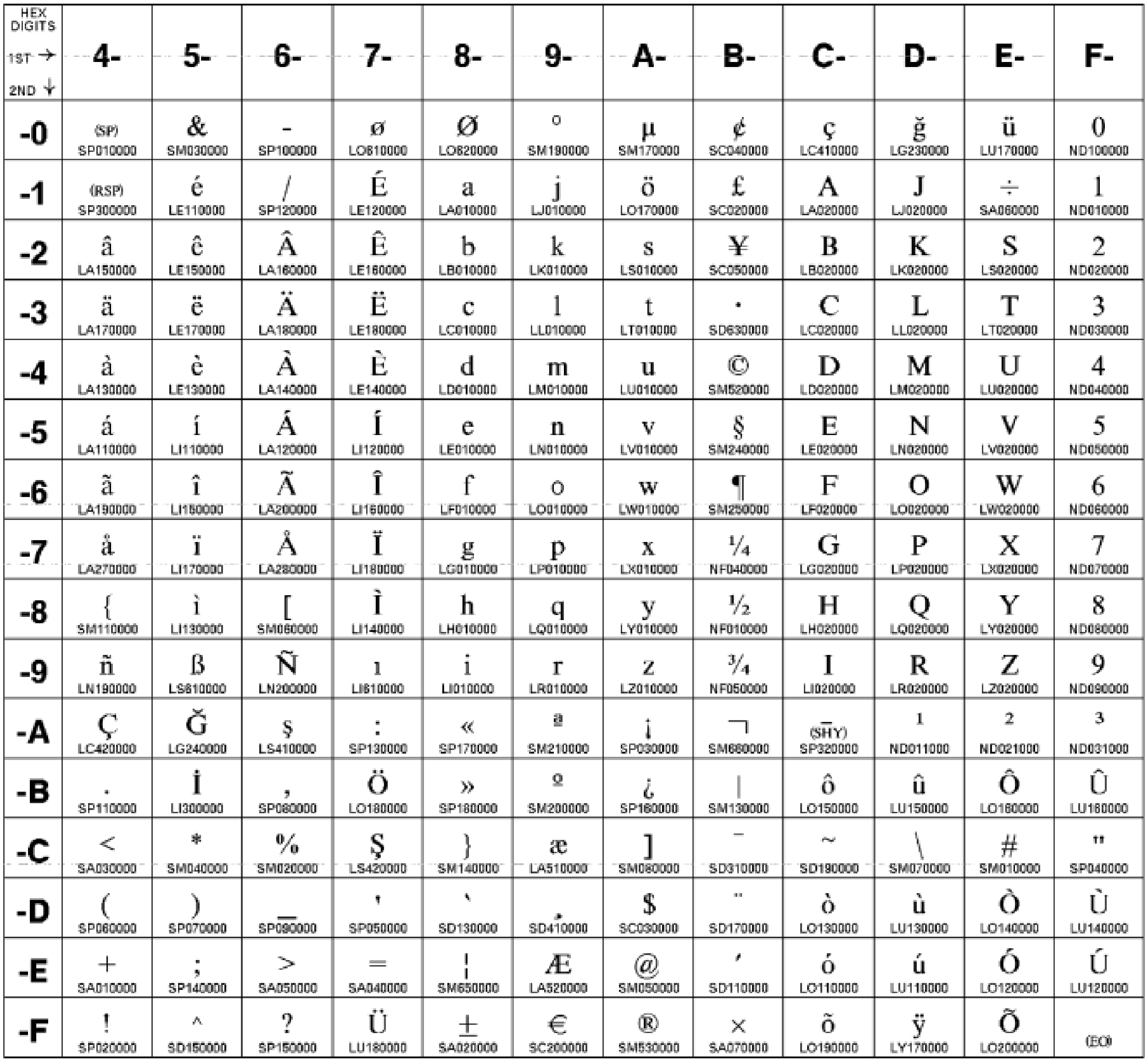
Code Page 01154

[Top of Page on page 1259 | Previous Page on page 1258 | Next Page on page 1260 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1155/1378 Latin 5 - Turkey

The column indicates the first digit, and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ç LC410000	ğ LG230000	ü LU170000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ö LO170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	{ SM110000	ì LI130000	[SM060000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	ı LI610000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ç LC420000	Ğ LG240000	ş LS410000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	İ LI300000	, SP080000	Ö LO180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ş LS420000	} SM140000	æ LA510000	] SM080000	¯ SD310000	~ SD190000	\ SM070000	# SM010000	" SP040000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	` SD130000	¸ SD410000	$ SC030000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	¦ SM650000	Æ LA520000	@ SM050000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	Ü LU180000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

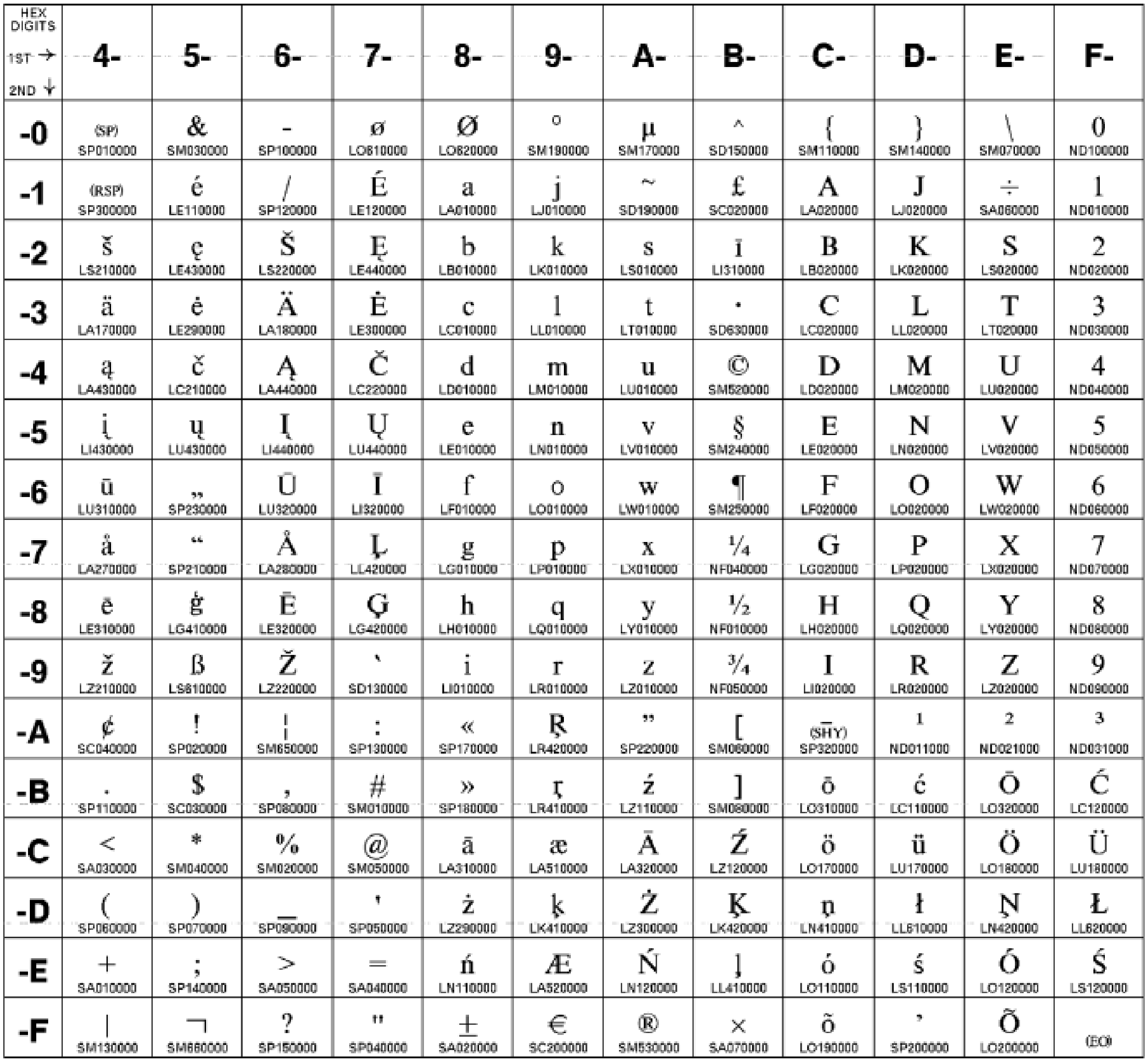
Code Page 01155

[Top of Page on page 1260 | Previous Page on page 1259 | Next Page on page 1261 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1156/1393 Latvia, Lithuania

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	š LS210000	ç LE430000	Š LS220000	Ę LE440000	b LB010000	k LK010000	s LS010000	ı LI310000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ė LE290000	Ä LA180000	Ė LE300000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ą LA430000	č LC210000	Ą LA440000	Č LC220000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	į LI430000	ų LU430000	Į LI440000	Ų LU440000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ū LU310000	„ SP230000	Ū LU320000	Ī LI320000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	“ SP210000	Å LA280000	Ļ LL420000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ē LE310000	ģ LG410000	Ē LE320000	Ģ LG420000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ž LZ210000	ß LS610000	Ž LZ220000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	Ŗ LR420000	” SP220000	[SM060000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	ŗ LR410000	ź LZ110000	] SM080000	ō LO310000	ć LC110000	Ō LO320000	Ć LC120000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ā LA310000	æ LA510000	Ā LA320000	Ź LZ120000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ż LZ290000	ķ LK410000	Ż LZ300000	Ķ LK420000	ņ LN410000	ł LL610000	Ņ LN420000	Ł LL620000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ń LN110000	Æ LA520000	Ń LN120000	ļ LL410000	ó LO110000	ś LS110000	Ó LO120000	Ś LS120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	õ LO190000	’ SP200000	Õ LO200000	(EO)

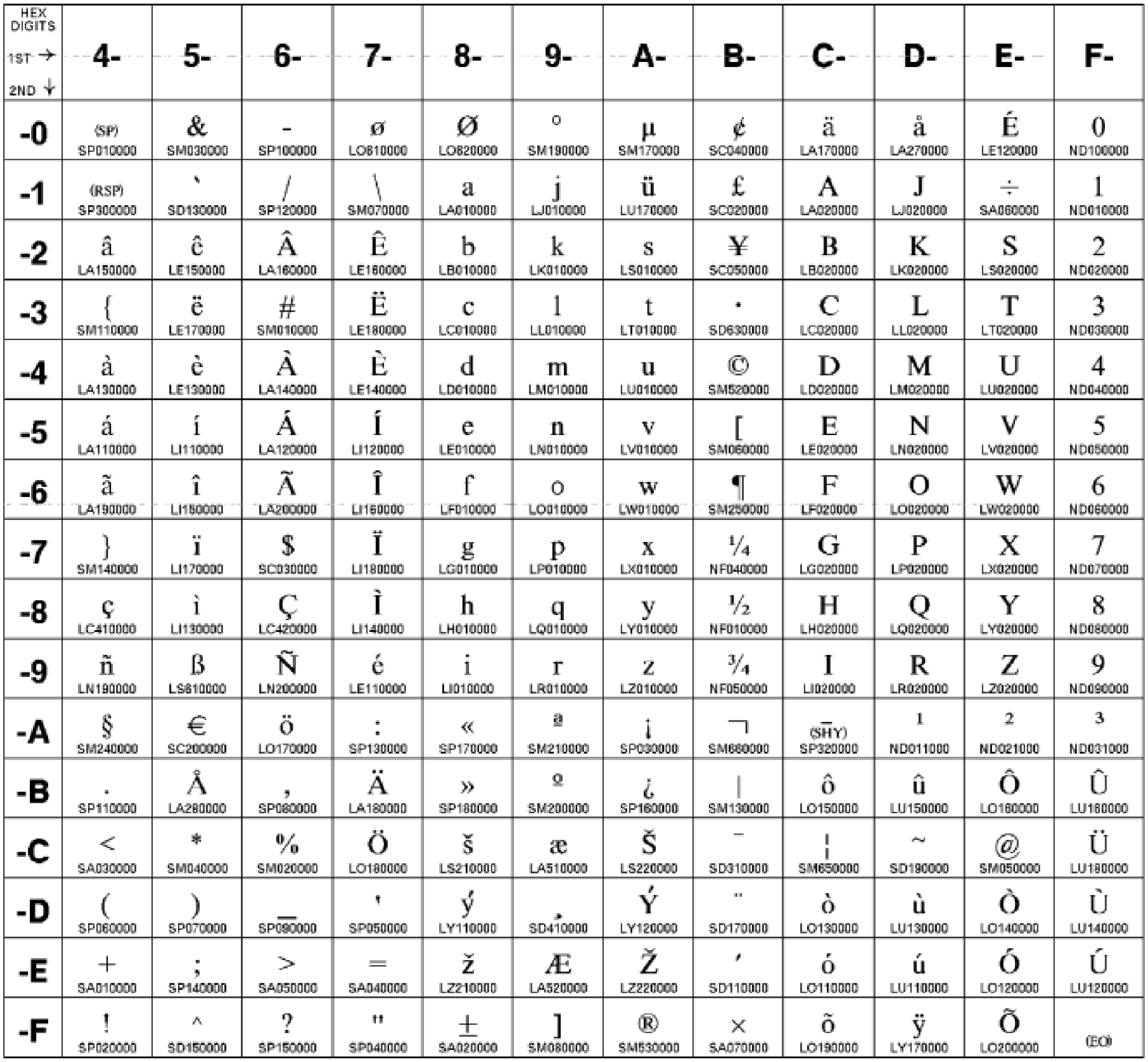
Code Page 01156

[Top of Page on page 1261 | Previous Page on page 1260 | Next Page on page 1262 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1157/1391 Estonia

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	å LA270000	É LE120000	0 ND100000
-1	(RSP) SP300000	` SD130000	/ SP120000	\ SM070000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	# SM010000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	[SM060000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	é LE110000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	§ SM240000	€ SC200000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Ä LA180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ö LO180000	š LS210000	æ LA510000	Š LS220000	¯ SD310000	¦ SM650000	~ SD190000	@ SM050000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ž LZ210000	Æ LA520000	Ž LZ220000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

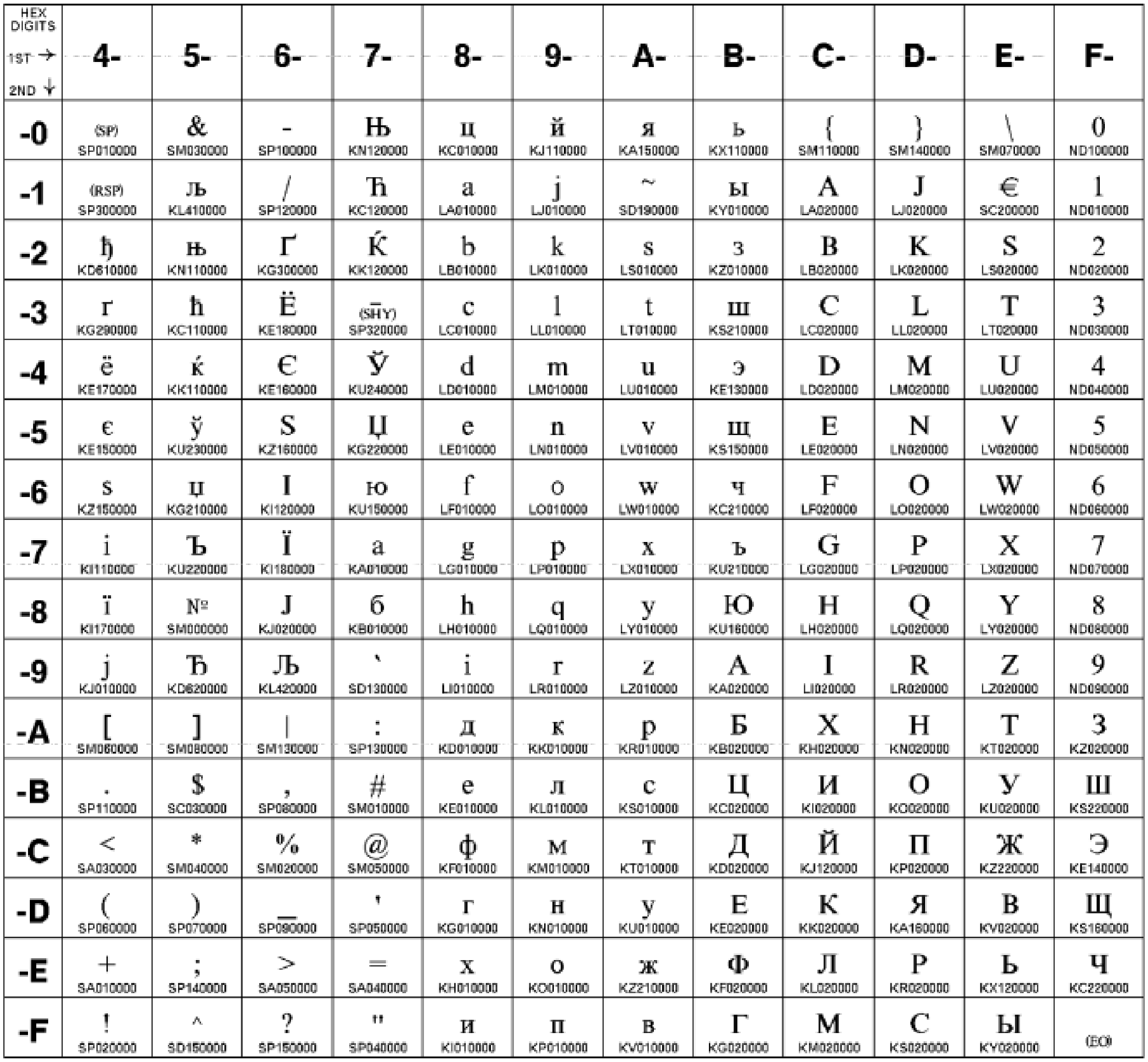
Code Page 01157

[Top of Page on page 1262 | Previous Page on page 1261 | Next Page on page 1263 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1158/1388 Ukraine

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	Њ KN120000	ц KC010000	й KJ110000	я KA150000	ь KX110000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	љ KL410000	/ SP120000	Ћ KC120000	a LA010000	j LJ010000	~ SD190000	ы KY010000	A LA020000	J LJ020000	€ SC200000	1 ND010000
-2	ђ KD610000	њ KN110000	Ґ KG300000	Ќ KK120000	b LB010000	k LK010000	s LS010000	з KZ010000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ґ KG290000	ћ KC110000	Ё KE180000	(SHY) SP320000	c LC010000	l LL010000	t LT010000	ш KS210000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ё KE170000	ќ KK110000	Є KE160000	Ў KU240000	d LD010000	m LM010000	u LU010000	э KE130000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	є KE150000	ў KU230000	Ѕ KZ160000	Џ KG220000	e LE010000	n LN010000	v LV010000	щ KS150000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ѕ KZ150000	џ KG210000	І KI120000	ю KU150000	f LF010000	o LO010000	w LW010000	ч KC210000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	і KI110000	Ъ KU220000	Ї KI180000	а KA010000	g LG010000	p LP010000	x LX010000	ъ KU210000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ї KI170000	№ SM000000	Ј KJ020000	б KB010000	h LH010000	q LQ010000	y LY010000	Ю KU160000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ј KJ010000	Ђ KD620000	Љ KL420000	` SD130000	i LI010000	r LR010000	z LZ010000	А KA020000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	| SM130000	: SP130000	д KD010000	к KK010000	р KR010000	Б KB020000	Х KH020000	Н KN020000	Т KT020000	З KZ020000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	е KE010000	л KL010000	с KS010000	Ц KC020000	И KI020000	О KO020000	У KU020000	Ш KS220000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ф KF010000	м KM010000	т KT010000	Д KD020000	Й KJ120000	П KP020000	Ж KZ220000	Э KE140000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	г KG010000	н KN010000	у KU010000	Е KE020000	К KK020000	Я KA160000	В KV020000	Щ KS160000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	х KH010000	о KO010000	ж KZ210000	Ф KF020000	Л KL020000	Р KR020000	Ь KX120000	Ч KC220000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	и KI010000	п KP010000	в KV010000	Г KG020000	М KM020000	С KS020000	Ы KY020000	(EO)

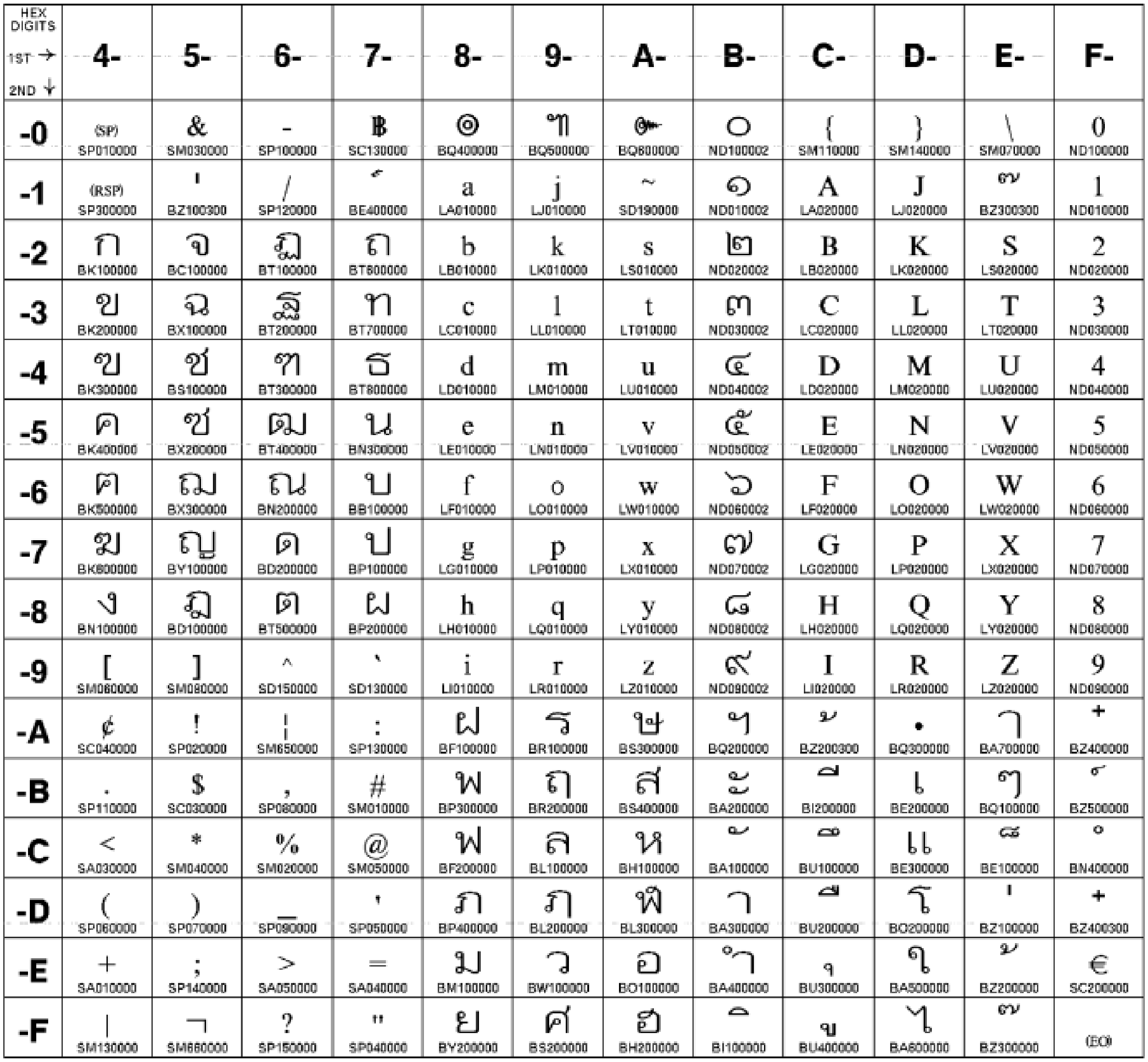
Code Page 01158

[Top of Page on page 1263 | Previous Page on page 1262 | Next Page on page 1264 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1160/1395 Thailand

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	฿ SC130000	๏ BQ400000	๚ BQ500000	๛ BQ600000	๐ ND100002	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	่ BZ100300	/ SP120000	๎ BE400000	a LA010000	j LJ010000	~ SD190000	๑ ND010002	A LA020000	J LJ020000	๊ BZ300300	1 ND010000
-2	ก BK100000	จ BC100000	ฏ BT100000	ถ BT600000	b LB010000	k LK010000	s LS010000	๒ ND020002	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ข BK200000	ฉ BX100000	ฐ BT200000	ท BT700000	c LC010000	l LL010000	t LT010000	๓ ND030002	C LC020000	L LL020000	T LT020000	3 ND030000
-4	ฃ BK300000	ช BS100000	ฑ BT300000	ธ BT800000	d LD010000	m LM010000	u LU010000	๔ ND040002	D LD020000	M LM020000	U LU020000	4 ND040000
-5	ค BK400000	ซ BX200000	ฒ BT400000	น BN300000	e LE010000	n LN010000	v LV010000	๕ ND050002	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ฅ BK500000	ฌ BX300000	ณ BN200000	บ BB100000	f LF010000	o LO010000	w LW010000	๖ ND060002	F LF020000	O LO020000	W LW020000	6 ND060000
-7	ฆ BK600000	ญ BY100000	ด BD200000	ป BP100000	g LG010000	p LP010000	x LX010000	๗ ND070002	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ง BN100000	ฎ BD100000	ต BT500000	ผ BP200000	h LH010000	q LQ010000	y LY010000	๘ ND080002	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	[SM060000	] SM080000	^ SD150000	` SD130000	i LI010000	r LR010000	z LZ010000	๙ ND090002	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	ฝ BF100000	ร BR100000	ษ BS300000	ฯ BQ200000	้ BZ200300	ฺ BQ300000	ๅ BA700000	๋ BZ400000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	พ BP300000	ฤ BR200000	ส BS400000	ะ BA200000	ี BI200000	เ BE200000	ๆ BQ100000	์ BZ500000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ฟ BF200000	ล BL100000	ห BH100000	ั BA100000	ึ BU100000	แ BE300000	็ BE100000	ํ BN400000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ภ BP400000	ฦ BL200000	ฬ BL300000	า BA300000	ื BU200000	โ BO200000	่ BZ100000	๋ BZ400300
-E	+ SA010000	; SP140000	> SA050000	= SA040000	ม BM100000	ว BW100000	อ BO100000	ำ BA400000	ุ BU300000	ใ BA500000	้ BZ200000	€ SC200000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	ย BY200000	ศ BS200000	ฮ BH200000	ิ BI100000	ู BU400000	ไ BA600000	๊ BZ300000	(EO)

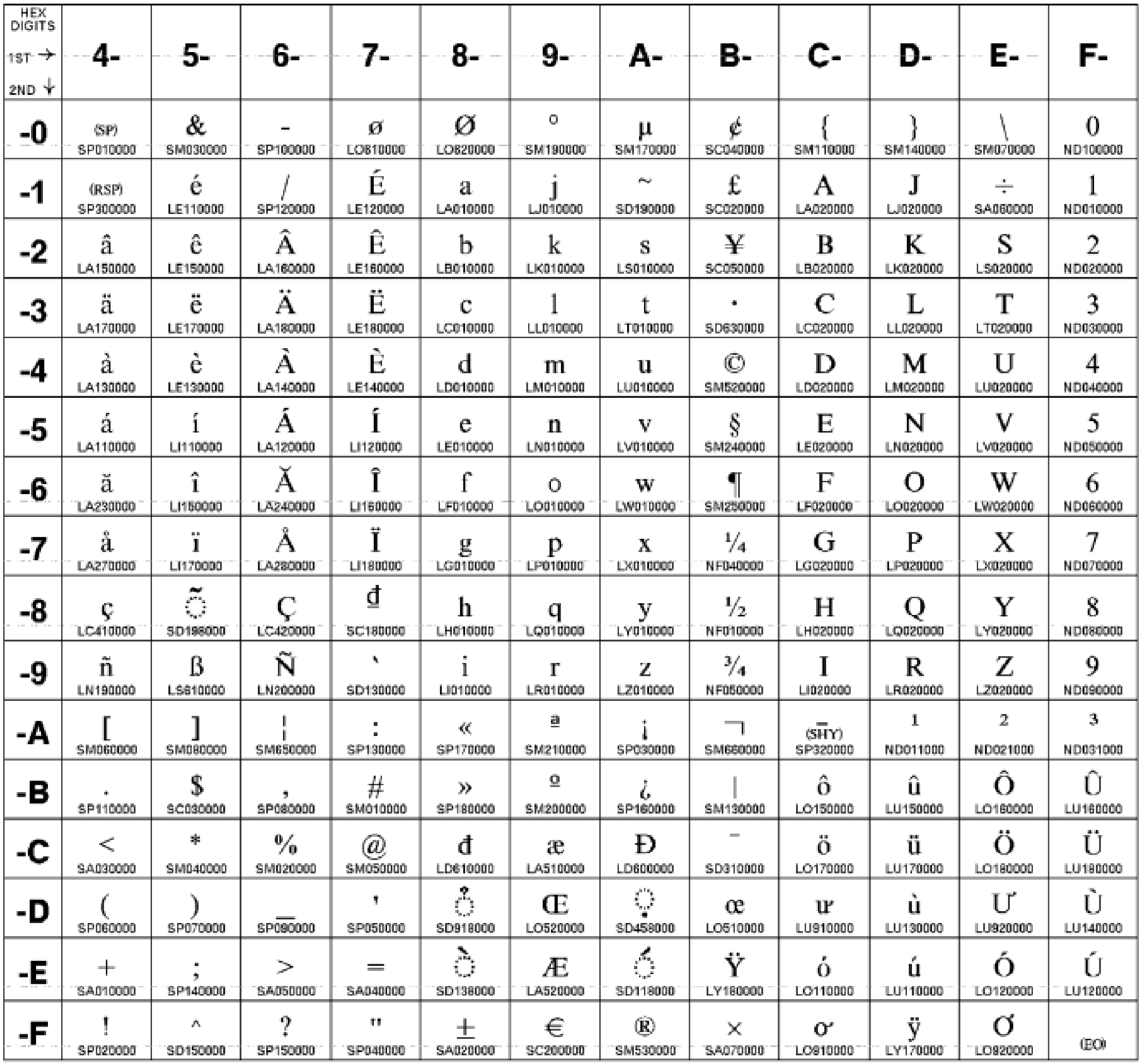
Code Page 01160

[Top of Page on page 1264 | Previous Page on page 1263 | Next Page on page 1265 | Table of Contents on page 1216]

Host Code Page Reference

Host Code Page 1164/1397 Vietnam

The column indicates the first digit and the row indicates the second digit.



HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ă LA230000	î LI150000	Ă LA240000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	◌̃ SD198000	Ç LC420000	₫ SC180000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	đ LD610000	æ LA510000	Đ LD600000	¯ SD310000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	◌̉ SD918000	Œ LO520000	◌̣ SD458000	œ LO510000	ư LU910000	ù LU130000	Ư LU920000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	◌̀ SD138000	Æ LA520000	◌́ SD118000	Ÿ LY180000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	€ SC200000	® SM530000	× SA070000	ơ LO910000	ÿ LY170000	Ơ LO920000	(EO)

Code Page 01164

[Top of Page on page 1265 | Previous Page on page 1264 | Next Page | Table of Contents on page 1216]

Chapter 3. PDF Library

PDF documentation for ZIE for Windows is also available on every online topic and click 📖 icon to download the document.

- Quick Beginnings Guide
- Installation Guide
- Emulator User's Reference Guide
- Administrator's Guide
- Emulator Programming Guide
- Host Access Class Library
- Reference Materials

The published edition is the most current version of the documentation and applies to any subsequent product releases and modifications until otherwise indicated in new editions.

Index