GXS EDI Services

Expedite Base/AIX for RS6000 Programming Guide

Version 4 Release 6

GC34-3280-02

Third Edition (November 2005)

This edition replaces document number GC34-3280-01.

Contents

To the reader

This book gives you the information necessary to program and use Expedite Base/AIX for your company's applications. Expedite Base/AIX is an Information Exchange-based communication program that enables you to transmit data files and messages to and from Information Exchange. Therefore Expedite Base/AIX customers can use the features of Information Exchange through a managed network.

The term *network*, as used in this book, refers to the communications network provided by *AT&T Global Network Services*. In other countries, GXS or another company may provide network services. Similarly, references in this document to your marketing representative refer to the representative at the company who provided your network services.

Who should read this book

This book is written for experienced AIX or UNIX® programmers familiar with Information Exchange who want to write application programs for Expedite Base/AIX. This book is also for users who want to use Expedite Base/AIX to transmit data, files, and messages to and from Information Exchange.

How to use this book

In this book, *baud rate* and *data rate* are synonymous.

The term *compression* as used in this book refers to data compression and decompression with the Comm-Press product, which may not be available in all countries.

In this book, *asynchronous communications through a network gateway*, *network gateway communication*, and *using a network communication gateway* refer to using COMMTYPE(A) on the TRANSMIT command.

Type conventions

Some type conventions are used in this book. Understanding what they mean can help you learn the material covered.

All application program interface (API) commands and record names are shown in small, uppercase letters; for example, SESSIONEND.

In the step-by-step instructions in this book, information that you must type is shown in bold faced type. For example:

> At the command shell, type **iebase**

In Chapter 8, "Using Expedite Base/AIX message commands," the command format examples have the following type conventions:

- Required parameters and values are boldfaced.
- Default values are underlined.
- Parameter values are italicized.

NOTE: When blank is listed as a variable, it refers to a blank space and not the actual typed word.

In general, you do not have to worry about case when typing commands and parameters, and can use upper or lowercase letters. However, there are two exceptions: file names and path names are case sensitive.

The following is an example of the type conventions described above:

send fileid(*file ID*) account(*account*) userid(*user ID*) class(*class*) priority(*blank*|*i*|*p*);

Words that are in the glossary are shown in italics the first time they are used in the body of the book.

How this book is organized

This book contains the following:

- Chapter 1, "Introducing Expedite Base/AIX," introduces Expedite Base/AIX and Information Exchange. It provides an overview of the functions in Expedite Base/AIX.
- Chapter 2, "Installing Expedite Base/AIX," provides hardware and software requirements and installation instructions, and describes the files in Expedite Base/AIX.
- Chapter 3, "Getting a quick start," explains how to run a session with Information Exchange using Expedite Base/AIX sample files. It provides instructions for copying, renaming, modifying, and running these files.
- Chapter 4, "Understanding Expedite Base/AIX," describes the API command syntax and discusses command and response files. It also provides examples of these files and discusses the interaction between Expedite Base/AIX and your application.
- Chapter 5, "Sending and receiving files," describes how to address files and explains Information Exchange's data recovery methods. It also provides information on sending and receiving text and binary files, and it provides examples that illustrate how you can use Expedite Base/AIX.

- Chapter 6, “Sending and receiving EDI data,” explains how Information Exchange's data recovery methods work when you transfer electronic data interchange (IDE) data. It also provides information on sending and receiving EDI-formatted data and provides examples that illustrate how you can use Expedite Base/AIX.
- Chapter 7, “Using Expedite Base/AIX profile commands,” explains how to create profiles and describes the profile commands and profile response records. It also provides information on changing your password and discusses the Extended Security Option (ESO).
- Chapter 8, “Using Expedite Base/AIX message commands,” provides detailed information about the message commands.
- Chapter 9, “Using Expedite Base/AIX message response records,” describes the message response records and their formats.
- Chapter 10, “Using additional features,” describes other features of Expedite Base/AIX, such as audit trails, mailbox queries, acknowledgments, libraries, and archiving.
- Chapter 11, “Communicating with users on different operating systems,” provides information on transferring files to other systems, including older Information Exchange interfaces, ASCII, and Extended Binary-Coded Decimal Interchange Code (EBCDIC).
- Chapter 12, “Displaying screen status,” describes in detail the session status picture and status messages.
- Chapter 14, “Using modem scripts,” provides information about modem commands and scripts.
- Chapter 13, “Using the connectivity log and trace files,” describes how to use the connectivity log and trace files to obtain detailed processing information on Expedite Base/AIX.
- Chapter 15, “Using SNA LU 6.2 communications,”Chapter 15, “Using SNA LU 6.2 communications,” describes installation prerequisites and provides installation and configuration instructions for SNA/LU6.2 communications.
- Chapter 16, “Using TCP/IP communications,” describes how to establish TCP/IP communications. It provides information on using Expedite Base/AIX for TCP/IP dial and leased line communications.
- Appendix A, “Expedite Base/AIX messages and codes,” contains all Expedite Base/AIX return codes with explanations and user actions.
- Appendix B, “Common data header,” provides a description of the CDH fields.
- Appendix C, “Reserved file names and user classes,” describes the reserved file names and user classes that Expedite Base/AIX may create or reference.

- Appendix D, “Information Exchange translate table,” provides Information Exchange ASCII-to-EBCDIC and EBCDIC-to-ASCII translate tables.
- Appendix E, “Using data compression,” describes the use of compression software to compress and decompress data with Expedite products.
- This book also includes a glossary and an index.

Related books

The following books contain information related to the topics covered in this book. For your convenience, these documents can be viewed and downloaded on the library page of the Application Services - EDI Services Web page at: **http://www.gxsolc.com/edi_bes.html**.

- *Expedite Base Programming Commands Quick Reference*, GC34-2328
- *Information Exchange Administration Services Messages and Codes*, GC34-2323
- *Information Exchange Messages and Formats,* GC34-2324
- *Information Exchange Charges Reference*, GX66-0653
- *Information Exchange Administration Services User’s Guide*, GC34-2221
- *Information Exchange Administration Mailbox Command Reference*, GC34-2260

Summary of changes

EDI Services Expedite software, formerly IBM EDI Services Expedite software adds Secure Sockets Layer TCP/IP support.

Overview

GXS Application Hosting, formerly IBM Application Hosting - EDI Services Expedite software for AIX is enhanced to support Secure Sockets Layer (SSL) TCP/IP connectivity to the Information Exchange mailbox component of EDI Services.

EDI Services customers who want to take advantage of the value-added features provided by an Expedite communication client can now connect to Information Exchange over the Internet through their Internet service provider using the new SSL TCP/IP connectivity. Communication between the Expedite clients and the Information Exchange service is performed using Client X.509 Certificate authenticated SSL for enhanced security.

Expedite Base/AIX for RISC System/6000 Version 4 Release 6 can support communication with Information Exchange using the following methods:

- Asychronous communication using a modem that supports Hayes AT command set, V.32, V34, X2. or V90 protocols
- TCP/IP communication using a properly configured TCP/IP connection to the appropriate network
- SNA LU 6.2 or TCP/IP leased-line communication using a multiprotocol adapter and a leased line

Customers can choose the communications protocol that is most appropriate for their needs.

Certificates

Secure communication between Expedite Base/AIX and Information Exchange is also accomplished by using the Internet Public Key Infrastructure (PKI) Client X.509 Certificate that is authenticated SSL for enhanced security.

Using the Information Exchange/File Transfer Protocol (IE/FTP) certificate generation process, you can request a new X.509 v3 certificate to use with the GXS Internet gateways. To request a certificate to access the Internet gateways. use the resources at the IE/FTP Web site, https://pki.services.ibm.com. Please note that you need the following:

- An Information Exchange account.

NOTE: If you do not currently access your Information Exchange mailbox via the Internet but wish to try Internet access, you can follow the self-registration process.

- Two Identification tokens (User number and Challenge token). These will have been sent to you by GXS via a secure postal method or sent to your Information Exchange mailbox if you used the self-registration process.
- An ftps client that conforms to RFC draft-murray-auth-ssl-10.txt.
- A browser, either Netscape 4.7 or Internet Explorer v5.0 or later.

Your self-registration request will be processed and two Identification tokens (User number and Challenge token) will be returned to your mailbox along with the address of the registration Web site. You can then go to the registration Web site and use your Identification tokens to request a new certificate.

Once you have your Identification tokens, follow the instructions for your browser.

GSKit environment

The GSKit environment is the solution that implements SSL using TCP/IP communication. The GSKit contains an application program interface (API) that is called from within the Expedite TCP/IP code when SSL is enabled.

File and command changes

- basein.pro

  Use the SSL command and the associated parameters to enable communication using TCP/IP in a secure environment.

- IDENTIFY

  Use this existing command with these three new parameters:

  - KEYRINGFILE
  - KEYRINGSTASHFILE
  - KEYRINGPASSWORD

  These parameters provide access to the GSKit environment in Expedite Base/AIX. The GSKit provides the ability to enable secure communications using Expedite Base/AIX.

- START

  Use this existing command with these three new parameters:

  - KEYRINGFILE
  - KEYRINGSTASHFILE
  - KEYRINGPASSWORD

  These parameters are only valid when the AUTOSTART AND AUTOEND parameters on the TRANSMIT command are set to **n** in the *basein.pro* file. These parameters provide access to the GSKit environment in Expedite Base/AIX.

New sample profile

Expedite Base/AIX for RS/6000 includes the following new sample profile:

- sslsamp.pro
  This is an SSL TCP/IP sample

Chapter 1

Introducing Expedite Base/AIX

Information Exchange is the electronic mailbox component of GXS, formerly IBM EDI Services, and enables you to send information to and receive information from trading partners. Expedite Base/AIX provides the interface that makes it possible to use Information Exchange from an AIX environment.

Expedite Base/AIX uses Information Exchange to deliver and receive data, such as electronic data interchange (EDI) data. Expedite Base/AIX runs as an application on an AIX system, which uses American National Standards Code for Information Interchange (ASCII). It can communicate with some Extended Binary Coded Decimal Interchange Code (EBCDIC) systems as well as with other ASCII computers through Information Exchange.

To communicate with Expedite Base/AIX and transfer data files to and from Information Exchange, you use application program interface (API) commands. These commands are contained in control files.

Understanding Information Exchange

Information Exchange is an international commerce engine feature of Application Hosting - EDI Services, offered by GXS. Information Exchange is an international storage and retrieval mechanism for electronic mailbox services and computer-to-computer networking and a common point of contact between you, your applications, and your trading partners. You can send and receive information of virtually any size in electronic form, from standard-format EDI transactions to free-format documents. Information Exchange receives transactions and documents from trading partners on a managed network, routes them to the recipients, and stores the data for retrieval.

Use Information Exchange to establish a computer-to-computer communication network between different locations to speed and simplify the delivery of e-mail, EDI envelopes, and other data.

Through a managed network, such as AT&T Global Network Services, Information Exchange can link the geographically scattered locations of a single company or of different companies; for example, a manufacturing company can use Information Exchange to communicate with its suppliers or distributors. Your computer can communicate with one or more Information Exchange addresses through a single Information Exchange session. The network assigns Information Exchange addresses during registration; each Information Exchange address is independent of other Information Exchange addresses.

You can access Information Exchange as shown in Figure 1. The figure shows how your user applications and Expedite Base/AIX—installed on a workstation running an AIX operating system—communicate with the network and Information Exchange through a modem.

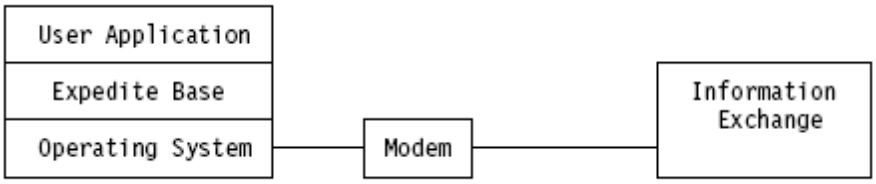


Figure 1. User access to Information Exchange

Using accounts, user IDs, and passwords

In order to connect to Information Exchange, you must have one or two sets of corresponding accounts, user IDs, and passwords defined in the Expedite Base/AIX profile, depending on the communications type you select. The first of these sets is required if you are using asynchronous communications, and is necessary to log on to the network. The individual fields in this set are the network account, user ID, and password. If you are using leased line access, this set is not necessary.

The second of these sets allows you to log on to Information Exchange on the network. This set is required for both asynchronous and leased-line communications. These fields are the Information Exchange account, user ID, and password. Expedite Base/AIX requires these values before you can connect to Information Exchange.

You provide both sets of accounts, user IDs, and passwords to Expedite Base/AIX in the IDENTIFY command. For more information on the IDENTIFY command, see Chapter 7, “Using Expedite Base/AIX profile commands.”

Understanding an Information Exchange session

This overview provides a list of the activities you perform and the activities Expedite Base/AIX performs. This book discusses each of these activities in detail.

You perform the following activities:

- Create a profile command file, basein.pro, or make the necessary changes to an existing one. The profile command file provides information about you and the information Expedite Base/AIX needs to connect to the network. For more detailed information about profiles, see Chapter 7, “Using Expedite Base/AIX profile commands.”

- Specify message commands, such as SEND and RECEIVE, in the message command file, *basein.msg*. The commands in this file tell Expedite Base/AIX what to do during a session. For detailed information on message commands, see Chapter 8, "Using Expedite Base/AIX message commands."
- Run the Expedite Base/AIX program in either of the following ways:
  - Type **iebase** at the operating system command prompt and press Enter
  - Call the program from another application.

When you run the program, Expedite Base/AIX does the following:

- Processes the profile commands in *basein.pro* and writes responses to the profile response file, *baseout.pro*.
- Establishes a connection to the network and logs on to the Service Manager using your network account, user ID, and password in *basein.pro*.
- Logs on to Information Exchange using your Information Exchange account, user ID, and password in *basein.pro* and starts a session.
- Processes the message commands in *basein.msg* and writes responses to the message response file, *baseout.msg*. For detailed information on message response records, see Chapter 9, "Using Expedite Base/AIX message response records."
- Ends the Information Exchange session, logs off the network, and terminates the connection.
- Writes the final return code to *baseout.msg*.

After an Information Exchange session, you should review *baseout.msg* to see that all commands processed correctly. If errors occurred, Expedite Base/AIX writes return codes and error messages to *baseout.msg*. For detailed information, see Appendix A, "Expedite Base/AIX messages and codes."

NOTE: An Information Exchange session begins when you log on to Information Exchange, includes processing the commands, and ends when you log off.

Sending and receiving data

Two sets of commands enable you to send and receive files, EDI data, and electronic mail (e-mail) using Expedite Base/AIX:

- SEND RECEIVE
- SENDEDI RECEIVEEDI

Use the SEND and RECEIVE commands in *basein.msg* for files and e-mail. Use the SENDEDI and RECEIVEEDI commands for EDI data.

You can use Expedite Base/AIX to send text or binary data to a trading partner. Text data can usually be read by a person while binary data can be read by a computer. Executable programs and computer drawings are examples of binary data. To send text or binary data, use the SEND command; to receive this data, use the RECEIVE command.

Use the SENDEDI command to transmit multiple EDI envelopes with different addresses from a single file with a single command. Information in this file can consist of any combination of ANSI X12, Uniform Communications Standard (UCS), Electronic Data Interchange For Administration, Commerce, and Transport (EDIFACT), and United Nations/Trade Data Interchange (UN/TDI) data. SENDEDI resolves the destinations of the different pieces of information from the EDI data, so you do not have to retype existing destination information.

Use the RECEIVEEDI command to receive multiple EDI envelopes containing different types of data with a single command.

For more information on the commands to send and receive data, see Chapter 8, "Using Expedite Base/AIX message commands."

Transferring files

A file is data that you name and store as a unit. The data in a file is usually related or grouped together; for example, you can store customer names and addresses in a file.

The method Expedite Base/AIX uses to send files depends on the data type. The type of data in a file on a workstation is in ASCII format, and can be either text data or binary data. Text data can be read by a person, while binary data is read by a computer. If the data is text, Expedite Base/AIX translates it from ASCII to EBCDIC when it sends the data to Information Exchange. If you indicate that the data is binary, Expedite Base/AIX does not translate the data.

For more information on transferring files, see Chapter 5, "Sending and receiving files."

Transferring EDI data

Electronic Data Interchange (EDI) is the electronic exchange of structured business documents between computer systems using a standard format. EDI uses data layout standards so that the sending and receiving systems can recognize the data format. Once trading partners agree to exchange data formatted to standards, they can exchange documents electronically instead of sending them through the mail.

For more information on transferring EDI data, see Chapter 6, "Sending and receiving EDI data."

Transferring electronic mail

Electronic mail (e-mail) is correspondence in the form of a file that you transmit over a computer network. Different software packages handle e-mail differently. The important thing is that the e-mail file looks the same to the receiver as it did to the sender.

As far as the Expedite family products are concerned, e-mail files are made up of 79-byte "records," padded with blanks if necessary. The 79-byte records are each followed by the character(s) that normally delimit records for the type of platform being used. For example, in Expedite Base/AIX each 79-byte record is delimited by the new line character.

In order to identify the file as being electronic mail, the Expedite family products use the user class FFMSG001. This way, the receiving system knows how to format the e-mail records when the data is received.

To create an e-mail file, use an editor to create the text for the file, making sure each line of text is not longer than 79 bytes and ends with a new line character. To send the file, use the SEND command with the FORMAT(Y) parameter. FORMAT(Y) tells Expedite Base/AIX to do the following:

- Pad each line of text with blanks up to 79 bytes, or split lines that are greater than 79 bytes.
- Add CRLF characters to each line.
- Send the file with a user class of FFMSG001.

When you receive an electronic mail file, use the RECEIVE command with the FORMAT(Y) parameter so that the file is properly received in the Expedite e-mail format for you to view.

NOTE: You can use the CLASS parameter on the SEND command to specify a user class other than FFMSG001. However, the receiving system will not automatically recognize the file as having the Expedite e-mail format.

Identifying Information Exchange error messages

When Information Exchange generates error messages, it places them in your mailbox with a sending account of *SYSTEM* and a user ID of *ERRMSG*. The most common reason for an error message is that Information Exchange is unable to deliver a file. For more information on Information Exchange error messages, see the *Information Exchange Messages and Formats*.

Requesting Information Exchange acknowledgments

Although they are not error messages, Information Exchange acknowledgments also have a sending account of *SYSTEM* and a user ID of *ERRMSG*. An acknowledgment is placed in your mailbox with information about the files that you sent. The three types of acknowledgments you can request using the ACK parameter in the SEND and SENDEDI commands are:

receipt	Information Exchange generates a receipt acknowledgment when a file reaches the receiver's mailbox after a successful Expedite Base/AIX session.
delivery	Information Exchange generates a delivery acknowledgment when a destination user receives a file from the Information Exchange mailbox.
purge	Information Exchange generates a purge acknowledgment when a file is purged from the receiver's mailbox.

To receive these acknowledgments, you must use the RECEIVE command. For more information on acknowledgments, see "Using acknowledgments" on page 268.

Providing security

Security is provided at the network-access level, the application-selection level, and the data-access level. You should understand that network security features operate within a widely used data processing environment. Information Exchange can protect users only if those users safeguard security controls. You must change passwords and profile authorizations at recommended intervals to ensure mailbox security.

Working with libraries

A library is a place to store information for an extended period of time. Unlike files in a user's mailbox, information in a library is not deleted automatically after a certain amount of time. Libraries are made up of library members, which contain the information you want to store. To use libraries in Expedite Base/AIX, do the following:

1. Create the library using Information Exchange Administration Services.
2. Use the GETMEMBER and PUTMEMBER commands to retrieve library members and place them in a mailbox, or place library members in a library.
3. Use the LISTLIBRARIES and LISTMEMBERS commands to identify libraries and members to which you have access.

For information on using acknowledgments with libraries, see "Using acknowledgments with libraries" on page 271. For more information on libraries, refer to "Working with libraries" on page 268 and the *Information Exchange Administration Services User's Guide*.

Connecting to the network

You can select one of four methods to connect to the network. The first method uses SNA LU 6.2 APPC communications, the second two use asynchronous communication, and the fourth uses TCP/IP communications.

Systems Network Architecture (SNA) Logical Unit 6.2 (LU 6.2) communications uses SNA services on AIX on the IBM RS/6000. This type of communication is only available for AIX Versions 5.1 or 5.2 on the IBM RS/6000. See "Using SNA LU 6.2 communications" on page 341 for more information on software requirements for SNA LU 6.2 communications.

Asynchronous communication involves the use of an asynchronous modem to dial the network. You should dial into the network communication gateway by specifying COMMTYPE(A) on the TRANSMIT command. An older protocol, worldwide asynchronous communication, is still supported on the network gateway but is not recommended, since it may increase the communication time.

For more information on communication protocols, see "TRANSMIT command" on page 156.

For more information on using TCP/IP, see Chapter 16, "Using TCP/IP communications."

Understanding Information Exchange Administration Services

Information Exchange Administration Services is part of Information Exchange. Information Exchange Administration Services is accessible as a panel-driven 3270 interface that is used to perform administrative tasks for Information Exchange users. For example, you can use Information Exchange Administration Services to enable users in an account to send mail back and forth to each other, set up distribution lists, and create libraries.

To access Information Exchange Administration Services, you need a full-screen emulator product or a network-attached terminal. Refer to the documentation for your operating system to find an application to use. It is strongly recommended that at least one person in each account has access to Information Exchange Administration Services.

Information Exchange Administration Services for the Web provides Internet Protocol (IP) customers with the ability to access many Information Exchange Administration Services functions and tasks without using 3270 emulation.

Expedite Base/AIX does not provide access to Information Exchange Administration Services, but it does provide the ability to perform some of its functions. The following is a summary of some of the functions Information Exchange Administration Services provides. The functions in italics are functions that Expedite Base/AIX also provides.

- Changing user profiles
- *Creating and updating distribution lists*
- *Creating and updating alias tables*
- Maintaining trading partner lists
- Maintaining payment levels
- Viewing and selecting archive information
- *Retrieving archived messages*
- *Retrieving audit trails*
- *Viewing the contents of a mailbox*
- Viewing session traces
- Creating libraries
- *Listing libraries and library members*
- *Retrieving library members*
- Resetting user passwords
- Resetting user sessions

For more information, see the *Information Exchange Administration Services User's Guide*.

Chapter 2

Installing Expedite Base/AIX

This chapter gives general information about hardware and software required to install and use Expedite Base/AIX, how to set up your terminal to work with Expedite Base/AIX, and the files included with Expedite Base/AIX.

Specific hardware and software requirements and installation instructions for AIX for the RISC System/6000 are included in this chapter.

Understanding what you need to use Expedite Base/AIX

The following sections discuss the Expedite Base/AIX for RS/6000 operating environment and connectivity requirements. The hardware and software requirements are listed below.

Hardware and software requirements

The following hardware is required to use Expedite Base/AIX for the RS/6000 Version 4.6:

- A RISC System/6000 capable of running AIX for the RISC System/6000 Version 5.
- 32 MB RAM
- 15 MB of disk storage.
- A color or monochrome display.
- If you are using asynchronous communication, you need a modem that supports the Hayes AT command set, V.32, V.34, X2, or V.90 protocols.

  NOTE: Only X2, and V.90 protocols support all of the bps rates supported by Expedite Base/AIX.

- If you are using an external modem for asynchronous communication, you need an asynchronous communication adapter or a native serial port.

- If you are using SNA LU 6.2 or TCP/IP leased-line communication, you need a multiprotocol adapter and a leased line.

The following software is required to run Expedite Base/AIX for the RS/6000 Version 4.6:

- AIX for the RISC System/6000 Versions 5.1 or 5.2.

NOTE: Because of the differences in operating system and compiler versions, Expedite Base/AIX does not run on AIX versions prior to Versions 5.1 or 5.2.

- Communication Server for AIX Version 6.0, if you are using SNA LU 6.2 leased-line communication.
- Client X.509 Certificate.

  The certificates are only required if you want SSL communication sessions using TCP/IP. You can obtain certificates from the following Web page: https://pki.services.ibm.com.

Understanding connectivity requirements

Expedite Base/AIX can communicate with Information Exchange in the following ways:

- Using asynchronous communication.
- Using SNA LU 6.2 communication through Communication Server for AIX Versions 5.1 or 5.2.
- Using TCP/IP communication.
- Using Secure Sockets Layer (SSL) TCP/IP communication.

Installing Expedite Base/AIX on the IBM RISC System/6000

For installation, you must have one of these AIX Versions: Version 5.1 or Version 5.2 to use Expedite Base/AIX for the RS/6000. If you plan to use SNA LU 6.2 communications over a leased-line, you must have Communication Server for AIX Version 6.0.

Expedite Base/AIX for the IBM RISC System/6000 is distributed as a downloadable .tar file. Use the steps below to work with the *expbase461.tar* file and the GSKit package that is included with that file. The GSKit provides the environment that Expedite Base/AIX needs to enable SSL (Secure Sockets Layer) communication.

To install Expedite Base/AIX for the IBM RISC System/6000 do the following:

1. Log in as user **root**.
2. Change to the directory where you want this product to be installed. The default directory is: /var/adm/expedite. (Create this directory if it does not currently exist.)

   cd /var/adm/expedite

3. Copy the *expbase461.tar* file into the adm/expedite directory; then type and enter the following command as root:

   tar -xvf expbase461.tar

   The program files are extracted into the current working directory.

4. Copy the executable files *iebase*, *iebaser*, *iebasepr*, and *iebasepo* to the executable path; or set your path statement to include the Expedite Base/AIX for the IBM RISC System/6000 install directory; or plan to invoke Expedite Base/AIX for the IBM RISC System/6000 by using the full path name:

   ```
   cp iebase /usr/bin/iebase
   cp iebaser /usr/bin/iebaser
   cp iebasepr /usr/bin/iebasepr
   cp iebasepo /usr/bin/iebasepo
   ```

   or

   ```
   export PATH=$PATH:/var/adm/expedite:
   ```

   or

   ```
   /var/adm/expedite/iebase
   ```

5. Make sure the port you want to use for Expedite Base/AIX for the IBM RISC System/6000 is available for dial out by using the PDISABLE AIX command:

   ```
   pdisable /dev/tty.
   ```

6. After you untar the Expedite Base/AIX for RISC System/6000 installation file, the GSKit runtime environment installation package file will be located in the /var/adm/expedite/gskinst directory. You need to be root to install the package.

   Change to the /var/adm/expedite/gskinst directory or change to the directory that you installed to and enter the **installp** command that is listed below.

   For example, if you installed Expedite Base/AIX for RISC System/6000 in the /var/admin/expedite directory, type and enter the following commands while you are logged in as root:

   cd /var/adm/expedite/gskinst
   installp -acgqw -d /var/adm/expedite/gaskinst gaskak.rte

7. If the file, *session.fil*, exists from a previous Expedite session, you need to allow that session to complete or reset the session. For more information, see “Restarting a session” on page 104 and “Resetting a session” on page 106.

8. If you modified any Expedite Base/AIX for the IBM RISC System/6000 files with the file type of *.scr* and want to retain the updates, make a copy of those files prior to installing Expedite Base/AIX for the IBM RISC System/6000.

Upgrading from Version 4.5 to 4.6 of Expedite Base/AIX

You must install the GSKit runtime environment to run Expedite Base/AIX Version 4.6 and to enable SSL before you attempt to run a session with Expedite Base/AIX Version 4.6. See the installation instructions under “Installing Expedite Base/AIX on the IBM RISC System/6000” on page 10.

Before you install Expedite Base/AIX for the IBM RISC System/6000, make sure all previous sessions with earlier versions of Expedite Base/AIX have completed successfully, and are not in checkpoint restart mode. Verify that you do not have a *session.fil* file in the directory where you use Expedite Base/AIX. Also, to be sure that all users on the system have no sessions in checkpoint restart, log in as user root and use the AIX *find* command:

```
find / -name session.fil print
```

If the checkpoint restart file *session.fi*l is found, start the current version of Expedite that you have installed and allow the session to complete. If you do not want to complete the session, then refer to “Resetting a session” on page 106 before erasing the session.fil files and proceeding with the upgrade.

Next, determine which version of expEDIte/AIX Base that you have installed (if you have not already done so). To do this, use the following command:

iebase -v

at the AIX command prompt, and see which version displays.

Using support files from a previous release

Expedite Base/AIX for the IBM RISC System/6000 includes enhancements to the display status and modem control files that were not available in previous releases. The function of these support files is essentially the same as in previous releases, but they are easier to modify to meet your needs. If you have modified your modem control or display status files, **cnnct.fil* or *message.fil*, it is recommended that you modify the Expedite Base/AIX for the IBM RISC System/6000 modem connect scripts or display status scripts in the same manner. However, if you want to continue to use the support files from a previous release with Expedite Base/AIX for the IBM RISC System/6000, you can use “compatibility mode.”

Installing for compatibility mode

If you have IBM Expedite Base/AIX Base Version 4.0, 4.3.3, or 4.5 installed, and you want to use the modem control or display status files you have modified, do the following:

1. Log in as user **root**.
2. Change to the directory where you have Expedite Base/AIX installed. The default directory for Versions 4.0 and 4.3.3. is /usr/bin/expedite. The default directory for Version 4.6 is /var/adm/expedite.

3. If you want to save a copy of the installed version, make a directory and copy the files from the installation directory to the backup directory. Make sure to copy the executable *iebase* file from the executable path where it is installed to the backup directory, if it is different than the current directory. For example:

```
mkdir oldexp
cp * oldexp
cp /usr/bin/iebase oldexp
```

4. If you want to use the modem control files but not the display status file, then remove the old display status file:

```
rm message.fil
```

5. If you want to use the old display status file, but not the old modem control files, then remove the old modem control files:

```
rm *cnnct.fil
```

6. Copy the executable files *iebase*, *iebaser*, *iebasepr*, and *iebasepo* to the same directory where you had the previous executable installed. For example:

```
cp iebase /usr/bin/iebase
cp iebaser /usr/bin/iebaser
cp iebasepr /usr/bin/iebasepr
cp iebasepo /usr/bin/iebasepo
```

7. Make sure the port you want to use for Expedite Base/AIX for the IBM RISC System/6000 is available for dial out by using the PDISABLE AIX command:

```
pdisable /dev/tty.
```

Switching out of compatibility mode

If you installed Expedite Base/AIX for the IBM RISC System/6000 to operate in compatibility mode, and then decide to use the enhanced display script or modem scripts included with Expedite Base/AIX for the IBM RISC System/6000, you must erase or rename the old display or modem control files. To switch from old modem control files to Expedite Base/AIX for the IBM RISC System/6000 modem scripts, you must also remove the modem type from your profile by specifying blank as the value of the MODEMTYPE parameter on the DIAL command in basein.pro.

Switching to the Expedite Base/AIX for the IBM RISC System/6000 display status script

If you want to use the new display status script, change to the directory where Expedite Base/AIX for the IBM RISC System/6000 is installed, and rename the old display control file as shown:

```
mv message.fil message.old
```

Make sure the new display status script, *display.scr*, is in the directory where Expedite Base/AIX for the IBM RISC System/6000 is installed.

Switching to Expedite Base/AIX for the IBM RISC System/6000 modem connect scripts

If you want to use the new modem control scripts:

1. Erase or rename the old modem control files. For example, if you used modem type *h*:

```
mv hcnnct.fil hcnnct.old
mv hdcnnct.fil hdcnnct.old
```

NOTE: The name of the modem control file to rename depends on the modem type you specified in basein.pro on the DIAL command. The names will be *xcnnct.fil* and *xdcnnct.fil* where *x* is the modem type you specified. If you did not specify a modem type, then the default is **h**.

2. Make sure the new modem scripts are in the directory where you have installed Expedite Base/AIX for the IBM RISC System/6000.

3. Specify blank as the value of modem type in your profile. Use the MODEMTYPE parameter on the DIAL command in basein.pro. For example:

```
DIAL modemtype( );
```

Setting up your terminal to work with Expedite Base/AIX

Expedite Base/AIX makes use of standard AIX terminal configurations to display the picture and status information. If either the picture or status information displays incorrectly, the cause can be the incorrect definition of your terminal under AIX.

In AIX 5.1 or 5.2, terminal information is defined in files in the directory */usr/share/lib/terminfo*. See the documentation for your operating system to find where the terminal information files are stored on your system. A number of files are provided which cover common terminal types. The names of these files are usually the terminal manufacturer's name with an extension of *.ti*.

One possible problem with the picture is that the box characters may display incorrectly. These characters are defined, in octal, in the line *box1=*. The characters are in the following order: top left corner, box horizontal character, top right corner, box side, bottom left corner, and bottom right corner. (Other box characters are not used by Expedite Base/AIX.)

If you have a system administrator who is responsible for modifying the system files on your system, this information is important for the system administrator. Do the following if you want to change any terminal characteristics:

- Copy your existing terminal information file into a new file.
- Edit the new file to change the incorrect configuration. (Change the terminal name for your configuration.)
- Compile the terminal information file, by using *tic* with your file name. Refer to your AIX system documentation for more information on using *tic*.
- In order to activate your configuration, type **set** TERM=<*your new terminal name*>.
- Type **export** TERM to make this information active.
- To activate TERM every time you log on, the last two commands should be added to your *.profile*.

Understanding Expedite Base/AIX files

The Expedite Base/AIX program is on a single diskette. The program diskette contains both files that are stored in the root Expedite program directory, and others that are stored in the \samples directory.

Files in the Expedite Base install directory

The following files are included in this directory:

- cnnct.scr

  This file contains the dial commands for a modem that supports the AT command set.

  NOTE: The script in this file is designed for use in Australia, Canada, Latin America, and the United States, and may require modification before it can be used in other countries.

- discnnct.scr

  This file contains the disconnect commands for a modem that supports the AT command set.

  NOTE: The script in this file is designed for use in Australia, Canada, Latin America, and the United States, and may require modification before it can be used in other countries.

- display.scr

  This file contains the status messages displayed by Expedite Base/AIX.

- errormsg.cmp

  This file contains the error messages written when Comm-Press programs encounter an error.

- errormsg.fil

  This file contains the error messages issued when Expedite Base/AIX encounters an error.

- errortxt.fil

  This file contains explanations and the appropriate user responses for the error messages in errormsg.fil.

- hostname.fil

  This file contains names and/or address and port numbers of the Information Exchange TCP/IP Relay and is needed for TCP/IP communication.

- IBM3270.XLT

  This is a translate table that Expedite Base/AIX can use to perform ASCII EBCDIC translation in the same manner as the IBM eNetwork Personal Communications program.

- iebase

This is the executable module for Expedite Base/AIX that calls *iebasepr*, *inmsgp*, *iebaser*, *iebasepo*, and *outmsgp* (the inmsgp and outmsgp files are present only if you have the supported compression software). Type and enter **iebase** to run this module from the command line.

NOTE: This is the program that should be called from any user program.

- iebasepo

  This is the file that checks the message response file, baseout.msg, for compressed files, and verifies that the *outmsgp* file (required for data decompression) exists.

- iebasepr

  This is the file that checks the COMPRESS() parameter, and verifies that the *inmsgp* file (required for data compression) exists.

- iebaser

  This is the renamed iebase file, which provides all the functions that iebase provided in the previous version of Expedite Base/AIX.

- lu62shrd.o

  This file is the SNA dynamic library required for LU 6.2 communications.

- NOXLATE.XLT

  This is a translate table you can use to send and receive data without ASCII and EBCDIC translation.

- readme

  This file contains information made available after the publication of this book. You should read the contents of this file before using Expedite Base/AIX.

Files in the samples directory

The following files are included in this directory:

- auditfmt

  This is a sample program you can use to format level 1, 2, or 3 audit records from Information Exchange. The format will look similar to the format of baseout.msg. The sample C-language code for this program is included in the auditfmt.c file.

- auditfmt.c

  This is a sample C-language program you can use to format level 1, level 2, or level 3 audit records from Information Exchange.

- basemsg.in

  This is a sample message command file you can use to run a sample Information Exchange session. For more information, see Chapter 3, “Getting a quick start.”

- basepro.in

  This is a sample profile command file you can use to run a sample Information Exchange session. For more information, see Chapter 3, “Getting a quick start.”

- ecnnct.scr

  This file contains a sample connect script for use in Austria, Belgium, Finland, France, Germany, Israel, Italy, Netherlands, and Sweden.

- edcnnct.scr

  This file contains a sample disconnect script for use in Austria, Belgium, Finland, France, Germany, Israel, Italy, Netherlands, and Sweden.

- expconn.sna

  This is the sample configuration for AIX Communication Server to use with SNA LU 6.2 communications.

- lu62samp.pro

  This is a sample profile command for LU 6.2 leased line communication with Information Exchange.

- makexlt.c

  This is a sample C-language program that can be used to build a translate table.

- psc.c

  This is a sample C-language program that you can use to encrypt and decrypt passwords.

- QUALTBL.TBL

  This is a sample EDI qualifier table. It specifies translation tables or centralized alias tables used to resolve EDI destinations.

- report.c

  This is a sample C-language program you can use to parse the contents of the message response file and create a summary of the session. It is not required for Expedite Base/AIX execution.

- samptest.fil

  This file contains sample data for a sample Information Exchange session. For more information, see Chapter 3, “Getting a quick start.”

- scnnct.scr

  This file contains a sample connect script for use in Switzerland and Slovenia.

- sdcnnct.scr

  This file contains a sample disconnect script for use in Switzerland and Slovenia.

- sslsamp.pro
  This is an SSL TCP/IP sample

- sysmsfmt

  This is a sample program you can use to format system error messages (or acknowledgments) from Information Exchange. The format will look similar to the format of baseout.msg. The sample C-language code for this program is included in the sysmsfmt.c file.

- sysmsfmt.c

  This is a sample C-language program you can use to format system error messages (or acknowledgments) from Information Exchange.

- tcpsamp.pro

  This is a sample profile command for TCP/IP communication with Information Exchange.

- tucnnct.scr

  This file contains a sample connect script for use by traveling users.

- ucnnct.scr

  This file contains a sample connect script for use in Denmark, Norway, South Africa, Spain, and the United Kingdom.

- udcnnct.scr

  This file contains a sample disconnect script for use in Denmark, Norway, South Africa, Spain, and the United Kingdom.

NOTE: None of the files in the samples subdirectory are needed for Expedite Base/AIX execution.

Chapter 3

Getting a quick start

One way to understand how Expedite Base/AIX works is to run a session with Information Exchange. The sample files included with Expedite Base/AIX can help you get a quick start. Sample files for LU 6.2 leased-line communication (*lu62samp.pro*) and for TCP/IP communication (*tcpsamp.pro*) are also provided. This chapter discusses how to use these files to run a sample session using asynchronous communication.

To work with the sample files, copy them from the samples subdirectory under the directory where you installed Expedite Base/AIX to your own directory.

1. Change to your home directory.
2. Enter the following command to create the subdirectory for the sample files:

   ```
   mkdir expedite
   ```

3. Change to the directory that you just created for the sample code by entering the following command:

   ```
   cd expedite
   ```

4. Copy the files from the installation directory using the CP command. Use the path where you installed Expedite Base/AIX, for example:

   ```
   cp /usr/bin/expedite/samples/base*.in
   cp /usr/bin/expedite/samples/samptest.fil
   ```

   The sample files are as follows:

   - basepro.in (profile command file)
   - basemsg.in (message command file)
   - samptest.fil (sample data file)

There are also sample profile files for TCP/IP communication, tcpsamp.pro, and SNA communication, *lu62samp.pro*.

NOTE: The sample file names are different from the Expedite Base/AIX file names so that you will not overwrite any files.

Rename the sample profile command file and message command file to the file names Expedite Base/AIX uses. If you have already created a profile command file, *basein.pro*, or message command file, *basein.msg*, rename those files so you do not lose any data when you run the sample files.

To rename the sample files, enter the following commands:

```
mv basepro.in basein.pro
mv basemsg.in basein.msg
```

Modifying the sample profile command file

To run a session with Information Exchange, Expedite Base/AIX needs information about the user and the method of communication. Expedite Base/AIX gets this information from the profile command file, *basein.pro*.

NOTE: When you order Expedite Base/AIX, you should receive your network and Information Exchange accounts, user IDs, passwords, and a telephone number to use to dial the network for asynchronous communication. If you do not have this information, contact your marketing representative.

In the sample *basein.pro* file, the terms in uppercase are the commands and parameters; the terms in lowercase are the values that pertain to you, except for file names. Using a text editor, replace the following values in the sample file with your accounts, user IDs, passwords, and telephone information.

inacct	Your network account
inuser01	Your network user ID
inpass	Your network password
ieacct	Your Information Exchange account
ieuser01	Your Information Exchange user ID
iepass	Your Information Exchange password
phone number	Your local telephone number, toll-free number, or fee number for the network
device	The fully qualified path to the device driver (for example, /dev/tty0)

Expedite Base/AIX uses this information to identify you to the network and Information Exchange and to establish a session with Information Exchange.

NOTE: If your telephone is on a PBX that requires an escape sequence to connect to an outside line, use the ESCAPE parameter on the DIAL command in basein.pro. For example, specify ESCAPE(9,) if you must dial a 9 before placing an outside telephone call.

The remaining parameters in the sample basein.pro file contain default values. These parameters and values illustrate complete commands. You do not have to specify a parameter if you choose to use its default value.

NOTE: The default value for the COMMTYPE parameter in the TRANSMIT command is **a**, which indicates the communication type is asynchronous through a network gateway. If you are outside the United States and cannot connect to a network gateway, specify COMMTYPE(W) on the TRANSMIT command for worldwide asynchronous communication. For information about other methods of communication, see “TRANSMIT command” on page 156.

NOTE: If you installed Expedite Base/AIX in a directory other than the default, you must enter the following line as the last line in the basein.pro file:

```
SESSION IEPATH(path);
```

where *path* is the directory where you installed Expedite Base/AIX.

Modifying the sample message command file

During an Information Exchange session, Expedite Base/AIX processes the commands you enter in the message command file, *basein.msg*. The sample message command file contains the following information:

```
SEND FILEID(samptest.fil) ACCOUNT(ieacct) USERID(ieuser01)
CLASS(test1);

RECEIVE FILEID(samptest.new) ACCOUNT(ieacct) USERID(ieuser01)
CLASS(test2);
```

Using a text editor, replace the values *ieacct* and *ieuser01* with your Information Exchange account and user ID.

The SEND command in this file tells Expedite Base/AIX to send the file, *samptest.fil*, to your mailbox with a user class of *test1*.

The RECEIVE command tells Expedite Base/AIX to receive any files that your account and user ID sent with a user class of *test1*. This is, of course, the file that you just sent to your own mailbox. When Expedite Base/AIX receives this file, it creates a new file, s*amptest.new*, on your workstation and places the received data in this file.

NOTE: If you wish to use SENDEDI and RECEIVEEDI commands in *basein.msg*, see Chapter 6, “Sending and receiving EDI data.”

Running a sample session

To run an Information Exchange session, follow these steps:

1. Attach your modem to the workstation and turn it on.
2. At the AIX command shell, type the command **iebase** and press Enter. You must be in the directory where you copied the sample files.

Viewing the picture and status display

When you run Expedite Base/AIX, the program displays a picture that represents a workstation and the network. It also displays status boxes that provide the following information about the session's progress.

Message in status box:	Session activity:
Dialing the network	Expedite Base/AIX is trying to connect with the network.
Successful xxxx connection	Expedite Base/AIX established a successful connection, where xxxx is the data rate of the connection for asynchronous communication. The status box shows the Help Desk telephone number and your terminal ID.
Successful network logon	Expedite Base/AIX logged on to the network Service Manager.
Started session with Information Exchange	Expedite Base/AIX has started its session with Information Exchange. A second status box displays within the first box. This box displays information about the files you send to and receive from Information Exchange. In this sample session, the status box shows one file sent and one file received.
Ending Information Exchange session	Expedite Base/AIX has completed the session with Information Exchange.
Disconnecting	Expedite Base/AIX is disconnecting from the network.

NOTE: If you do not get these results or if you do not get a 00000 return code, copy, edit, and run the sample files again or see Appendix A, “Expedite Base/AIX messages and codes.”

Verifying the session results

After the session, verify the following new files in the directory where you installed Expedite Base/AIX:

baseout.pro

This file contains the processing results of the profile command file, *basein.pro*.

iebase.pro

This file maintains profile information for Expedite Base/AIX internal processing.

baseout.msg

This file contains the processing results of the message command file, *basein.msg*.

samptest.new

This file contains the data Expedite Base/AIX received from Information Exchange.

baseout.pro

Use your editor to view baseout.pro. This file shows the profile commands from basein.pro along with their associated return codes. The following is a subset of the information you should see in your baseout.pro file.

```
IDENTIFY INACCOUNT(inacct)INUSERID(inuser01)INPASSWORD(inpass)
IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass);
RETURN(00000);
DIAL PHONE(phone number);
RETURN(00000);
PROFILERC(00000);
```

Each command should have processed with a 00000 return code. The PROFILERC record also processed with a 00000 return code, which indicates all profile commands completed successfully.

iebase.pro

When Expedite Base/AIX processes basein.pro, it creates the file *iebase.pro*. Because this is an internal file Expedite Base/AIX uses, you do not need to review it. Note, however, that Expedite Base/AIX creates this file and updates it when you make changes to *basein.pro*.

baseout.msg

Use your editor to view *baseout.msg*. This file shows the message commands from *basein.msg* along with their associated return codes.

```
AUTOSTART SESSIONKEY(xxxxxxxx);
STARTED LASTSESS(0)RESPCODE(00000)SESSIONKEY(377DL87) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(samptest.fil) ACCOUNT(ieacct) USERID(ieuser01)
CLASS(TEST1);
SENT UNIQUEID(xxxxxxxx) LENGTH(xx);
RETURN(00000);
RECEIVE FILEID(samptest.new) ACCOUNT(ieacct) USERID(ieuser01)
CLASS(TEST1);
RECEIVED ACCOUNT(ieacct) USERID(ieuser01) CLASS(TEST1) CHARGE(1)
LENGTH(xxx)
FILEID(samptest.new) MSGDATE(xxxxxx) MSGDATELONG(xxxxxxxx)
MSGTIME(xxxxxx)
MSGSEQO(xxxxxx) SESSIONKEY(xxxxxxxx) DELIMITED(E) SYSNAME(xxxxxxxx)
SYSLEVEL(xxxx) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(samptest.fil) SENDERLOC(/home/dir)
FILEDATE(xxxxxx) FILEDATELONG(xxxxxxxx) FILETIME(xxxxxx) RECFM(????)
RECLEN(00000) RECDLM(C) UNIQUEID(xxxxxxxx) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

In this file, the *x* values are determined during transmission. The following is an explanation of the commands and responses in *baseout.msg*.

AUTOSTART

Indicates that Expedite Base/AIX started the Information Exchange session automatically. This record has a 00000 return code.

STARTED

Provides a response to the AUTOSTART command. This record indicates the SESSIONKEY, Information Exchange version and release, the return code for the last session, and the response code for the AUTOSTART command.

SEND

Is an echo of the SEND command in *basein.msg*.

SENT

Provides a response to the SEND command. This record indicates that Expedite Base/AIX sent the file and assigned a unique ID. It also shows the length of the file and has a 00000 return code.

RECEIVE

Is an echo of the RECEIVE command in *basein.msg*.

RECEIVED

Provides a response to the RECEIVE command. This record indicates that Expedite Base/AIX received the file.

AUTOEND

Indicates that Expedite Base/AIX ended the Information Exchange session automatically. This record has a 00000 return code.

SESSIONEND

Indicates the overall processing results of the message commands. This record has a 00000 return code.

samptest.new

Contains the data Expedite Base/AIX received from Information Exchange. This record is identical to the file *samptest.fil* you sent to Information Exchange.

Chapter 4

Understanding Expedite Base/AIX

Expedite Base/AIX uses the following files to perform its functions:

- Profile command file (basein.pro)
- Profile response file (baseout.pro)
- Message command file (basein.msg)
- Message response file (baseout.msg)
- Temporary response file (tempout.msg)

You use Expedite Base/AIX to place requests into command files, run the Expedite Base/AIX program, and then examine the appropriate response files to see if the requests completed successfully. For a detailed description of all profile commands and their parameters, see Chapter 7, “Using Expedite Base/AIX profile commands.” For a detailed description of all message commands and their parameters, see Chapter 8, “Using Expedite Base/AIX message commands.”

This chapter describes the Expedite Base/AIX command syntax and discusses the command and response files. It also discusses user and design considerations for your application interface.

The following figure shows how Expedite Base/AIX uses the command and response files to move data between Information Exchange and your application.

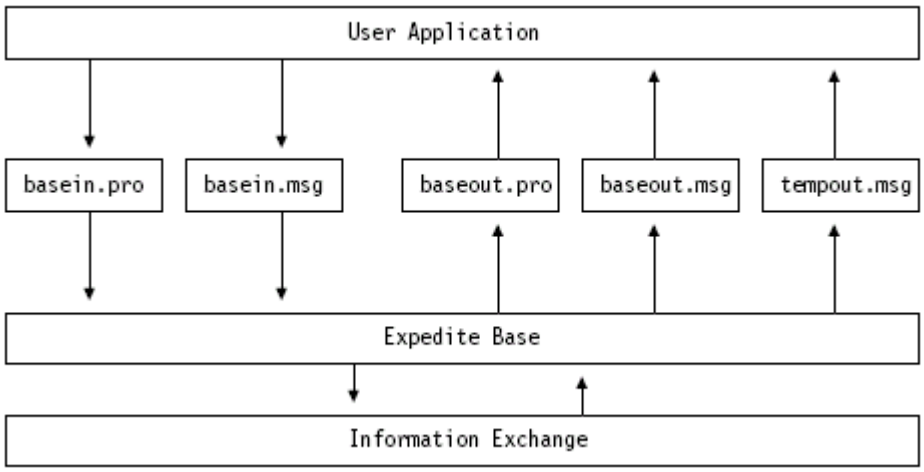


Figure 2. Application design diagram

Understanding command syntax

This is an example of Expedite Base/AIX command syntax.

```
#Comment or description
command parameter(value) parameter(value) ...parameter(value);
```

Defines or delimits comments. Type any information after **#**. The program ignores the characters after **#** until it encounters a new line. You do not have to end comments with **#**. Expedite Base/AIX considers the end of the line in which the comment exists to be the end of that comment. If you want a comment to continue, begin a new line with **#**.

If you place a command *after* the comment on the same line, Expedite Base/AIX ignores that command. To ensure that your command is processed, you should either place the command on a new line or place the command *before* the comment.

Some file names and description statements contain **#**. If you include a **#** in a parameter value, Expedite Base/AIX knows from the context that the **#** is part of a command and does not ignore the parameter value or the characters that follow.

The following example shows how to place a command and a comment on the same line. In this case, Expedite Base/AIX ignores all the information after #. The program begins processing again at the ACCOUNT parameter because this parameter is on a new line that does not begin with **#**.

```
SEND FILEID(sample.fil) #this is a new file
account(acct2) userid(user2) class(class1);
```

command Identifies the Expedite Base/AIX command.

parameter	Identifies a parameter associated with the command preceding it.
value	Defines the value associated with the parameter.
...	Indicates in the example above that you can specify as many parameters as necessary. This is not part of the syntax.
;	Ends the command.

In general, you can type Expedite Base/AIX commands and parameters in uppercase or lowercase letters. However, path and file names are case sensitive.

The commands and parameters can span several lines in a command file. However, the following limitations apply:

- You must enter the entire command name (for example, IDENTIFY) on a single input line.
- You must enter the entire parameter name (for example, IEACCOUNT) on a single input line.
- A left parenthesis must immediately follow each parameter. Do not use spaces between parameter names and values. For example, type **ieaccount(acct)** rather than **ieaccount (acct).**
- You must end each command with a semicolon.

Understanding the profile command file

The profile command file, *basein.pro*, is a free-format command file that controls communication between Expedite Base/AIX and the network. You must not change the name of *basein.pro*; it is the primary Expedite Base/AIX file. Place *basein.pro* in the current directory or in a directory you specify with the -P command line option. For a detailed description of all profile commands and their parameters, see Chapter 7, “Using Expedite Base/AIX profile commands.”

For most of the parameters in basein.pro, you can use the default values. However, there are some parameters for which you must provide values. The following list describes these required parameters:

- INACCOUNT, INUSERID, and INPASSWORD parameters on the IDENTIFY command

  Use these parameters to log on to the network if you are a dial user. Enter your network account, user ID, and password. The format is alphanumeric.

  For information on specifying Information Exchange passwords, see the *Information Exchange Messages and Formats*.

- IEACCOUNT, IEUSERID, and IEPASSWORD parameters on the IDENTIFY command

  Use these parameters to start a session with Information Exchange. Enter your Information Exchange account, user ID, and password. The format is alphanumeric.

  For information on specifying Information Exchange passwords, see the *Information Exchange Messages and Formats*.

- KEYRINGFILE, KEYRINGSTASHFILE, KEYRINGPASSWORD parameters on the IDENTIFY command.

Use these parameters when you start a session with Information Exchange and are required to supply the GSKit code with the data needed for secure communication.

- PHONEn parameter on the DIAL command

  Use this parameter for dial access to the network. Enter at least one phone number. If you have more than one number, enter the numbers in the order you want Expedite Base/AIX to dial them.

- DEVICEx parameter on the DIAL command

  Use this parameter to indicate the communications device driver on your system through which the modem connects to Information Exchange. The value should be the fully qualified path to the device driver. You can specify a total of five device drivers. Expedite Base/AIX will cycle through the list of device drivers and use the first one that responds.

This list contains some of the other profile parameters you may want to include in basein.pro. These are not required parameters.

- NINPASSWORD parameter on the IDENTIFY command

  This parameter is used to change your network password. After you change the password, copy the value in NINPASSWORD to the INPASSWORD parameter and remove the NINPASSWORD parameter from *basein.pro*. Expedite Base/AIX stores this password in an encrypted format in an internal file.

- NIEPASSWORD parameter on the IDENTIFY command

  Use this parameter to change your Information Exchange password. After you change the password, copy the value in NIEPASSWORD to the IEPASSWORD parameter and remove the NIEPASSWORD parameter from *basein.pro*. Expedite Base/AIX stores this password in an encrypted format in an internal file.

- ESCAPE parameter on the DIAL command

  Use this parameter to obtain an outside line when you have an auto-dial modem and you are using dial access. Expedite Base/AIX appends the parameter to all telephone numbers so the modem can dial the correct sequence to obtain an outside line.

- BAUDRATEn and DIALCOUNTn parameters on the DIAL command

  Use these parameters with the PHONEn parameter when you have more than one phone number. Enter at least one local number and one backup number. You can use up to five different sets of phone numbers, modem speeds (data rates), and dial counts.

- DBAUDRATEx parameter on the DIAL command

  Use this parameter with the DEVICEx parameter. Make sure you specify DBAUDRATEx if your modem supports a different data rate than 2400 bps.

- COMMTYPE parameter on the TRANSMIT command

  Use this parameter to specify your communication type: asynchronous communication, worldwide asynchronous communication, SNA LU 6.2 communication, or TCP/IP communications. The default value is **a** for asynchronous communication.

- RECONNECT parameter on the TRANSMIT command

  Use this parameter to specify how many times Expedite Base/AIX should automatically redial the network if it loses contact. The default value is **5**.

- COMMITDATA parameter on the TRANSMIT command

  Use this parameter to specify the amount (bytes) of data you want Expedite Base/AIX to send Information Exchange between checkpoints (also known as commits). The default value is **`37000`**. You should only use values lower than 37000 if poor telephone conditions are causing frequent disconnection with Information Exchange. Otherwise, low values in this parameter cause frequent checkpoints with Information Exchange and slow data transfer. If you are using SNA communications over a leased line, you can specify 141000 to improve performance.

- MSGSIZE

  Use this parameter to specify the size of the segment for sending data. Your trading partner can take checkpoints only for the message size you specify with this parameter. The default value is **`37000`**. If you are using SNA LU 6.2 communications over a leased line, you can specify 47000 for MSGSIZE and 141000 for COMMITDATA to improve performance when sending data.

  NOTE: MSGSIZE must be less than or equal to COMMITDATA.

- CYCLE and WAIT parameters on the DIAL command

  Use the CYCLE parameter to specify how many times Expedite Base/AIX cycles through the telephone number list if it cannot establish a connection in the first cycle. This is for dial access users.

  Use the WAIT parameter to specify how long Expedite Base/AIX pauses between cycles. For example, if the values are *`4`* for CYCLE (first cycle plus four additional cycles for a total of five) and *`0030`* for WAIT, Expedite Base/AIX attempts to connect to the network one cycle every 30 minutes for a total of five cycles. If Expedite Base/AIX cannot establish a connection, it issues an unsuccessful return code.

- MODEM parameter on the TRACE command

  Use this parameter to produce a trace to determine why Expedite Base/AIX cannot establish a connection with Information Exchange using dial access. Expedite Base/AIX places the trace information in *iebase.trc*. This file shows command processing information from the modem control file and shows responses from the modem.

  Other parameters of the TRACE command are CNNCT, IOFILE, LINK, MODEM, PROTOCOL, LOGIC, and DISPLAY. Expedite Base/AIX stores the trace information for LINK in *s1dcl.trc* and stores the trace information for the other parameters in *iebase.trc*.

  NOTE: Although it is not necessary to set these traces on permanently, you should consider giving users access to all trace parameters to help them with problem determination. If users do not have access to these traces, it severely limits the ability of the Help Desk to determine the cause of a problem.

- OVERWRITE parameter on the SESSION command

  Use this parameter to tell Expedite Base/AIX whether or not to overwrite existing files during the receive process. If this parameter value is *n* and an existing file and a received file have the same file name, Expedite Base/AIX does not overwrite the existing file. Instead, it appends the received file to the end of the existing file.

Expedite Base/AIX places the processing results of the profile commands in the profile response file, *baseout.pro*.

NOTE: If you are going to switch between different communication types (asynchronous dial, TCP/IP, SNA, SSL), use separate profiles, *basein.pro*, for each type of communication. Make sure you erase *iebase.pro* before you switch profiles.

Reviewing examples of basein.pro

The following examples illustrate possible profile command files. The examples address asynchronous communication, TCP/IP communication, trace information, and delayed transmission. In each example, the terms in uppercase are the commands and parameters; the terms in lowercase are the values that pertain to the user. An explanation of the commands, parameters, and values follows each example.

Example 1

This is a simple profile for asynchronous communication that contains the IDENTIFY and DIAL commands.

```
IDENTIFY INACCOUNT(inacct) INUSERID(inuser01) INPASSWORD(inpass)
IEACCOUNT(ieacct) IEUSERID(ieuser01)IEPASSWORD(iepass);

DIAL PHONE1(555-1234) DEVICEA(/dev/tty0) DBAUDRATEA(9600);
```

The IDENTIFY command provides the information that Expedite Base/AIX needs to log on to the network and Information Exchange. This information includes your network and Information Exchange accounts, user IDs, and passwords.

The DIAL command specifies the telephone number and the modem that Expedite Base/AIX uses to connect to the network.

NOTE: The COMMTYPE parameter on the TRANSMIT command indicates the type of communication you are using. Because the default is asynchronous communication, you do not need to specify the TRANSMIT command in this profile. However, you could specify it as follows:

```
TRANSMIT COMMTYPE(A);
```

Example 2

This is a more complex profile for asynchronous communication.

```
IDENTIFY INACCOUNT(inacct) INUSERID(inuser01) INPASSWORD(inpass)
IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass)
NIEPASSWORD(newpass);

DIAL PHONE1(555-1234) DIALCOUNT1(3) BAUDRATE1(9600)
PHONE2(1-800-555-4321) DIALCOUNT2(1) BAUDRATE2(2400)
PHONE3(1-800-555-1111) DIALCOUNT3(0) BAUDRATE3(2400)
DEVICEA(/dev/tty0) DBAUDRATEA(9600) CYCLE(2) WAIT(0030) ESCAPE(9,);
```

In addition to providing account, user ID, and password information, the IDENTIFY command also provides new password information. The NIEPASSWORD parameter tells Expedite Base/AIX to change the Information Exchange password. When Expedite Base/AIX starts a session, it changes the password specified in the IEPASSWORD parameter to the one specified in the NIEPASSWORD parameter. After the password is changed, you must change the password in the IEPASSWORD parameter to the new password and remove the NIEPASSWORD parameter.

The DIAL command specifies the telephone numbers and other telephone information Expedite Base/AIX uses to connect to the network. The DIALCOUNT1 parameter indicates that Expedite Base/AIX can dial the telephone number in PHONE1 up to three times if it cannot make a successful connection. The BAUDRATE1 parameter sets the communication at a data rate of 9600 bps.

If Expedite Base/AIX dials the first telephone number three times without a successful connection, it dials the telephone number in PHONE2. It dials this number only once at a data rate of 2400 bps. The PHONE3 parameter specifies a third telephone number, but Expedite Base/AIX cannot dial it because the DIALCOUNT3 is 0.

At this point, if Expedite Base/AIX does not establish a successful connection, it uses the CYCLE and WAIT parameters to attempt to connect to the network again. If you specified a value greater than 0 for CYCLE and WAIT, Expedite Base/AIX attempts to connect after the time specified in the WAIT parameter and continues making attempts for the number of times specified in the CYCLE parameter.

NOTE: A CYCLE value of 2 results in a total of three cycles: the original and two additional cycles.

The ESCAPE parameter indicates that the telephone system requires Expedite Base/AIX to dial 9 before dialing an external telephone number. The single comma in this parameter value tells the modem to wait one second after dialing 9 before it dials the external number.

Example 3

This is a profile that requests a larger number of characters between commits and a larger maximum message size to improve performance for SNA LU 6.2 communications on the IBM RISC System/6000.

```
IDENTIFY IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass);

TRANSMIT COMMTYPE(s) COMMITDATA(141000) MSGSIZE(47000);
```

The IDENTIFY command provides the information Expedite Base/AIX needs to establish a session with Information Exchange.

The TRANSMIT command indicates to Expedite Base/AIX that SNA LU 6.2 communications should be used. To improve performance over the leased line, the TRANSMIT command specifies that Expedite Base/AIX should allow 141,000 characters between commits, and use a message size of 47,000.

NOTE: The SNACOMM command must be specified if you want to use anything other than the default value.

Example 4

This is a profile for TCP/IP communication.

```
IDENTIFY IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass);
TRANSMIT COMMTYPE(T);
```

The IDENTIFY command provides the information Expedite Base/AIX needs to log on to Information Exchange. This information includes your Information Exchange account, user ID, and password.

The value **T** in the COMMTYPE parameter on the TRANSMIT command indicates TCP/IP communication with the network.

Example 5

This is a profile for TCP/IP communication where you want to enable the SSL (Secure Sockets Layer) command.

```
IDENTIFY IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass)
KEYRIGNGFILE(filename)KEYRINGPASSWORD(password);
TRANSMIT COMMTYPE(T)
SSL ENABLED (Y);
```

OR

```
IDENTIFY IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass)
KEYRIGNGFILE(filename) KEYRINGSTASHFILE(filename)
TRANSMIT COMMTYPE(T)
SSL ENABLED (Y);
```

Use the KEYRINGFILE, KEYRINGSTASH, and KEYRINGPASSWORD parameters when you specify the valueT in the COMMTYPE parameter on the TRANSMIT command to indicate TCP/IP communication with the network.

The KEYRINGFILE requires a password that you can supply either in a stash file or in the command as a parameter.

You can supply only one KEYRINGPASSWORD at a time. If the KEYRINGSTASHFILE parameter is present and you specify the filename that contains the KEYRINGPASSWORD, you cannot use the KEYRINGPASSWORD parameter. The same is true in reverse.

Understanding the profile response file

When Expedite Base/AIX processes *basein.pro*, it echoes the profile commands, along with their associated return codes, to the profile response file, *baseout.pro*. The RETURN records in this file contain the return codes for each profile command. The PROFILERC record contains the return code for the processing results of the entire basein.pro file. For more information on profile response records, see “PROFILERC record” on page 161.

You can examine the RETURN record to see if a particular profile command processed correctly. However, if the PROFILERC record does not have a zero return code, Expedite Base/AIX did not save the changes requested in *basein.pro*. When this happens, correct any incorrect commands and reissue all of the commands in *basein.pro*.

Reviewing an example of baseout.pro

The following is an example of a profile response file. An explanation of the response records follows the example.

```
IDENTIFY INACCOUNT(inacct) INUSERID(inuser01) INPASSWORD(inpass)
  IEACCOUNT(ieacct) IEUSERID(ieuser01) IEPASSWORD(iepass);
RETURN(00000); DIAL PHONE1(555-1234)
DIALCOUNT1(3)
  PHONE2(555-4321) DIALCOUNT2(2)
  PHONE3(555-5555) DIALCOUNT3(0)
  DEVICEA(/dev/tty0) DBAUDRATEA(9600) CYCLE(2) WAIT(0030) ESCAPE(9,);
RETURN(00000);
PROFILERC(00000);
```

The 00000 return codes in the RETURN records indicate that the commands completed successfully. The 00000 return code in the PROFILERC record indicates that all the profile commands completed successfully.

Understanding the message command file

You must enter commands for Expedite Base/AIX in the message command file (basein.msg). You can enter commands to:

- Start a session with Information Exchange
- Define a distribution list
- Define alias names and alias tables
- Send e-mail
- Send an unformatted text or binary file
- Send an EDI-formatted file
- Receive e-mail
- Receive an unformatted text or binary file
- Receive an EDI-formatted file
- Request that audit records be placed in your mailbox
- Delete a specific file from a user's mailbox. Work with libraries
- End a session

Expedite Base/AIX places the processing results of these commands in the message response file, *baseout.msg*.

Reviewing examples of basein.msg

The following examples show message command files. In each example, the terms in uppercase are the commands and parameters; the terms in lowercase are the values that pertain to the user and the files. An explanation of the commands, parameters, and values follows each example.

Example 1

This example includes the commands for sending and receiving a file.

```
SEND FILEID(sample.snd) ACCOUNT(acct) USERID(user1) CLASS(test);
RECEIVE FILEID(sample.rcv) ACCOUNT(acct) USERID(user1) CLASS(test);
```

The message command file shows that you want to send one file and receive one file. The file you are sending is *sample.snd*. You are sending it to account `acct` and user ID `user1`. The user class is `test`.

The file you are receiving was sent from account `acct` and user ID `user1` with a user class of `test`. You want to receive the file into *sample.rcv*.

Example 2

This example includes the commands for sending a file and receiving system error messages.

```
SEND FILEID(sample.snd) ACCOUNT(acct) USERID(user2);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

The message command file shows that you want to send one file and receive any system error messages. The file you are sending is *sample.snd*. You want to send it to account `acct` and user ID `user1`, but you accidentally specify the user ID as `user2`.

The user ID `user2` is not a valid user ID, but the syntax of the SEND command is correct, so Expedite Base/AIX processes the command without error. However, when Information Exchange receives the file, it detects the error and sends a system error message to your mailbox.

Because you are receiving system error messages in the file *errors.fil*, you get the message that tells you Information Exchange could not deliver the file because `user2` is not a valid user ID. If you had not been receiving error messages, you would assume Information Exchange sent the file to your trading partner. This is why it is important that you always attempt to receive system error messages when you send files. You may not receive the system error message in the same session, but in a later session when you request system messages.

NOTE: To verify that a destination account ID and user ID exist on the Information Exchange system, use the VERIFY parameter on the SEND command. For more information, see “SEND command” on page 216.

Understanding the message response file

When Expedite Base/AIX processes *basein.msg*, it echoes the message commands, along with response records and their associated return codes, to the message response file, *baseout.msg*. The RETURN records in this file contain the return codes for each message command.

Your application interface should read and process the response file. If a session restart is necessary and commands have not completed, your application interface should make any necessary decisions or changes based on information in *baseout.msg* and the information in the temporary response file, *tempout.msg*. Refer to “Processing the message response file” on page 38 for more information about processing the message response file.

Reviewing examples of baseout.msg

The following examples show message response files. An explanation of the response records follows each example.

Example 1

This example includes the response file for a SEND and RECEIVE command.

```
AUTOSTART SESSIONKEY(377DLL87);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(377DLL87)IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(sample.snd) ACCOUNT(ACCT) USERID(USER1) CLASS(TEST);
SENT UNIQUEID(54232365) LENGTH(5334);
RETURN(00000);

RECEIVE FILEID(sample.rcv) ACCOUNT(ACCT) USERID(USER1) CLASS(TEST);
RECEIVED ACCOUNT(ACCT) USERID(USER1) CLASS(TEST) CHARGE(1)LENGTH(5334)
FILEID(sample.rcv) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(100232)
MSGSEQO(234523) SESSIONKEY(377DLL87) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED) SENDERFILE(samptest.fil)
SENDERLOC(/home/dir) FILEDATE(980601) FILEDATELONG(19980601)
FILETIME(120023) RECFM(????) RECLEN(00000) RECDLM(C)
UNIQUEID(89337949) SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates the session started successfully.

Before writing the SENT and RECEIVED records, Expedite Base/AIX echoes the SEND and RECEIVE commands from *basein.msg*.

The SENT record indicates that Expedite Base/AIX sent the file. It also provides information about the unique ID that Expedite Base/AIX assigned the file and the length of the file. The 00000 return code in the RETURN record indicates the command completed successfully.

The RECEIVED record indicates that Expedite Base/AIX received the file and provides information about the file. The 00000 return code in the RETURN record indicates the command completed successfully.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN record indicates the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Example 2

This example shows the response file for a SEND and RECEIVE command. The data that is received is a system error message.

```
AUTOSTART SESSIONKEY(DJJL9200);
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(DJJL9200) IEVERSION(04)
IERELEASE(05);
RETURN(00000);

SEND FILEID(sample.snd) ACCOUNT(ACCT) USERID(USER2);
SENT UNIQUEID(97459230) LENGTH(4898);
RETURN(00000);

RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RECEIVED ACCOUNT(*SYSTEM*) USERID(*ERRMSG*) CHARGE(6) LENGTH(4898)
FILEID(errors.fil) MSGDATE(980603) MSGDATELONG(19980603)
 MSGTIME(073614)
MSGSEQO(001954) SESSIONKEY(DJJL9200) DELIMITED(N) SYSNAME(IBMIE)
SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED) RECFM(????)
 RECLEN(00000)
RECDLM(N) UNIQUEID(12682030) SYSTYPE(12) SYSVER(0);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates the session started successfully.

The SENT record indicates that Expedite Base/AIX sent the file. It also provides information about the unique ID that Expedite Base/AIX assigned the file and the length of the file. The 00000 return code in the RETURN record indicates the command completed successfully.

The RECEIVED record indicates that Expedite Base/AIX received a system error message in the file *errors.fil* and provides information about the file. The 00000 return code in the RETURN record indicates the command completed successfully.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN record indicates the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Even though SESSIONEND indicates all commands completed successfully, the message from Information Exchange in the *errors.fil* file tells you that user ID `user2` is not a valid user ID. The message indicates that Information Exchange could not deliver the file.

Understanding the common data header

Most Information Exchange interfaces can use a *common data header* (CDH) to send and receive detailed information about the contents of a file. The CDH typically contains the following information:

- Type of file (text or binary)
- Record delimiters used in the file; for example, carriage-return and line-feed (CRLF) characters or record lengths
- Whether or not the data is EDI formatted
- Original name of the file on the sending system
- Free-format text description of the file
- Other information describing the original file

When Expedite Base/AIX receives a file, it can use the information in the CDH to format the file. Expedite Base/AIX places the CDH information for received data in the response file in the RECEIVED record or the AVAILABLE record. For more information, see “RECEIVED record” on page 249 or “AVAILABLE record” on page 235.

You do not have to do anything to send or receive the CDH because Expedite Base/AIX does it automatically. For more detailed information on the use and format of the CDH, see Appendix B, “Common data header.”

Processing the message response file

The following sections contain information to help you ensure that Expedite Base/AIX processed the message command file and message commands completely and successfully. Information is also provided to help you understand the SENT and RECEIVED records in the message response file

Checking the session return code

To ensure Expedite Base/AIX finished processing the message command file, check the return code in the SESSIONEND record. This is also the return code of the Expedite Base/AIX program.

If the return code indicates that Expedite Base/AIX did not finish processing the message command file (for example, the return code is not 00000 or 28xxx), correct the error and reinvoke Expedite Base/AIX. The SESSIONEND record often includes an error description. You can find detailed descriptions of errors in Appendix A, “Expedite Base/AIX messages and codes.”

Checking the command RETURN records

Once Expedite Base/AIX finishes processing the message command file, you should examine the RETURN records in the message response file to see if commands completed successfully. If an error occurred, the message response file will also have error description records.

Checking the SENT and RECEIVED records

Expedite Base/AIX produces SENT records for every file sent to Information Exchange and RECEIVED records for every file received from Information Exchange. These records provide detailed information about files you sent and received.

A single RECEIVE or RECEIVEEDI command can often produce multiple RECEIVED records, indicating that multiple files were received with a single command. Also, a single SENDEDI command can produce several SENT records, indicating that several EDI envelopes were sent with a single command.

NOTE: SENT and RECEIVED records can be present even if the RETURN record is not present. If the RETURN record is not present, the command is not complete. Whenever possible, correct the error that prevented Expedite Base/AIX from completing the command and reinvoke Expedite Base/AIX. For more information on response file records, see “Working with profile response records” on page 163 and Chapter 9, “Using Expedite Base/AIX message response records.”

Using the temporary response file

Expedite Base/AIX echoes commands and response records for commands processed since the last Information Exchange checkpoint to the temporary response file (tempout.msg). For more information on Information Exchange checkpoints, see Chapter 5, “Sending and receiving files,” and Chapter 6, “Sending and receiving EDI data.”

If an Information Exchange session ends in error, processing information may be in tempout.msg as well as baseout.msg. Looking at tempout.msg can often help you to determine the cause of an error (for example, a syntax error). The tempout.msg file contains the RETURN record for most errors. It also can contain the command that caused the error.

The following example shows a command with a syntax error and the resulting tempout.msg file.

SYNTAX ERROR: In this example, the parameter userid was incorrectly typed **`USER&!`**.

```
send fileid(test.fil) account(acct) user&!(user01) class(test);
```

tempout.msg:

```
AUTOSTART SESSIONKEY(366899D6);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(366899D6)
 IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(test.fil) ACCOUNT(acct) USER&!(user01);
RETURN(15030) ERRDESC(Invalid parameter found.);
```

Designing your interface

Before you design an application interface to Expedite Base/AIX, you need to know who is going to use the interface and what activities the users expect the interface to perform. You also need to decide how the interface will run Expedite Base/AIX—attended or unattended.

Understanding the users

To help you determine who your users are and what they need, ask yourself these questions:

- Are the users sophisticated AIX users capable of reacting to system problems?
- Do the users expect a simplistic approach, such as sending and receiving files with the same file names?
- Do the users want to set up the send and receive requests, or do they want the interface to create them?

When you answer these questions, you can then decide if your interface should run Expedite Base/AIX attended or unattended, or run it using a combination of both.

Attended operation: With attended operation, the users can interact with Expedite Base/AIX if an Information Exchange session does not complete successfully. Users should be able to restart and reset sessions, correct errors in the profile command file and message command file, and cancel sessions.

Unattended operation: With unattended operation, the interface runs Expedite Base/AIX. It completes Information Exchange sessions with little or no assistance from the users. Therefore, the interface must be capable of correcting error conditions.

Understanding how your interface interacts with Expedite Base/AIX

To design an interface to Expedite Base/AIX, you must understand the interaction between your interface and Expedite Base/AIX during an Information Exchange session. The following information briefly describes the activities your application interface performs and the activities Expedite Base/AIX performs.

- Your application interface performs the following activities:
  - Builds the profile command file
  - Builds the message command file
  - Calls the iebase program
- Expedite Base/AIX performs these activities:
  - Processes the profile command file and writes responses to the profile response file
  - Establishes a connection to the network and logs on to the Service Manager
  - Logs on to Information Exchange
  - Processes the message command file and writes responses to the message response file

  - Ends the Information Exchange session, logs off the network, and terminates the connection
  - Writes the final return code to the message response file

- When the Information Exchange session ends, your application interface performs the following activities:
  - Checks the final return code in the message response file
  - Processes the responses in the message response file
  - Provides a method for you to correct errors
  - Decides whether a session restart or reset is necessary when a session ends in error

Reviewing an example of an application interface

This is an example of an application interface that interacts with Expedite Base/AIX. It may give you an idea of some things to consider for your interface.

Company A is an insurance agency that sells policies and processes claims. At the end of each day, the company uses an application program to send all transactions to the home office. The program provides the following menu options:

1. Process policies
2. Process claims
3. Update profile information
4. Send to home office

When a user selects “Process policies” or “Process claims,” a panel appears for the user to enter information. The program stores this information in a database.

When the user selects “Update profile information,” a panel appears for the user to enter profile information that Expedite Base/AIX requires. On that panel, the user can update the Expedite Base/AIX profile command file, *basein.pro*, without using a text editor.

When the user selects “Send to home office,” the application program performs the following activities:

- Extracts all of the policy information from the database and builds the file policy.fil
- Extracts all of the claim information from the database and builds the file claims.fil
- Reads the Expedite Base/AIX profile information from the database and builds the file basein.pro
- Builds the message command file, *basein.msg*, to send *policy.fil* and *claims.fil* to the home office
- Calls the iebase program to dial the network so Expedite Base/AIX can send the files to the home office Information Exchange mailbox
- Reads the message response file, *baseout.msg*, to analyze the results of the Information Exchange session
- Stores the results of the previous session in a database

- Creates a panel to display information about the previous session
- Stores information about the previous session so the user has a record of the information sent to the home office

Using this application program, any user can send files to the home office without having detailed knowledge of how Expedite Base/AIX works.

Other considerations for your application

The following are two features that you might like to consider incorporating in your application:

1. You may find it useful to include a feature that allows users to receive and install program updates and enhancements through Information Exchange. For example, in every session include a request for a specific file; that file can be a package that contains the program updates and installation procedures. If no files are received, you incur no charge. If you do receive a file, and the return code from Expedite Base/AIX is 00000, your application can unpack the file and use the install program to install the changes.

   This installation would be transparent to the user, and would keep all users at a consistent program level. This could be useful for both your application and for Expedite Base/AIX.

2. To enhance problem determination, you may find it useful to record problems that occur during a transmission, or during processing by your application. You could send this report to a central support center during the next Information Exchange session.

   A trace file, *iebase.trc* or *s1dcl.trc*, is sometimes necessary for problem determination for Expedite Base/AIX. You may want to include in your interface a feature that, at the user's request, will send the trace file to your support center or to the Help Desk through Information Exchange. Alternatively, if a session cannot be established, you may want to copy the trace file to diskette.

   ATTENTION: When you send a trace file through Information Exchange, you must rename that trace file. Otherwise it may be overwritten by the trace file for the current session.

For more information

For detailed information on sending and receiving files, see Chapter 5, “Sending and receiving files.” For detailed information on sending and receiving EDI data, see Chapter 6, “Sending and receiving EDI data.” The remaining chapters provide information on the structure of Expedite Base/AIX commands and response records and describe other features of Expedite Base/AIX.

Chapter 5

Sending and receiving files

You can use Expedite Base/AIX to send and receive text and binary files. This chapter explains how you work with files and discusses the use of the common data header (CDH). It also describes the procedures for sending and receiving text and binary files and explains how to use the Information Exchange translate table. In addition, this chapter provides user scenarios to help you learn more about sending and receiving files.

For information on sending and receiving EDI data, see Chapter 6, “Sending and receiving EDI data.”

For information on sending and receiving compressed data, see Appendix E, “Using data compression.”

Addressing files

The address you specify when you send files to Information Exchange serves the same purpose as the address on a letter. For a letter to be delivered, it must contain the proper address information and so must the files you send to Information Exchange. When you send files, specify the Information Exchange mailbox address in the SEND command. For more information, see “SEND command” on page 216.

The following sections discuss the various methods you can use to address files.

Using accounts and user IDs

You can send files using accounts and user IDs. The Information Exchange mailbox address has two parts, the account and the user ID. The account and user ID can be from 1 to 8 characters in length.

NOTES:

1. The account and user ID combination is unique to each Information Exchange system.
2. If you are sending intersystem messages, you use a third part, the system ID.

Using centralized Information Exchange alias tables

You can also send files using an *alias table*. An alias table is a list of alternate names that you can use to send files to other users. Alias tables are permanent tables that reside within Information Exchange. You can make them available to all Information Exchange users (global alias table), members of a particular account (organization alias table), or a single user (private alias table).

You can create and maintain alias tables in two ways:

1. Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*).

2. Using the DEFINEALIAS command (see "DEFINEALIAS command" on page 178).

Using distribution lists

A *distribution list* is another way you can send files. You can send the same file to more than one person at a time by making a list of users and sending the file to the list. There are two types of distribution lists, permanent and temporary.

Permanent distribution lists are permanently stored in Information Exchange. The advantage of using these lists is that many people using different types of computers can use them.

Temporary distribution lists last only for the duration of your Information Exchange session. When your Information Exchange session ends, the system deletes your temporary lists.

To create permanent or temporary lists, use the LIST command. You can also create a permanent list by using Information Exchange Administration Services. For more information, see "LIST command" on page 188.

ATTENTION: If you use the LIST command with session-level recovery, do not use the same distribution list name on more than one LIST command in the same session.

Sending and receiving e-mail

Electronic mail (*e-mail) is* correspondence in the form of a file that you transmit over a computer network. Different software packages handle e-mail differently. The important thing is that the e-mail file looks the same to the receiver as it did to the sender.

As far as the Expedite family products are concerned, e-mail files are made up of 79-byte "records," padded with blanks if necessary. The 79-byte records are each followed by the characters that normally delimit records for the type of platform being used. For example, in Expedite Base/AIX each 79-byte record is delimited by the newline characters.

In order to identify the file as being electronic mail, the Expedite family products use the user class ffmsg001. This way, the receiving system knows how to format the e-mail records when the data is received.

To create an e-mail file, you use an editor to create the text for the file, making sure each line of text is not longer than 79 bytes and ends with newline characters. To send the file, use the SEND command with the FORMAT(Y) parameter. FORMAT(Y) tells Expedite Base/AIX to do the following:

- Pad each line of text with blanks up to 79 bytes, or split lines that are greater than 79 bytes.
- Add CRLF characters to each line.
- Send the file with a user class of ffmsg001.

When you receive an electronic mail file, use the RECEIVE command with the FORMAT(Y) parameter so that the file is properly received in the Expedite e-mail format for you to view.

NOTE: You can use the CLASS parameter on the SEND command to specify a user class other than ffmsg001. However, the receiving system will not automatically recognize the file as having the Expedite e-mail format.

Understanding ASCII text and binary files

There are two general types of files on the workstation: ASCII text and binary. ASCII text files contain text that a person can read. They usually contain only characters that you can type from a computer keyboard, such as A through Z, 0 through 9, and special characters. Binary files contain data that are intended to be read by a computer. Executable computer programs are a common type of binary file. You cannot enter the characters that are in binary files from a computer keyboard.

Sending and receiving text files

Readable text files consist of ASCII characters; readable text on most mainframe computers consists of EBCDIC characters. When Expedite Base/AIX sends a file, it does not know the type of system that will receive that file. Because the Information Exchange application resides on a host system, Expedite Base/AIX translates all ASCII (text) files to EBCDIC and marks the files as EBCDIC in their CDHs when it sends them.

When Expedite Base/AIX receives a file, it checks the CDH to see if the file is EBCDIC or binary. If the file is EBCDIC, it is translated to ASCII. If the file type is unknown because there is not a CDH, Expedite Base/AIX assumes the file is EBCDIC and translates it to ASCII.

Sending and receiving binary files

Binary files do not contain readable characters. Binary data is in a format that can be read and used by a computer. For example, an executable program consists of binary data.

When sending binary data to Information Exchange, it is usually best to avoid translating the binary data from ASCII to EBCDIC, since any changes to the binary data may render it unusable. Expedite Base/AIX allows you to specify that the file is in a binary format and will not translate the data from ASCII to EBCDIC when sending it to Information Exchange. Use the DATATYPE parameter of the SEND command to indicate that the file is binary and that translation should not be done.

When Expedite Base/AIX receives a file, it checks the CDH to see if the file is binary. If so, it does not translate the data from EBCDIC to ASCII when receiving it.

Understanding the translate table

You can use the Information Exchange translate table or an alternative for ASCII EBCDIC translation. The name of the Information Exchange translate table is iestdtbl. You cannot alter this table. It is contained within Expedite Base/AIX; it is not included on your program diskette. For a description of the Information Exchange translate table, see Appendix D, “Information Exchange translate table.”

The Expedite Base/AIX program provides two alternate translate tables. The first is a table matching the IBM eNetwork Personal Communications 4.2 program. It is called IBM3270.XLT. The second is a table that provides no translation at all. This table is called NOXLATE.XLT. When Expedite Base/AIX receives a file that does not have a CDH, it assumes the EBCDIC to ASCII translation is necessary. If your trading partner is using a product that does not support the CDH and is sending you a binary file, use the NOXLATE.XLT translate table when you receive the file. Expedite Base/AIX will use the translate table but will not alter the binary file.

To change translate tables, use the TRANSLATE parameter in the TRANSMIT, SEND, SENDEDI, RECEIVE, or RECEIVEEDI commands.

NOTE: When you send a file to another workstation, the alternate translate table you use to send the file must be available on the receiving workstation. If the alternate translate table is not available, the data will be translated incorrectly and will not be usable.

Reviewing an example of the TRANSLATE parameter

The following example illustrates how to specify an alternate translate table using the TRANSLATE parameter in the SEND command.

Company A uses a workstation to send files to a trading partner. The trading partner uses a mainframe to receive the files and downloads them to a workstation using a 3270 emulation program. Because Company A knows how its trading partner receives files, it uses the IBM3270.XLT alternate translate table so that the translation on the sending system is the same as that on the receiving system. This prevents damage to data during translation. The following example shows the SEND command.

```
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(user01)
TRANSLATE(IBM327.);
```

The TRANSLATE parameter tells Expedite Base/AIX to use IBM3270.XLT to translate the data in the files from ASCII to EBCDIC. When the trading partner receives the files on the mainframe and downloads them to the workstation, the data looks the same on its workstation as it did on Company A's workstation.

Recovery levels

The default recovery level for a session with Information Exchange is checkpoint. This is usually the best way to exchange data when using a dial line or when transferring a large number of files or a large amount of data.

Checkpoint-level recovery ensures that if the line is disconnected or noisy, Expedite Base/AIX can pick up the data transfer at the last checkpoint rather than start the transfer over from the beginning.

When using a leased line, you can select session-level recovery, because it is very unlikely the line will go down during the data transfer. When you use session-level recovery, there is less processing required to recover after a failed session, as it is an all-or-nothing data transfer method.

When using a dial line, you can also select session-level recovery if you are transmitting only a small amount of data. If the line is disconnected, simply start the session over from the beginning.

Using session-level recovery, however, has its disadvantages. With checkpoint-level recovery, Information Exchange commits the data sent or received when a checkpoint takes place during the session. The commit processing is done in short intervals throughout the session. For a session-level recovery session, this processing is all done at the end of the session. If the communication line gets disconnected, Expedite Base/AIX must start from the beginning the next time a connection is made.

Using checkpoint-level, file-level, and user-initiated recovery

The most important job of your application interface is processing the Expedite Base/AIX response file. To work with Expedite Base/AIX, you need to understand how it recovers from an error during an Information Exchange session.

Checkpoint-level, file-level, and user-initiated recovery are Information Exchange methods that Expedite Base/AIX can use to recover data at specific checkpoints. When you use session-level recovery and an error occurs (See “Using session-level recovery” on page 62), Expedite Base/AIX must retransmit all data for the session. If you are sending large amounts of data, retransmission can take several hours. But when you select checkpoint-level, file-level, or user-initiated recovery, Expedite Base/AIX can recover data more efficiently.

The following table shows when Expedite Base/AIX takes checkpoints for each of these recovery methods:

Checkpoint-level recovery:	File-level recovery:	User-initiated recovery:
• after sending the number of bytes you specify in the COMMITDATA parameter of the TRANSMIT command (default is 37000 bytes) • at the end of each SEND, SENDEDI, or PUTMEMBER command, if the next command is not a SEND, SENDEDI, or PUTMEMBER command • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command	• after each file sent as a result of a SEND, SENDEDI, or PUTMEMBER command • after each file is received • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command	• after each COMMIT command, unless there is nothing to commit • at the end of each session, even if you have not specified a COMMIT command • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command

Checkpoint-level recovery is the default in Expedite Base/AIX. To request one of the other recovery methods, use the RECOVERY parameter on the TRANSMIT command with one of the following values:

s Session-level recovery

f File-level recovery

u User-initiated recovery

The processes for using checkpoint-level, file-level, and user-initiated recovery are very similar. Expedite Base/AIX uses the same work files for these recovery methods. Considerations for restarting after an error and resetting the Expedite Base/AIX session, described later in this chapter, also apply to all three recovery methods.

ATTENTION: You should never run multiple sessions for the same Information Exchange account and user ID from different machines. If you start an Information Exchange session using checkpoint-level, file-level, or user-initiated recovery while another Information Exchange session with the same account and user ID is running, Information Exchange ends the first session and continues the second session. The results in the first session depend on whether a checkpoint ended successfully.

- If a checkpoint ended successfully, Information Exchange delivers any data sent prior to the checkpoint and deletes any data from the mailbox that was received prior to the checkpoint.
- If a checkpoint did not end successfully, Information Exchange does not deliver any data and does not delete any received data from the mailbox. This means that data received in the first session may be received again in error.

In either case, you may get an error when you restart the first session.

Understanding post-session processing for checkpoint-level, file-level, and user-initiated recovery

The following sections describe what to do when a session ends in error. Post-session processing activities for checkpoint-level recovery include:

- Restarting a session
- Resetting a session
- Checking the SENT and RECEIVED response records
- Checking return codes

Restarting a session

It is important that your application process the responses in the response file correctly. Therefore, you need to understand the difference between session *restart* and session *reset*.

You initiate a session restart when an Information Exchange session ends in error and you want Expedite Base/AIX to resume the session at the last checkpoint. Before you restart a session, correct any problems that caused the previous session to end in error. Do not alter the response file, *baseout.msg*, or the session file, *session.fil*. You can correct a syntax error in the command file, *basein.msg*, to allow the session to continue, but do not add or delete lines from it.

If the session completes abnormally but you have return records for all of your commands with 0 return codes, then restart the session to allow it to complete normally. Do not reset the session, or lost or duplicate data may occur.

ATTENTION: Do not erase or alter the session file (session.fil) at any time. If session.fil is altered or erased during a restart, data may be duplicated or lost. If session.fil does not exist, Expedite Base/AIX starts processing at the beginning of the basein.msg file. When this happens, any data sent before the last successful checkpoint in the previous session is sent again. In addition, any data that was received during the previous session may be overwritten or erased. Files and messages received before the last successful checkpoint in the previous session are no longer available from Information Exchange. Issuing the same RECEIVE command may cause data already received, but not processed, to be overwritten by the results of the most recent RECEIVE command.

If you have altered or erased your *session.fil* file, you should review the contents of *baseout.msg* to see which commands processed successfully. Remove these commands from *basein.msg* and reset the session by entering **`iebase -r`**.

Changing files on restart

You can change some files before you restart a session. The following list indicates which files you cannot change and which you can change with limitations. You can change any files that are not on the list.

- **Message command file, basein.msg**

  You cannot change any commands or parameters in basein.msg that have been echoed to baseout.msg. This includes characters such as blanks and carriage returns that occur between commands and parameters. You can change commands and parameters in basein.msg that Expedite Base/AIX has written to tempout.msg or that are not shown in tempout.msg or baseout.msg.

- **Message response file, baseout.msg**

  Do not change baseout.msg.

- **Profile command file, basein.pro**

  You can change basein.pro if you want to modify a profile value. However, do not modify the COMMTYPE parameter on the TRANSMIT command.

- **Profile response file, baseout.pro**

  There is no need to change baseout.pro because Expedite Base/AIX creates a new baseout.pro when you restart.

- **Profile information file, iebase.pro**

  Never change iebase.pro. If you erase it, Expedite Base/AIX must create another iebase.pro using the commands in basein.pro. This means you need to provide the required profile information again, such as your account, user ID, and password, using profile commands. If you erase iebase.pro before restarting and basein.pro is present, Expedite Base/AIX can still restart.

- **Session work file, session.fil**

  Never change this file before restarting. Expedite Base/AIX uses it to restart the session.

- **Receive name file, rcvfiles.fil**

  Never change this file before restarting. Expedite Base/AIX uses this control file to track files it receives during the session.

- **Receive offset file, rcvofset.fil**

  Never change this file before restarting. It enables data to be appended correctly after restart.

- **Files being sent or received**

  Do not modify files you are sending with the SEND or PUTMEMBER command. Do not modify files you are receiving. Changes may cause unpredictable data to be sent or received.

Reviewing examples of session restart

Example 1

This example illustrates a session in which some commands did not process successfully, and you need to restart the session.

The files you are sending are *test1.snd* and *test2.snd* from the current directory. They have a user class of *test*. In the SEND command for *test2.snd*, you did not enter an ACCOUNT parameter. The following example shows the command file, *basein.msg*.

```
SEND    FILEID(test1.snd)
        ACCOUNT(ACCT)
        USERID(USER1)
        CLASS(TEST)
        DATATYPE(B);

SEND    FILEID(test2.snd)
        USERID(USER1)
        CLASS(TEST);
```

When the session is complete, the return code in the SESSIONEND record is 02806. This return code indicates a missing ACCOUNT parameter in the SEND command. Post-processing of the response file *baseout.msg* shows you what the error is but does not show you which command is in error.

The following example shows the response file, *baseout.msg*.

```
AUTOSTART SESSIONKEY(D9936DAP);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(D9936DAP)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND    FILEID(test1.snd)
        ACCOUNT(ACCT)
        USERID(USER1)
        CLASS(TEST)
        DATATYPE(B);
SENT    UNIQUEID(27854371) LENGTH(22);
RETURN(00000);

SESSIONEND(02806)
ERRDESC(Incomplete destination on SEND command.)
ERRTEXT(EXPLANATION: You specified an incomplete destination in the SEND)
ERRTEXT(command. The destination must be an ACCOUNT and USERID;)
ERRTEXT(SYSID, ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.)
ERRTEXT(USER RESPONSE: Check the message response file,
baseout.msg, or)
ERRTEXT(response work file, tempout.msg, to determine which SEND command)
ERRTEXT(produced the error. Correct the SEND command in the message command)
ERRTEXT(file, basein.msg, and retry the program.);
```

To determine which command is in error, examine the temporary response work file (tempout.msg). It shows the error is in the SEND command for the second file, test2.snd. Correct the error by entering an ACCOUNT parameter and restart Expedite Base/AIX. Processing will begin at the SEND command for test2.snd. The following example shows the temporary response work file, tempout.msg.

```
SEND    FILEID(test2.snd)
        USERID(USER1)
        CLASS(TEST);
RETURN(02806)

ERRDESC(Incomplete destination on SEND command.)
ERRTEXT(EXPLANATION: You specified an incomplete destination in the SEND)
ERRTEXT(command. The destination must be an ACCOUNT and USERID;)
ERRTEXT(SYSID, ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.)
ERRTEXT(USER RESPONSE: Check the message response file, baseout.msg, or)
ERRTEXT(response work file, tempout.msg, to determine which SEND command)
ERRTEXT(produced the error. Correct the SEND command in the message command)
ERRTEXT(file, basein.msg, and retry the program.);
```

Example 2

This example illustrates a session using user-initiated recovery. The session does not end successfully because of a syntax error on the RECEIVE command. The RECEIVE command was entered as **RECVE**. The following example shows the command file, basein.msg.

```
SEND FILEID(test1.snd) ACCOUNT(ACCT) USERID(USER1);
RECEIVE FILEID(testone.rcv);
COMMIT;
SEND FILEID(test2.snd) ACCOUNT(ACCT) USERID(USER2);
RECVE FILEID(testtwo.rcv);
COMMIT;
```

```
SEND FILEID(test3.snd) ACCOUNT(ACCT) USERID(USER3);
```

The session ends with a 15040 error on the **RECVE** command and a 15040 error on the SESSIONEND record. Because this is a syntax error, the command in error is found in the tempout.msg file. Post processing of the response file shows which commands completed successfully. The following example shows the message response file, baseout.msg, and the response work file, tempout.msg.

baseout.msg

```
AUTOSTART SESSIONKEY(STUPU7DO);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(STUPU7DO)IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(test1.snd) ACCOUNT(ACCT) USERID(USER1);
SENT UNIQUEID(STUPWAJN) LENGTH(2400);
RETURN(00000);
RECEIVE FILEID(testone.rcv);
RECEIVED ACCOUNT(*SYSTEM*) USERID(*ERRMSG*) CHARGE(6) LENGTH(257)
FILEID(testone.rcv) MSGDATE(980422) MSGDATELONG(199804022) MSGTIME(192954)
MSGSEQO(001687) SESSIONKEY(STUPU7DO) DELIMITED(N) SYSNAME(IBMIE)
SYSLEVEL(0405) TIMEZONE(L) DATATYPE(A) RECFM(????) RECLEN(0)
RECDLM(N) UNIQUEID(00000000) SYSTYPE(01) SYSVER(0);
RETURN(00000);

COMMIT;
RETURN(00000);

SESSIONEND(15040) ERRDESC(Command not recognized.)
ERRTEXT(EXPLANATION: You specified an unrecognized command in the)
ERRTEXT(command file.)
ERRTEXT(USER RESPONSE: Check the message response file, baseout.msg,)
ERRTEXT(profile response file, baseout.pro, or response work file,)
ERRTEXT(tempout.msg, to determine which command produced the error.)
ERRTEXT(Correct the appropriate command file and retry the program.);
```

tempout.msg

```
SEND FILEID(test2.snd) ACCOUNT(ACCT) USERID(USER2);
SENT UNIQUEID(STUPWEX6) LENGTH(13746);
RETURN(00000);

RECVE
RETURN(15040) ERRDESC(Command not recognized.)
ERRTEXT(EXPLANATION: You specified an unrecognized command in the)
ERRTEXT(command file.)
ERRTEXT(USER RESPONSE: Check the message response file, baseout.msg,)
ERRTEXT(profile response file, baseout.pro, or response work file,)
ERRTEXT(tempout.msg, to determine which command produced the error.)
ERRTEXT(Correct the appropriate command file and retry the program.);
```

The RETURN(00000) records indicate that the first three commands before the COMMIT command completed successfully. The RETURN(00000) and SENT records in tempout.msg for the second SEND command indicate that this command completed successfully. However, because the second COMMIT command was not processed due to the syntax error, Information Exchange will not deliver the file for the second SEND command.

The **RECVE** command should be corrected in the basein.msg file and the session restarted. Expedite will continue processing after the last successful COMMIT command, which in this example is after the first RECEIVE command. Expedite Base/AIX will complete processing for the second SEND command, the corrected RECEIVE command, and the final SEND.

Resetting a session

You initiate a session reset when the session ends in error and you do not want Expedite Base/AIX to continue the Information Exchange session. When you reset a session, Expedite Base/AIX acts as if a session were never active and begins processing at the beginning of the command file. There is some risk in using the same command file when you reset a session. Any data sent before the last successful checkpoint in the previous session is sent again.

Files and messages received before the last successful checkpoint in the previous session are no longer available from Information Exchange. You need to process the data from these received files and messages before you use the command file again. Otherwise, the results of the most recent RECEIVE commands may overwrite the data in the received files. When an existing session ends because of an error, partially processing the baseout.msg file can help you determine the commands that completed before the session ended. To partially process the baseout.msg file, process the commands in baseout.msg that completed successfully, build a new basein.msg file with the commands that were not processed, and start Expedite Base/AIX by typing **`iebase -r`** on the command line.

NOTE: If the session file, *session.fil*, does not exist, then there were no checkpoints taken during the previous session and Expedite Base/AIX begins processing at the start of the *basein.msg* file. Therefore, partial processing of the *baseout.msg* file is not necessary.

Reviewing examples of session reset

The following examples illustrate when a session reset is necessary. In these examples, the return code is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. This is usually caused by accessing Information Exchange using the same account and user ID on different machines at the same time.

Example 1

In this example you are sending files and the session ends in error.

You are sending 10 files to account act1 and user ID user01. The following example shows the basein.msg file.

```
SEND FILEID(order1.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order2.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order3.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order4.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order5.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order6.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order7.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order8.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order9.fil)  ACCOUNT(ACT1) USERID(USER01);
SEND FILEID(order10.fil) ACCOUNT(ACT1) USERID(USER01);
```

When the session ends in an error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows which commands completed successfully. The RETURN(00000) records indicate that the first five SEND commands completed successfully and the files were sent to Information Exchange. However, the remaining SEND commands were not processed. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(3774IO90);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3774IO90)
IEVERSION(04)IERELEASE(07);
RETURN(00000);
SEND FILEID(order1.fil) ACCOUNT(ACT1) USERID(USER01);
SENT UNIQUEID(43495778) LENGTH(6572);
RETURN(00000);
SEND FILEID(order2.fil) ACCOUNT(ACT1) USERID(USER01);
SENT UNIQUEID(74469581) LENGTH(5342);
RETURN(00000);
SEND FILEID(order3.fil) ACCOUNT(ACT1) USERID(USER01);
SENT UNIQUEID(67856334) LENGTH(3456);
RETURN(00000);
SEND FILEID(order4.fil) ACCOUNT(ACT1) USERID(USER01);
SENT UNIQUEID(19600628) LENGTH(9865);
RETURN(00000);
SEND FILEID(order5.fil) ACCOUNT(ACT1) USERID(USER01);
SENT UNIQUEID(37941045) LENGTH(9745);
RETURN(00000);
SEND FILEID(order6.fil) ACCOUNT(ACT1) USERID(USER01);
SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not
match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level
recovery, the check-)
```

```
ERRTEXT(point numbers for the send or receive side of the session
do not match)
ERRTEXT(the values Information Exchange recorded. Your session
file,)
ERRTEXT(session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the r command line
para)
ERRTEXT(meter on the IEBASE command. Also, make sure there is not
another)
ERRTEXT(user using this user ID. If the problem persists, contact
the)
ERRTEXT(Help Desk. Before starting the next session, review the)
ERRTEXT(message response file, baseout.msg, to see which commands
were)
ERRTEXT(processed successfully. Remove these commands from the)
ERRTEXT(message command file, basein.msg, so they are not
processed again.)
ERRTEXT(Warning: If you reset the session using the r command
line para)
ERRTEXT(meter you will no longer be able to continue the previous
session.)
ERRTEXT(Failure to modify the message command file, basein.msg,
before)
ERRTEXT(resetting the session may result in some data being lost
or duplicated.);
```

You need to edit the command file and delete the first five SEND commands. Then reset the session by entering **`iebase -r`** on the command line.

NOTE: If you do not remove the first five SEND commands before you reset the session, Expedite Base/AIX sends these files again.

Example 2

In this example you are receiving files and the session ends in error.

You are receiving four files from your Information Exchange mailbox. Each file has a different user class. The following example shows the command file.

```
RECEIVE FILEID(rcv1.fil) CLASS(TEST1);
RECEIVE FILEID(rcv2.fil) CLASS(TEST2);
RECEIVE FILEID(rcv3.fil) CLASS(TEST3);
RECEIVE FILEID(rcv4.fil) CLASS(TEST4);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows which commands completed successfully. The RETURN(00000) records indicate that the first three RECEIVE commands completed successfully. However, the last RECEIVE command was not processed.

The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(366830KH);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(366830KH)IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVE FILEID(rcv1.fil) CLASS(TEST1);
RECEIVED ACCOUNT(ACT1) USERID(USER01) CLASS(TEST1) CHARGE(5)
LENGTH(3467)
FILEID(rcv1.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(366830KH) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd1.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv2.fil) CLASS(TEST2);
RECEIVED ACCOUNT(ACT2) USERID(USER02) CLASS(TEST2) CHARGE(5) LENGTH(9771)
FILEID(rcv2.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(366830KH) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd2.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv3.fil) CLASS(TEST3);
RECEIVED ACCOUNT(ACT3) USERID(USER.3) CLASS(TEST3) CHARGE(5) LENGTH(2342)
FILEID(rcv3.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(366830KH) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd3.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv4.fil) CLASS(TEST4);
SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery, the check)
ERRTEXT(point numbers for the send or receive side of the session do not
match)
ERRTEXT(the values Information Exchange recorded. Your session file,)
ERRTEXT(session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the r command line para)
ERRTEXT(meter on the IEBASE command. Also, make sure there is not another)
ERRTEXT(user using this user ID. If the problem persists, contact the)
ERRTEXT(Help Desk.)
ERRTEXT(Before starting the next session, review the message response)
ERRTEXT(file, baseout.msg, to see which commands were processed)
ERRTEXT(successfully. Remove these commands from the message command)
ERRTEXT(file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the r command line para)
ERRTEXT(meter you will no longer be able to continue the previous session.)
ERRTEXT(Failure to modify the message command file, basein.msg, before)
```

```
ERRTEXT(resetting the session may result in some data being lost or
duplicated.);
```

You need to edit the basein.msg file and delete the first three RECEIVE commands. Then reset the session by entering **`iebase -r`** on the command line.

ATTENTION: If you specified OVERWRITE(Y) on the SESSION command or omitted the OVERWRITE parameter from the SESSION command in basein.pro, and you reset this session without modifying the command file, the first three files will be deleted and you will lose that data.

If you specified OVERWRITE(N), then new data received is appended to the existing files with the same name, which may make the data difficult to use.

Example 3

In this example you are receiving multiple files with one RECEIVE command and the session ends in error.

You are receiving six files from your Information Exchange mailbox using the MULTFILES(Y) parameter in the RECEIVE command. You want to receive the first file in *rcvfile* and each subsequent file in a new file named by numbering the file extensions starting with 002. The following example shows the basein.msg file.

```
RECEIVE FILEID(rcvfile) CLASS(TEST4) MULTFILES(Y);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows that Expedite Base/AIX received three files from your Information Exchange mailbox and stored them in rcvfile, rcvfile.002, and rcvfile.003. The absence of RETURN(00000) before the SESSIONEND record indicates that more files are in your mailbox.

The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(3377849);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3377849) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVE FILEID(rcvfile) CLASS(TEST4);
RECEIVED ACCOUNT(ACT1) USERID(USER01) CLASS(TEST4) CHARGE(5) LENGTH(6991)
FILEID(rcv.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377849) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd1.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

RECEIVED ACCOUNT(ACT2) USERID(USER02) CLASS(TEST4) CHARGE(5) LENGTH(2882)
FILEID(rcvfile.002) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377849) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd2.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

RECEIVED ACCOUNT(ACT3) USERID(USER.3) CLASS(TEST4) CHARGE(5)
LENGTH(6287)
FILEID(rcvfile.003) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377849) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd3.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery, the check)
ERRTEXT(point numbers for the send or receive side of the session do not
match)
ERRTEXT(the values Information Exchange recorded. Your session file,)
ERRTEXT(session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the r command line para)
ERRTEXT(meter on the IEBASE command. Also, make sure there is not another)
ERRTEXT(user using this user ID. If the problem persists, contact the)
ERRTEXT(Help Desk. Before starting the next session, review the)
ERRTEXT(message response file, baseout.msg, to see which commands were)
ERRTEXT(processed successfully. Remove these commands from the)
ERRTEXT(message command file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the r command line para)
ERRTEXT(meter you will no longer be able to continue the previous session.)
ERRTEXT(Failure to modify the message command file, basein.msg, before)
ERRTEXT(resetting the session may result in some data being lost or
duplicated.);
```

You need to edit the basein.msg file and delete the first three RECEIVE commands. Then reset the session by entering **`iebase -r`** on the command line. If Expedite Base/AIX appends data to an existing file, the data may be difficult to use. If it overwrites existing files, the original data is lost.

ATTENTION: If you specified OVERWRITE(Y) on the SESSION command or omitted the OVERWRITE parameter from the SESSION command in basein.pro, and you reset this session without modifying the basein.msg file, the first three files will be deleted and you will lose that data.

If you specified OVERWRITE(N) then new data received is appended to the existing files with the same name, which may make the data difficult to use. Therefore, you need to consider one of the following actions before you restart the session.

- Process the data in rcvfile, rcvfile.002, and rcvfile.003; for example, store the data in a database. Then erase the files or specify OVERWRITE(Y) in the SESSION command.
- Rename the files rcvfile, rcvfile.002, and rcvfile.003 so that Expedite Base/AIX does not overwrite them or append data to them when it receives the remaining files.
- Change the name of the FILEID parameter in the RECEIVE command to something other than rcvfile so that Expedite Base/AIX uses new file names when it receives the remaining files.

Checking the SENT and RECEIVED response records

You cannot determine what files Expedite Base/AIX sent or received by examining only the RETURN record. You also need to examine the SENT and RECEIVED records. SENT response records follow SEND commands. If no SENT record exists, Expedite Base/AIX did not send a file.

RECEIVED records follow RECEIVE commands. A RECEIVED record is written for each file received from Information Exchange. Files that have RECEIVED records are no longer in your Information Exchange mailbox and you cannot receive them again. If no RECEIVED record exists for a file, Expedite Base/AIX did not receive it.

NOTE: You should look only at the SENT and RECEIVED records in baseout.msg. You should not rely on SENT and RECEIVED records in tempout.msg. The tempout.msg file contains SENT and RECEIVED records processed since the last Information Exchange checkpoint.

Checking return codes

When a session completes, Expedite Base/AIX provides two numeric codes that identify the activities it performed. The first code is a five-digit return code that Expedite Base/AIX displays in the SESSIONEND or RETURN response record in baseout.msg. These return codes are grouped into categories, such as message command syntax errors and profile command syntax errors.

The second code is an exit code that Expedite Base/AIX returns to the operating system. You can use the operating system exit codes and Expedite Base/AIX return codes to decide what actions, if any, to take in the next session. For descriptions of the exit codes and return codes, see “Expedite Base/AIX messages and codes” on page 357.

The decision to restart or reset a session is based in the return code value in the SESSIONEND response record in baseout.msg. The following return codes are grouped into four categories:

- The return code is 00000, session completed normally. The operating system exit code is 0.

  If the return code is 0, you may still need to process the responses in the baseout.msg file. For example, if you receive multiple files from your mailbox to your system using separate file names, you need to know the names of the files and how many you received. If you receive files from your mailbox to your system using the original file names, you may need to check the file names indicated in the CDH. If you request system messages, such as error messages and acknowledgments, you may need to check this information.

  Information Exchange places error messages in your mailbox in a fixed format message with account *SYSTEM* and user ID *ERRMSG*. To receive these messages, you need to issue a RECEIVE command for this account and user ID and process the information in the baseout.msg file. This is also how you receive acknowledgments.

- The return codes are 16000-16999, or 28000-28020, session ended but incomplete. The operating system exit code is 112.

  Errors 16000-16999 indicate a problem trying to send the information to the specified destination. Error 28000 indicates that a warning was generated. Error 28010 indicates that one or more of the commands in the basein.msg file was not processed because of an error. The error number is shown in the RETURN response record immediately following the command that caused the error. A description of the error is in the ERRDESC and ERRTEXT records in the baseout.msg file immediately following the SESSIONEND response record.

  Error 28020 indicates an error occurred during the disconnect process. The number of the error that occurred is shown in a WARNING record following the SESSIONEND record. The WARNING record is followed by ERRDESC and ERRTEXT records describing the error.

  The information Expedite Base/AIX displays in the ERRDESC and ERRTEXT records is similar to the information in Appendix A, “Expedite Base/AIX messages and codes.”

- The return code is one of the following and indicates that a session restart is necessary.

Return code:	Category description:	Exit code:
11863, 26996	Wait and try again later	110
02000–04999	Message command syntax errors	111
05000–09999	Profile command syntax errors	104
11000–11999	Network errors	111
12000–12199	Modem script syntax errors	111, 114
12200–12399	Display status script syntax errors	111
13000–13999	Communication device driver errors	111
14000–15999	Parser errors	111
17000–18999	EDI parsing, send, or receive errors	111
19000–19999	TCP/IP communication errors	111
20000–23999	General environment errors	111
24000–24699	Session start and end errors	113
25000	PF key exit	111
26000–26999	Internal communications errors	111, 114
27000–27099	Old message.fil errors	111, 114
28100–28200	Miscellaneous command processing errors	111
29960–29990	SNA LU 6.2 communications errors	111, 114
29998	Modem command processor error	111
30006 - 30120	SSL communications errors	111
31400	Unexpected program interrupt error	Unknown

- The return code indicates that a session reset is necessary. The operating system exit code is 113 or 114. If the exit code is 114, you may be able to resolve the problem simply by redialing. If this does not resolve the problem, you need to reset the session.
  - Session start and end error return codes are 24000–24699.
  - Unexpected errors and commit error return codes are 31000–31339.

In addition, the Help Desk may suggest that the session be reset for other reasons. Your application interface should offer an easy way to reset the session and partially process the baseout.msg file.

Using session-level recovery

When using a leased line, you can select session-level recovery, because it is very unlikely the line will go down during the data transfer. When you use session-level recovery, there is less processing required to recover after a failed session, as it is an all-or-nothing data transfer method.

When using a dial line, you can also select session-level recovery if you are transmitting only a small amount of data. If the line is disconnected, simply start the session over from the beginning.

When you use session-level recovery to transmit data and an error occurs, Expedite Base/AIX stops transmission and produces a SESSIONEND record with a return code. You can check this return code to determine the cause of the error and correct the problem.

With session-level recovery, if data transmission stops, you need to send and receive all files again. Although it takes time to retransmit large amounts of data, there are advantages to using session-level recovery. For example, you do not need to be concerned with determining which files Expedite Base/AIX sent successfully and which files you need to resend.

If you use multiple START and END commands in basein.msg, there are certain precautions you need to take. See “Using multiple START and END commands with session-level recovery” on page 67 for more information.

If the session completes abnormally but you have return records for all of your commands with 0 return codes, then restart the session to allow it to complete normally. Do not reset the session, or lost or duplicate data may occur.

ATTENTION: If you start an Information Exchange session using session-level recovery while another Information Exchange session with the same account and user ID is running, Information Exchange ends the first session and starts the second session. Information Exchange does not deliver data sent in the first session and does not delete received data from the mailbox. This means that data received in the first session may be received again in error. The results when the first session ends are unpredictable.

Using session-level recovery, however, has disadvantages. With checkpoint-level recovery, Information Exchange commits the data sent or received when a checkpoint takes place during the session. The commit processing is done in short intervals throughout the session. For a session-level recovery session, this processing is all done at the end of the session.

So when Expedite Base/AIX sends the SESSIONEND command, Information Exchange does not return the SESSIONEND response until the commit processing is completed. If a large number of files are transferred during the session, the processing takes longer, especially during prime business hours.

During the processing, it is possible for the line to be disconnected because of a time-out. Expedite Base/AIX ends the session with a 29999 return code, “Session end response failure.” In this case, it is not clear whether or not the session ended successfully.

There are several ways to determine if the session was successful or not. For example, the CHECK parameter on the START command indicates to Expedite Base/AIX that you only want to check the status of the previous session. If you specify CHECK on the START command, do not specify any other commands except the END command in the input file. See “START command” on page 229 for more information.

Expedite Base/AIX also provides information about the previous session on the STARTED record. This record is written to the output file as a result of a START or AUTOSTART command. “STARTED record” on page 261 for more information.

After a session fails with 29999, follow these steps:

1. Specify AUTOSTART(N), AUTOEND(N), and RECOVERY(S) on your TRANSMIT command in the Expedite profile.
2. Create an input file containing START and END records; an example follows:

```
START CHECK(Y);
END;
```

NOTE: Do not specify any other commands in the input file if you specify CHECK(Y) on the session start command.

3. Run Expedite Base/AIX.No data is transferred in the above example, and you are not charged for this inquiry.
4. Examine the output file to check the LASTSESS parameter value on the STARTED record.

LASTSESS(0)	Indicates the previous session was successful. No further recovery is required.
LASTSESS(1)	Indicates the previous session was not successful.

If Expedite Base/AIX reported the 29999 SESSIONEND return code for a session-level recovery session, you should switch to checkpoint-level, file-level, or user-level recovery for future sessions with a similar number of commands.

If you were only receiving files from your Information Exchange mailbox, make sure all data was received by verifying that it is no longer in your mailbox. You can do this by viewing your mailbox with Information Exchange Administration Services or by running a QUERY command to get a list of AVAILABLE response records for each file in your mailbox. If the data is still in your mailbox, switch from session-level recovery to checkpoint-level recovery and run the session again to receive the data.

If you were sending files, you need to check your audit trail to see if the files were sent. You can do this by using Information Exchange Administration Services, or by using Expedite Base/AIX to request an audit be sent to your mailbox. Refer to Chapter 8, “Using Expedite Base/AIX message commands,’’ and “Using audit trails” on page 264 for more information. If the files were not sent, switch to checkpoint-level recovery and run the session again.

When a large number of files is being sent or received, session-level recovery is not recommended. Customers have experienced time-out problems when sending or receiving more than 700 files (the size of the files does not matter). Use checkpoint, user, or file-level recovery instead.

Understanding post-session processing for session-level recovery

The following sections describe what to do when a session ends with an error condition. Post-session processing activities for session-level recovery include:

- Processing the baseout.msg file records
- Checking return codes

Processing the response file records

When you transmit data, Expedite Base/AIX processes your message command file and creates a message response file, baseout.msg. The response records in baseout.msg are free-format records. Their syntax is the same as that defined for commands. Response records always start at the beginning of a line. However, parameters in response records may occur in any position and in any order. In addition, Expedite Base/AIX may not show all parameters in a response record. When examining response records, consider the following:

- Assume a default value if you do not get a response record parameter you are expecting. This is not an error.
- Truncate the parameter if a response record parameter is longer than you expect.

NOTE: You should be prepared for the possibility of new parameters in existing response records and entirely new response records that may be provided in the future.

Checking return codes

To ensure that Expedite Base/AIX finished processing the message command file, basein.msg, check the return code in the SESSIONEND record. Detailed return code descriptions are included in Appendix A, “Expedite Base/AIX messages and codes.”

The following is a list of the SESSIONEND return codes.

- The return code is 00000, session completed normally. The operating system exit code is 0.

  If the return code is 0, Expedite Base/AIX processed all commands and all command RETURN records contain zero return codes.

- The return codes are 28000–28020, session ended but is incomplete. The operating system exit code is 112.

  These errors indicate that one or more of the commands in the basein.msg file was not processed because of a basein.msg file error or because an error occurred during the disconnect process. If the problem was with the basein.msg file, correct the command that caused the error and run the program again. If the problem is in the disconnect process, the session error return code will be 28020. The session completed successfully but you should correct the problem so that future sessions disconnect from the network properly. The error number is shown in the RETURN response record immediately following the command that caused the error. The errors are described in ERRDESC and ERRTEXT records files immediately following the SESSIONEND response record. If the error is caused by a problem with a specific command, such as a syntax error, the command in error is followed by a RETURN record with the same return code as the SESSIONEND record.

- The return code is not 00000, 28010, or 28020. The operating system exit code is 110, 111, 113, or 114.

  This error indicates that Expedite Base/AIX did not finish processing the basein.msg file, the Information Exchange session failed, and none of the file transfer requests in the message command file completed. Expedite Base/AIX did not place any mail in your trading partner's Information Exchange mailbox or remove any from your mailbox. The SESSIONEND record may include an error description to help you find the problem. A command RETURN record may contain the same code and description. If baseout.msg does not contain a RETURN record, check baseout.pro for the error. If the return code was 28020, then all commands in the basein.msg file were processed.

  NOTE: While Expedite Base/AIX is receiving data from Information Exchange, it saves the data in files on your workstation. Even if a session does not complete successfully, Expedite Base/AIX may have received and saved data during the session. However, since you are using session-level recovery, both Information Exchange and Expedite Base/AIX ignore the files that were sent and received during the unsuccessful session. The next time a session is started, all of the data will be sent and received again.

Requests other than file transfer may complete even if the Expedite Base/AIX SESSIONEND is not 28010, 28020, or 00000. These requests include:

- ARCHIVEMOVE
- AUDIT
- CANCEL
- DEFINEALIAS
- GETMEMBER
- LIST
- LISTLIBRARIES
- LISTMEMBERS
- PURGE

If these requests are followed by a RETURN(00000) record in baseout.msg, they completed and you do not need to reissue them.

Reviewing examples of session-level recovery

The following examples illustrate session-level recovery.

Example 1

This example illustrates a session that ends in error when you are sending files.

You are sending five files to account *ACCT* and user ID *user01*. The following example shows the basein.msg file.

```
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(USER01);
SEND FILEID(file2.fil) ACCOUNT(ACCT) USERID(USER01);
SEND FILEID(file3.fil) ACCOUNT(ACCT) USERID(USER01);
SEND FILEID(file4.fil) ACCOUNT(ACCT) USERID(USER01);
SEND FILEID(file5.fil) ACCOUNT(ACCT) USERID(USER01);
```

When the session ends in error, the return code in the SESSIONEND record is 26805. This return code indicates that the carrier was lost during data transmission. The two SENT records indicate that Expedite Base/AIX sent the first two files before the session ended. However, since you are using session-level recovery, Information Exchange will not deliver these files to the trading partner's mailbox until the session ends successfully. You need to resend the files. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(34773663);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(34773663)IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(USER01);
SENT UNIQUEID(07580371) LENGTH(500);
RETURN(00000);
SEND FILEID(file2.fil) ACCOUNT(ACCT) USERID(USER01);
SENT UNIQUEID(78207881) LENGTH(300);
RETURN(00000);
SEND FILEID(file3.fil) ACCOUNT(ACCT) USERID(USER01);

SESSIONEND(26805)
ERRDESC(Lost carrier.)
ERRTEXT(EXPLANATION: The carrier was lost during data transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem)
ERRTEXT(persists, contact the Help Desk.);
```

To resend the files, run IEBASE again. When Expedite Base/AIX establishes a session, it processes all the commands in the basein.msg file again so you resend all five files.

Example 2

This example illustrates a session that ends in error when you are receiving files.

You want to receive four files from your Information Exchange mailbox. Each file has a different user class. The following example shows the basein.msg file.

```
RECEIVE FILEID(rcv1.fil) CLASS(TEST1);
RECEIVE FILEID(rcv2.fil) CLASS(TEST2);
RECEIVE FILEID(rcv3.fil) CLASS(TEST3);
RECEIVE FILEID(rcv4.fil) CLASS(TEST4);
```

When the session ends in error, the return code in the SESSIONEND record is 26996. This return code indicates that Expedite Base/AIX timed out while waiting for a response from the network. The three RECEIVED records indicate that you received the first three files before the session ended, but you did not receive all four files. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(3377856H);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3377856H) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVE FILEID(rcv1.fil) CLASS(TEST1);
RECEIVED ACCOUNT(ACT1) USERID(USER01) CLASS(TEST1) CHARGE(5) LENGTH(8279)
FILEID(rcv1.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377856H) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd1.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);
```

```
RECEIVE FILEID(rcv2.fil) CLASS(TEST2);
RECEIVED ACCOUNT(ACT2) USERID(USER02) CLASS(TEST2) CHARGE(5) LENGTH(437.)
FILEID(rcv2.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377856H) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd2.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv3.fil) CLASS(TEST3);
RECEIVED ACCOUNT(ACT3) USERID(USER.3) CLASS(TEST3) CHARGE(5) LENGTH(1789)
FILEID(rcv3.fil) MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717)
MSGSEQO(001955) SESSIONKEY(3377856H) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(snd3.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1) TRANSLATE(TESTDTBL);
RETURN(00000);

SESSIONEND(26996)
ERRDESC(Timedout while waiting for a response.)
ERRTEXT(EXPLANATION: DCL timed out while waiting for a response from the)
ERRTEXT(network gateway. This can occur if the line is dropped)
ERRTEXT(and the operating system does not return the lostcarrier)
ERRTEXT(condition to the program.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists, it may be)
ERRTEXT(that the Asynchronous Relay is down. Try the program again in)
ERRTEXT(about 30 minutes. If the problem still occurs, contact the)
ERRTEXT(Help Desk.);
```

Because you are using session-level recovery, Information Exchange ignores the three files it sent to you in the previous incomplete session. To receive all the files, run IEBASE again. When Expedite Base/AIX establishes a session, it processes all the commands in the basein.msg file again so that you receive all four files.

NOTE: If you specified OVERWRITE(N) in the SESSION command in basein.pro and you run IEBASE again without deleting the three files you originally received, Expedite Base/AIX appends the three files you receive the second time to the three files you received the first time. For more information, see “SESSION command” on page 148.

Using multiple START and END commands with session-level recovery

It is important to note the difference between an Expedite Base/AIX session and an Information Exchange session. An Expedite Base/AIX session consists of all commands specified in basein.msg that are issued during a single network connection. An Information Exchange session consists of the commands issued between a START command and an END command in basein.msg. The Expedite Base/AIX session is the same as the Information Exchange session when there is only one START command and one END command in basein.msg. However, Expedite Base/AIX allows the user to start and end multiple Information Exchange sessions within a single Expedite Base/AIX connection.

If you use multiple START and END commands in basein.msg, you create an environment similar to that of checkpoint-level recovery. Each END command stops an Information Exchange session. Requests in each Information Exchange session complete even if a subsequent Information Exchange session ends in error. Only Information Exchange sessions that end in error require you to send or receive data again.

If you specify multiple START and END commands in basein.msg, and an error occurs before all commands in basein.msg have completed, you need to take special measures to process your basein.msg and baseout.msg files before restarting. These measures are similar to those you need to take when doing checkpoint-level recovery.

That is, you need to review the contents of baseout.msg to determine which of the Information Exchange sessions completed successfully, and which need to be run again. Commands in the sessions that were successful must be removed from basein.msg to avoid sending duplicate data or losing received data.

ATTENTION: Failure to remove commands for successfully completed Information Exchange sessions from basein.msg may result in duplicate or lost data in subsequent Expedite Base/AIX sessions. When using session-level recovery with multiple START and END commands, you need to process your basein.msg and baseout.msg files similar to the way required for checkpoint-level recovery. It is recommended that you use session-level recovery with a single START and END command to avoid the need to process these files after incomplete sessions. If you need to run multiple Information Exchange sessions within a single Expedite Base/AIX session, you may consider using checkpoint-level recovery instead of session-level recovery.

When an Information Exchange session has completed, Expedite Base/AIX will write two records to baseout.msg. The first is the END record, which is echoed from basein.msg. The second record Expedite Base/AIX will write is the RETURN record, which shows the return code for the END command. When you see the END and RETURN records in baseout.msg, all commands in that Information Exchange session have been completed. Before starting Expedite Base/AIX again, you should remove all commands processed successfully from basein.msg.

NOTE: When Expedite Base/AIX has completed all commands in basein.msg, it writes a return code for the Expedite Base/AIX session. The return code is specified in the SESSIONEND record in baseout.msg. There will only be one SESSIONEND record in baseout.msg, regardless of the number of Information Exchange sessions started and ended within basein.msg.

Reviewing examples using multiple Information Exchange sessions with session-level recovery

Example 1

This example illustrates three Information Exchange sessions within a single Expedite Base/AIX session. The following shows the contents of basein.msg.

```
START;
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(USER01);
END;
START;
```

```
RECEIVE FILEID(rcv.fil);
END;
START;
SEND FILEID(file2.fil) ACCOUNT(ACCT) USERID(USER02);
END;
```

Following are the contents of baseout.msg when Expedite Base/AIX has completed processing.

```
START;
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(SOVSHX25) IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(SOVSHX25);
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(USER01);
SENT UNIQUEID(73133557) LENGTH(500);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(77EIIGH2) IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(77EIIGH2);
RECEIVE FILEID(rcv.fil);
RECEIVED ACCOUNT(ACCT) USERID(USER02) CLASS(TEST1) CHARGE(5) LENGTH(4101)
FILEID(rcv.fil) MSGDATE(980809) MSGDATELONG(19980809) MSGTIME(134011)
MSGSEQO(001988) SESSIONKEY(77EIIGH2) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(SENDER.FIL) SENDERLOC(/home/expedite) FILEDATE(980412)
FILEDATELONG(19980412) FILETIME(120000) RECFM(????) RECLEN(0) RECDLM(C)
UNIQUEID(73133557) SYSTYPE(15) SYSVER(4) TRANSLATE(IESTDTBL);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(SDK84MMN) IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(SDK84MMN);
SEND FILEID(file2.fil) ACCOUNT(ACCT) USERID(USER02);
SENT UNIQUEID(00959571) LENGTH(1335);
RETURN(00000);
END;
RETURN(00000);
SESSIONEND(00000);
```

Note that each of the END records is followed by a RETURN record. This means that each of the Information Exchange sessions completed. Further, the return code 00000 in the RETURN records indicates that each session completed successfully. Finally, the SESSIONEND record indicates the completion of the Expedite Base/AIX session.

Example 2

This example uses the same input file as Example 1. However, in this example, the output file indicates that a problem occurred during the connection. Following are the contents of baseout.msg when Expedite Base/AIX has completed processing.

```
START;
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(77FGHH55) IEVERSION(04)
```

```
IERELEASE(07);
RETURN(00000) SESSIONKEY(77FGHH55);
SEND FILEID(file1.fil) ACCOUNT(ACCT) USERID(USER01);
SENT UNIQUEID(73133557) LENGTH(500);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(0000) SESSIONKEY(337FJK73) IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(337FJK73);
RECEIVE FILEID(rcv.fil);

SESSIONEND(26805)
ERRDESC(Lost carrier.)
ERRTEXT(EXPLANATION: The carrier was lost during data transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists, contact
the)
ERRTEXT(Help Desk.);
```

In this example, the first Information Exchange session completed successfully, as indicated by the END and RETURN(00000) records. However, the second session did not complete successfully. The baseout.msg file shows that there are no END or RETURN records associated with the second Information Exchange session. The RECEIVE command is, instead, followed by a SESSIONEND record indicating the end of the Expedite Base/AIX session. In addition, there are ERRDESC and ERRTEXT records with information about the problem.

Because you are using session-level recovery, Expedite Base/AIX will begin processing at the beginning of basein.msg the next time it is run. If no changes are made to basein.msg, the file that was successfully sent the first time will be sent again, resulting in duplicate data sent to Information Exchange. Therefore, before you restart Expedite Base/AIX, you should remove the commands associated with this Information Exchange session from basein.msg The new basein.msg should look as follows:

```
START;
RECEIVE FILEID(rcv.fil);
END;
START;
SEND FILEID(file2.fil) ACCOUNT(ACCT) USERID(USER02);
END;
```

Receiving multiple files

When Expedite Base/AIX processes a RECEIVE command, it uses the parameters in the command to determine which files to receive from Information Exchange. You can specify that Expedite Base/AIX receives all files in the mailbox, all files sent from a specific account and user ID, all files with a specific user class, or any combination of these. See “RECEIVE command” on page 199 for detailed information.

In the RECEIVE command, you need to specify the name of the file in which Expedite Base/AIX is to place the received file. If you have more than one file in your mailbox, you can receive the files from the mailbox in a single file or receive each file in a separate file.

When you receive multiple files from your mailbox in a single file, Expedite Base/AIX appends the files in the order it receives them. This is the default for receiving multiple files.

To receive multiple files from your mailbox in separate files, specify the value **y** in the MULTFILES parameter. This tells Expedite Base/AIX to place the first file in the file you specified and place subsequent files in new files by numbering the file extensions starting with 002.

The new extension will be added after any existing extension on the file name; any original extension will not be truncated. If more than 999 files are received, the extension becomes four digits: .1000, .1001, .1002, and so on. If more than 9999 are received, the extension becomes five digits: .10000, .10002, and so on. If more than 99999 are received, the rest of the files are appended to the file name in the FILEID with the extension *.ovf*.

For example, if you specify FILEID(testmsg) and three files are received, Expedite Base/AIX names the files as follows:

```
File 1 = testmsg
File 2 = testmsg.002
File 3 = testmsg.003
```

Receiving specific files

Previous sections of this chapter have demonstrated that you can specify certain criteria on the RECEIVE command to limit the files that you receive. For example, you can receive all files from a particular user, or all files with a particular user class.

You can use the RECEIVE command to specify a date and time range for files you want to receive. Information Exchange checks the date and time the files were sent to you, and gives you those files that fall within your specified date and time range.

For example, suppose you wanted to receive only those files sent to you between noon and 6:00 p.m. on June 14, 1998. You would include the following on your RECEIVE command:

```
STARTDATE(980614) STARTTIME(120000) ENDDATE(980614) ENDTIME(180000)
TIMEZONE(L)
```

Expedite Base/AIX also allows you to receive a single, specific file even if other files in your mailbox are from the same sender or have the same user class. Each file in your mailbox has a unique message key that distinguishes the file from all others. You can issue a RECEIVE command, using the MSGKEY parameter to specify the unique message key of the file you want to receive.

For example, suppose there were three files in your mailbox from the same user, with the same user class. The files were sent to your mailbox on three consecutive days. However, you are only interested in receiving the first file, which has a unique message key of 887A9DE0021FA9C236F8. Your RECEIVE command might look as follows:

```
RECEIVE FILEID(first.fil) MSGKEY(887A9DE0021FA9C236F8);
```

As a result of this command, Expedite Base/AIX receives only the file with this message key.

To find out what the message key is for a specific file, you can use Information Exchange Administration Services, or use the Expedite Base/AIX QUERY command in basein.msg. As a response to the QUERY command, Expedite Base/AIX provides information about each of the files in your mailbox, including the message key for each file. “Querying a mailbox” on page 266 provides more information about using the QUERY command. “RECEIVE command” on page 199 provides information about the format of the RECEIVE command.

SEND and RECEIVE file number limits

Information Exchange limits the number of files that can be sent and received between commits because of the processing requirements involved. The current limit of 1,000 files is an Information Exchange value that can be set differently in different Information Exchange installations. Contact your marketing representative to determine the maximum for Information Exchange installations outside the United States.

There are also limitations on the number of files that can be sent to and received from Expedite Base/AIX, which depend upon the type of recovery you are using. This section discusses limitations which you should take into consideration for your installation.

User-level recovery

If you are using user-level recovery, you need not specify more than 1,000 SEND, SENDEDI, or PUTMEMBER commands without specifying a COMMIT command. Do not send more than 1,000 EDI envelopes within a single file because each envelope counts as a file.

Checkpoint-level recovery

If you use checkpoint-level recovery, Expedite Base/AIX will perform a COMMIT after sending or receiving the number of characters specified in the COMMITDATA parameter on the TRANSMIT command in basein.pro. The default value for this parameter is 141000. You need not attempt to send or receive more than 1,000 files whose combined size is less than the value in the COMMITDATA parameter. If this is the case, you can either lower the value of the COMMITDATA parameter or decrease the number of files being sent or received. This will allow a COMMIT to be performed before the 1,000 file limit is reached.

File-level recovery

If you use file-level recovery, there is no limitation to the number of files you can send and receive. This is because each file is committed as it is sent or received, and the maximum number of files between commits is 1. If you are sending or receiving many small files, you can get better performance using checkpoint-level recovery.

Session-level recovery

If you use session-level recovery and you try to send more files than the limit, you will receive an error from Information Exchange, which Expedite Base/AIX reports to you as SESSIONEND return code 31360. In this case, break your input file into multiple input files and run Expedite Base/AIX for each of the input files. Follow the instructions in Chapters 6 and 7 to recover the session that ended with the 31360 return code.

If you have more than 1,000 files in your mailbox that match your receive request, Information Exchange stops sending them when the 1,000 file limit is reached. If you have more files in your mailbox, Expedite Base/AIX returns a 28171 value in the RETURN parameter after the last file was received. The SESSIONEND code is 28010, indicating the session completed successfully but not all commands were processed. The data you already received is no longer in your Information Exchange mailbox, but there are still additional files in the mailbox which match your receive request. Before using Expedite Base/AIX again to receive the remaining files, be sure to

process the data already received in order to ensure that the files are not overwritten during the next session with Information Exchange. “Using session-level recovery” on page 62 for more information about processing received files when using session-level recovery.

Learning more about Expedite Base/AIX

The following examples illustrate how companies can use Expedite Base/AIX. These examples may help you implement this application in your company.

Example 1

Company A is an insurance agency that sells policies and processes claims. At the end of each business day, Company A runs an application program to collect information about the day's transactions and prepare the information for transmission to the home office. The program writes all claim information to the file claims.fil and writes all policy information to policy.fil. Every night the company sends these two files to the home office, and the home office sends the agency updated claim information in update.fil.

The activities that Company A needs to perform include:

- Create a standard message command file (basein.msg) to send and receive the same files every night and receive any system error messages.
- Specify a delayed transmission for the middle of the night when telephone and Information Exchange charges are less.
- Specify checkpoint-level recovery since the files contain large amounts of data.
- Write an application interface to:
  - Read the message command file, basein.msg and write any errors to an error log.
  - Process the message command file, basein.msg and modify the message command file as needed when a session does not complete.
- Review the error log and database to determine the results of the Information Exchange session.

The following example shows the message command file, basein.msg.

```
SEND FILEID(claims.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(CLAIMS);
SEND FILEID(policy.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(POLICY);
RECEIVE FILEID(update.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(UPDATE);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

In this file Company A is entering commands to send claims.fil to account *home* and user ID *office* with the user class *claims* and policy.fil to the same account and user ID with the user class *policy.*

The company also wants to receive any files in its mailbox from account *home* and user ID *office* with the user class update. In addition, it wants to receive any system error messages that Information Exchange places in its mailbox. The system error messages have an account of *SYSTEM* and a user ID of *ERRMSG*. Company A wants to receive any error messages in errors.fil.

The following example shows the message command file, basein.msg.

```
AUTOSTART SESSIONKEY(DKK68390);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DKK68390) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(claims.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(CLAIMS);
SENT UNIQUEID(44063134) LENGTH(215432);
RETURN(00000);
SEND FILEID(policy.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(POLICY);
SENT UNIQUEID(64554922) LENGTH(142154);
RETURN(00000);
RECEIVE FILEID(update.fil) ACCOUNT(HOME) USERID(OFFICE) CLASS(UPDATE);
RECEIVED ACCOUNT(HOME) USERID(OFFICE) CLASS(UPDATE) CHARGE(5) LENGTH(2203)
FILEID(update.fil) MSGDATE(980112) MSGDATELONG(19980112) MSGTIME(113314)
MSGSEQO(001950) SESSIONKEY(DKK68390) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(update0a) SENDERLOC(/home/expedite) FILEDATE(980110)
FILEDATELONG(19980110) FILETIME(160340) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(16662366) SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This file shows the commands from the message command file, basein.msg, along with their associated response records and return codes.

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session started successfully.

The SENT records indicate that Expedite Base/AIX sent the files. These records also provide information about the unique ID Expedite Base/AIX assigned the files and the length of the files. The 00000 return code in the RETURN records indicate that the commands completed successfully.

The RECEIVED record indicates that Expedite Base/AIX received one file and provides information about the file. The 00000 return code in the RETURN record indicates that the command completed successfully.

The second RECEIVE command is not followed by any RECEIVED records. Instead, it is immediately followed by a return record with a 00000 return code. This means that the command completed successfully, but there were no system error messages to receive.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Example 2

Company B is a hardware store that sends orders electronically to hardware manufacturers. The company places orders once a week to all of the manufacturers on the same day and requests delivery acknowledgments when the manufacturers receive the orders.

Since the order files contain small amounts of data, Company B decides to simplify its programming tasks by using session-level recovery. This means that if a session ends unexpectedly, the company must resend all the files.

NOTE: Because Company B is using session-level recovery, it does not need to partially process the message command file, basein.msg, when a session ends unexpectedly. However, Company B should still review the message response file to review what happened in the previous session and to check for error codes and error messages.

The following summarizes the activities Company B needs to perform:

- Create a panel-driven program that a manager can use to order products. The program needs to create the necessary order files and the SEND commands in basein.msg for each file. It also needs to read the message file and write any errors to an error log when the session ends.
- Specify session-level recovery.

The following example shows the message command file, basein.msg.

```
SEND FILEID(widgets.fil) ACCOUNT(WIDG) USERID(WIDGETS) CLASS(COMPANYB);
SEND FILEID(gidgets.fil) ACCOUNT(GIDG) USERID(GIDGETS) CLASS(COMPANYB);
SEND FILEID(gadgets.fil) ACCOUNT(GADG) USERID(GADGITS) CLASS(COMPANYB);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

In this file, Company B is entering commands to send three order files with the user class *companyb*. The company also wants to receive any system error messages that Information Exchange places in its mailbox. The system error messages have an account of *SYSTEM* and a user ID of *ERRMSG*. Company B wants to receive any error messages in errors.fil.

The following example shows the message command file, basein.msg.

```
AUTOSTART SESSIONKEY(DI88SNKY);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DI88SNKY) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(widgets.fil) ACCOUNT(WIDG) USERID(WIDGETS)
CLASS(COMPANYB);
SENT UNIQUEID(09383850) LENGTH(1525);
RETURN(00000);
SEND FILEID(gidgets.fil) ACCOUNT(GIDG) USERID(GIDGETS)
CLASS(COMPANYB);
SENT UNIQUEID(54803437) LENGTH(1267);
RETURN(00000);
SEND FILEID(gadgets.fil) ACCOUNT(GADG) USERID(GADGITS)
CLASS(COMPANYB);
SENT UNIQUEID(09874578) LENGTH(856);
RETURN(00000);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This file shows the commands from the message command file, basein.msg, along with their associated response records and return codes.

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session started successfully.

The SENT records indicate that Expedite Base/AIX sent the files. These records also provide information about the length of the files and the unique ID Expedite Base/AIX assigned the files. The 00000 return code in the RETURN records indicate that the commands completed successfully.

The RECEIVE command is not followed by any RECEIVED records. Instead, it is followed by a return record with a 00000 return code. This means that the command completed successfully, but there were no system error messages to receive.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN indicates that the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Example 3

Company C is an engineering firm that sends files that contain CAD/CAM drawings, which are in a special data format that must be treated like binary data. The company also sends free-format messages and receives requests for proposals (RFPs). The employees use a panel-driven program to send these files and messages. The program stores the e-mail messages in email.fil.

On occasion, the company receives RFPs from different sources. Because it does know when to expect these requests, the company queries its mailbox every day.

Expedite Base/AIX performs these activities when an employee initiates a session with Information Exchange:

- Sends CAD/CAM files as binary data
- Sends the free-format message file
- Sends a QUERY command to obtain a list of the files in the company mailbox
- Receives any system error messages
- Uses checkpoint-level recovery because the files contain large amounts of data

Company C's program performs these activities at the end of each session:

- Reads the message command file, basein.msg, and writes any errors to an error log
- Optionally updates a database to indicate which files Expedite Base/AIX sent
- In case of an incomplete session, processes the message command file, basein.msg, and modifies the message command file as needed
- Displays a list of files that are in the company mailbox

The following example shows the message command file, basein.msg.

```
SEND FILEID(cadcam.fil) ACCOUNT(ABCD) USERID(ENGIN1) CLASS(DRAWING)
DATATYPE(B);
SEND FILEID(email.fil) ACCOUNT(ABCD) USERID(ENGIN1) FORMAT(Y);
QUERY;
AUDIT STARTDATE(980601) ENDDATE(980701);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

In this file Company C is entering commands to send cadcam.fil to account *abcd* and user ID *engin1* with the user class *drawing*. Using the DATATYPE parameter, the company indicates that Expedite Base/AIX should treat the data in this file as binary data (*b*), which does not get translated to EBCDIC when it is sent to Information Exchange. The company is also sending an e-mail file (email.fil) to the same user to which it is sending cadcam.fil. The value **y** in the FORMAT parameter tells Expedite Base/AIX to send the file to Information Exchange with a user class of *FFMSG001*.

In addition, Company C wants to obtain a list of files in its mailbox. It also wants audit records for messages and files from June 1, 1998, to July 1, 1998, and wants to receive any system error messages that Information Exchange places in the mailbox. The system error messages have an account of *SYSTEM* and a user ID of *ERRMSG*. Company C receives error messages in errors.fil.

The following example shows the message command file, basein.msg.

```
AUTOSTART SESSIONKEY(4KK.9SGT);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(4KK09SGT) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SEND FILEID(cadcam.fil) ACCOUNT(ABCD) USERID(ENGIN1) CLASS(DRAWING)
DATATYPE(B);
SENT UNIQUEID(87578890) LENGTH(1264898);
RETURN(00000);

SEND FILEID(email) ACCOUNT(ABCD) USERID(ENGIN1) FORMAT(Y);
SENT UNIQUEID(35879068) LENGTH(948);
RETURN(00000);

QUERY;
AVAILABLE ACCOUNT(WXYZ) USERID(WXYZ001) MSGKEY(A876BC932D75E9712D88)
CLASS(RFP) MSGDATE(980601) MSGDATELONG(19980601) MSGTIME(081522)
LENGTH (5000) SYSNAME(EB/AIX) SYSLEVEL(0450) DATATYPE(A)
EDITYPE(UNFORMATTED) SENDERFILE(filemeg.txt) SENDERLOC(/home/expedite)
FILEDATE(980528) FILEDATELONG(19980528) FILETIME(123636) RECFM(????)
RECLEN(0) RECDLM(C) UNIQUEID(16706410) SYSTYPE(17) SYSVER(4)
  TRANSLATE(IESTDTBL);
AVAILABLE ACCOUNT(WXYZ) USERID(WXYZ001) MSGKEY(F5831D55E729A597B732)
CLASS(RFP) MSGDATE(980601) MSGDATELONG(19980601) MSGTIME(081522)
LENGTH(5000) SYSNAME(EB/AIX) SYSLEVEL(0450);
EDITYPE(UNFORMATTED) SENDERFILE(filemeg.txt) SENDERLOC(/home/expedite)
FILEDATE(980621) FILEDATELONG(19980621) FILETIME(123636) RECFM(????)
RECLEN(0) RECDLM(C) UNIQUEID(13857218) SYSTYPE(17) SYSVER(4)
  TRANSLATE(IESTDTBL);
AVAILABLE ACCOUNT(WXYZ) USERID(WXYZ001) MSGKEY(94843039838303893000)
CLASS(FFMSG001) MSGDATE(980601) MSGDATELONG(19980601) MSGTIME(083015)
LENGTH(256) SYSNAME(EB/AIX) SYSLEVEL(0450);
EDITYPE(UNFORMATTED) SENDERFILE(filemeg.txt) SENDERLOC(/home/expedite)
FILEDATE(980314) FILEDATELONG(19980314) FILETIME(123636) RECFM(????)
RECLEN(0) RECDLM(C) UNIQUEID(38235082) SYSTYPE(17) SYSVER(4)
TRANSLATE(IESTDTBL);
RETURN(00000);

AUDIT STARTDATE(980601) ENDDATE(980701);
RETURN(00000);

RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This file shows the commands from the message command file, basein.msg, along with their associated response records and return codes.

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session started successfully.

The SENT records indicate that Expedite Base/AIX sent the files. These records also provide information about the length of the files and the unique ID Expedite Base/AIX assigned the files. The 00000 return code in the RETURN records indicate that the commands completed successfully.

The responses to the QUERY command are the AVAILABLE records. The AVAILABLE records indicate that there are three files in your Information Exchange mailbox ready to be received. The three AVAILABLE records indicate that Information Exchange placed three files in the company mailbox. The user class (*ffmsg001*) in the last AVAILABLE record indicates that it is a free-format message. Company C can use the MSGKEY value in the AVAILABLE records to receive that specific file by using the MSGKEY parameter in the RECEIVE (or RECEIVEEDI) command. For more information, see “RECEIVE command” on page 199.

The 00000 return code in the RETURN record after the AUDIT command indicates that Information Exchange placed an audit file in the company mailbox with an account of *SYSTEM* and a user ID of *AUDITS*. To receive this file, Company C needs to enter a RECEIVE command in the basein.msg file and run another Information Exchange session.

The RECEIVE command is not followed by any RECEIVED records. Instead, it is followed by a RETURN record with a 00000 return code. This means that the command completed successfully, but there were no system error messages to receive.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Example 4

Company D is a manufacturer that sells widgets to hardware stores. Each day Company D receives orders from hardware stores electronically. All widget orders in the company mailbox have a user class of widgets. The company receives these order files in a single file so that a clerk can enter the information in the order entry system.

On occasion, the clerk erases a widget order accidentally. When this happens, the clerk requests the archived copy of the file from Information Exchange.

Company D also receives company profiles from new hardware stores. These files all have the user class profile. When Company D receives more than one profile, it receives each profile in a different file.

Company D needs to create a message command file (basein.msg) to:

- Receive all order files in a single file
- Receive company profiles in separate files
- Move a file from archive to the company mailbox
- Receive any system error messages

The following example shows the message command file, basein.msg.

```
RECEIVE FILEID(orders.fil) CLASS(WIDGETS) ARCHIVEID(980601W) MULTFILES(N);
RECEIVE FILEID(profiles) CLASS(PROFILES) ARCHIVEID(980601P) MULTFILES(Y);
ARCHIVEMOVE ARCHIVEID(980531P);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

In this file Company D is entering commands to receive files with the user class *widgets*. The value n in the MULTFILES parameter tells Expedite Base/AIX to place all of these files in a single file (orders.fil). The value in the ARCHIVEID parameter is the archive reference identifier the company wants Information Exchange to assign the file.

Company D also wants to receive any files with the user class *profiles*. The value *y* in the MULTFILES parameter tells Expedite Base/AIX to place the first file in *profiles* and create new separate files for subsequent files and name them by numbering the extensions starting with 002.

In addition, the company wants Information Exchange to copy a file from the archive to the company mailbox, and receive any system error messages.

The following example shows the message command file, basein.msg.

```
AUTOSTART SESSIONKEY(DJJK4889);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DJJK4889) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVE FILEID(orders.fil) CLASS(WIDGETS) ARCHIVEID(980601W) MULTFILES(N);
RECEIVED ACCOUNT(AAAA) USERID(AAAA01) CLASS(WIDGETS) CHARGE(1) LENGTH(2513)
FILEID(orders.fil) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(4890289) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(aaaa.fil) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);

RECEIVED ACCOUNT(BBBB) USERID(BBBB01) CLASS(WIDGETS) CHARGE(1) LENGTH(1561)
FILEID(orders.fil) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(489028) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(bbbb.fil) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);
```

```
RECEIVED ACCOUNT(CCCC) USERID(CCCC01) CLASS(WIDGETS) CHARGE(1) LENGTH(2221)
FILEID(orders.fil) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(489028) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(cccc.fil) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

RECEIVE FILEID(profile.fil) CLASS(PROFILES) ARCHIVEID(980601P) MULTFILES(Y);
RECEIVED ACCOUNT(DDDD) USERID(DDDD01) CLASS(PROFILES) CHARGE(1) LENGTH(2451)
FILEID(profiles) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(489028) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(dddd.pro) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);

RECEIVED ACCOUNT(EEEE) USERID(EEEE01) CLASS(PROFILES) CHARGE(1) LENGTH(3314)
FILEID(profiles.002) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(489028) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(dddd.pro) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);

RECEIVED ACCOUNT(FFFF) USERID(FFFF01) CLASS(PROFILES) CHARGE(1) LENGTH(1345)
FILEID(profiles.003) MSGDATE(980603) MSGDATELONG(19980603) MSGTIME(120132)
MSGSEQO(489028) SESSIONKEY(DJJK4889) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(UNFORMATTED)
SENDERFILE(dddd.pro) SENDERLOC(/home/expedite) FILEDATE(980601)
FILEDATELONG(19980601) FILETIME(123000) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(34092819)) SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

ARCHIVEMOVE ARCHIVEID(930531P);
MOVED NUMBER(1);
RETURN(00000);

RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This file shows the commands from the message command file, basein.msg, along with their associated response records and return codes.

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session started successfully.

The 00000 return code in the RETURN record after the third RECEIVED record indicates that Expedite Base/AIX received three files in orders.fil.

The next three RECEIVED records indicate that Expedite Base/AIX received three files with the user class profiles (*profiles*, *profiles.002*,and *profiles.003*).

The MOVED record after the ARCHIVEMOVE record indicates that Information Exchange copied one file from Information Exchange archive to the company mailbox.

The last RECEIVE command is not followed by any RECEIVED records. Instead, it is followed by a return record with a 00000 return code. This means that the command completed successfully, but there were no system error messages to receive.

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

Example 5

Company E is a widget manufacturer that receives its software from a vendor. The vendor is responsible for setting up the company's workstations, installing all software, and providing code updates. The vendor sends the code update file to the company's Information Exchange mailbox, and Company E receives the file using the ORIGFILE parameter. This parameter tells Expedite Base/AIX to receive the updates in a file using the file name from the sending system. The original file name from the sending system is in the CDH. Therefore, both the vendor and Company E must be using versions of the Expedite products that support the CDH.

NOTE: When Company E uses the ORIGFILE parameter, it must use the FILEID parameter in case the file does not have a CDH.

Company E needs to create a message command file (basein.msg) to:

- Receive code updates in a file using the file name from the sending system
- Receive any system error messages

The following example shows the message command file, basein.msg.

```
RECEIVE FILEID(/home/mycode/code.fil) CLASS(CODE) ORIGFILE(Y);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

In this file, Company E is entering commands to receive files with the user class *code*. The value *y* in the ORIGFILE parameter tells Expedite Base/AIX to place this data in files using the file names from the sending system. The value */home/mycode/code* in the FILEID parameter tells Expedite Base/AIX to place the files in code.fil in the *mycode* directory if the files do not have CDHs. If the files have CDHs, and if the original file name is a valid AIX file name, then Expedite Base/AIX ignores the file name code.fil in the FILEID parameter, but uses the path.

Company E also wants to receive any system error messages that Information Exchange places in its mailbox. The system error messages have an account of *SYSTEM* and a user ID of *ERRMSG*. The company wants to receive any error messages in errors.fil.

The following example shows the message command file, basein.msg.

```
AUTOSTART SESSIONKEY(227D94JK);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(227D94JK) IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVE FILEID(/home/mycode/code.fil) CLASS(CODE) ORIGFILE(Y);
RECEIVED ACCOUNT(VEND) USERID(VENDOR) CLASS(CODE) CHARGE(5) LENGTH(5241)
FILEID(/home/mycode/order) MSGDATE(980603) MSGDATELONG(19980603)
MSGTIME(115306) MSGSEQO(001952) SESSIONKEY(227D94JK)
DELIMITED(N) SYSNAME(EB/AIX) SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A)
EDITYPE(UNFORMATTED) SENDERFILE(order) SENDERLOC(/usr/bin/fix)
FILEDATE(980601) FILEDATELONG(19980601) FILETIME(160340) RECFM(????)
RECLEN(00000) RECDLM(C) UNIQUEID(.9566269) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);
RETURN(00000);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This file shows the commands from the message command file, basein.msg, along with their associated response records and return codes.

The AUTOSTART record indicates that Expedite Base/AIX started the Information Exchange session automatically. The 00000 return code in the RETURN record indicates that the session started successfully.

The RECEIVED record and the 00000 return code in the RETURN record following it indicate that Expedite Base/AIX received one file in *order* in the *mycode* directory. Because the file has a CDH, Expedite Base/AIX uses the original file name of *order* instead of code.fil and places the file in the specified directory, */home/mycode*.

The second RECEIVE command is not followed by any RECEIVED records. Instead, it is followed by a return record with a 00000 return code. This means that the communication completed successfully, but there were no system error messages to receive.

The AUTOEND record indicates that Expedite Base/AIX ended Information Exchange session automatically. The 00000 return code in the RETURN indicates that the session ended successfully. The 00000 return code in the SESSIONEND record indicates that all the commands completed successfully.

ATTENTION: Workstation viruses are commonly passed from one workstation to another when you exchange executable files and then run them. You should exercise caution when receiving files using original file names, in order to protect your workstation from viruses. If you use the ORIGFILE parameter, be sure you know what file you are receiving and that it comes from a trusted source.

Chapter 6

Sending and receiving EDI data

You can use Expedite Base/AIX to send and receive data formatted for electronic data interchange (EDI). Expedite Base/AIX provides a single set of commands for all EDI data transmission—SENDEDI and RECEIVEEDI. The following sections explain how these commands work.

For information on sending and receiving non-EDI data, see Chapter 5, “Sending and receiving files.”

For information on sending and receiving compressed EDI data,see Appendix E, “Using data compression.”

Understanding how the network sends EDI data

Chapter 5, “Sending and receiving files,” describes how Information Exchange uses a two-part address to deliver mail to a trading partner's mailbox. The address consists of the account and user ID. The same chapter also discusses how the destination address is specified in the SEND command.

Information Exchange handles EDI data differently. When you send an EDI file, the data in the file contains the destination address.

- In X12 data, the destination is stored in the ISA segment.
- In UCS data, the destination is stored in the BG segment.
- In EDIFACT data, the destination is stored in the UNB segment.
- In UN/TDI data, the destination is stored in the STX segment.

Information Exchange must have the account and user ID of the destination mailbox in order to deliver the file. However, EDI standards allow you to specify addresses using different conventions. For example, you can specify a Dun and Bradstreet (DUNS) number for the address. You can also use phone numbers. Expedite Base/AIX and Information Exchange allow you to continue to use these kinds of addressing conventions. However, information must be provided so that the various addresses can be converted to the account and user ID that Information Exchange needs to deliver the file. see “Resolving EDI destinations” on page 87 for additional information.

Understanding how Expedite Base/AIX sends EDI data

The SENDEDI command is used to send data formatted in X12, UCS, EDIFACT, and UN/TDI to Information Exchange. You do not need to specify the destination address in the SENDEDI command because Expedite Base/AIX can get the address from the data. For information about the structure of the SENDEDI command, see Chapter 8, “Using Expedite Base/AIX message commands.”

A group of EDI transactions with a single destination address is an EDI envelope. An EDI *envelope* consists of the EDI header, the data in the EDI transactions, and the EDI trailer. The EDI header contains the destination address for the data within the envelope. The format of the headers, data, and trailers differs depending on what type of EDI data is sent. See “Using EDI envelopes” on page 86 for more details about the different types of EDI envelopes.

Expedite Base/AIX can transmit multiple EDI envelopes with different addresses contained in a single file with a single SENDEDI command. It can also transmit multiple types of EDI data from a single file. You can combine X12, UCS, EDIFACT, and UN/TDI data in one file and transmit it to multiple Information Exchange destinations with one SENDEDI command.

Using the SENDEDI command, you can send data to Information Exchange without specifying an Information Exchange destination as part of the command. SENDEDI can match EDI destinations contained in the EDI data to Information Exchange destinations.

SENDEDI determines where to send an EDI envelope by examining the envelope header. The envelope definition (the type of EDI data you are transmitting) determines the location of the destination within an EDI envelope. Expedite Base/AIX must read the envelope header to extract the destination address. Expedite Base/AIX converts that address to a valid Information Exchange account and user ID mailbox address. “Using EDI envelopes” discusses the EDI envelopes. “Resolving EDI destinations” on page 87 explains how Expedite Base/AIX converts the EDI address.

Using EDI envelopes

When you send EDI transactions, you can group the EDI transactions for a single destination within a single envelope. The EDI envelope definitions for each EDI data type are described below:

This EDI data type:	Uses this EDI envelope definition:
X12	Data between and including the ISA and IEA segments.
EDIFACT	Data between and including the UNA (or UNB) and UNZ segments.
UN/TDI	Data between and including the SCH (or STX) and END segments.
UCS	Data between and including the BG and EG segments.

The type of EDI data you are transmitting determines the location of the destination within an EDI header. Expedite Base/AIX locates the EDI destination as follows:

This EDI data type:	Contains the EDI destination in this segment:
X12	ISA. SENDEDI takes the actual EDI destination from the interchange receiver ID element (ISA08). It takes the EDI qualifier from the interchange ID qualifier element (ISA07).
EDIFACT	UNB. SENDEDI takes the destination from data element 0010 in composite data element S003 (Interchange Recipient). It takes the EDI qualifier from the data element 0007 in composite data element S003 (Interchange recipient).
UN/TDI	STX. SENDEDI takes the destination from the first subelement of the UNTO element (the recipient code address). If it does not find the recipient code, it uses the second subelement of the UNTO element (the recipient clear address) as the actual Information Exchange account and user ID.
UCS	BG. SENDEDI takes the destination from the application receiver's code (BG04) element.

Resolving EDI destinations

Before sending EDI data, you need to understand how EDI destinations are converted to Information Exchange addresses.

Each Information Exchange mailbox is identified by a unique address. When you send data to Information Exchange, you must provide information so that Information Exchange can determine the correct destination mailbox address. Information Exchange and the Expedite products understand three different forms of addresses:

1. Account and user ID

   Information Exchange must have the account and user ID in order to deliver the data to the proper mailbox.

2. Alias table and alias name

   Information Exchange uses tables to convert the alias table and alias name combination that you provide to the corresponding account and user ID.

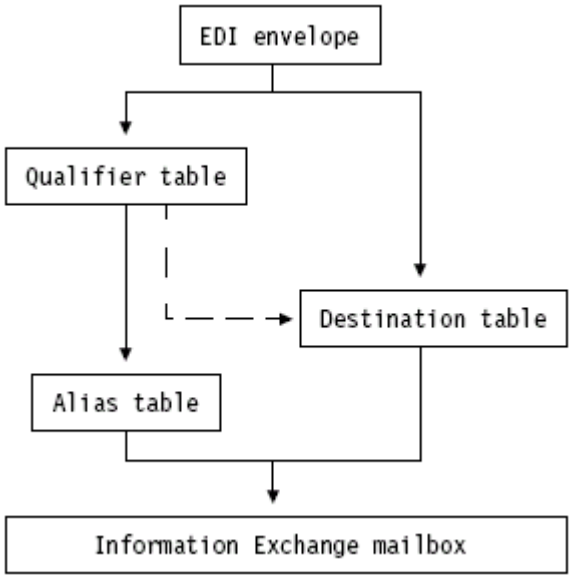
3. List

   When you send to a list, you should have defined a list of accounts and user IDs or alias tables and alias names.

The destination address in the EDI data can be specified in terms of an Information Exchange account and user ID. This is the simplest scenario, because Expedite Base/AIX does not have to convert the address. "Bypassing tables" on page 88 discusses this scenario and provides examples.

However, EDI standards define sets of rules for specifying destination addresses. When you send EDI data to Information Exchange using a destination address other than an Information Exchange account and user ID, you must provide additional information so that Expedite Base/AIX and Information Exchange can convert an EDI address to an address that Information Exchange understands. You provide this information using several tables. An overview of these tables is provided in the following flowchart. A discussion of how these tables work is provided in the following sections.

This flowchart illustrates how the SENDEDI command locates EDI destinations in most cases.



SENDEDI can use three tables to determine Information Exchange destination from the receiver ID specified in the EDI data: qualifier, destination, and alias tables.

Bypassing tables

If you need to send only EDIFACT, X12, or UN/TDI data to an Information Exchange destination, and you specify an Information Exchange destination in the EDI header, you can bypass the tables and send the EDI data directly to an Information Exchange destination.

To send EDIFACT or X12 data to an Information Exchange destination contained within the EDI data, follow these steps:

1. Place ZZ in the receiver ID qualifier of the EDI header.
2. Use the Information Exchange account and user ID as the actual EDI destination. For X12 data, you should separate the account and user ID by at least one blank. Otherwise, Expedite Base/AIX will use the first seven characters for the account and the last eight characters for the user ID. For EDIFACT data, you must separate the account and user ID by a period (.), slash (/), or blank.

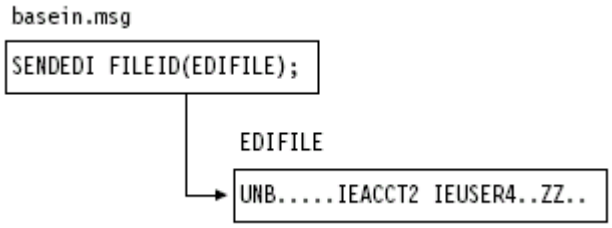
3. Use the SENDEDI command to send the file containing your EDI data.

NOTE: When you use a ZZ qualifier, Expedite Base/AIX tries to resolve the destination by searching the tables. When it does not locate the destination in the tables, or the tables do not exist, Expedite Base/AIX sends the data to the specified Information Exchange account and user ID. If you do not want Expedite Base/AIX to refer to the tables first, you can use a blank qualifier instead of a ZZ qualifier in the EDI header for X12 data. For EDIFACT data, a blank qualifier is treated the same way as any other qualifier and does not result in the tables being bypassed.

To send UN/TDI data to an Information Exchange destination contained within EDI data, follow these steps:

1. Do not specify the recipient code (UNTO:1).
2. Place the Information Exchange account and user ID in the recipient clear address (UNTO:2).
3. Use the SENDEDI command to send the file containing your EDI data.

The following example shows how Expedite Base/AIX sends an EDIFACT file to a trading partner with an EDI destination specified as an Information Exchange account and user ID of *ieacct2*, *ieuser4*.



Expedite Base/AIX sends the data in this example to account *ieacct2* and user ID *ieuser4*.

- You can use intersystem addressing when using the SENDEDI command to transmit EDIFACT or UN/TDI data. To do this, you place an Information Exchange address in the EDIFACT or UN/TDI header, specifying the appropriate identifying information in the following order:

- System ID (not required if the sender and receiver are using the same system)
- Account ID
- User ID

All of the IDs must be separated by one of the following:

- Period (.)
- Slash (/)
- One or more spaces

For EDIFACT data, SENDEDI splits the receiver code, which is data element 0010 in composite data element S003 (Interchange Recipient), into the system, account, and user ID.

For UN/TDI data, SENDEDI splits the recipient clear code (UNTO:2) into the system, account, and user ID.

If you are sending UCS data, you cannot bypass the use of tables. You must have one of the following in order to send UCS data:

- A ttable01.tbl file that specifies an Information Exchange account and user ID for the UCS destination address.
- A qualtbl.tbl file that identifies an Information Exchange alias table. The alias table must have an entry with the UCS destination address and the associated Information Exchange account and user ID.
- A qualtbl.tbl file that identifies a destination table to be used to translate the UCS destination address to an Information Exchange account and user ID.

When SENDEDI cannot find a destination or qualifier table, SENDEDI determines the Information Exchange destination for each EDI data type as follows:

If the EDI data is:	SENDEDI determines Information Exchange destination as follows:
X12 and the qualifier is ZZ or blank	The SENDEDI command splits the receiver ID into an account and user ID. Expedite Base/AIX will look for a blank character as the separator between account and user ID. Otherwise, it uses the first seven characters for the account and the last eight characters for the user ID.
X12 and the qualifier is not ZZ or blank	This is an error, and Expedite Base/AIX does not send the data.
EDIFACT and the qualifier, which is the data element 0007 in the composite data element S003 (Interchange Recipient), is ZZ	SENDEDI splits the receiver code, which is data element 0010 in composite data element S003 (Interchange Recipient), into the system, account, and user ID. The system, account, and user ID are separated by a period (.), slash (/), or by one or more blank spaces. The system ID is optional if the sender and receiver are using the same system.
EDIFACT and the qualifier, which is the data element 0007 in the composite data element S003 (Interchange Recipient), is not ZZ	This is an error, and Expedite Base/AIX does not send the data.
UN/TDI and the recipient code (UNTO:1) is not specified	SENDEDI splits the recipient clear code (UNTO:2) into the system, account, and user ID. The system, account, and user ID are separated by a period (.), slash (/), or by one or more blank spaces. The system ID is optional if the sender and receiver are using the same system.
UN/TDI and UNTO:1 was specified	This is an error, and Expedite Base/AIX does not send the data.
UCS	This is an error, and Expedite Base/AIX does not send the data.

Using EDI destination tables

If you do not specify Information Exchange destinations in the EDI header, you must first create an EDI destination table so that the EDI destination can be converted to an Information Exchange address.

Think of an EDI destination table as a list of EDI destinations paired with Information Exchange destinations. “Understanding the EDI destination table entry format” on page 100 provides details about how to build the EDI destination tables. The SENDEDI command resolves destinations by searching for an EDI destination and then using the corresponding Information Exchange destination as the actual address for an envelope.

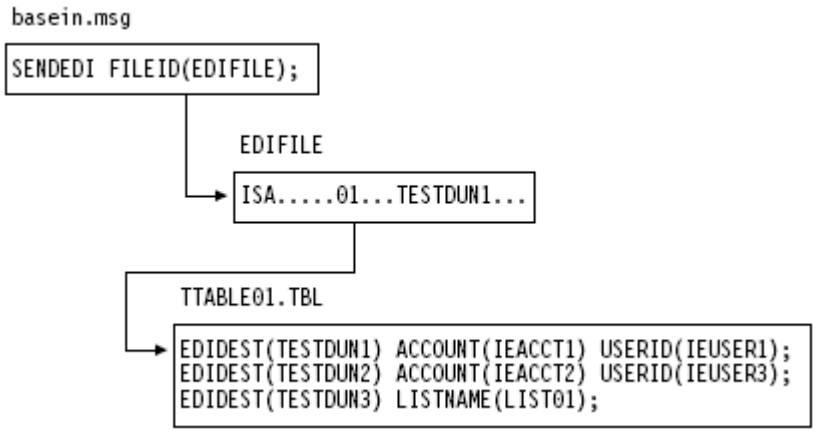
EDI destination tables have the following default naming convention:

This EDI data type:	Defaults to this EDI destination table:
X12	ttablexx.tbl where *xx* is the 2-character ID qualifier. A blank qualifier always defaults to account and user ID. SENDEDI never treats //TTABLExx as a match if *xx* is blank.
EDIFACT	ttablexx.tbl where *xx* is the first 2 characters of the ID qualifier. A blank qualifier defaults to TTABLE_ _.
UN/TDI	ieuntdi.tbl
UCS	ttable01.tbl

To send EDI data to a destination you define in an EDI destination table, follow these steps:

1. Build an EDI destination table containing your EDI destination and the corresponding Information Exchange destination. This table has the file name ttablexx.tbl where xx is the receiver ID qualifier in the EDI header.
2. Use the SENDEDI command to send the file containing your EDI destination.

The following example shows the tables Expedite Base/AIX uses to send an X12 file to a trading partner with an EDI destination of testdun1 and a qualifier of 01. This example shows how to use an EDI destination table without a qualifier table. “Using EDI qualifier tables” describes how to use a qualifier table.

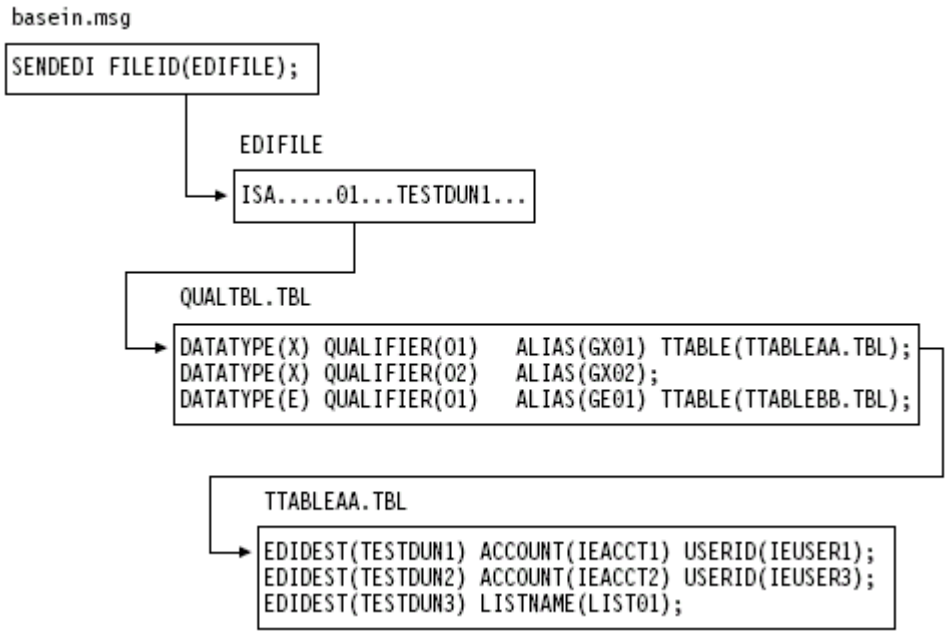


In this example, the receiver ID is *testdun1* and the receiver ID qualifier is *01*. An EDI qualifier table is not used. Expedite Base/AIX searches the EDI destination table ttable01.tbl for the receiver ID *testdun1* and converts the address to Information Exchange account *ieacct1* in Information Exchange user ID *ieuser1*. The table ttable01.tbl is the default destination table for X12 data using a *01* receiver ID qualifier.

Using EDI qualifier tables

"Using EDI destination tables" on page 91 describes how Expedite Base/AIX EDI destination tables are used to convert from an EDI destination to an Information Exchange address destination.

EDI qualifier tables tell Expedite Base/AIX which EDI destination table to use for this conversion. If you do not provide an EDI qualifier table, Expedite Base/AIX uses the default naming convention specified in "Using EDI destination tables" to determine which EDI destination table to use. If you find that the default naming convention is unsatisfactory, you can override the default destination table file names using the EDI qualifier table. The EDI qualifier table specifies a list of EDI destination tables which should be used for specific types of EDI data. "Creating tables for destination resolution" on page 99 for details about how to build an EDI qualifier table.



In this example, X12 data is sent to receiver ID *testdun1* with receiver ID qualifier *01*. An EDI qualifier table is used. Expedite Base/AIX searches the EDI qualifier table for a data type of *x* and qualifier 01. This entry indicates that ttableaa.tbl should be used to convert the EDI destination to a proper Information Exchange address. Expedite Base/AIX proceeds to search ttableaa.tbl for the receiver ID testdun1 and converts the address in Information Exchange account *ieacctl* to Information Exchange user ID *ieuser1.*

If the EDI destination cannot be resolved using the local TTABLExx.TBL, the EDI destination is passed to Information Exchange for resolution via the alias table specified in QUALTBL.TBL.

Using centralized Information Exchange alias tables

You may find it time consuming to maintain EDI destination tables in multiple locations. With the SENDEDI command, you can use centralized Information Exchange *alias tables*. These permanent tables reside within Information Exchange and convert EDI destinations into Information Exchange destinations.

You can make them available to all Information Exchange users (*global alias tables*), members of a particular account (*organization alias tables*), or a single user (*private alias tables*). The EDI qualifier table and the EDI destination table determine which, if any, of these alias tables Expedite Base/AIX should use.

NOTES: You can create and maintain alias tables in two ways:

1. Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*)
2. Using the DEFINEALIAS command ("DEFINEALIAS command" on page 178)

To send EDI data to a destination defined in an Information Exchange centralized alias table, follow these steps:

1. Add the target EDI destination to an Information Exchange centralized alias table.
2. Build an EDI qualifier table that contains the name of the Information Exchange centralized alias table and specifies the EDI data type.

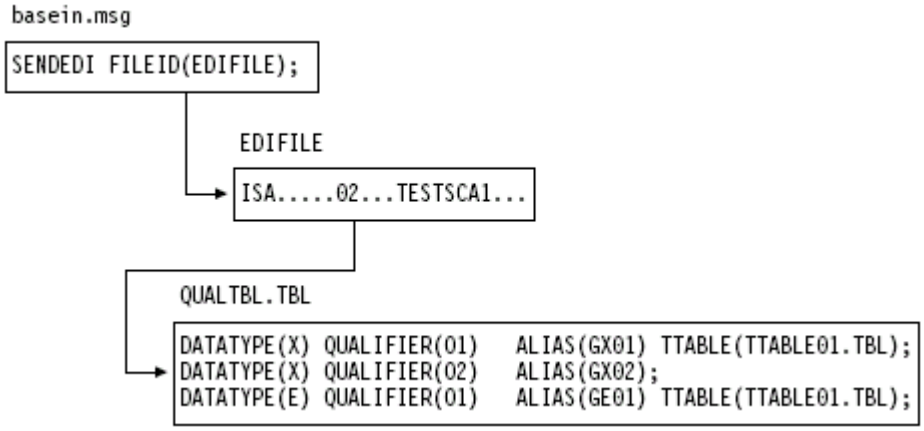
   NOTE: Expedite Base/AIX contains a sample EDI qualifier table. This table defines standard, centralized alias tables for all types of EDI data. In some Information Exchange installations (in the United States), these standard, centralized alias tables already exist, and you can add your EDI destinations to these tables to resolve your destinations.

   If you use a centralized alias table, make sure Expedite Base/AIX cannot resolve your target EDI destination locally on the workstation.

3. Use the SENDEDI command to send the file containing your EDI data.

   The following example shows the tables Expedite Base/AIX uses to send an X12 file to a trading partner with an EDI destination of `testsca1` and a qualifier of `02`. This example does not include an EDI destination table.

Expedite Base/AIX sends the data in this example to alias name *testsca1* in Information Exchange alias table *GX02*. Information Exchange alias table *GX02* is used to resolve the alias to an Information Exchange account and user ID.



Using Information Exchange distribution lists

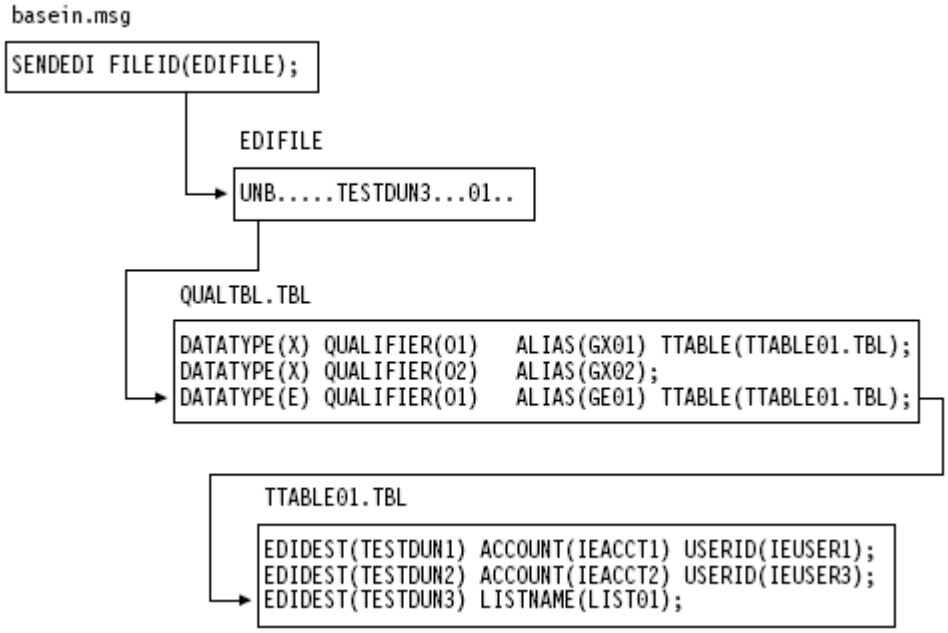
To send EDI data to an Information Exchange list, follow these steps:

1. Define an Information Exchange list.You can define a list using either Expedite Base/AIX or Information Exchange Administration Services.
2. Build an EDI destination table containing your EDI destination and the corresponding Information Exchange list name.
3. If you are not using a default EDI destination table file name, build an EDI qualifier table that contains the name of your EDI destination table and the type of EDI data it references.

   NOTE: If you use the default name for your EDI destination table, you do not have to build an EDI qualifier table.

4. Use the SENDEDI command to send the file containing your EDI data.

The following example shows the tables Expedite Base/AIX uses to send an EDIFACT file to an Information Exchange distribution list called `list01`.



Expedite Base/AIX sends the data in this example to Information Exchange list name `list01`.

To create a temporary list that only exists during your current Information Exchange session or a permanent list in Information Exchange, use the LIST command described in “LIST command” on page 188.

You can also use Information Exchange Administration Services to create a permanent list. For more information, see the *Information Exchange Administration Services User's Guide*.

Specifying Information Exchange control fields

You can specify Information Exchange control fields by using parameters within the SENDEDI command. *Control fields* allow you to assign information to the files that you send, and which can be used by you and your trading partner. Following are the Information Exchange control fields you can use:

- Message name (MSGNAME) an alphanumeric identifier for your file.
- Message sequence number (MSGSEQNO) a numeric identifier for your file.
- User class (CLASS) an alphanumeric identifier for your file. The user class can be used by your trading partner when receiving the file.

Providing a message name (MSGNAME)

When you use the SENDEDI command and provide a MSGNAME parameter, SENDEDI uses this value for Information Exchange MSGNAME. If you do not provide a MSGNAME parameter, the SENDEDI command generates Information Exchange MSGNAME based on the EDI data type. The following table describes how the SENDEDI command generates the MSGNAME.

This EDI data type:	Generates this MSGNAME:
EDIFACT data	Expedite Base/AIX takes the MSGNAME from the data element 0020 (Interchange Control Reference) of the EDI data. If the element exceeds eight bytes in length, Expedite Base/AIX uses the first eight bytes. If the element is fewer than eight bytes in length, Expedite Base/AIX places it in the MSGNAME field, left-justified and padded with blanks.
UN/TDI data	Expedite Base/AIX takes the MSGNAME from the sender's reference field (SNRF) of the EDI data. If the SNRF exceeds eight bytes in length, Expedite Base/AIX uses the first eight bytes. If the SNRF is fewer than eight bytes in length, Expedite Base/AIX places it in the MSGNAME field, left-justified and padded with blanks.
X12 data	Expedite Base/AIX takes the MSGNAME from the last eight bytes of the interchange control number of the X12 data.
UCS data	Expedite Base/AIX takes the MSGNAME from the interchange control number. Because the UCS interchange control number has a maximum length of five bytes, Expedite Base/AIX places the interchange control number in the MSGNAME field, left-justified and padded with blanks.

Providing a message sequence number (MSGSEQNO)

When you use the SENDEDI command and provide a MSGSEQNO parameter, SENDEDI uses this value for the Information Exchange message sequence number. If you do not provide the MSGSEQNO parameter, the SENDEDI command generates an Information Exchange MSGSEQNO in the following manner for all EDI data types.

The SENDEDI command counts each EDI envelope it transmits from a single data file. SENDEDI formats MSGSEQNO as a series of numeric characters ranging from 00001 to 99999. It assigns and places the count of each EDI envelope in the MSGSEQNO of the corresponding Information Exchange messages. For example, three EDI envelopes SENDEDI sent from a single file would have the following MSGSEQNO values:

- 00001 for the first EDI envelope in the file
- 00002 for the second EDI envelope in the file
- 00003 for the last EDI envelope in the file

When the MSGSEQNO reaches 99999, it automatically resets to 00001. Also, the MSGSEQNO counter resets to 00001 each time you use the SENDEDI command for a new file of EDI envelopes.

Providing a user class (CLASS)

If you do not provide a CLASS parameter in the SENDEDI command, SENDEDI generates a parameter value based on the EDI data type.

EDIFACT and UN/TDI data

For EDIFACT and UN/TDI data, Expedite Base/AIX takes the CLASS parameter from the application reference field (APRF) of the EDI data. The APRF field is in data element 0026 in the UNB for EDIFACT messages and in the APRF element of the STX for UN/TDI messages. If the APRF exceeds eight bytes in length, Expedite Base/AIX uses the first eight bytes. If the APRF is fewer than eight bytes in length, Expedite Base/AIX places the APRF in the CLASS parameter, left-justified and padded with blanks. If the APRF is not present, Expedite Base/AIX sets CLASS as follows:

This EDI data type:	Defaults to this CLASS parameter:
EDIFACT	#EE
UN/TDI	#EU

X12 and UCS data

For X12 and UCS data, if you do not specify the CLASS parameter, Expedite Base/AIX sets CLASS as follows:

This EDI data type:	Defaults to this CLASS parameter:
EDIFACT	#EE
UN/TDI	#EU

Inserting blanks following EDI segments

If you insert blanks at the end of EDI segments when preparing EDI information, Expedite Base/AIX does not consider the blanks part of the EDI data. The SENDEDI command accepts EDI data with blanks after the segment terminators, but it removes these blanks before transmitting the data to Information Exchange.

Using SENDEDI response records

The SENT record keeps track of the EDI envelopes SENDEDI sent. For a description of the format of this record, see “SENT record” on page 257.

The NOTSENT record provides a record of the EDI envelopes not sent by the SENDEDI command due to a destination verification failure. NOTSENT records are only provided when you specify VERIFY(C) or VERIFY(G). For a description of the format of this record, “NOTSENT record” on page 247.

NOTE: You can use the SENT records in the message response file (baseout.msg) to determine which EDI envelopes were sent if the SENDEDI command does not complete successfully, and you can use the NOTSENT records to determine which envelopes were not sent.

Receiving EDI data

You use the RECEIVEEDI command to receive EDI data from Information Exchange. You can receive multiple EDI envelopes containing different types of data with a single RECEIVEEDI command.

The RECEIVEEDI command is similar to the RECEIVE command, but it includes the ability to reformat received data based on EDI segment delimiters. The EDIOPT parameter in the RECEIVEEDI command controls this function. For more information on the format of this command, see “RECEIVEEDI command” on page 207.

It is recommended that you receive EDI data using the RECEIVEEDI command. Also, it is possible to receive and process EDI data using the RECEIVE command as long as your trading partner sends you EDI data with a CDH and you use the AUTOEDI parameter in the RECEIVE command. This way, the RECEIVE command can recognize the EDI format of the data and format it accordingly. See Chapter 8, “Using Expedite Base/AIX message commands,’’ for additional information about the RECEIVE command. Expedite Base/AIX always prepares a CDH when sending data. Any Information Exchange interface before Release 3.0 does not support the CDH.

If the system sending the EDI data does not support the CDH, there is no way to guarantee that the data you receive is EDI. However, you can make arrangements with your trading partner to help ensure that you receive EDI data. For example, you and your trading partner can agree that all EDI data is sent with a user class of `edidata`. All non-EDI data must have a different user class. That way, your RECEIVEEDI command can receive from user class `edidata` and the data is EDI. The following example shows a RECEIVEEDI command using the CLASS parameter to receive all files with the class `edidata` from the mailbox.

```
receiveedi fileid(editest) class(edidata);
```

NOTE: This method does not guarantee that the data in the file is EDI. For example, your trading partner might mistakenly send a non-EDI file to your mailbox with a user class of `edidata`. This method only improves the chances that the data is really EDI.

If the person sending you data uses the default user classes in the SENDEDI command, you can use the wild card receive feature of Information Exchange to simplify receipt. For example, by specifying **#E?** as the user class in the RECEIVEEDI command, you can ask Information Exchange to return only files that have a user class beginning with `#E`. This includes all files sent with the default EDI user classes.

If your trading partner uses a version of an Expedite product that supports the CDH, there is a better method of ensuring the data you receive is EDI. You can use the EDIONLY parameter in the RECEIVEEDI command to receive the data marked in the CDH as EDI data. If your trading partner sent the data using the SENDEDI command, then the DFORMAT field in the CDH indicates that the file is EDI.

The following example shows a RECEIVEEDI command using the EDIONLY parameter to receive all the files in the mailbox that are marked in the CDH as EDI data:

```
receiveedi fileid(edidata) edionly(y);
```

If the CDH indicates the data is not EDI, the file will not be received.

This method is especially useful if you intend to use a translator to translate from EDI standard format to proprietary format. Some translators run into problems if they try to translate data that is not really in EDI format. The EDIONLY parameter on the RECEIVEEDI command can help avoid such problems.

If there is no CDH, RECEIVEEDI attempts to process the files as EDI data. If this is not possible, the program writes the data without reformatting.

NOTE: The SENDEDI command places a single EDI envelope in an Information Exchange file. The RECEIVEEDI command expects each EDI envelope to be contained in a separate Information Exchange file. If you put multiple EDI envelopes in a single file and send them using the SEND (not the SENDEDI) command, the entire file will be sent to the Information Exchange account and user ID specified in the SEND command regardless of the destinations specified in the EDI envelopes within the file.

Creating tables for destination resolution

The following sections provide information on the format of the EDI qualifier table and EDI destination table. When you are sending data, these tables enable Expedite Base/AIX to resolve destinations. Use these formats to create your tables.

Understanding the EDI qualifier table entry format

The file qualtbl.tbl contains the EDI qualifier table. Each entry in this table indicates an EDI destination table, a centralized alias table, or both for a given EDI qualifier or EDI data type. The format for an EDI qualifier table entry is:

```
datatype(data type) qualifier(EDI qualifier) ttable(ttable ID)
alias(alias);
```

datatype

Indicates the EDI data type. Use one of the following values:

blank	For all EDI data types. This entry matches any supported EDI data type. This is the default.
x	For X12 data.
c	For UCS data.
e	For EDIFACT data.
u	For UN/TDI data.

qualifier

Indicates the EDI qualifier. If the qualifier parameter is blank, this entry matches any EDI qualifier. Use 1 to 4 alphanumeric characters. The default is blank.

ttable

Indicates the name of the EDI destination table. If you specify the TTABLE parameter, SENDEDI uses that table to try to resolve the Information Exchange destination. If you do not specify a destination table, SENDEDI does not use a table to match EDI data and resolve the destination. Use a standard Expedite Base/AIX file name, 1- to 12-alphanumeric characters.

alias

Indicates the name of the centralized alias table. If you specify this parameter and SENDEDI does not find the destination in the EDI destination table, SENDEDI uses this alias table with the EDI receiver ID as the alias name.

blank	No centralized alias table. This is the default
g*xxx*	Global alias table, where xxx identifies a 1- to 3-character table
o*xxx*	Organizational alias table, where xxx identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where xxx identifies a 1- to3-character table name.
NOTE:	You can include comments in the qualifier table. See “Understanding command syntax” on page 26 for more information on comment rules.

The following is an example of a qualifier table entry.

```
# Destination table for southeast trading partners. datatype(x)
qualifier(01) ttable(TTABLE010TBL) alias(gx01);
```

Understanding the EDI destination table entry format

Use the TTABLE parameter in the EDI qualifier to specify the name of your EDI destination table. Each entry in this table indicates the Information Exchange destination associated with a given EDI receiver ID. The format of an entry is:

```
edidest(EDI destination)
alias(alias) aliasname(alias name);
```

or

```
sysid(system ID) account(account) userid(user ID);
```

or

```
account(account) userid(user ID);
```

or

```
listname(list name);
```

edidest

Indicates the EDI receiver ID. If this parameter matches the receiver ID from the EDI data, Expedite Base/AIX sends the file to the Information Exchange destination using the parameters you select. Use 1 to 35 alphanumeric characters.

alias

Indicates the alias table type and table name.

blank	No alias table name. This is the default
g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to3-character table name.

aliasname

Indicates an alias name defined in the alias table. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of a single-destination user ID. You need this only if the account and user ID you specify reside on another Information Exchange system. Use this parameter only with the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the Information Exchange account name of a single-destination user ID. Use 1- to 8-alphanumeric characters.

userid

Indicates the Information Exchange destination user ID. Use 1 to 8 alphanumeric characters.

listname

Indicates the name of a previously defined list of account and user IDs. Use 1 to 8 alphanumeric characters.

NOTE: You can include comments in the destination table. See “Understanding command syntax” on page 26 for more information on comment rules.

The following is an example of an EDI destination table entry.

```
# Information Exchange address for XYZ Company in Baltimore
edidest(testdun1) account(ieacct1) userid(ieuser1);
```

Recovery levels

The default recovery level for a session with Information Exchange is checkpoint. This is usually the best way to exchange data when using a dial line or when transferring a large number of files or a large amount of data.

Checkpoint-level recovery ensures that if the line is disconnected or noisy, Expedite Base/AIX can pick up the data transfer at the last checkpoint rather than start the transfer over from the beginning.

When using a leased line, you can select session-level recovery, because it is very unlikely the line will go down during the data transfer. When you use session-level recovery, there is less processing required to recover after a failed session, as it is an all-or-nothing data transfer method.

When using a dial line, you can also select session-level recovery if you are transmitting only a small amount of data. If the line is disconnected, simply start the session over from the beginning.

Using session-level recovery, however, has its disadvantages. With checkpoint-level recovery, Information Exchange commits the data sent or received when a checkpoint takes place during the session. The commit processing is done in short intervals throughout the session. For a session-level recovery session, this processing is all done at the end of the session.s If the communication line gets disconnected, Expedite Base/AIX must start from the beginning the next time a connection is made.

Using checkpoint-level, file-level, and user-initiated recovery

The most important job of your application interface is processing the Expedite Base/AIX response file. To work with Expedite Base/AIX, you need to understand how it recovers from an error during an Information Exchange session.

Checkpoint-level, file-level, and user-initiated recovery are Information Exchange methods that Expedite Base/AIX can use to recover data at specific checkpoints. When you use session-level recovery and an error occurs (see “Using session-level recovery” on page 116), Expedite Base/AIX must retransmit all data for the session. If you are sending large amounts of data, retransmission can take several hours. But when you select checkpoint-level, file-level, or user-initiated recovery, Expedite Base/AIX can recover data more efficiently.

The following table shows when Expedite Base/AIX takes checkpoints for each of these recovery methods:

Checkpoint-level recovery:	File-level recovery:	User-initiated recovery:
• after sending the number of bytes you specify in the COMMITDATA parameter of the TRANSMIT command (default is 37000 bytes) • at the end of each SEND, SENDEDI, or PUTMEMBER command, if the next command is not a SEND, SENDEDI, or PUTMEMBER command • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command	• after each file sent as a result of a SEND, SENDEDI, or PUTMEMBER command • after each file is received • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command	• after each COMMIT command, unless there is nothing to commit • at the end of each session, even if you have not specified a COMMIT command • while receiving data for a RECEIVE or RECEIVEEDI command, if the file was sent with checkpoints • at the end of each RECEIVE or RECEIVEEDI command

Checkpoint-level recovery is the default in AIX. This is usually the best way to exchange data when using a dial line or when transferring a large number of files or a large amount of data. Checkpoint-level recovery ensures that if the line is disconnected or noisy, Expedite can pick up the data transfer at the last checkpoint without starting the session over.

To request one of the other recovery methods, use the recovery parameter on the TRANSMIT command with one of the following values:

s	Session-level recovery
f	File-level recovery
u	User-initiated recovery

The processes for using checkpoint-level, file-level, and user-initiated recovery are very similar. Expedite Base/AIX uses the same work files for these recovery methods. Considerations for restarting after an error and resetting the Expedite Base/AIX session, described later in this chapter, also apply to all three recovery methods.

ATTENTION: You should not run multiple sessions for the same account and user ID from different machines. If you start an Information Exchange session using checkpoint-level, file-level, or user-initiated recovery while another Information Exchange session with the same account and user ID is running, Information Exchange ends the first session and continues the second session. The results in the first session depend on whether a checkpoint ended successfully.

- If a checkpoint ended successfully, Information Exchange delivers any data sent prior to the checkpoint and deletes any data from the mailbox that was received prior to the checkpoint.
- If a checkpoint did not end successfully, Information Exchange does not deliver any data and does not delete any received data from the mailbox. This means that data received in the first session may be received again in error.

In either case, you may get an error when you restart the first session. You should not run multiple sessions for the same account and user ID from different machines.

Understanding post-session processing for checkpoint-level, file-level, and user-initiated recovery

The following sections describe what to do when a session ends in error. Post-session processing activities for checkpoint-level recovery include:

- Restarting a session
- Resetting a session
- Checking the SENT, NOTSENT, and RECEIVED response records
- Checking return codes

Restarting a session

It is important that your application process the responses in the response file correctly. Therefore, you need to understand the difference between session *restart* and session *reset*.

You initiate a session restart when an Information Exchange session ends in error and you want Expedite Base/AIX to resume the session at the last checkpoint. Before you restart a session, correct any problems that caused the previous session to end in error. Do not alter the basein.msg command file, the baseout.msg response file, or the session.fil session file. You can correct a syntax error in the command file to allow the session to continue, but do not add or delete lines from it.

If the session completes abnormally but you have return records for all of your commands with 0 return codes, then restart the session to allow it to complete normally. Do not reset the session, or lost or duplicate data may occur.

ATTENTION: Do not erase or alter the session file (session.fil) at any time. If session.fil is altered or erased during a restart, data may be duplicated or lost. If session.fil does not exist, Expedite Base/AIX starts processing at the beginning of the basein.msg file. When this happens, any data sent before the last successful checkpoint in the previous session is sent again. In addition, any data that was received during the previous session may be overwritten or erased. Files and messages received before the last successful checkpoint in the previous session are no longer available from Information Exchange. Issuing the same RECEIVEEDI command may cause data already received, but not processed, to be overwritten by the results of the most recent RECEIVEEDI command.

If you have altered or erased your session.fil file, you should review the contents of baseout.msg to see which commands processed successfully. Remove these commands from basein.msg and reset the session by entering **iebase -r**.

Changing files on restart

You can change some files before you restart a session. The following list indicates which files you cannot change and which you can change with limitations. You can change any files that are not in the list.

- **Message command file, basein.msg**

  You cannot change any commands or parameters in basein.msg that have been echoed to baseout.msg. This includes characters such as blanks and carriage returns that occur between commands and parameters. You can change commands and parameters in basein.msg that Expedite Base/AIX has written to tempout.msg or that are not shown in tempout.msg or baseout.msg.

- **Message response file, baseout.msg**

  Do not change baseout.msg.

- **Profile command file, basein.pro**

  You can change basein.pro if you want to modify a profile value. However, do not modify the COMMTYPE parameter of the TRANSMIT command.

- **Profile response file, baseout.pro**

  There is no need to change baseout.pro since Expedite Base/AIX creates a new baseout.pro when you restart.

- **Profile information file, iebase.pro**

  Never change iebase.pro. If you erase it, Expedite Base/AIX must create another iebase.pro using the commands in basein.pro. This means you must provide the required profile information again, such as your account, user ID, and password, using profile commands. If you erase iebase.pro before restarting and basein.pro is present, Expedite Base/AIX can still restart.

- **Session work file, session.fil**

  Never change this file before restarting. Expedite Base/AIX uses it to restart the session.

- **EDI work file, ediwork.fil**

  Never change this file before restarting. Expedite Base/AIX uses it to restart the session.

- **Receive name file, rcvfiles.fil**

  Never change this file before restarting. Expedite Base/AIX uses this control file to track files it receives during the session.

- **Receive offset file, rcvofset.fil**

  Never change this file before restarting. It enables data to be appended correctly after restart.

- **EDI qualifier and destination tables**

  Change these files only if there is a syntax error in the file. If you change the destination used for a SENDEDI command while the command is in progress, unpredictable results can occur.

- **Files being sent or received**

  You can modify the EDI file you are sending with the SENDEDI command to correct a problem with the EDI format in data that was not sent, or to correct a destination that was found to be invalid if you specified VERIFY(Y) on the SENDEDI command.

  Do not modify files you are sending with the SEND command. Do not modify files you are receiving. Changes may cause unpredictable data to be sent or received.

Reviewing an example of session restart

This example illustrates a session in which some commands did not process successfully, and you must restart the session.

The files you are sending are test1.x12, test2.x12, and test3.x12 from the current directory. On the SENDEDI command for test2.x12, the FILEID parameter is misspelled. The following example shows the basein.msg file.

```
SENDEDI FILEID(test1.x12);
SENDEDI FILID(test2.x12);
SENDEDI FILEID(test3.x12);
```

When the session is complete, the return code in the SESSIONEND record is 15030. This return code indicates an invalid parameter in the command file. Post-processing of the response file shows you what the error is but does not show you which command is in error. The following example shows the baseout.msg file.

```
SESSIONEND(15030)
ERRDESC(Invalid parameter found.)
ERRTEXT(EXPLANATION: You specified an invalid parameter in the
command)
ERRTEXT(file.)
ERRTEXT(USER RESPONSE: Check the message response file, baseout.msg,)
ERRTEXT(profile response file, baseout.pro, or response work file,)
ERRTEXT(tempout.msg, to determine which command produced the error.)
ERRTEXT(Correct the appropriate command file and retry the program.);
```

To determine which command is in error, examine the temporary response file (tempout.msg). It shows the error is in the SENDEDI command for the second file, test2.x12. Correct the error by correctly specifying the FILEID parameter and restart Expedite Base/AIX. Processing begins at the SENDEDI command for test2.x12. The following example shows the tempout.msg file.

```
AUTOSTART SESSIONKEY(DF883762);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DF883762)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(test1.x12);
SENT UNIQUEID(39416704) LENGTH(2262) ACCOUNT(ACCT) USERID(USER1)
EDITYPE(X12)
DESTINATION(ACCT USER1) QUALIFIER(ZZ) CONTROLNUM(000022223)
CLASS(#E2)
MSGNAME(00022223) MSGSEQNO(00001);
RETURN(00000);
SENDEDI FILID(test2.x12)
RETURN(15030)
ERRDESC(Invalid parameter found.)
ERRTEXT(EXPLANATION: You specified an invalid parameter in the
command file.)
ERRTEXT(USER RESPONSE: Check the message response file, baseout.msg,)
ERRTEXT(profile response file, baseout.pro, or response work file,)
ERRTEXT(tempout.msg, to determine which command produced the error.)
ERRTEXT(Correct the appropriate command file and retry the program.);
```

Resetting a session

You initiate a session reset when the session ends in error and you do not want Expedite Base/AIX to continue the Information Exchange session. When you reset a session, Expedite Base/AIX acts as if a session were never active and begins processing at the beginning of the command file. There is some risk in using the same command file when you reset a session. Any data sent before the last successful checkpoint in the previous session is sent again. Files and messages received before the last successful checkpoint in the previous session are no longer available from Information Exchange. You must process the data from these received files and messages before you use the command file again. Otherwise, the results of the most recent RECEIVEEDI commands may overwrite the data in the received files.

When an existing session ends because of an error, partially processing the response file can help you determine the commands that completed before the session ended. To partially process the response file, process the commands in baseout.msg that completed successfully, build a new basein.msg file with the commands that were not processed, and start Expedite Base/AIX by typing iebase r on the command line.

NOTE: If the session file, session.fil, does not exist, then there were no checkpoints taken during the previous session and Expedite Base/AIX begins processing at the start of the command file. Therefore, partial processing of the response file is not necessary.

Reviewing examples of session reset

The following examples illustrate when a session reset is necessary. In these examples, the return code is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. This is usually caused by accessing Information Exchange using the same account or user ID on different machines at the same time.

Example 1

In this example you are sending files and the session ends in error.

The following example shows the basein.msg file.

```
SENDEDI FILEID(order1.x12);
SENDEDI FILEID(order2.x12);
SENDEDI FILEID(order3.x12);
SENDEDI FILEID(order4.x12);
SENDEDI FILEID(order5.x12);
SENDEDI FILEID(order6.x12);
SENDEDI FILEID(order7.x12);
SENDEDI FILEID(order8.x12);
SENDEDI FILEID(order9.x12);
SENDEDI FILEID(order10.x12);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows which commands completed successfully. The RETURN(00000) record indicates that the first five SENDEDI commands completed successfully and the files were sent to Information Exchange. However, the remaining SENDEDI commands were not processed. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(47793LKU);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(47793LKU)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(order1.x12);
SENT UNIQUEID(43495778) LENGTH(6572);
RETURN(00000);
SENDEDI FILEID(order2.x12);
SENT UNIQUEID(74469581) LENGTH(5342);
RETURN(00000);
SENDEDI FILEID(order3.x12);
SENT UNIQUEID(67856334) LENGTH(3456);
```

```
RETURN(00000);
SENDEDI FILEID(order4.x12);
SENT UNIQUEID(19600628) LENGTH(9865);
RETURN(00000);
SENDEDI FILEID(order5.x12);
SENT UNIQUEID(37941045) LENGTH(9745);
RETURN(00000);
SENDEDI FILEID(order6.x12);
SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery,)
ERRTEXT(the checkpoint numbers for the send or receive side of the)
ERRTEXT(session do not match the values Information Exchange
recorded.)
ERRTEXT(Your session file, session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the -r)
ERRTEXT(command line parameter on the IEBASE command.)
ERRTEXT(Also, make sure there is not another user using this user
ID.)
ERRTEXT(If the problem persists, contact the Help Desk. Before)
ERRTEXT(starting the next session, review the message response)
ERRTEXT(file, baseout.msg, to see which commands were processed)
ERRTEXT(successfully. Remove these commands from the message)
ERRTEXT(command file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the -r)
ERRTEXT(command line parameter you will no longer be able to
continue)
ERRTEXT(the previous session. Failure to modify the message command)
ERRTEXT(file, basein.msg, before resetting the session may result in)
ERRTEXT(some data being lost or duplicated.);
```

You need to edit the command file and delete the first five SENDEDI commands. Then reset the session by entering **iebase -r** on the command line.

NOTE: If you do not remove the first five SENDEDI commands before you reset the session, Expedite Base/AIX sends these files again.

Example 2

In this example you are receiving files and the session ends in error.

You are receiving four files from your Information Exchange mailbox. Each file has a different user class. The following example shows the basein.msg file.

```
RECEIVEEDI FILEID(rcv1.x12) CLASS(TEST1);
RECEIVEEDI FILEID(rcv2.x12) CLASS(TEST2);
RECEIVEEDI FILEID(rcv3.x12) CLASS(TEST3);
RECEIVEEDI FILEID(rcv4.x12) CLASS(TEST4);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows which commands completed successfully. The RETURN(00000) record indicates that the first three RECEIVEEDI commands completed successfully. However, the last RECEIVEEDI command was not processed. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(26637987);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(26637987)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVEEDI FILEID(rcv1.x12) CLASS(TEST1);
RECEIVED ACCOUNT(ACT1) USERID(USER01) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(249000065) CLASS(TEST1)
CHARGE(5) LENGTH(1429) FILEID(rcv1.x12)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001955)
SESSIONKEY(26637987) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(O450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(snd1.X12) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv2.X12) CLASS(TEST2);
RECEIVED ACCOUNT(ACT2) USERID(USER02) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(338359825) CLASS(TEST2)
CHARGE(5) LENGTH(7113) FILEID(rcv2.x12)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001956)
SESSIONKEY(26637987) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(O450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(snd2.X12) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(38576390) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);
RETURN(00000);

RECEIVE FILEID(rcv3.x12) CLASS(TEST3);
RECEIVED ACCOUNT(ACT3) USERID(USER.3) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(385872839) CLASS(TEST3)
CHARGE(5) LENGTH(8219) FILEID(rcv3.x12)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001957)
SESSIONKEY(26637987) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(O450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(snd3.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(15837294) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);
RETURN(00000);

SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery,)
ERRTEXT(the checkpoint numbers for the send or receive side of the)
ERRTEXT(session do not match the values Information Exchange
recorded.)
ERRTEXT(Your session file, session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the r)
ERRTEXT(command line parameter on the IEBASE command.)
ERRTEXT(Also, make sure there is not another user using this user
ID.)
```

```
ERRTEXT(If the problem persists, contact the Help Desk.)
ERRTEXT(Before starting the next session, review the message
response)
ERRTEXT(file, baseout.msg, to see which commands were processed)
ERRTEXT(successfully. Remove these commands from the message command)
ERRTEXT(file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the r)
ERRTEXT(command line parameter you will no longer be able to
continue)
ERRTEXT(the previous session. Failure to modify the message command)
ERRTEXT(file, basein.msg, before resetting the session may result in)
ERRTEXT(some data being lost or duplicated.);
```

You need to edit the command file and delete the first three RECEIVEEDI commands. Then reset the session by entering **iebase -r** on the command line.

ATTENTION: If you specified OVERWRITE(Y) on the SESSION command or omitted the OVERWRITE parameter from the SESSION command in basein.pro, and you reset this session without modifying the command file, the first three files will be deleted and you will lose that data.

If you specified OVERWRITE(N), then new data received is appended to the existing files with the same name, which may make the data difficult to use.

Example 3

In this example you are receiving multiple files with one RECEIVEEDI command and the session ends in error.

You are receiving six files from your Information Exchange mailbox using the MULTFILES parameter of the RECEIVEEDI command. You want to receive the first file in rcvx12 and each subsequent file in a new file named by numbering the file extensions starting with 002. The following example shows the basein.msg file.

```
RECEIVEEDI FILEID(rcvx12) CLASS(TEST4) MULTFILES(Y);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows that Expedite Base/AIX received three files from your Information Exchange mailbox and stored them in rcvx12, rcvx12.002, and rcvx12.003. The absence of RETURN(00000) before the SESSIONEND record indicates that more files are in your mailbox. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(CCK839FH);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(CCK839FH)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVEEDI FILEID(rcvx12) CLASS(TEST4);
RECEIVED ACCOUNT(ACT1) USERID(USER01) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(385937203) CLASS(TEST4)
CHARGE(5) LENGTH(3503) FILEID(rcvx12)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001955)
SESSIONKEY(CCK839FH) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
```

```
SENDERFILE(snd1.X12) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

RECEIVED ACCOUNT(ACT2) USERID(USER02) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(395730298) CLASS(TEST4)
CHARGE(5) LENGTH(5320) FILEID(rcvx120002)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001956)
SESSIONKEY(CCK839FH) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(snd3.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

RECEIVED ACCOUNT(ACT3) USERID(USER.3) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(305930298) CLASS(TEST4)
CHARGE(5) FILEID(rcv3.fil)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001957)
SESSIONKEY(CCK839FH) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(snd2.fil) SENDERLOC(/home/expedite) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
RECDLM(C) UNIQUEID(43495778) SYSTYPE(12) SYSVER(1)
TRANSLATE(TESTDTBL);

SESSIONEND(24100)
ERRDESC(Session and Information Exchange checkpoints do not match.)
ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery,)
ERRTEXT(the checkpoint numbers for the send or receive side of the)
ERRTEXT(session do not match the values Information Exchange
recorded.)

ERRTEXT(Your session file, session.fil, may be damaged.)
ERRTEXT(USER RESPONSE: Reset the session using the r)
ERRTEXT(command line parameter on the IEBASE command.)
ERRTEXT(Also, make sure there is not another user using this user
ID.)
ERRTEXT(If the problem persists, contact the Help Desk.)
ERRTEXT(Before starting the next session, review the message
response)
ERRTEXT(file, baseout.msg, to see which commands were processed)
ERRTEXT(successfully. Remove these commands from the message command)
ERRTEXT(file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the r)
ERRTEXT(command line parameter you will no longer be able to
continue)
ERRTEXT(the previous session. Failure to modify the message command)
ERRTEXT(file, basein.msg, before resetting the session may result in)
ERRTEXT(some data being lost or duplicated.);
```

You need to edit the command file and delete the first three RECEIVEEDI commands. Then reset the session by entering **iebase -r** on the command line.

ATTENTION: If you specified OVERWRITE(Y) on the SESSION command or omitted the OVERWRITE parameter from the SESSION command in basein.pro, and you reset this session without modifying the command file, the first three files will be deleted and you will lose that data.

If you specified OVERWRITE(N), then new data received is appended to the existing files with the same name, which may make the data difficult to use. Therefore, you need to consider one of the following actions before you reset the session:

- Process the data in rcvx12, rcvx12.002, and rcvx13.003; for example, store the data in a database. Then erase the files or specify OVERWRITE(Y) in the SESSION command.
- Rename the files rcvx12, rcvx12.002, and rcvx12.003 so that Expedite Base/AIX does not overwrite them or append data to them when it receives the remaining files.
- Change the name of the FILEID parameter of the RECEIVEEDI command to something other than rcvx12 so that Expedite Base/AIX uses new file names when it receives the remaining files.

Example 4

In this example you are sending six EDI envelopes within a single file, using a single SENDEDI command, when the session ends in error. The following example shows the basein.msg file.

```
SENDEDI FILEID(edidata.fil);
```

When the session ends in error, the return code in the SESSIONEND record is 24100. This return code indicates that the session and Information Exchange checkpoints do not match. Post processing of the response file shows which envelopes were sent successfully. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(37748987);
  STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(37748987)
  IEVERSION(04)
  IERELEASE(07);
  RETURN(00000);
  SENDEDI FILEID(edidata.fil);
  SENT UNIQUEID(87503678) LENGTH(3984);
  SENT UNIQUEID(34987349) LENGTH(14.3);
  SENT UNIQUEID(37803293) LENGTH(3202);
  SESSIONEND(24100)
  ERRDESC(Session and Information Exchange checkpoints do not match.)
  ERRTEXT(EXPLANATION: In a session using checkpoint-level recovery,)
  ERRTEXT(the checkpoint numbers for the send or receive side of the)
  ERRTEXT(session do not match the values Information Exchange
  recorded.)
  ERRTEXT(Your session file, session.fil, may be damaged.)
  ERRTEXT(USER RESPONSE: Reset the session using the r)
  ERRTEXT(command line parameter on the IEBASE command.)
  ERRTEXT(Also, make sure there is not another user using this user
  ID.)
  ERRTEXT(If the problem persists, contact the Help Desk.)
```

```
ERRTEXT(Before starting the next session, review the message
response)
ERRTEXT(file, baseout.msg, to see which commands were processed)
ERRTEXT(successfully. Remove these commands from the message command)
ERRTEXT(file, basein.msg, so they are not processed again.)
ERRTEXT(Warning: If you reset the session using the r)
ERRTEXT(command line parameter you will no longer be able to
continue)
ERRTEXT(the previous session. Failure to modify the message command)
ERRTEXT(file, basein.msg, before resetting the session may result in)
ERRTEXT(some data being lost or duplicated.);
```

The response file shows three SENT records, indicating three of the six EDI envelopes were sent to Information Exchange before the error occurred. You need to edit the EDI file, edidata.fil, to remove the first three EDI envelopes before you reset the session. Then reset the session by entering **iebase -r** on the command line.

NOTE: If you do not remove the first three EDI envelopes from edidata.fil before you reset the session, Expedite Base/AIX sends the first three envelopes again.

Checking the SENT, NOTSENT, and RECEIVED response records

You cannot determine what files Expedite Base/AIX sent or received by examining only the RETURN record. You must also examine the SENT, NOTSENT, and RECEIVED records.

SENT response records follow SENDEDI commands. Expedite Base/AIX writes a SENT record for each EDI envelope that is sent. If no SENT record exists, Expedite Base/AIX did not send any EDI envelope.

NOTSENT response records follow SENDEDI commands (with VERIFY(C) or VERIFY(G) specified) for each EDI envelope that could not be successfully sent due to a destination verification failure.

RECEIVED records follow RECEIVEEDI commands. A RECEIVED record is written for each file received from Information Exchange. Files that have RECEIVED records are no longer in your Information Exchange mailbox and you cannot receive them again. If no RECEIVED record exists for a file, Expedite Base/AIX did not receive it.

NOTE: You should look only at the records in baseout.msg. You should not rely on the records in tempout.msg, which contain information processed since the last checkpoint.

Checking return codes

When a session completes, Expedite Base/AIX provides two numeric codes that identify the activities it performed. The first code is a five-digit return code that Expedite Base/AIX displays in the SESSIONEND or RETURN response record in baseout.msg. These return codes are grouped into categories, such as message command syntax errors and profile command syntax errors.

The second code is a three-digit exit code that Expedite Base/AIX returns to the operating system. You can use the operating system exit codes and Expedite Base/AIX return codes to decide what actions, if any, to take in the next session. For descriptions of the exit codes and return codes, see Chapter Appendix A, “Expedite Base/AIX messages and codes.”

The decision to restart or reset a session is based on the return code value in the SESSIONEND response record in baseout.msg. The following return codes are grouped into four categories:

- The return code is 00000, session completed normally. The operating system exit code is 0.

  If the return code is 0, you may still need to process the responses in the response file. For example, if you receive multiple files from your mailbox to your system using separate file names, you need to know the names of the files and how many you received. If you receive files from your mailbox to your system using the original file names, you may need to check the file names indicated in the CDH. If you request system messages, such as error messages and acknowledgments, you may need to check this information.

  Information Exchange places error messages in your mailbox in a fixed format message with account *SYSTEM* and user ID *ERRMSG*. To receive these messages, you must issue a RECEIVE command for this account and user ID and process the information in the response file. This is also how you receive acknowledgments.

- The return codes are 1600016999, or 2800028020, session ended but was incomplete. The Expedite Base/AIX error-level code is 112.

Errors 1600016999 indicate a problem trying to send the information to the specified destination. Error 28000 indicates that a warning was generated.

Error 28010 indicates that one or more of the commands in the command file was not processed because of an error. The error number is shown in the RETURN response record immediately following the command that caused the error. The errors are described in the ERRDESC and ERRTEXT records in the response file immediately following the SESSIONEND response record.

Error 28020 indicates an error occurred during the disconnect process. The number of the error that occurred is shown in a WARNING record following the SESSIONEND record. The WARNING record is followed by ERRDESC and ERRTEXT records describing the error.

The information Expedite Base/AIX displays in the ERRDESC and ERRTEXT records is similar to the information in Appendix A, "Expedite Base/AIX messages and codes."

- The return code is one of the following and indicates that a session restart is necessary.

Return code:	Category description:	Exit code:
11863, 26996	Wait and try again later	110
02000–04999	Message command syntax errors	111
05000–09999	Profile command syntax errors	104
11000–11999	network errors	111
12000–12199	Modem script syntax errors	111, 114
12200–12399	Display status script syntax errors	111
13000–13999	Communication device driver errors	111
14000–15999	Parser errors	111
17000–18999	EDI parsing, send, or receive errors	111
19000–19999	TCP/IP communication errors	111
20000–23999	General environment errors	111
24000–24699	Session start and end errors	113
25000	PF key exit	111
26000–26999	Internal communications errors	111, 114
27000–27099	Old message.fil errors	111, 114
28100–28200	Miscellaneous command processing errors	111
29960–29990	SNA LU 6.2 communications errors	111, 114
29998	Modem command processor error	
31400	Unexpected program interrupt error	

- The return code indicates that a session reset is necessary. The operating system exit code is 113 or 114. If the exit code is 114, you may be able to resolve the problem simply by redialing. If this does not resolve the problem, you must reset the session.

  - Session start and end error return codes 24000–24699
  - Unexpected errors and commit error return codes 31000–31339

In addition, the Help Desk may suggest that the session be reset for other reasons. Your application interface should offer an easy way to reset the session and partially process the response file.

Using session-level recovery

When using a leased line, you can select session-level recovery, because it is very unlikely the line will go down during the data transfer. When you use session-level recovery, there is less processing required to recover after a failed session, as it is an all-or-nothing data transfer method.

When using a dial line, you can also select session-level recovery if you are transmitting only a small amount of data. If the line is disconnected, simply start the session over from the beginning.

When you use session-level recovery to transmit data and an error occurs, Expedite Base/AIX stops transmission and produces a SESSIONEND record with a return code. You can check this return code to determine the cause of the error and correct the problem.

With session-level recovery, if data transmission stops, you must send and receive all files again. Although it takes time to retransmit large amounts of data, there are advantages to using session-level recovery. For example, you do not need to be concerned with determining which files Expedite Base/AIX sent successfully and which files you need to resend.

If you use multiple START and END commands in basein.msg, there are certain precautions you must take. See “Using multiple START and END commands with session-level recovery” on page 122 for more information.

If the session completes abnormally but you have return records for all of your commands with 0 return codes, then restart the session to allow it to complete normally. Do not reset the session, or lost or duplicate data may occur.

ATTENTION: If you start an Information Exchange session using session-level recovery while another Information Exchange session with the same account and user ID is running, Information Exchange ends the first session and starts the second session. Information Exchange does not deliver data sent in the first session and does not delete received data from the mailbox. This means that data received in the first session may be received again in error. The results when the first session ends are unpredictable.

Using session-level recovery, however, has disadvantages. With checkpoint-level recovery, Information Exchange commits the data sent or received when a checkpoint takes place during the session. The commit processing is done in short intervals throughout the session. For a session-level recovery session, this processing is all done at the end of the session. So when Expedite Base/AIX sends the SESSIONEND command, Information Exchange does not return the SESSIONEND response until the commit processing is completed. If a large number of files are transferred during the session, the processing takes longer, especially during prime business hours.

During the processing, it is possible for the line to be disconnected because of a time-out. Expedite Base/AIX ends the session with a 29999 return code, “Session end response failure.” In this case, it is not clear whether or not the session ended successfully.

There are several ways to determine if the session was successful or not. For example, the CHECK parameter on the START command indicates to Expedite Base/AIX that you only want to check the status of the previous session. If you specify CHECK on the START command, do not specify any other commands except the END command in the input file. See “START command” on page 229 for more information.

Expedite Base/AIX also provides information about the previous session on the STARTED record. This record is written to the output file as a result of a START or AUTOSTART command. See “STARTED record” on page 261 for more information.

After a session fails with 29999, follow these steps:

1. Specify AUTOSTART(N), AUTOEND(N), and RECOVERY(S) on your TRANSMIT command in the Expedite profile.

2. Create an input file containing START and END records; an example follows:

```
START CHECK(Y);
  END;
```

NOTE: Do not specify any other commands in the input file if you specify CHECK(Y) on the session start command.

3. Run Expedite Base/AIX. No data is transferred in the above example, and you are not charged for this inquiry.

4. Examine the output file to check the LASTSESS parameter value on the STARTED record.

   LASTSESS(0) Indicates the previous session was successful. No further recovery is required.

   LASTSESS(1) Indicates the previous session was not successful.

If Expedite Base/AIX reported the 29999 SESSIONEND return code for a session-level recovery session, you should switch to checkpoint-level, file-level, or user-initiated level recovery for future sessions with a similar number of commands.

If you were only receiving files from your Information Exchange mailbox, make sure all data was received by verifying that it is no longer in your mailbox. You can do this by viewing your mailbox with Information Exchange Administration Services or by running a QUERY command to get a list of AVAILABLE response records for each file in your mailbox. If the data is still in your mailbox, switch from session-level recovery to checkpoint-level recovery and run the session again to receive the data.

If you were sending files, you must check your audit trail to see if the files were sent. You can do this by using Information Exchange Administration Services, or by using Expedite Base/AIX to request an audit be sent to your mailbox. Refer to Chapter 8, “Using Expedite Base/AIX message commands,” and “Using audit trails” on page 264 for more information. If the files were not sent, switch to checkpoint-level recovery and run the session again.

When a large number of files is being sent or received, session-level recovery is not recommended. Customers have experienced time-out problems when sending or receiving more than 700 files (the size of the files does not matter). Use checkpoint, user-initiated, or file-level recovery instead.

Understanding post-session processing for session-level recovery

The following sections describe what to do when a session ends with an error condition. Post-session processing activities for session-level recovery include:

- Processing the response file records
- Checking return codes

Processing the baseout.msg response file records

When you transmit data, Expedite Base/AIX processes your message command file and creates a message response file, baseout.msg. The response records in baseout.msg are free-format records. Their syntax is the same as that defined for commands. Response records always start at the beginning of a line. However, parameters in response records may occur in any position and in any order. In addition, Expedite Base/AIX may not show all parameters in a response record. When examining response records, consider the following:

- Assume a default value if you do not get a response record parameter you are expecting. This is not an error.
- Truncate the parameter if a response record parameter is longer than you expect.
- Be prepared to handle parameters that are split across records. Splitting can occur if the parameter is longer than the record length of your response file.

NOTE: You should be prepared for the possibility of new parameters on existing response records and entirely new response records that may be provided in the future.

Checking return codes

To ensure that Expedite Base/AIX finished processing the message command file, check the return code in the SESSIONEND record. Detailed return code descriptions are included in Appendix A, “Expedite Base/AIX messages and codes.”

The following is a list of the SESSIONEND return codes:

- The return code is 00000, session completed normally. The operating system exit code is 0.

  If the return code is 0, Expedite Base/AIX processed all commands and all command RETURN records contain 0 return codes.

- The return codes are 28000–28020, session ended but was incomplete. The operating system exit code is 112.

  Error 28000 indicates that a warning was generated. Error 28010 indicates that one or more of the commands in the command file was not processed because of a command file error. If the return code is 28020, all commands in the command file were processed, and an error occurred during the disconnect process. If the problem was with the command file, correct the command that caused the error and run the program again. If the problem was in the disconnect process, the session completed successfully but you should correct the problem so that future sessions disconnect from the network properly.

The error number is shown in the RETURN response record immediately following the command that caused the error. The errors are described in ERRDESC and ERRTEXT records files immediately following the SESSIONEND response record. If the error is caused by a problem with a specific command, such as a syntax error, the command in error is followed by a RETURN record with the same return code as the SESSIONEND record.

- The return code is not 00000, 28010, or 28020. The operating system exit code is 111 or 113.

  This error indicates that Expedite Base/AIX did not finish processing the command file, the Information Exchange session was not successful, and none of the file transfer requests in the message command file completed. Expedite Base/AIX did not place any mail in your trading partner's Information Exchange mailbox or remove any from your mailbox. The SESSIONEND record can include an error description to help you find the problem. A command RETURN record can contain the same code and description. If baseout.msg does not contain a RETURN record, check baseout.pro for the error.

  NOTE: While Expedite Base/AIX is receiving data from Information Exchange, it saves the data in files on your workstation. Even if a session does not complete successfully, Expedite Base/AIX may have received and saved data during the session. However, since you are using session-level recovery, both Information Exchange and Expedite Base/AIX ignore the files that were sent and received during the unsuccessful session. The next time a session is started, all of the data will be sent and received again.

Requests other than file transfer may complete even if the Expedite Base/AIX SESSIONEND is not 28010, 28020, or 00000. These requests include:

- ARCHIVEMOVE
- AUDIT
- CANCEL
- DEFINEALIAS
- GETMEMBER
- LIST
- LISTLIBRARIES
- LISTMEMBERS
- PURGE

If these requests are followed by a RETURN(00000) record in baseout.msg or tempout.msg, they completed and you do not need to reissue them.

Reviewing examples of session-level recovery

The following examples illustrate session-level recovery.

Example 1

This example illustrates a session that ends in error when you are sending files.

You are sending five EDI files. The following example shows the basein.msg file.

```
SENDEDI FILEID(file1.x12);
SENDEDI FILEID(file2.x12);
SENDEDI FILEID(file3.x12);
SENDEDI FILEID(file4.x12);
SENDEDI FILEID(file5.x12);
```

When the session ends in error, the return code in the SESSIONEND record is 26805. This return code indicates that the carrier was lost during the send process. The two SENT records indicate that Expedite Base/AIX sent the first two files before the session ended. However, since you are using session-level recovery, Information Exchange will not deliver these files to the trading partner's mailbox until the session ends successfully. You must resend the files. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(3379DLK8);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3379DLK8)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(file1.x12);
SENT UNIQUEID(07580371) LENGTH(500);
RETURN(00000);
SENDEDI FILEID(file2.x12);
SENT UNIQUEID(78207881) LENGTH(300);
RETURN(00000);
SENDEDI FILEID(file3.x12);

SESSIONEND(26805)
ERRDESC(Lost carrier.)
ERRTEXT(EXPLANATION: The carrier was lost during the data
transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the program persists,)
ERRTEXT(contact the Help Desk.);
```

Because you are using session-level recovery, Information Exchange discards the files sent to it in the previous incomplete session. To send all the files, run the program again. When Expedite Base/AIX establishes a session, it processes all the commands in the command file again so you send all five files.

Example 2

This example illustrates a session that ends in error when you are receiving files.

You want to receive four files from your Information Exchange mailbox. Each file has a different user class. The following example shows the basein.msg file.

```
RECEIVEEDI FILEID(rcvx12) CLASS(TEST1);
RECEIVEEDI FILEID(rcv2x12) CLASS(TEST2);
RECEIVEEDI FILEID(rcv3x12) CLASS(TEST3);
RECEIVEEDI FILEID(rcv4x12) CLASS(TEST4);
```

When the session ends in error, the return code in the SESSIONEND record is 26996. This return code indicates that Expedite Base/AIX timed out while waiting for a response from the network. The three RECEIVED records indicate that you received the first three files before the session ended, but you did not receive all four files. The following example shows the baseout.msg file.

```
AUTOSTART SESSIONKEY(3363789H);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3363789H)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
RECEIVEEDI FILEID(rcvx12) CLASS(TEST1);
RECEIVED ACCOUNT(ACT1) USERID(USER01) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(398203520) CLASS(TEST1)
CHARGE(5) LENGTH(1429) FILEID(rcvx12)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001955)
SESSIONKEY(3363789H) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) RECEIVER(ACT 1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(000022228)
EDITYPE(X12) SENDERFILE(SND1.X12) SENDERLOC(/home/expedite)
FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
FILETIME(111414) RECFM(????) RECLEN(00000) RECDLM(C)
UNIQUEID(43495778)
SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

RECEIVEEDI FILEID(rcv2x12) CLASS(TEST2);
RECEIVED ACCOUNT(ACT2) USERID(USER02) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(183759283) CLASS(TEST2)
CHARGE(5) LENGTH(3423) FILEID(rcv2x12)
RECEIVER(ACT1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(003857620)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001955)
SESSIONKEY(3363789H) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) RECEIVER(ACT 1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(003857620)
EDITYPE(X12) SENDERFILE(SND2.FIL) SENDERLOC(/home/expedite)
FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
FILETIME(111414) RECFM(????) RECLEN(00000) RECDLM(C)
UNIQUEID(43495778)
SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
```

```
RECEIVEEDI FILEID(rcv3x12) CLASS(TEST3);
RECEIVED ACCOUNT(ACT3) USERID(USER.3) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(8274.9657) CLASS(TEST3)
CHARGE(5) LENGTH(2949) FILEID(rcv3x12)
RECEIVER(ACT1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(001837650)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(111717) MSGSEQO(001955)
SESSIONKEY(3363789H) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L) DATATYPE(A) RECEIVER(ACT 1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(001837650)
EDITYPE(X12) SENDERFILE(SND3.X12) SENDERLOC(/home/expedite)
FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(111414) RECFM(????) RECLEN(00000)
FILETIME(111414) RECFM(????) RECLEN(00000) RECDLM(C)
UNIQUEID(43495778)
SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

SESSIONEND(26996)
ERRDESC(Timedout while waiting for a response.)
ERRTEXT(EXPLANATION: DCL timedout while waiting for a response from)
ERRTEXT(the network gateway. This can occur if the line)
ERRTEXT(is dropped and the operating system does not return the)
ERRTEXT(lostcarrier condition to the program.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists,)
ERRTEXT(it may be that the Asynchronous Relay is down. Try the)
ERRTEXT(program again in about 3. minutes. If the problem still)
ERRTEXT(occurs, contact the Help Desk.);
```

Because you are using session-level recovery, Information Exchange ignores the three files it sent to you in the previous incomplete session. To receive all the files, run the program again. When Expedite Base/AIX establishes a session, it processes all the commands in the command file again so that you receive all four files.

NOTE: If you specified OVERWRITE(N) in the SESSION command in basein.pro and you run the program again without deleting the three files you originally received, Expedite Base/AIX appends the three files you receive the second time to the three files you received the first time. For more information, “SESSION command” on page 148.

Using multiple START and END commands with session-level recovery

It is important to note the difference between an Expedite Base/AIX session and an Information Exchange session. An Expedite Base/AIX session consists of all commands specified in basein.msg that are issued during a single network connection. An Information Exchange session consists of the commands issued between a START command and an END command in basein.msg. The Expedite Base/AIX session is the same as the Information Exchange session when there is only one START command and one END command in basein.msg. However, Expedite Base/AIX allows the user to start and end multiple Information Exchange sessions within a single Expedite Base/AIX connection.

If you use multiple START and END commands in basein.msg, you create an environment similar to that of checkpoint-level recovery. Each END command stops an Information Exchange session. Requests in each Information Exchange session complete even if a subsequent Information Exchange session ends in error. Only Information Exchange sessions that end in error require you to send or receive data again.

If you specify multiple START and END commands in basein.msg, and an error occurs before all commands in basein.msg have completed, you must take special measures to process your basein.msg and baseout.msg files before restarting. These measures are similar to those you must take when doing checkpoint level, file-level, or user-initiated recovery. That is, you must review the contents of baseout.msg to determine which of the Information Exchange sessions completed successfully, and which need to be run again. Commands in the sessions that were successful must be removed from basein.msg to avoid sending duplicate data or losing received data.

ATTENTION: Failure to remove commands for successfully completed Information Exchange sessions from basein.msg may result in duplicate or lost data in subsequent Expedite Base/AIX sessions. When using session-level recovery with multiple START and END commands, you must process your basein.msg and baseout.msg files similar to the way required for checkpoint-level, file-level, and user-initiated recovery. We recommend you use session-level recovery with a single START and END command to avoid the need to process these files after incomplete sessions. If you need to run multiple Information Exchange sessions within a single Expedite Base/AIX session, you may consider using checkpoint-level recovery instead of session-level recovery.

When an Information Exchange session has completed, Expedite Base/AIX will write two records to baseout.msg. The first is the END record, which is echoed from basein.msg. The second record Expedite Base/AIX will write is the RETURN record, which shows the return code for the END command. When you see the END and RETURN records in baseout.msg, all commands in that Information Exchange session have been completed. Before starting Expedite Base/AIX again, you should remove all commands processed successfully from basein.msg.

NOTE: When Expedite Base/AIX has completed all commands in basein.msg, it writes a return code for the Expedite Base/AIX session. The return code is specified in the SESSIONEND record in baseout.msg. There will only be one SESSIONEND record in baseout.msg, regardless of the number of Information Exchange sessions started and ended within basein.msg.

Reviewing examples using multiple Information Exchange sessions with session level recovery

Example 1

This example illustrates three Information Exchange sessions within a single Expedite Base/AIX session. The following shows the contents of basein.msg.

```
START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(33877898)
IEVERSION(04)
IERELEASE(07);
SENDEDI FILEID(file1x12);
END;
```

```
START;
RECEIVEEDI FILEID(rcvx12);
END;
START;
SENDEDI FILEID(file2x12);
END;
```

Following are the contents of baseout.msg when Expedite Base/AIX has completed processing.

```
START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(33877898)
IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(33877898);
SENDEDI FILEID(file1x12);
SENT UNIQUEID(73133557) LENGTH(500);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(SOVSHX25)
IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(SOVSHX25);
RECEIVEEDI FILEID(rcvx12);
RECEIVED ACCOUNT(ACCT) USERID(USER02) RECEIVER(ATAP DLAVMN)
RECVQUAL(22)
SENDER(ATAP PTRSND) SENDQUAL(22) CONTROLNUM(763857283) CLASS(TEST1)
CHARGE(5) LENGTH(4101)
RECEIVER(ACT1 DEPT01) RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(000013856)
FILEID(rcvx12) MSGDATE(980809) MSGDATELONG(19980809) MSGTIME(134011)
MSGSEQO(001988) SESSIONKEY(SOVSHX25) DELIMITED(N) SYSNAME(EB/AIX)
SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A) EDITYPE(X12)
SENDERFILE(SENDER.FIL) SENDERLOC(/home/expedite) FILEDATE(980412)
FILEDATELONG(19980412) FILETIME(120000) RECFM(????) RECLEN(0)
RECDLM(C)
UNIQUEID(73133557) SYSTYPE(15) SYSVER(4) TRANSLATE(IESTDTBL);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(44783KL9)
IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(44783KL9);
SENDEDI FILEID(file2x12);
SENT UNIQUEID(00959571) LENGTH(1335);
RETURN(00000);
END;
RETURN(00000);
SESSIONEND(00000);
```

Note that each of the END records is followed by a RETURN record. This means that each of the Information Exchange sessions completed. Further, the return code 00000 in the RETURN records indicates that each session completed successfully. Finally, the SESSIONEND record indicates the completion of the Expedite Base/AIX session.

Example 2

This example uses the same input file as Example 10 However, in this example, the output file indicates that a problem occurred during the connection. Following are the contents of baseout.msg when Expedite Base/AIX has completed processing.

```
START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(337874687)
IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONKEY(337874687);
SENDEDI FILEID(file1x12);
SENT UNIQUEID(73133557) LENGTH(500);
RETURN(00000);
END;
RETURN(00000);

START;
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(337898DH)
IEVERSION(04)
IERELEASE(07);
RETURN(00000) SESSIONEKEY(337898DH);
RECEIVEEDI FILEID(rcvx12);
SESSIONEND(26805)
ERRDESC(Lost carrier.)
ERRTEXT(EXPLANATION: The carrier was lost during data transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists,
contact)
ERRTEXT(the Help Desk.);
```

In this example, the first Information Exchange session completed successfully, as indicated by the END and RETURN(00000) records. However, the second session did not complete successfully. The baseout.msg file shows that there are no END or RETURN records associated with the second Information Exchange session. The RECEIVE command is, instead, followed by a SESSIONEND record indicating the end of the Expedite Base/AIX session. In addition, there are ERRDESC and ERRTEXT records with information about the problem.

Because you are using session-level recovery, Expedite Base/AIX will begin processing at the beginning of basein.msg the next time it is run. If no changes are made to basein.msg, the file that was successfully sent the first time will be sent again, resulting in duplicate data sent to Information Exchange. Therefore, before you restart Expedite Base/AIX, you should remove the commands associated with this Information Exchange session from basein.msg The new basein.msg should appear as follows:

```
START;
RECEIVEEDI FILEID(rcvx12);
END;
START;
SENDEDI FILEID(file2x12);
END;
```

Receiving multiple files

In the RECEIVEEDI command, you must specify the name of the file in which Expedite Base/AIX is to place the received file. If you have more than one file in your mailbox, you can receive the files from the mailbox in a single file or receive each file in a separate file.

When you receive multiple files from your mailbox in a single file, Expedite Base/AIX appends the files in the order it receives them. This is the default for receiving multiple files.

When you receive multiple files from your mailbox in separate files, specify the value y in the MULTFILES parameter. This tells Expedite Base/AIX to place the first file in the file you specified and place subsequent files in new files by numbering the file extensions starting with 002. For example, if you specify FILEID(test.msg) and three files are received, Expedite Base/AIX names the files as follows:

```
File 1 = TEST.MSG
File 2 = TEST.002
File 3 = TEST.003
```

Receiving specific files

Previous sections of this chapter have demonstrated that you can specify certain criteria on the RECEIVEEDI command to limit the files that you receive. For example, you can receive all files from a particular user, or all files with a particular user class.

You can use the RECEIVEEDI command to specify a date and time range for files you want to receive. Information Exchange checks the date and time the files were sent to you, and gives you those files that fall within your specified date and time range.

For example, suppose you wanted to receive only those files sent to you between noon and 6:00 p.m. on June 14, 1998. You would include the following on your RECEIVE command:

```
STARTDATE(980614) STARTTIME(120000) ENDDATE(980614) ENDTIME(180000)
TIMEZONE(L)
```

Expedite Base/AIX also allows you to receive a single, specific file even if other files in your mailbox are from the same sender or have the same user class. Each file in your mailbox has a unique message key that distinguishes the file from all others. You can issue a RECEIVEEDI command, using the MSGKEY parameter to specify the unique message key of the file you want to receive.

For example, suppose there were three files in your mailbox from the same user, with the same user class. The files were sent to your mailbox on three consecutive days. However, you are only interested in receiving the first file, which has a unique message key of 887A9DE0021FA9C236F8. Your RECEIVEEDI command might look as follows:

```
RECEIVEEDI FILEID(first.fil) MSGKEY(887A9DE0021FA9C236F8);
```

As a result of this command, Expedite Base/AIX receives only the file with this message key.

To find out what the message key is for a specific file, you can use Information Exchange Administration Services, or use the Expedite Base/AIX QUERY command in basein.msg. As a response to the QUERY command, Expedite Base/AIX provides information about each of the files in your mailbox, including the message key for each file. “Querying a mailbox” on page 266 provides more information about using the QUERY command. “RECEIVEEDI command” on page 840 provides information about the format of the RECEIVEEDI command.

SENDEDI and RECEIVEEDI file number limits

Information Exchange limits the number of files that can be sent and received between commits because of the processing requirements involved. The current limit of 1,000 files is an Information Exchange value that can be set differently in different Information Exchange installations. Contact your marketing representative to determine the maximum for Information Exchange installations outside the U.S.

There are also limitations on the number of files that can be sent to and received from Expedite Base/AIX, which depend upon the type of recovery you are using. This section discusses limitations which you should take into consideration for your installation.

User-level recovery

If you are using user-level recovery, you must not specify more than 1,000 SEND, SENDEDI, or PUTMEMBER commands without specifying a COMMIT command. Do not send more than 1,000 EDI envelopes within a single file because each envelope counts as a file.

Checkpoint-level recovery

If you use checkpoint-level recovery, Expedite Base/AIX will perform a COMMIT after sending or receiving the number of characters specified in the COMMITDATA parameter on the TRANSMIT command in basein.pro. The default value for this parameter is 141000. You must not attempt to send or receive more than 1,000 files whose combined size is less than the value in the COMMITDATA parameter. If this is the case, you can either lower the value of the COMMITDATA parameter or decrease the number of files being sent or received. This will allow a COMMIT to be performed before the 1,000 file limit is reached.

File-level recovery

If you use file-level recovery, there is no limitation to the number of files you can send and receive. This is because each file is committed as it is sent or received, and the maximum number of files between commits is 1. If you are sending or receiving many small files, you can get better performance using checkpoint-level recovery.

Session-level recovery

If you use session-level recovery and you try to send more files than the limit, you will receive an error from Information Exchange, which Expedite Base/AIX reports to you as SESSIONEND return code 31360. In this case, break your input file into multiple input files and run Expedite Base/AIX for each of the input files.

If you have more than 1,000 files in your mailbox that match your receive request, Information Exchange stops sending them when the 1,000 file limit is reached. If you have more files in your mailbox, Expedite Base/AIX returns a 28171 value in the RETURN parameter after the last file was received. The SESSIONEND code is 28010, indicating the session completed successfully but not all commands were processed. The data you already received is no longer in your Information Exchange mailbox, but there are still additional files in the mailbox which match your receive request. Before using Expedite Base/AIX again to receive the remaining files, be sure to process the data already received in order to ensure that the files are not overwritten during the next session with Information Exchange.

See “Using session-level recovery” on page 116 for more information about processing received files when using session-level recovery.

Integrating with an EDI translator

The purpose of Electronic Data Interchange (EDI) is to provide common standards for the exchange of data so that the output from one computer application can be the input to a trading partner's application. However, few computer programs are written that provide data formatted according to an EDI standard. Data is usually stored in a proprietary format to meet the needs of the business.

This is where EDI translators become important. A business can purchase or build an EDI translator that can read the proprietary data and produce properly formatted EDI data that is sometimes referred to as documents. The trading partner would also use a translator that can receive the EDI data and reformat it to its proprietary format. The following example illustrates this:

A business application can be designed so that when a file is designated to be sent to a trading partner, it is first read into the translator so that a properly formatted EDI document is produced. This document can then be stored on the workstation with other EDI documents until the user is ready to send it to the trading partners.

Another option might be to translate all of the proprietary data to be sent to trading partners at once and store them all in a single file. Remember that an EDI document starts with a predefined header and ends with a trailer. Multiple documents can be sent to Information Exchange as a single file. Expedite Base/AIX separates the individual documents so that they are sent to the proper destination mailboxes.

The application programmer must make a decision about when the translator is invoked to translate from proprietary format to EDI documents. The application programmer must also build the proper Expedite Base/AIX input file so that the EDI data can be properly sent.

Learning more about sending and receiving EDI data

The following examples illustrate how you can use SENDEDI and RECEIVEEDI commands to send and receive EDI data.

Example 1

The XYZ supply store has an inventory and ordering system. When the inventory of a particular product is low, the system automatically generates an electronic purchase order to order more. The XYZ company uses the ANSI X12 purchase order standard (the 850 transaction) when formatting its electronic purchase orders. An EDI translator is used to translate the data from its format in the inventory and ordering system to the 850 standard format. The translator also builds the proper levels of EDI envelopes, including the outermost ISAIEA envelope.

During the day, the system continues to create electronic purchase orders destined for different trading partners. All of the data is stored in a single file. At the end of the day, the system runs Expedite Base/AIX to send the data to Information Exchange. The following is the format for the input file:

Sample input file basein.msg

```
SENDEDI FILEID(orders.fil);
RECEIVEEDI FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

Sample output file baseout.msg

```
AUTOSTART SESSIONKEY(DDJ84987);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DDJ84987)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(orders.fil);
SENT UNIQUEID(51219934) LENGTH(791) ACCOUNT(AAAA) USERID(AAAA01)
EDITYPE(X12)
DESTINATION(AAAA AAAA01) QUALIFIER(ZZ) CONTROLNUM(000022223)
CLASS(#E2)
MSGNAME(00022223) MSGSEQNO(00001);
SENT UNIQUEID(13352145) LENGTH(856) ACCOUNT(BBBB) USERID(BBBB01)
EDITYPE(X12)
DESTINATION(BBBB BBBB01) QUALIFIER(ZZ) CONTROLNUM(000022224)
CLASS(#E2)
MSGNAME(00022224) MSGSEQNO(00002);
SENT UNIQUEID(66864241) LENGTH(543) ACCOUNT(CCCC) USERID(CCCC01)
EDITYPE(X12)
DESTINATION(CCCC CCCC01) QUALIFIER(ZZ) CONTROLNUM(000022225)
CLASS(#E2)
MSGNAME(00022225) MSGSEQNO(00003);
RETURN(00000);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

Results: In this example there were three purchase orders sent to Information Exchange. Each purchase order has an ISAIEA envelope and each went to a different trading partner. The first went to account *aaaa* user ID `aaaa01`, the second went to account `bbbb` user ID `bbbb01`, and the third went to account `cccc` user ID `cccc01`.

A CLASS parameter was not specified in the SENDEDI command. The result is that a default user class of `#E2` was assigned to these files. This is different from the SEND command where no specified user class results in a blank user class for the file.

System error messages were requested, but none were received.

Example 2

In this example there is a problem with the transmission from the XYZ supply store one night. This example shows what happens if the XYZ store uses session-level recovery. Session-level recovery means that Information Exchange doesn't put any data in your trading partners' mailboxes until there is a successful session end. For additional information, see “Using session-level recovery” on page 116. The following input file is the same as the input file in example 1.

Sample input file basein.msg

```
SENDEDI FILEID(orders.fil);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

Sample output files

baseout.msg

```
SESSIONEND(26805)
ERRDESC(Lost carrier.);
```

tempout.msg

```
AUTOSTART SESSIONKEY(C9938UJ6);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(C9938UJ6)
IEVERSION(04) IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(orders.fil);
SENT UNIQUEID(62323675) LENGTH(791) ACCOUNT(AAAA) USERID(AAAA01)
EDITYPE(X12)
DESTINATION(AAAA AAAA01) QUALIFIER(ZZ) CONTROLNUM(000022223)
CLASS(#E2) MSGNAME(00022223) MSGSEQNO(00001);
SENT UNIQUEID(35245245) LENGTH(856) ACCOUNT(BBBB) USERID(BBBB01)
EDITYPE(X12)
DESTINATION(BBBB BBBB01) QUALIFIER(ZZ) CONTROLNUM(000022224)
CLASS(#E2) MSGNAME(00022224) MSGSEQNO(00002);
RETURN(26805)
ERRDESC(Lost carrier.) ERRTEXT(EXPLANATION: The carrier was lost
during data transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists,)
ERRTEXT(contact the Help Desk.);
```

Results: In this example only two of the three purchase orders were sent to Information Exchange before the session ended unexpectedly due to a lost carrier. This is apparent because the output file tempout.msg shows two SENT records. Each purchase order has its own ISAIEA envelope and is destined for a different trading partner. Because the sender is using session-level recovery, Information Exchange discards the two purchase orders that it received since the session did not complete successfully. The next time Expedite Base/AIX has a session with Information Exchange, it starts sending from the beginning of basein.msg and resends all of the data previously sent. When the session ends successfully, Information Exchange puts the purchase orders in the three trading partners' mailboxes.

Example 3

In this example there is a problem with the transmission from the XYZ supply store one night and the XYZ store used checkpoint level-recovery. When checkpoint-level recovery is used, checkpoints are taken during the data transmission. If a session ends unexpectedly, when Expedite Base/AIX next reestablishes a connection, the data transmission continues at the previous checkpoint. This is in contrast to session-level recovery, where the data transmission starts at the beginning. See “Using checkpoint-level, file-level, and user-initiated recovery” on page 102 for additional information. The following input file is the same as the input file in example 1.

Sample input file basein.msg

```
SENDEDI FILEID(orders.fil);
RECEIVE FILEID(errors.fil) ACCOUNT(*SYSTEM*) USERID(*ERRMSG*);
```

Sample output file baseout.msg

```
AUTOSTART SESSIONKEY(3387JK7J);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(3387JK7J)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(orders.fil);
SENT UNIQUEID(62323675) LENGTH(791) ACCOUNT(AAAA) USERID(AAAA01)
EDITYPE(X12)
DESTINATION(AAAA AAAA01) QUALIFIER(ZZ) CONTROLNUM(000022223)
CLASS(#E2)
MSGNAME(00022223) MSGSEQNO(00001);
SENT UNIQUEID(35245245) LENGTH(856) ACCOUNT(BBBB) USERID(BBBB01)
EDITYPE(X12)
DESTINATION(BBBB BBBB01) QUALIFIER(ZZ) CONTROLNUM(000022224)
CLASS(#E2)
MSGNAME(00022224) MSGSEQNO(00002);
RETURN(26805)
ERRDESC(Lost carrier.)
ERRTEXT(EXPLANATION: The carrier was lost during data transmission.)
ERRTEXT(USER RESPONSE: Retry the program. If the problem persists,)
ERRTEXT(contact the Help Desk.);
```

Results: In this example only two of the three purchase orders were sent to Information Exchange before the session ended unexpectedly due to a lost carrier. This is apparent because the output file baseout.msg shows 2 SENT records. Each purchase order has its own ISAIEA envelope and is destined for a different trading partner. Since the sender is using checkpoint-level recovery, Information Exchange delivers the two purchase orders to the trading partners' mailboxes even though the session did not complete successfully. The next time Expedite Base/AIX has a session with Information Exchange, it starts sending from where it left off, which is the third ISA envelope in the file orders.fil.

Example 4

The ABC company trades EDI data with multiple trading partners. ABC uses an EDI translator to translate their system data to EDI standard data when they send documents and to translate from EDI standard data to their system data when they receive documents.

Each time the translator creates an electronic document, it stores the data in a separate file. All files are stored in the directory called EDI.

Different translators have different requirements for the data they work with. For example, ABC's translator will run into difficulties if it is expecting EDI standard data but receives non-EDI data instead. It also expects the received EDI standard data to be formatted with newlines at the end of each segment. ABC can structure its input file commands to handle these requirements.

Sample input file basein.msg

```
SENDEDI FILEID(order1.fil);
SENDEDI FILEID(order2.fil);
SENDEDI FILEID(order3.fil);
RECEIVEEDI FILEID(edidata) CLASS(EDI) EDIONLY(Y) MULTFILES(Y)
EDIOPT(Y);
```

Sample output file baseout.msg

```
AUTOSTART SESSIONKEY(CCKL6378);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(CCKL6378)
IEVERSION(04)
IERELEASE(07);
RETURN(00000);
SENDEDI FILEID(order1.fil);
SENT UNIQUEID(0573893.) LENGTH(1265);
RETURN(00000);
SENDEDI FILEID(order2.fil);
SENT UNIQUEID(67323552) LENGTH(765);
RETURN(00000);
SENDEDI FILEID(order3.fil);
SENT UNIQUEID(08747849) LENGTH(3566);
RETURN(00000);

RECEIVEEDI FILEID(edidata) CLASS(DATA) EDIONLY(Y) MULTFILES(Y)
EDIOPT(Y);
RECEIVED ACCOUNT(XXXX) USERID(XXXX01) RECEIVER(ACT1 DEPT01)
RECVQUAL(01)
SENDER(ACT2 DEPT02) SENDQUAL(01) CONTROLNUM(000057372) CLASS(DATA)
CHARGE(1) LENGTH(4290) FILEID(edidate)
MSGDATE(980701) MSGDATELONG(19980701) MSGTIME(020132) MSGSEQO(489028)
```

```
SESSIONKEY(CCKL6378) DELIMITED(E) SYSNAME(EB/AIX) SYSLEVEL(0450)
TIMEZONE(L)
DATATYPE(A) EDITYPE(X12) SENDERFILE(UPDATE0A) SENDERLOC(/home/
expedite)
FILEDATE(980602) FILEDATELONG(19980602) FILETIME(102544) RECFM(????)
RECLEN(00000) RECDLM(E) UNIQUEID(28700977) SYSTYPE(12) SYSVER(1)
TRANSLATE(IESTDTBL);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

Results : Three files containing EDI data were sent to three trading partners. The SENDEDI command does not require a destination address. The address is contained within the EDI envelope in the data.

The RECEIVEEDI command was issued to receive files from user class *edi* and store the data in file *edidata.fil*.

MULTFILES(Y) indicates to Expedite Base/AIX that if more than one file is received, the data from each received file would be stored in a separate file. The first received file would be stored in edidata, the second file in edidata.002, the third in edidata.003, and so forth. In the example, only one file was received.

EDIONLY(Y) indicates to Expedite Base/AIX that only EDI files should be received with this command. Information about whether this file is EDI or not is stored in the CDH. Without this parameter, any file in the mailbox with a user class of edi would be received. If a non-EDI file is received and read by the ABC translator, it could cause problems. This parameter helps to avoid such a situation.

EDIOPT(Y) indicates to Expedite Base/AIX to insert newlines after each EDI segment.

Example 5

The ABC company in Example 4 identifies trading partners by their DUNS numbers. When the translator creates the ISA envelope, it uses the DUNS number as the address of the trading partner. Information Exchange uses accounts and user IDs to identify individual mailboxes. “Using EDI destination tables” on page 91 describes how Expedite Base/AIX uses translate tables or EDI destination tables to translate a non-network address to an account and user ID address.

This example shows a sample ISA envelope and how Expedite Base/AIX uses translate tables to get the account and user ID address. Below is part of a sample ISA header. Most of the information in this sample ISA is replaced by *xxx* since it is not important to this example. What is important are the sender and receiver IDs.

```
ISA*xx*xxxxxxxxxx*xx*xxxxxxxxxx*01*1111111111     *01*2222222222   *xxxxxx*;
                                  ↑       ↑          ↑        ↑
                        Sender ID    Sender    Receiver ID  Receiver
                        Qualifier      ID      Qualifier      ID
```

The ABC company is the sender. ABC's DUNS number is 1111111111. Its trading partner’s DUNS number is 2222222222.

Expedite Base/AIX uses the ID qualifiers to determine how to translate the *ID*. The translation is done using an EDI destination table with the name ttabl*xx*.tbl where the *xx* is replaced by the *ID* qualifier. In this example, Expedite Base/AIX uses the EDI destination table ttabl01.tbl to translate the trading partner's DUNS number to an account and user ID. The following example is an EDI destination table:

```
EDIDEST(2222222222) ACCOUNT(acct) USERID(partner);
```

Using this information, Expedite Base/AIX sends this EDI data to the account *acct* and user ID *partner*.

Chapter 7

Using Expedite Base/AIX profile commands

To use Expedite Base/AIX, you need to create an Expedite Base/AIX profile. This chapter explains how to create profiles, explains the command syntax, and describes the profile commands and response records. It also explains how to change your network and Information Exchange passwords and discusses the *Extended Security Option (ESO)*.

Creating profiles

To create a profile, enter profile commands in basein.pro using a text editor. When you run the program, Expedite Base/AIX echoes the profile commands in basein.pro, along with response records and their associated return codes, to the profile response file (baseout.pro). You can review baseout.pro to verify successful completion of the profile commands.

Once these commands complete successfully, Expedite Base/AIX stores their values in an internal profile called iebase.pro. Therefore, these values remain in effect until you change them in basein.pro. You cannot change these values by deleting them from basein.pro or erasing basein.pro. You must use a profile command in basein.pro to change them. For example, if you do not want Expedite Base/AIX to dial the second telephone number you specified for the network, change the value of the DIALCOUNT2 parameter to 0. If you simply remove the DIALCOUNT2 parameter from basein.msg, it will not change iebase.pro and you will not get the desired result. If you want to remove a parameter value from the profile, you can change the parameter to specify a value of blank.

Understanding command syntax examples

The command syntax examples in this chapter show required parameters in bold, parameter values in italics, and default parameter values underlined.

Working with profile commands

Use the following commands to create and maintain your profile.

The profile commands are:

- DIAL, page 137. Use this command to enter modem and telephone (dial) information to connect to the network.
- IDENTIFY, page 144. Use this command to set up the fields in the profile related to your account, user ID, and password information.
- SESSION, page 148. Use this command to specify session-related information, such as exit key and file overwrite.
- SNACOMM, page 150. Use this command to specify the connection name for SNA APPC communications.
- SSL, page 151. Use this command to enable a secure TCP/IP connection using Secure Sockets Layer (SSL).
- TCPCOMM, page 151. Use this command to specify the connection name for TCP/IP communication.
- TRACE, page 154. Use this command to specify the information that Expedite Base/AIX records in the trace file during a session. Possible traces are BASE, CNNCT, DISPLAY, IOFILE, LINK, MODEM, and PROTOCOL.
- TRANSMIT, page 156. Use this command to specify the level of recovery Expedite Base/AIX will use to control the amount of data sent between checkpoints. You can also use it to specify whether Expedite Base/AIX starts and ends an Information Exchange session automatically.

The following sections provide detailed information on each of these commands.

DIAL command

Use the DIAL command to specify port, modem, and telephone (dial) information to connect to the network. You can enter up to five sets of phone numbers, connect and disconnect scripts; dial counts, and data rates. You correlate the members of the set by specifying a number from 1 to 5 at the end of the parameter.

You can also enter five sets of device drivers, device data rates, initialization and reset scripts, and modem initialization strings. You correlate the members of the set by specifying a letter from A to E at the end of the parameter.

The following example shows the syntax of the DIAL command:**`dial`**

```
phonen(phone number) baudraten(baudrateN) dialcountn(dialcountN)
devicex(deviceX) dbaudratex(dbaudrateX)
modeminit(modem init string) modeminitx(modem init stringX)
modemreset(modem reset string) modemresetx(modem reset stringX)
cnnctscr(connect script) discnnctscr(disconnect script)
initscr(initialization script) resetscr(reset script)
netinit(secondary network initialization string)
netpw(secondary network password)
netaddr(secondary network address) cycle(cycle) wait(wait)
escape(escape sequence) phonetype(phone type)
   dclversion(1|2) lockdir(lock directory) usern(userdefined variable);
```

phone*n*

Indicates the telephone number Expedite Base/AIX uses to dial the network. If you are using asynchronous dial, you must specify at least one telephone number in the profile. The total number you can specify is five. A telephone number can contain control characters recognized by your modem in a dial command. For example, you can include a comma in your telephone number to indicate a pause if you are using a Hayes-compatible modem. This value is substituted for the %PHONE% variable in the connect script. Use 1- to 36-alphanumeric characters.

The valid values for *n* are **1** to **5**.

baudrate*n*

Indicates the data rate (in bits per second) for communication between Expedite Base/AIX and the modem. This value is substituted for the %BAUD% variable in the connect script. The valid values are:

300
1200
2400
4800
9600
19200
38400

The default value is **2400**. The valid values for *n* are **1** to **5**.

NOTE: The maximum data rate depends on the telephone number you dial and the data rate of the modem. Consult your marketing representative for more information on the data rate.

dialcount*n*
Indicates the maximum number of times Expedite Base/AIX dials the corresponding telephone number. The valid values are **0** to **9**. If you place **0** in this parameter, Expedite Base/AIX does not dial the corresponding telephone number. The default value is **2**.

The valid values for *n* are **1** to **5**.

device*x*
Indicates the communications device driver on your system through which the modem connects to Information Exchange. The value should be the fully qualified path to the device driver. You can specify a total of five device drivers. Expedite Base/AIX will cycle through the list of device drivers and use the first one that responds. The valid values for x are **A** to **E**.

`dbaudrate`*x*

Indicates the speed of the corresponding device driver. The valid values are:

`300`
`1200`
`2400`
`4800`
`9600`
`19200`
`38400`

The default value is **2400**. The valid values for *x* are **A** to **E**.

`modeminit`

Indicates the initialization string Expedite Base/AIX uses to initialize the modem. This value will be substituted for the %INIT% variable in the default connect script. Use up to 40 characters. The default value is **ATL1X1V1Q0&C1&D2**.

`modeminit`*x*

Indicates the initialization string that Expedite Base/AIX uses to initialize DEVICE*x*. If you specify MODEMINIT and MODEMINIT*x*, the MODEMINIT*x* values override the default indicated by MODEMINIT. Therefore, you can indicate MODEMINIT for all modems except the one in which MODEMINIT*x* is specified.

The valid values for *x* are **A** to **E**. Use up to 40 characters. The default value is **ATL1X1V1Q0&C1&D2**.

`modemreset`

Indicates the string Expedite Base/AIX uses to reset the modem to the factory configuration, or to some other known state. The default modem connect script refers to this parameter as the %RESET% variable value. You can use up to 8 characters. The default is **`AT&F.`**

modemreset*x*

Indicates the string Expedite Base/AIX uses to reset DEVICEx to the factory configuration, or some other known state. The default modem connect script refers to this parameter as the %RESET% variable value. You can use up to 8 characters. The default is **AT&F**.

cnnctscr

Indicates the modem script Expedite Base/AIX uses to connect to the network. The connect script is run each time a phone number is dialed. The connect script is used to dial the phone number and establish the connection. Use a valid operating system file name, 1 to 14 characters. The default value is **cnnct.scr**.

NOTE: To access Information Exchange remotely, use the value fucnnct.scr in place of the default value. For additional information on the Traveling User feature, refer to Chapter 10, "Using additional features."

Place this script file in the directory where Expedite Base/AIX is installed or in the path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name. For more information on reserved file names, see Appendix C, "Reserved file names and user classes."

discnnctscr

Indicates the modem script Expedite Base/AIX uses to return the modem to an on-hook state after disconnecting from the network. Use a valid operating system file name, 1 to 14 characters. The default value is **discnnct.scr**.

Place this script file in the directory where Expedite Base/AIX is installed or in the path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name. For more information on reserved file names, see Appendix C, "Reserved file names and user classes."

initscr

Indicates the modem script Expedite Base/AIX uses to initialize the modem. Normally, Expedite Base/AIX uses cnnct.scr to initialize the modem, but you can provide a separate script to handle the modem initialization. Expedite Base/AIX uses your script the first time it tries to access the modem but not on redials. Use a valid operating system file name, 1 to 14 characters.

Place this script file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name. For more information on reserved file names, see Appendix C, "Reserved file names and user classes."

resetscr

Indicates the modem script Expedite Base/AIX uses to reset the modem. Normally, Expedite Base/AIX uses discnnct.scr to reset the modem, but you can provide a separate script to handle modem reset. Expedite Base/AIX uses your script after the last time it disconnects the modem but not on redials. Use a valid operating system file name, 1 to 14 characters.

Place this script file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name. For more information on reserved file names, see Appendix C, "Reserved file names and user classes."

cnnctscr*n*

Indicates the connection scripts Expedite Base/AIX uses for different telephone numbers. For example, if PHONE1 requires PAD connect processing and PHONE2 does not, you can specify different connect scripts for each phone number using CNNCTSCR1 and CNNCTSCR2 parameters. Use a valid operating system file name, 1 to 14 characters. If you do not use this parameter, the default is the value you specify in the CNNCTSCR parameter.

Place this script file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

The valid values for *n* are **1** to **5**.

discnnctscr*n*

Indicates the disconnect scripts Expedite Base/AIX uses for different telephone numbers. For example, if PHONE1 requires PAD disconnect processing and PHONE2 does not, you can specify different disconnect scripts for each phone number using DISCNNCTSCR1 and DISCNNCTSCR2 parameters. Use a valid operating system file name, 1 to 14 characters. If you do not use this parameter, the default is the value you specify in the DISCNNCTSCR parameter.

Place this script file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

The valid values for *n* are **1** to **5**.

initscr*x*

Indicates a different modem initialization script for different modems attached. If you specify INITSCR and INITSCRx, the INITSCRx values override the default indicated by INITSCR. Therefore, you can indicate INITSCR for all modems except the one for which INITSCRx is specified. Use a valid operating system file name, 1 to 14 characters. The valid values of *x* are from **A** to **E**.

Place this file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name.

resetscr*x*

Indicates the modem script Expedite Base/AIX uses to reset the modems for different modems attached. If RESETSCR and RESETSCR*s* is specified, RESETSCR*x* values override the default indicated by RESETSCR. Therefore, you can indicate RESETSCR for all modems except the one for which RESETSCRx is specified. Use a valid operating system file name, 1 to 14 characters. Valid values for *x* are from **A** to **E**.

Place this script file in the directory where Expedite Base/AIX is installed or in a path you specify in the IEPATH parameter of the SESSION command in basein.pro.

NOTE: Do not specify a reserved file name.

netinit

Indicates the initialization string Expedite Base/AIX uses to access the network through a secondary network. The value will be substituted for %NETINIT% in the connect script. Use up to 40 characters.

netpw

Indicates the secondary network password when you access the network through a secondary network. The value will be substituted for %NETPW% in the connect script. If you specify ENCRYPT(Y) in the IDENTIFY command, you must encrypt this password and it must be 8 characters long. Use up to 8 alphanumeric characters.

netaddr

Indicates the secondary network address when you access the network through a secondary network. The value will be substituted for %NETADDR% in the connect script. Use up to 15 alphanumeric characters.

cycle

Indicates the number of times Expedite Base/AIX redials a list of telephone numbers after an unsuccessful attempt to connect to the network. The valid values are **0** to **9**. The default value is **0**.

wait

Indicates the time Expedite Base/AIX waits between connect cycles. The format for this field is *HHMM*. All information entered is right-justified, and padded on the left with zeros. The default value is **0000**.

escape

Indicates a character sequence to obtain an outside line from a local PBX. The escape sequence can contain control characters recognized by your modem in a DIAL command. For example, you can include a comma in your escape sequence to indicate a pause if you use a Hayes-compatible modem. This value will be substituted for the %ESC% variable in the connect script. Use 1 to 8 alphanumeric characters.

phonetype

Indicates your telephone type. This value will be substituted for the %PTYPE% variable in the modem script.

p	Indicates a pulse dial telephone (rotary dial).
t	Indicates a touchtone telephone. This is the default.

dclversion

Indicates which Data Control Layer (DCL) block size will be used.

When you set the data rate to 2400 bps or lower, Expedite Base/AIX communicates with the network gateway using a smaller DCL block size.

At data rates higher than 2400 bps, Expedite Base/AIX can use a larger DCL block size. If you specify DCLVERSION(2), which is the default, Expedite Base/AIX automatically uses the larger DCL block size if the data rate is higher than 2400 bps and the network gateway supports the larger DCL block size. This reduces the number of acknowledgments between Expedite Base/AIX and the network gateway, thereby improving throughput. If you want to prevent use of the larger DCL block size, specify DCLVERSION(1).

1	Indicates that the smaller DCL block size is to be used.
2	Indicates that the larger DCL block size is to be used when the data rate is higher than 2400 bps and the network gateway supports the larger DCL block size. This is the default.

lockdir

Indicates the lock directory to be used if your system requires that the device lock file be written to a different directory than the default. Expedite Base/AIX writes a lock file with the name `LCK..nnnnn`, where `nnnnn` is the device name. The file contains the process ID for the currently-executing copy of Expedite Base/AIX. The lock file usage follows the AIX convention to keep multiple applications from trying to access the modem at the same time.

To specify the lock directory, use the full path where the lock file should be written, up to 128 characters, with the last character the directory delimiter "/." For example:

```
DIAL ... LOCKDIR(/usr/lib/uucp/) ...;
```

(In this example, "..." indicates that other DIAL parameters may be present.) If you want to run Expedite Base/AIX with a user ID other than root, you may need to change the permission of the lock file directory.

The default values are:

```
AIX: /etc/locks/
```

user*n* *n* **parameter**

Indicates a user-defined value that will be substituted for a variable in a modem script.

user 1	Value substituted for %USER1% in the modem script. Use 1 to 32 characters.
user 2	Value substituted for %USER2% in the modem script. Use 1 to 32 characters.
user 3	Value substituted for %USER3% in the modem script. Use 1 to 32 characters.

DIAL command example

The following is an example of the DIAL command:

```
dial escape(9,)
phone1(1234567) baudrate1(9600) dialcount1(2)
phone2(2345678) baudrate2(9600) dialcount2(0)
    devicea(/dev/tty0) dbaudratea(9600);
```

Results: Expedite Base/AIX will attempt to communicate through /dev/tty0. Expedite Base/AIX will dial 9 1234567 at the data rate of 9600 bps. It will dial up to two times to make the connection. Because the dial count specified in DIALCOUNT2 is 0, the phone number 2345678 will not be dialed.

IDENTIFY command

Use the IDENTIFY command to set up the network and Information Exchange account names, user IDs, passwords, and other information.

The following example shows the syntax of the IDENTIFY command:

identify

```
inaccount(network account) inuserid(network user ID)
inpassword(network password) ninpassword(new network password)
ieaccount(IE account) ieuserid(IE userid)
iepassword(IE password) niepassword(new IE password)
keyringfile(filename) keyringpassword(password)
    keyringstashfile(filename)
product(product) timezone(time zone)
    encrypt(n|y);
```

inaccount

Indicates the network account name. Use 1 to 8 alphanumeric characters.

inuserid

Indicates the network user ID. Use 1 to 8 alphanumeric characters.

inpassword

Indicates the network password. Use 1 to 8 alphanumeric characters, if you specify ENCRYPT(N). If you specify ENCRYPT(Y), this value must be 8 characters long and must be encrypted.

The following three parameters enable access to the Expedite Base/AIX GSKit environment for use with a Secure Sockets Layer (SSL) TCP/IP connection.

keyringfile

Indicates the file name of the keyring file. The keyring file requires a password. The maximum length is 256 characters.

NOTE: The KEYRINGFILE requires a password that you can supply either in a stash file or in the command as a parameter. You can use only one of these options. If you use the KEYSTASHFILE parameter, you cannot use the KEYRINGPASSWORD option.

keyringstashfile

The file name of the stash file that stores the password for the KEYRINGFILE parameter. The maximum length is 256 characters. If you use this parameter, you do not need to specify the KEYRINGPASSWORD parameter.

keyringpassword

The password for the KEYRINGFILE. Maximum length is 16 characters. If you use this parameter, you do not need to specify the keyringstashfile parameter.

ninpassword

Indicates the new network password. Expedite Base/AIX changes the network password on the next network connection, even if the Information Exchange session has an error. Use 1 to 8 alphanumeric characters, if you specify ENCRYPT(N). If you specify ENCRYPT(Y), this value must be 8 characters long and must be encrypted.

For more information on changing passwords, “Changing passwords” on page 162.

ieaccount

Indicates the Information Exchange account ID. You do not need to specify this parameter if you use the START command in basein.msg. Otherwise, this parameter is required. Use 1 to 8 alphanumeric characters.

ieuserid

Indicates the Information Exchange user ID. You do not need to specify this parameter if you use the START command in basein.msg. Otherwise, this parameter is required. Use 1 to 8 alphanumeric characters.

iepassword

Indicates the Information Exchange password. Use 1 to 8 alphanumeric characters, if you specify ENCRYPT(N). If you specify ENCRYPT(Y), this value must be 8 characters long and must be encrypted.

niepassword

Indicates the new Information Exchange password. Expedite Base/AIX changes the password upon the successful completion of the next Information Exchange session. If the Information Exchange session has an error, Expedite Base/AIX does not change the password. Use 1 to 8 alphanumeric characters, if you specify ENCRYPT(N). If you specify ENCRYPT(Y), this value must be 8 characters long and must be encrypted.

For more information on changing passwords, “Changing passwords” on page 162.

product

Indicates the name of the Information Exchange product on your network Product Selection menu. Use 1 to 8 alphanumeric characters. The default is **infoexch**.

`timezone`

Indicates your local time zone. You can enter one of the time zone codes listed here, or you can specify an offset from Greenwich mean time (GMT) by indicating the number of hours and minutes east or west of the Greenwich meridian. The format for specifying the hours and minutes is `ehhmm` or `whhmm`.

Type:	To indicate:
e	east
w	west
hh	hours
mm	minutes

For example, to specify eastern daylight time, you can enter either **`edt`** or **`w0400`**, where *`w`* indicates west, and 0400 indicates 4 hours and 0 minutes.

Type:	To indicate:
ahs	W1000 (Hawaii standard time)
ast	W0400 (Atlantic standard time)
bst	E0100 (British summer time)
cdt	W0500 (Central daylight time)
cst	W0600 (Central standard time)
ead	E1000 (Eastern Australia daylight time)
edt	W0400 (Eastern daylight time)
emt	E0200 (Eastern Mediterranean time)
est	W0500 (Eastern standard time)
gmt	E0000 (Greenwich mean time)
jst	E0900 (Japanese standard time)
mdt	W0600 (Mountain daylight time)
mst	W0700 (Mountain standard time)
pdt	W0700 (Pacific daylight time)
pst	W0800 (Pacific standard time)
wed	E0200 (Western Europe daylight time)
wes	E0100 (Western Europe standard time)
ydt	W0800 (Alaska daylight time)
yst	W0900 (Alaska standard time)

Use 1 to 5 alphanumeric characters. The default is `gmt`.

encrypt

Indicates whether the passwords entered in the IDENTIFY and START commands are encrypted. You cannot encrypt some passwords and not encrypt others.

n	The passwords are not encrypted. This is the default.
y	The passwords are encrypted with a value of 190.

NOTE: If you specify *y*, you must encrypt all passwords before entering them in a command. In addition, be sure to specify the ENCRYPT(Y) parameter before any of the password parameters. Otherwise, if Expedite Base/AIX reads an encrypted password before reading the ENCRYPT(Y) parameter, it will not know that the password is encrypted and will not be able to properly process it. see "Encryption/decryption of passwords" on page 164 for more information on password encryption.

IDENTIFY command example

The following is an example of the IDENTIFY command:

```
identify inaccount(acct) inuserid(user01) encypt(n)
inpassword(mypass) ieaccount(acct) ieuserid(user01)
iepassword(mypass);

or

identify inaccount(acct) inuserid(user01) encrypt(Y) inpassword(mypass)
    ieaccount(acct) ieuserid(user01) iepassword(mypass)
    keyringfile(filename) keyringpassword(mypass);


    or

identify inaccount(acct) inuserid(user01) encrypt(Y) inpassword(mypass)
    ieaccount(acct) ieuserid(user01) iepassword(mypass)
    keyringfile(filename) keyringstashfile(filename);
```

Results: The user has the same account and user ID for both the network and Information Exchange, as well as the same password. Expedite Base/AIX uses this information to log on to the network and start a session with Information Exchange.

The user set of AUTOSTART(Y) and ENABLESSL(Y) establishes a secure communication connection to the network for communication with Information Exchange.

SESSION command

Use the SESSION command to change the default session information. You can use this command to:

- Specify a different exit key
- Identify a different directory where Expedite Base/AIX will find program files
- Not display the session picture
- Not display session status
- Identify a program that runs after Expedite Base/AIX completes
- Tell Expedite Base/AIX whether or not to overwrite existing files when receiving from Information Exchange

The following example shows the syntax of the SESSION command:

```
session
exitkey(exitkey) iepath(iepath)
picture(y|n) status(y|n)
    nextprogram(nextprogram) overwrite(y|n);
```

exitkey

Indicates the function key you use to exit Expedite Base/AIX. The valid values are **2** to **10**. The default value is **3**.

iepath

Indicates the directory in which the Expedite Base/AIX program files are installed. The default is /var/adm/expedite for the IBM RISC System/6000, and /usr/bin/expedite; otherwise, see “Reserved file names for IEPATH parameter” on page 461 for a list of the specific files affected by this parameter.

picture

Indicates whether Expedite Base/AIX draws the Information Exchange status picture.

n	Do not draw the Information Exchange status picture.
y	Draw the Information Exchange status picture. This is the default.

status

Indicates whether Expedite Base/AIX displays the Information Exchange connection and communication status.

n	Do not display the Information Exchange connection and communication status.
y	Display the Information Exchange connection and communication status. This is the default.

nextprogram

Names a program your system invokes after an Information Exchange session. Only use this field if Expedite Base/AIX does not return to the calling application automatically. For example, this option is necessary if you invoke Expedite Base/AIX with a C language "execl" statement. Use 1 to 14 alphanumeric characters. The default value is **blank**.

overwrite

Indicates whether or not Expedite Base/AIX should overwrite existing files when receiving data from Information Exchange.

y	If the file specified in the FILEID parameter of the RECEIVE or RECEIVEEDI command already exists on the workstation, overwrite that file with the data received from Information Exchange. This is the default.
n	Do not overwrite the file specified in the FILEID in the RECEIVE or RECEIVEEDI command. If the file already exists, then the data received from Information Exchange will be appended to the existing file. NOTE: If Expedite Base/AIX receives two files with the same name in the same session, it never overwrites one of these files with the other. Instead, it appends the second file to the first.

SESSION command example

The following is an example of the SESSION command:

```
session exitkey(10) overwrite(n);
```

Results: This SESSION command indicates that the user can press the F10 key to exit the program. In addition, when files are received, they will not overwrite any existing files with the same name.

SNACOMM command

Use the SNACOMM command to specify the connection name for SNA APPC communications.

The following example shows the syntax of the SNACOMM command:

snacomm

```
connection(connection);
```

connection

Indicates the name of the connection profile in Communication Server to be used. The name can be 1 to 31 characters. The default is **expconn**.

SNACOMM command example

The following is an example of the SNACOMM command:

```
snacomm connection(mycnnct);
```

Results: This SNACOMM command tells Expedite Base/AIX to use the Communication Server connection profile named *mycnnct* instead of the default **expconn**.

SSL command

Use the SSL profile command to specify whether you want to communicate with Information Exchange using a secure Secure Sockets Layer (SSL) TCP/IP connection. To use this command, you need to do the following:

- Request a connection using TCP/IP
- Point to the correct TCP/IP address
- Have an IBM Global Services Public Key Infrastructure. Refer to the certification information, “Client X.509 Certificate.” on page 10
- Have your AIX environment set up properly to use SSL

The following example shows the syntax of the SSL command:

```
SSL
enablessl (n/y)
```

Parameters

enablessl

Indicates whether you want to communicate with Information Exchange using a secure session.

n	Disables SSL. This is the default.
y	Enables SSL

NOTE: This command is valid only when the TRANSMIT command COMMTYPE parameter is set for TCP/IP communication as follows:

```
TRANSMIT COMMTYPE(T);
```

SSL command example

The following is an example of the SSL command:

```
SSL enablessl(y);
TRANSMIT autostart(y)
IDENTIFY keyringfile(filename) keyringpassword(password)
```

or

```
SSL enablessl(y);
TRANSMIT autostart(y)
IDENTIFY keyringfile(filename) keyringstashfile(filename)
```

or

```
SSL enablessl(y);
TRANSMIT autostart(n)
```

with

```
START...keyringfile(filename) keyringpassword(password) in the basein.msg
    file.
```

or

```
SSL enablessl(y);
   TRANSMIT autostart(n)
```

with

```
START...keyringfile(filename) keyringstashfile(filename) in the
   basein.msg file.
```

Results: The SSL command tells Expedite Base/AIX whether or not to establish a secure connection to Information Exchange. The default is **n**.

TCPCOMM command

Use the TCPCOMM command to specify parameters for TCP/IP communications. The following example shows the syntax of the TCPCOMM command:

TCPCOMM

timeout(*minutes*);

timeout

Indicates the maximum number of minutes that Expedite Base/AIX should wait when communicating with Information Exchange. If this activity timeout is reached before Expedite Base/AIX can send or receive new data from Information Exchange, Expedite Base/AIX will assume that the connection has been dropped. If the WAIT parameter is specified on the RECEIVE or RECEIVEEDI command and is greater than the value specified for timeout on the TCPCOMM command, the TCPCOMM timeout is used. The valid values are 2 through 10. The default value is **10**.

TRACE command

Use the TRACE command to specify what information the trace file records during a session. If you request BASE, CNNCT, DISPLAY, IOFILE, MODEM, or PROTOCOL, Expedite Base/AIX places the trace information in the trace file (iebase.trc). If you request LINK, Expedite Base/AIX places the trace information in the link trace file (s1dcl.trc).

If you do not request trace, Expedite Base/AIX still creates trace files but places minimal information in them.

The following example shows the syntax of the TRACE command:

`trace`

cnnct(*n*|*y*) display(*n*|*y*) modem(*n*|*y*)

protocol(*n*|*y*) link(*n*|*y*) base(*n*|*y*) iofile(*n*|*y*);

`cnnct`

Indicates whether the trace file contains syntax information for the modem script. Use this trace if you have problems modifying a modem script.

n	Do not include modem script information in the trace file. This is the default.
y	Include modem script information in the trace file.

`display`

Indicates whether the trace file contains the display status file processing information. Use this trace file if you have problems modifying display.scr.

n	Do not include the draw picture information in the trace file. This is the default.
y	Include the draw picture information in the trace file.

`modem`

Indicates whether the trace file contains the commands sent to the modem.

Use this trace file if your modem is not working properly.

n	Do not include the modem command information in the trace file. This is the default.
y	Include the modem command information in the trace file.

protocol

Indicates whether the trace file contains Information Exchange protocol information. The Help Desk uses this trace file for problem determination.

n	Do not include the Information Exchange protocol information in the trace file. This is the default.
y	Include the Information Exchange protocol information in the trace file.

link

Indicates whether Expedite Base/AIX traces data control layer information. Expedite Base/AIX places LINK protocol information in s1dcl.trc. The Help Desk uses this trace for problem determination. Expedite Base/AIX places LINK protocol information for TCP/IP communication in link.trc.

n	Do not include the link information in the trace file. This is the default.
y	Include the link information in the trace file.

base

Indicates whether the trace file contains the Expedite Base/AIX module information. The Help Desk uses this trace file for problem determination.

n	Do not include the Expedite Base/AIX module information in the trace file. This is the default.
y	Include the Expedite Base/AIX module information in the trace file.

iofile

Use this trace file if you are having problems creating or modifying basein.pro and basein.msg.

n	Do not include the command parsing information in the trace file. This is the default.
y	Include the command parsing information in the trace file.

TRACE command example

The following is an example of the TRACE command:

```
trace protocol(y) link(y);
```

Results: During the session, Information Exchange protocol trace information is written to the iebase.trc file. The data control layer information is written to the s1dcl.trc file.

TRANSMIT command

You can use the TRANSMIT command to specify a level of data recovery, the date and time of a delayed transmission, the block size, the maximum number of messages, and other information. You can also use this command to specify whether Expedite Base/AIX starts and ends an Information Exchange session automatically.

The following example shows the syntax of the TRANSMIT command:

`transmit`

```
autostart(y|n) reconnect(reconnect)
autoend(y|n) msgsize(message size)
commitdata(commit data) blocksize(block size)
translate(translate table) maxmsgs(max msg segments)
commtype(a|s|t|w) recovery(c|s|f|u);
```

`autostart`

Indicates whether Expedite Base/AIX starts an Information Exchange session automatically. The only time it starts a session automatically is when it begins to process basein.msg. If you want to start multiple Information Exchange sessions in basein.msg, you must specify both AUTOSTART(N) and AUTOEND(N).

y — Start the Information Exchange session automatically. You do not include the Expedite Base/AIX START command in the message command file. This is the default value.

n — Set this parameter to n for the following options:

1. Do not start the Information Exchange session automatically. You need to include the Expedite Base/AIX START command in the message command file.
2. Start a single session with Information Exchange with multiple Information Exchange user IDs. You need to include the Expedite Base/AIX START command in the message command file.

`reconnect`

Indicates the number of times Expedite Base/AIX attempts to reconnect to the network if it loses contact with the network after a successful logon. The valid values are **`0`** to **`9`**. The default value is **`5`**.

autoend

Indicates whether Expedite Base/AIX sends an Information Exchange session end command when it finishes processing your command file. The only time it ends a session automatically is when it finishes processing basein.msg. If you want to start multiple Information Exchange sessions in basein.msg, you must specify both AUTOSTART(N) and AUTOEND(N).

y — End the Information Exchange session automatically. You do not include the Expedite Base/AIX END command in the message command file. This is the default value.

n — Set this parameter to n for the following options:

1. Do not start the Information Exchange session automatically. You need to include the Expedite Base/AIX START command in the message command file.
2. Start a single session with Information Exchange with multiple Information Exchange user IDs. You need to include the Expedite Base/AIX START command in the message command file.

msgsize

Segments the data for sending. Your trading partner can take checkpoints only for the message size you specify with this parameter. If you use a large value, your trading partner cannot take frequent checkpoints.

Valid values are **1000** through **47000**. The default is **47000** bytes for TCP/IP and **37000** for all other communication types. Lower values permit the receiving interface to complete more frequent checkpoints while receiving the data.

NOTE: MSGSIZE must be less than or equal to COMMITDATA.

commitdata

This parameter applies only to checkpoint-level recovery. It is ignored for session-level, file-level, or user-initiated recovery.

This parameter indicates the maximum number of bytes of data the program sends between checkpoints. For maximum efficiency, the COMMITDATA value should be an even multiple of the MSGSIZE value. Lower values can result in less retransmission of data if there is a communication failure. Higher values provide faster data transmission. Valid values are **1000** to **141000**. The default value is **141000** bytes for TCP/IP and **37000** for all other communication types.

This value must be at least as large as the value you specified in MSGSIZE.

blocksize

Indicates the size of the blocks Expedite Base/AIX breaks Information Exchange messages into for line transmission. Valid values are **256** to **3500** for asynchronous communication and **256** to **1024** for worldwide asynchronous communication. The default value is 1024 if COMMTYPE is *w* and **2000** for all other COMMTYPEs. Do not change the default value unless you are losing contact with Information Exchange frequently.

translate

Indicates the default table Expedite Base/AIX uses for ASCII to EBCDIC and EBCDIC to ASCII translation. Use 1 to 8 characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the file name of the translate table. The translate table you specify must exist in the directory where Expedite Base/AIX is installed or in the path specified by IEPATH in the SESSION command. If you do not specify a table, the default is the standard Information Exchange translate table.

NOTE: This value can be overridden by the TRANSLATE parameter in the SEND, SENDEDI, RECEIVE, or RECEIVEEDI commands.

maxmsgs

Maximum number of message segments the program can receive between Information Exchange commits. The valid values are 1 to 10. The larger the number specified, the more data Information Exchange sends to you without committing it. A lower number causes more frequent data commits. The default value is **10**. Do not change the default unless you are losing contact with Information Exchange frequently.

commtype

Indicates the type of communication protocol Expedite Base/AIX uses to transmit data.

a — Error-corrected asynchronous communications

This option requires a connection with the network gateway. The gateway is commonly available in the United States but has limited availability in other countries.

s — SNA LU communications version of SNA

This option requires IBM AIX SNA Communication Server 4.2 and a leased-line attachment to the network.

t — TCP/IP communications

This option uses Transfer Control Protocol/Internet Protocol to connect to the network.

This option also enables secure network communication using the SSL command. For additional information, see “SSL command” on page 151.

w — Asynchronous communications without the network gateway

This option is generally used outside the United States. It can be used in the United States but it is not recommended.

`recovery`

Indicates the type of recovery level Expedite Base/AIX uses when it transmits data.

c	Checkpoint-level recovery. This is the default. For non-EDI data, see page 72 for more information. For EDI data, see page 101 for more information.
s	Session-level recovery. For non-EDI data, see page 62 for more information. For EDI data, see page 116 for more information.
f	File-level recovery. For non-EDI data, page 47 for more information. For EDI data, see page 102 for more information.
u	User-initiated recovery. This method requires the use of the COMMIT command (see "COMMIT command" on page 177). For non-EDI data, also see page 47 for more information. For EDI data, also see page 102 for more information.

TRANSMIT command example

The following is an example of the TRANSMIT command:

```
transmit commtype(a) reconnect(9) commitdata(15000) msgsize(5000);
```

Results: Expedite Base/AIX will attempt to communicate with the network using asynchronous communication. While sending data to Information Exchange, Expedite Base/AIX will take a commit checkpoint after every 15,000 bytes. If the connection to Information Exchange is lost during the transmission process, Expedite Base/AIX will attempt to reconnect a maximum of nine times.

Working with profile response records

The profile response file (baseout.pro) contains an echo of the profile commands and their response records.

The profile response records are:

- PROFILERC, page 161

  This record indicates the completion of basein.pro.

- RETURN, page 161

  This record indicates the completion of a command in basein.pro.

- WARNING, page 162

  This record indicates a minor error that did not stop the command from completing, but is an error of which you should be aware.

The following sections provide detailed information on each of these records.

PROFILERC record

The PROFILERC record is the last record in baseout.pro. The PROFILERC record indicates the processing of the profile commands is complete. A zero value indicates that all the profile commands completed successfully.

The following example shows the format of the PROFILERC record:

```
PROFILERC(return code) ERRDESC(error description)
    ERRTEXT(error text);
```

NOTE: There is only one PROFILERC record in each baseout.pro.

profilerc

Indicates whether Expedite Base/AIX processed the profile commands successfully. If the return code is 0, the commands completed successfully. If the return code is not 0, the program displays an error number along with ERRDESC and ERRTEXT records. The program displays the same return code on the SESSIONEND record in baseout.msg. This parameter contains 5 numeric characters.

errdesc

Provides a short description of an error. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

errtext

Provides a detailed description of an error, and may suggest steps to correct the problem. There may be multiple error text records in the file. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

RETURN record

The RETURN record indicates the completion of a command in basein.pro. A 0 value indicates that the command completed successfully.

The following example shows the format of the RETURN record:

```
RETURN(return) ERRDESC(error description)
    ERRTEXT(error text)...;
```

return

Indicates completion of an Expedite Base/AIX command. If the return code is 0, the command completed successfully. If the return code is not 0, the program displays an error number along with ERRDESC and ERRTEXT records. This parameter contains 5 numeric characters.

errdesc

Provides a short description of an error. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

errtext

Provides a detailed description of an error, and may suggest steps to correct the problem. There may be multiple error text records in the file. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

WARNING record

The WARNING record indicates a low-severity problem that does not prevent the profile command from completing.

The following example shows the format of the WARNING record:

```
WARNING(warning) ERRDESC(error description)
    ERRTEXT(error text);
```

warning

Indicates the warning code. This parameter contains 5 numeric characters.

errdesc

Provides a short description of an error. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

errtext

Provides a detailed description of an error, and may suggest steps to correct the problem. There may be multiple error text records in the file. If the return code is 0, this parameter is not in baseout.pro or baseout.msg. This parameter contains 1- to 76-alphanumeric characters.

Changing passwords

Your network passwords must have the following characteristics:

- Be 5 to 8 characters long
- Contain at least 3 different characters
- Not be the same as any of your three previous passwords
- Begin with an alphabetic character
- Not be the word *cancel*

To change your network and Information Exchange passwords, use the NINPASSWORD and NIEPASSWORD parameters in the IDENTIFY command in basein.pro. The following figure shows how to use these parameters.

```
identify    inaccount(acct) inuserid(user01) inpassword(inpwd)
            ninpassword(newinpwd)
            ieaccount(acct) ieuserid(user01) iepassword(iepwd)
            niepassword(newiepwd);
```

After you add new passwords, rerun the program to process the password change. When Expedite Base/AIX establishes a connection with the network and Information Exchange, it uses the original passwords to log on and then changes the passwords to the values specified in the NINPASSWORD and NIEPASSWORD parameters.

Before you run the program again, you must modify the IDENTIFY command in basein.pro. Remove the NINPASSWORD and NIEPASSWORD parameters and modify the INPASSWORD and IEPASSWORD parameters to reflect the new passwords. The following figure shows how to modify the IDENTIFY command.

```
identify    inaccount(acct) inuserid(user01) inpassword(newinpwd)
            ieaccount(acct) ieuserid(user01) iepassword(newiepwd);
```

If you run the program before you modify the IDENTIFY command, you receive an error from Expedite Base/AIX.

Selecting the Extended Security Option

ESO provides additional password and Information Exchange mailbox security. The ESO USER flag may be turned on by the Information Exchange Service Administrator, using Information Exchange Administration Services. Users with ESO can send files and messages to users with ESO and users without ESO.

For information on using ESO, see the *Information Exchange Administration Services User's Guide.*

ESO contains the following security features:

- Your Information Exchange Service Administrator must change your ESO password if it is the same as your user ID. If you do not provide a new password in the START command, your Information Exchange session will not be successful.
- Your new ESO password must conform to the following rules:
  - Cannot be the same as the Information Exchange user ID
  - Must contain 5 to 8 characters
  - Must begin with an alphabetic character
  - Must contain at least 3 different characters
  - Cannot be the same as the five previous passwords
  - Must not be the word *cancel*
  - Can be only alphanumeric plus national characters @ # $. & ( ) ! ' ? : ; " .

  If your new password does not conform to these rules, it is considered invalid, and your Information Exchange session will not be successful.
- Information Exchange revokes your ESO user ID if you make three consecutive attempts to start an Information Exchange session with an invalid password. All further attempts to start a session will be unsuccessful until your Information Exchange Service Administrator resets your password.
- The Information Exchange Service Administrator can use the Information Exchange Administration Services password reset function to reinstate an ESO user ID.

NOTE: Passwords are reset immediately after resetting the ESO option to *Y*. When the administrator sets the ESO option to *Y*, all affected user IDs must change their password at their next logon. This is true even if the administrator subsequently resets the ESO option back to *N*.

Encryption/decryption of passwords

Expedite Base/AIX provides the capability to encrypt and decrypt passwords. The encryption process converts ASCII characters to special unreadable characters, while decryption performs the reverse process. If you choose to encrypt passwords, all the passwords that Expedite Base/AIX uses must be encrypted. These passwords include:

- Network password
- New network password
- Information Exchange password
- New Information Exchange password
- Secondary network password

If you specify ENCRYPT(Y) on the IDENTIFY command in basein.pro, Expedite Base/AIX expects all passwords except the keyringpassword to be specified in encrypted format.

NOTE: If you will be specifying encrypted passwords, be sure to specify the ENCRYPT(Y) parameter *before* any of the password parameters on the IDENTIFY command. Otherwise, if Expedite Base/AIX reads an encrypted password before reading the ENCRYPT(Y) parameter, it will not know that the password is encrypted and will not be able to properly process it.

Encryption/decryption routines

The encryption/decryption routines are written in C language and are located on the Expedite Base/AIX program diskette.

To call the password encryption routine, specify:

```
psc(function,key,textlen,ptext,ctext)
```

where:

function	Is an integer *1* For encryption *2* decryption
key	Is an integer that specifies the encryption to be used.
textlen	Is an integer that is the length of the password.
ptext	Is the password to be encrypted or decrypted.
ctext	Is the place to put the encryption. The length of this field must be at least equal to *textlen*.

Encryption keys

For the network password and new network password, the key you must specify is 144. For the Information Exchange password and new Information Exchange password, the key you must specify is 151. The key you must specify for secondary network password is 167.

Chapter 8

Using Expedite Base/AIX message commands

To use Expedite Base/AIX message commands, you need to understand the command syntax. You also need to know when and how to use the commands. This chapter describes the message commands and provides examples.

Understanding command syntax examples

The command syntax examples in this chapter show required parameters in bold, parameter values in italics, and default parameter values underlined.

In some examples, you have a choice between parameters or groups of parameters. The word or separates the choices and two blank lines separate the choices from the other parameters in the example. Choose one line of parameters from any lines separated this way. If required parameters are in more than one of the choices, you still need to choose only one line.

Working with message commands

Use the message commands to send and receive files, control your Information Exchange mailbox, and define lists of users. You must enter message commands for Expedite Base/AIX in basein.msg. When Expedite Base/AIX processes basein.msg, it echoes the message commands, along with their associated return codes, to baseout.msg.

The message commands are:

- ARCHIVEMOVE, page 168

  This command copies files from the Information Exchange short-term archive to your mailbox. Expedite Base/AIX places a MOVED record in the response file as a result of the ARCHIVEMOVE command.

- AUDIT, page 169

  This command retrieves an audit trail from Information Exchange and places it in your mailbox.

- CANCEL, page 173

  This command cancels previously sent files if the receiver has not received them from their mailbox.

- COMMIT, page 177

  This command sends a commit to Information Exchange and processes the response.

- DEFINEALIAS, page 178

  This command defines a new alias, redefines an existing alias, or changes or deletes an existing alias table.

- END, page 182

  This command ends an Information Exchange session.

- GETMEMBER, page 183

  This command copies a library member from an existing Information Exchange library to an Information Exchange mailbox.

- LIST, page 188

  This command sets up a list of account and user IDs that you can use to send and receive files.

- LISTLIBRARIES, page 191

  This command returns a list of Information Exchange libraries to which you have access.

- LISTMEMBERS, page 192

  This command returns a list of members within an Information Exchange library.

- PURGE, page 193

  This command deletes a specific file from your mailbox.

- PUTMEMBER, page 194

  This command adds a member to an existing Information Exchange library.

- QUERY, page 198

  This command enables you to get information about all the files in your mailbox. You can also use QUERY to see the CDH information associated with each file.

- RECEIVE, page 199

  This command enables you to receive all files or specific files, including e-mail.

- RECEIVEEDI, page 207

  This command enables you to receive EDI-formatted files. Expedite Base/AIX supports X12, UCS, EDIFACT, and UN/TDI standards.

- SEND, page 216

  This command enables you to send files including e-mail that you save in a file, to a user or a list of users

- SENDEDI, page 223

  This command enables you to send EDI-formatted files. Expedite Base/AIX supports X12, UCS, EDIFACT, and UN/TDI standards.

- START, page 229

  This command starts an Information Exchange session.

The following sections provide detailed information on each of these commands.

ARCHIVEMOVE command

Use the ARCHIVEMOVE command to copy files from the Information Exchange short-term archive to your mailbox. The message response records associated with the ARCHIVEMOVE command are the MOVED record and the RETURN record.

The following example shows the syntax of the ARCHIVEMOVE command:

archivemove

archiveid(*archive ID*);

archiveid

Indicates the archive reference identifier for the files you want to copy from archive to your mailbox. The archive reference identifier is the value you specify in the ARCHIVEID parameter in the RECEIVE or RECEIVEEDI command when you receive a file. If you do not specify an ARCHIVEID in the RECEIVE or RECEIVEEDI commands when you receive a file, Information Exchange assigns an archive reference identifier equivalent to the session key. To see what the session key is, look at the SESSIONKEY parameter in the RECEIVED record in baseout.msg or use Information Exchange Administration Services. Use 1 to 8 alphanumeric characters. This is a required parameter.

ARCHIVEMOVE command example

The following is an example of the ARCHIVEMOVE command:

archivemove archiveid(*myrefid*);

Results: Files stored in the Information Exchange archives with the archive ID of *myrefid* are copied to your mailbox.

AUDIT command

Use the AUDIT command to retrieve an audit trail from Information Exchange and place the information in your mailbox. When an audit is available, you must issue a RECEIVE command to retrieve it. The audit information comes from the *SYSTEM* account and the *AUDITS* user ID in user class #SAUDIT. The message response record associated with the AUDIT command is the RETURN record. For details on audit information, see Chapter 10, "Using additional features."

NOTE: Information Exchange does not make audits available immediately. Therefore, you cannot successfully issue the RECEIVE command to receive the audit file immediately after issuing the AUDIT command. Audits normally are available during a subsequent session.

The following example shows the syntax of the AUDIT command:

audit

```
account(account) userid(user ID)
```

or

```
account(account) userid(user ID) sysid(system ID)
```

or

```
alias(alias) aliasname(alias name)
altacct(alternate account)
altuserid(alternate user ID|?)
msgtype(b|s|r) class(class)
startdate(start date) enddate(end date) status(blank|u|p|d)
timezone(l|g) level(1|2|3);
```

account

Indicates the account of an Information Exchange trading partner. Expedite Base/AIX uses this field in conjunction with the USERID parameter to indicate that you want only audit records for file exchanges with this account and user ID. If you specify a USERID parameter, you must also specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange trading partner. Expedite Base/AIX uses this parameter in conjunction with the ACCOUNT parameter to indicate that you want only audit records for file exchanges with this user ID and account. If you specify an ACCOUNT parameter, you must also specify a USERID parameter. Use 1 to 8 alphanumeric characters.

sysid

Indicates the system ID of an Information Exchange trading partner to whom you are sending data. You need the system ID only if you specify the ACCOUNT and USERID parameters for a trading partner on another Information Exchange system. If you specify a SYSID parameter, you must also specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to indicate that you want only audit records for file exchanges with a single user.

blank	An alias name was not used. This is the default.
gxxx	Global alias table, where *xxx* identifies a 1- to 3-character table name.
oxxx	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
pxxx	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify a ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTES: You can create and maintain alias tables in two ways:

1. Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*).
2. Using the DEFINEALIAS command (see "DEFINEALIAS command" on page 178.).

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to indicate that you want only audit records for file exchanges with a single user. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

altacct

Indicates the alternate account for which you want to receive an audit. ALTACCT can be 1 to 8 characters. If you specify ALTACCT, you must specify ALTUSERID.

This field allows you to request audit records for an account and user ID other than your own. If left blank, audit records for your own account will be returned. You must have authority to view audit records for the alternate account and user ID.

NOTE: ALTACCT is only valid for expansion level 3 audit level requests.

altuserid

Indicates that you want to retrieve audit records for another user or users in your account. To retrieve audit records for another user in your account, put the other user ID here. To retrieve audit records for all user IDs in your account, use**?** as the parameter value.

If you want to retrieve audit records for several specific users, you must specify each user ID in a separate AUDIT command. Use 1 to 8 alphanumeric characters.

NOTE: Information Exchange ignores this field and places an error message in your mailbox if you do not have service administrator authorization.

msgtype

Indicates the type of audit record to retrieve.

b	Retrieve all audit records for files you have sent and that have been sent to you. This is the default value.
s	Retrieve all audit records for files you sent.
r	Retrieve all audit records for files sent to you.

class

Indicates the user class of the files for which you want to retrieve audit records. Use 1 to 8 alphanumeric characters.

startdate

Indicates the start date of a time range for the messages and files for which you want audit records. The format is *YYYYMMDD* or *YYMMDD*. The default value is determined by Information Exchange and is currently 000102 (January 2, 1900).

enddate

Indicates the end date of a time range for the messages and files for which you want audit records. The format is *YYYYMMDD* or *YYMMDD*. The default value is determined by Information Exchange and is currently **420916** (September 16, 2042).

status

Indicates that you only want audit records for messages with a specified status.

blank	Retrieve audit records for all files. This is the default.
u	Retrieve audit records for undelivered files only.
p	Retrieve audit records for purged files only.
d	Retrieve audit records for delivered files only.

`timezone`

Indicates the date and time reference in the STARTDATE and ENDDATE parameters.

l	Your local time, as specified on the TIMEZONE parameter of the IDENTIFY command. This is the default value.
g	GMT (Greenwich mean time).

`level`

Indicates the style of an audit report.

1	Retrieve audit report in the original style. This is the default.
2	Retrieve audit report in the enhanced style.
3	Retrieve audit report in the enhanced style with the EDI exchange control number.

AUDIT command example

The following is an example of the AUDIT command:

```
audit account(acct) userid(user01) class(#e2);
```

Results: This command places the appropriate level audit report in your mailbox. This audit will include information about all files with a class of *#e2* sent to and received from account *acct* and user ID *user01*, or any files that user ID might have purged.

CANCEL command

Use the CANCEL command to cancel files you sent to a single account and user ID, a list of users, or a destination specified by an alias name. You can cancel these files only if the receiver has not retrieved them from Information Exchange. Only the sender of a file can request to cancel the file from the receiver's mailbox.

The message response record associated with the CANCEL command is the RETURN record.

The following example shows the syntax of the CANCEL command:

cancel

account(*account*) **userid**(*user ID*)

or

alias(*alias*) **aliasname**(*alias name*)

or

listname(*list name*)
priority(*blank*|*p*) msgname(*message name*)
msgseqno(*message sequence number*) class(*class*)
timezone(*l*|*g*) ack(*blank*|*h*|*t*)
startdate(*start date*) starttime(*start time*)
enddate(*end date*) endtime(*end time*);

account

Indicates the account of an Information Exchange user to whom you sent data. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user to whom you sent data. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user to whom you sent data.

blank	An alias name was not used. This is the default.
g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in either of two ways:

- Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*).
- Using the DEFINEALIAS command (see "DEFINEALIAS command" on page 178).

aliasname

Indicates an alias name in the alias table you specified on the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user to whom you sent data. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

listname

Indicates the name of a list of accounts and user IDs. Expedite Base/AIX uses this field to identify a list of users to whom you sent data. Use 1 to 8 alphanumeric characters.

priority

Indicates the priority of the files you want to cancel.

blank	Indicates normal priority. This is the default.
p	Indicates high priority.

msgname

Indicates the message name of the files you want to cancel. Use 1 to 8 alphanumeric characters.

msgseqno

Indicates the message sequence number of the files you want to cancel. Use 1 to 5 alphanumeric characters.

class

Indicates the user class of the files you want to cancel. Use 1 to 8 alphanumeric characters.

timezone

Indicates the time reference in the STARTTIME and ENDTIME parameters.

l	Your local time, as specified on the TIMEZONE parameter of the IDENTIFY command. This is the default value.
g	GMT (Greenwich mean time).

ack

Indicates the types of acknowledgment messages you want to receive regarding the cancellation.

blank	No acknowledgments for the cancellations of files. However, Information Exchange may create other types of acknowledgments. This is the default.
h	Acknowledgments include only header information.
t	Acknowledgments include both header and text information.

startdate

Indicates the start date of a time range for files sent to Information Exchange that you want to cancel. For a file to qualify for cancellation, the date the file was sent to Information Exchange must fall within this date range. Format the start date as either *YYYYMMDD* or *YYMMDD*. The default value is determined by Information Exchange and is currently 000201 (January 2, 1900).

starttime

Indicates the start time of a time range for files sent to Information Exchange that you want to cancel. For a file to qualify for cancellation, the time the file was submitted must fall within this time range. Format the start time *HHMMSS*. The default value is determined by Information Exchange and is currently **000000** (00:00:00).

enddate

Indicates the ending date of a date range for files sent to Information Exchange that you want to cancel. For a file to qualify for cancellation, the date the file was submitted must fall within this date range. Format the end date as either *YYYYMMDD* or *YYMMDD*. The default value is determined by Information Exchange and is currently **420916** (September 16, 2042).

endtime

Indicates the ending time of a time range for the files sent to Information Exchange that you want to cancel. For a file to qualify for cancellation, the time the file was submitted must fall within this time range. Format the end time *HHMMSS*. The default value is determined by Information Exchange and is currently **235347** (23:53:47).

CANCEL command example

The following is an example of the CANCEL command:

```
cancel account(acct) userid(user01) class(special);
```

Results: This command removes any files in the mailbox for account *acct* and user ID *user01*, that you sent with a class of *special* and that have not yet been received.

COMMIT command

Use the COMMIT command to send a commit to Information Exchange and process the response. In order for Expedite Base/AIX to perform the commit, a SEND, SENDEDI, or PUTMEMBER command must be requested between COMMIT commands or between a session start and a COMMIT command.

Checkpoints are taken automatically:

- After each COMMIT command, unless there is nothing to commit
- At the end of each session, even if you have not specified a COMMIT command
- While receiving data for a RECEIVE or RECEIVEEDI command, if Information Exchange requests a checkpoint
- At the end of each RECEIVE or RECEIVEEDI command

The COMMIT command is valid only for user-initiated recovery. The TRANSMIT profile command determines the type of recovery. To specify user-initiated recovery, ensure that the RECOVERY parameter of TRANSMIT is set to *u*.

On restart, data transmission resumes after the last successful COMMIT command. If the error occurred while processing the first COMMIT command in the message command file and a checkpoint had not occurred, Information Exchange does not deliver any data and does not delete received data from the mailbox. In this case, Expedite Base/AIX retransmits all data upon restart.

The COMMIT command has no parameters. Type the COMMIT command as follows:

commit;

DEFINEALIAS command

Use the DEFINEALIAS command to:

- Create an alias table
- Add a new alias
- Redefine an existing alias
- Change or delete an existing alias table

NOTE: Your user ID must be authorized to update the alias table, or you will receive a system error message in your mailbox.

Although you can generally specify parameters in any order, the DEFINEALIAS command entries must include a DEFINENAME parameter and one of the following groups of parameters:

- ACCOUNT and USERID
- SYSID, ACCOUNT, and USERID
- ALIAS and ALIASNAME

You must pair these entries correctly. You must specify a DEFINENAME parameter with the ACCOUNT and USERID parameters or the SYSID, ACCOUNT, and USERID parameters. Another option is to specify the DEFINENAME parameter with the ALIAS and ALIASNAME parameters.

You cannot specify another DEFINENAME parameter unless you complete the definition for the previous DEFINENAME.

If the value of the FUNCTION parameter is *e* for erase, you cannot specify a DEFINENAME parameter.

NOTE: Do not define an alias more than once in a session, or the first alias will be overwritten.

The message response record associated with the DEFINEALIAS command is the RETURN record.

Information Exchange does not perform table updates while you are in session. Therefore, it is possible that you may receive a zero return code for the DEFINEALIAS command but the alias table is not updated. An example of this is if you attempt to add an alias to a nonexistent table. In this situation, Information Exchange places a system error message in your mailbox.

The following example shows the syntax of the DEFINEALIAS command:

definealias

aliastable(*alias table*) **function**(*a*|*n*|*c*|*d*|*e*)

authority(*p*|*a*|*g*)

definename(*define alias name 1*)

account(*account 1*) userid(*user ID 1*)

or

```
sysid(system ID 1) account(account 1) userid(user ID 1)
```

or

```
alias(alias 1) aliasname(alias name 1)
definename(define alias name n)
account(account n) userid(user ID n)
```

or

```
sysid(system ID n) account(account n) userid(user ID n)
```

or

```
alias(alias n) aliasname(alias name n);
```

aliastable

Indicates the alias table type and table name.

g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
pxxx	Private alias table, where *xxx* identifies a 1- to 3-character table name.

function

Indicates the type of operation you want Expedite Base/AIX to request for the alias table.

a	Add the following entries to the alias table. This is the default.
n	Create a new alias table using the name specified in the ALIASTABLE parameter.
c	Change the following entries in the alias table.
d	Delete the following entries from the alias table. If you list all entries in the table, you get the same results as with option **e**.
e	Erase the entire alias table. If *e* is specified for FUNCTION, no other parameters can be used in the DEFINEALIAS command.

NOTE: If you specify FUNCTION(N) to create a new alias table, and there is already an existing alias table with that name, you will receive a system error message in your Information Exchange mailbox saying that the table already exists.

authority

Indicates the authorization level of the alias table you are referencing. Authority is valid only if FUNCTION is *n*. You cannot change the authority of an existing table.

p	Only the owner of the table can update the alias table. This is the default.
a	Any administrator in the account can update the alias table.
g	Any user can update the alias table. You cannot specify *g* for AUTHORITY if you are defining a private or organizational alias table.

definename

Indicates the alias name you want to add, change, or delete in the table specified by ALIASTABLE. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of an Information Exchange user. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another Information Exchange system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the account of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user.

blank	An alias name was not used. This is the default.
gxxx	Global alias table, where *xxx* identifies a 1- to 3-character table name.
oxxx	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
pxxx	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

DEFINEALIAS command example

The following is an example of the DEFINEALIAS command:

```
definealias
aliastable(gx01) function(a)
definename(duns01) account(act1) userid(user1)
definename(duns02) sysid(eur) account(act2) userid(user2)
definename(duns03) alias(gx02) aliasname(duns51);
```

Results: This command adds three new aliases to alias table GX01.

- Alias name *duns01* is associated with Information Exchange account *act1* and user ID *user1*.
- Alias name *duns02* is associated with Information Exchange system *eur*, account *act2*, and user ID *user2*.
- Alias name *duns03* is associated with another alias *duns51*, defined in alias table *gx02*.

NOTE: You can define alias names that refer to other alias names.

The following is an example of an invalid DEFINEALIAS command:

```
definealias
aliastable(gx01) function(a)
definename(duns01) account(act1) #invalid entry: no userid
definename(duns02) account(act2) userid(user2)
userid(user1) #invalid entry: 2 userids
definename(duns03) alias(gx02) aliasname(dun51);
```

Results: This command is not valid because:

- Alias name *duns01* has an associated account, but no user ID.
- Alias name *duns02* has two user IDs associated with it.

END command

Use the END command to end an Information Exchange session. The message response record associated with the END command is the RETURN record.

The END command has no parameters. Type and enter it as follows:

end;

NOTE: If you specify AUTOEND(Y) in the TRANSMIT command in basein.pro to tell Expedite Base/AIX to end the Information Exchange session automatically, you cannot specify END in basein.msg. Specifying END in basein.msg results in error 03620 if you use **y** as the AUTOEND value in basein.pro.

GETMEMBER command

Use the GETMEMBER command to copy a member from an existing Information Exchange library to an Information Exchange user's mailbox. When the member is available in the mailbox, specify a RECEIVE command as the next command to receive it from the mailbox. When you issue a GETMEMBER command, the library member may not be immediately available in your mailbox. If you do not receive the member immediately, do not issue another GETMEMBER command if the first one was successful. You know it was successful if there was a RETURN(00000) in the output file. Try to receive the member in a subsequent session.

If you leave the destination blank in the GETMEMBER command, the default is your own mailbox. If you do not specify the MSGNAME, MSGSEQNO, and CLASS parameters, the values default to the ones the member had when it was stored in the library.

You may get a warning record on a GETMEMBER command if you don't have access to the library, if the library does not exist, or if the member does not exist in the library. You also cannot get a member from a library on another Information Exchange system. You can get a member from a library on your Information Exchange system into a mailbox for a user on another Information Exchange system.

The message response record associated with the GETMEMBER command is the RETURN record.

The following example shows the syntax of the GETMEMBER command:

```
getmember
library(library name)
member(member name)
owner(library account)
account(account)
userid(user ID)
```

or

```
sysid(system ID)
account(account) userid(user ID)
```

or

```
alias(alias)
aliasname(alias name)
```

or

```
listname(list name)
msgname(message name) msgseqno(message sequence number)
class(class) charge(1|3|5|6)
ack(blank|a|b|c|d|e|f|r) retain(retention period);
```

library

Indicates the library from which Information Exchange copies the member. Use 1 to 8 alphanumeric characters.

member

Indicates the library member Information Exchange copies from the library. Use 1 to 8 alphanumeric characters.

owner

Indicates the account of the library owner. The owner account is used to distinguish between libraries with the same name belonging to different accounts. The default is your own account. Use 1 to 8 alphanumeric characters.

sysid

Indicates the system ID of an Information Exchange user receiving the member. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another Information Exchange system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the account of an Information Exchange user receiving the member. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters. The default is your own account.

userid

Indicates the user ID of an Information Exchange user receiving the member. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters. The default is your own user ID.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user receiving the member.

blank	An alias name was not used. This is the default.
gxxx	Global alias table, where *xxx* identifies a 1- to 3-character table name.
oxxx	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
pxxx	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in either of two ways:

- Using Information Exchange Administration Services (see *Information Exchange Administration Services User's Guide*)
- Using the DEFINEALIAS command ("DEFINEALIAS command" on page 178)

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user receiving the member. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

listname

Indicates the name of a list of accounts and user IDs. Expedite Base/AIX uses this field to identify a list of users receiving the member. Use 1 to 8 alphanumeric characters.

msgname

Indicates the name you assign the file as an identifier. This name will override the MSGNAME used by the originator when this member was put into the library. If you do not specify a MSGNAME, the default is the MSGNAME specified when the member was stored in the library. Use 1 to 8 alphanumeric characters.

msgseqno

Indicates the control number you assign the file. This number will override the MSGSEQNO used by the originator when this member was put into the library. If you do not specify a MSGSEQNO, the default is the MSGSEQNO specified when the member was stored in the library. Use 1 to 5 alphanumeric characters.

class

Indicates the user class you assign to the files. A receiver can use this name to receive only files of this class. This user class will override the class used by the originator when this member was put into the library. If you do not specify a class, the default is the user class specified when the member was stored in the library. Use 1 to 8 alphanumeric characters.

charge

Indicates to Information Exchange how the receiver wants to pay the receive charge.

1	Indicates the receiver pays the receive charge.
3	Indicates the receiver pays the receive charge if agreed to by the receiver. If not, the library owner and the receiver split the charges, if agreed to by the library owner. Otherwise, the library owner pays all the charges.
5	Indicates the library owner pays the receive charge if agreed to by the library owner. Otherwise, the receiver pays all charges. This is the default value.
6	Indicates the library owner pays the receive charge.

ack

Indicates the type of acknowledgment you want to receive. Information Exchange puts these acknowledgments in your mailbox.

blank	No acknowledgment.
a	Information Exchange creates only purge acknowledgment.
b	Information Exchange creates both receipt and delivery acknowledgments.
c	Information Exchange creates both receipt and purge acknowledgments.
d	Information Exchange creates only delivery acknowledgments.
e	Information Exchange creates either purge or delivery acknowledgments.
f	Information Exchange creates receipt acknowledgments and either purge or delivery acknowledgments.
r	Information Exchange creates only receipt acknowledgments.

For more information, refer to “Using acknowledgments” on page 268.

NOTE: If the library owner is paying for the receive charges for the member, the library owner receives the acknowledgment. Otherwise, the individual who issued the GETMEMBER request receives the acknowledgment.

retain

Indicates the number of days Information Exchange keeps the file in the mailbox if no one receives it. Valid values are **blank** and **0** through **180**.

The maximum retention period and the default retention period can be different on different Information Exchange systems. These periods are system dependent. The default retention period is determined when Information Exchange is installed. For installations in the U.S., the default retention period is 30 days. Contact your marketing representative for more information on these values.

If you specify **0** or **blank**, Information Exchange retains the file for the default retention period. If you specify a retention period that is longer than the maximum retention period for your system, Information Exchange retains the file for the default retention period.

GETMEMBER command example

The following is an example of the GETMEMBER command:

```
getmember owner(act1) library(mylib)
member(book1)
account(act2) userid(user01)  #put in mailbox for act2/user01
class(book);                  #override sender's class
```

Results: This command gets member *book1* from the library *mylib* under account *act1*. It places the member in the mailbox for account *act2* and user ID *user01*, with a user class of *book*.

LIST command

Use the LIST command to create a distribution list of account and user IDs when sending and receiving files. The message response record associated with the LIST command is the RETURN record.

The following example shows the syntax of the LIST command. The dots between lines indicate that you can list as many accounts, user IDs, and aliases as necessary:

```
list
listname(list name) function(n|a|d|e) listtype(t|p|a|g)
account(account 1) userid(user ID 1)
.    .
.    .
alias(alias 1) aliasname(alias name 1)
.    .
.    .
alias(alias n) aliasname(alias name n)
.    .
.    .
sysid(system ID n) account(account n) userid(user ID n);
```

A LIST command can include as many ALIAS and ALIASNAME entries or ACCOUNT and USERID entries as required to create the list. Although you can generally specify parameters in any order, the LIST command has the following constraints:

- LIST command entries should include either an ACCOUNT and USERID; a SYSID, ACCOUNT, and USERID; or ALIAS and ALIASNAME. You must pair these entries correctly. For example, you must specify an ACCOUNT parameter next to a USERID parameter.
- If you specify the SYSID parameter, you must specify it either before the ACCOUNT and USERID parameters to which it belongs or between them. It cannot follow the ACCOUNT and USERID parameters.
- At least one list entry is required for functions a (add entries), d (delete entries), or n (new list). No entries entries are permitted for function e (erase entire list).

ATTENTION: If you use the LIST command with session-level recovery, do not use the same distribution list name on more than one LIST command.

listname

Indicates the name of the list you wish to define. Use 1 to 8 alphanumeric characters.

function

Indicates the type of operation you want Expedite Base/AIX to perform on the list.

n Create a new list using the name specified in the LISTNAME parameter and add the entries to the list.

ATTENTION: If you specify FUNCTION(N) to create a new list, and there is already an existing list by that name, you will overwrite the contents of the existing list with the new list that you are defining.

a Add the following entries to the list.

d Delete the following entries from the list. If you include all the names from the list, you get the same result as with option **e**.

e Erase the entire list.

listtype

Indicates the list type.

t The list is temporary; it goes away when your session ends. This is the default value.

p The list is a permanent, private list.

a The list is a permanent list accessible to all members of your account.

g The list is a permanent list with account level grouping. This type of list is used to limit communication with other users. For more information, see *Information Exchange Administration Services User's Guide*.

NOTE: If you want the table to exist after the session, you must specify an option other than **t**.

account

Indicates the account of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user.

blank	An alias name was not used. This is the default.
g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in either of two ways:

- Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*)
- Using the DEFINEALIAS command (see "DEFINEALIAS command" on page 812)

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of an Information Exchange user. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another Information Exchange system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

LIST command example

The following is an example of the LIST command:

```
list listname(mylist) function(n)
account(act1) userid(user01)
sysid(eur) account(act2) userid(user02)
alias(gtbl) aliasname(alias1)
account(act1) userid(user03);
```

Results: This command creates a list called *mylist* with four addresses. Two of the addresses are identified by account and user ID. One address lists system, account, and user ID. The other gives an alias; since no LISTTYPE parameter was specified, the list is temporary and is discarded when the session ends.

LISTLIBRARIES command

Use the LISTLIBRARIES command to request a list of Information Exchange libraries. Expedite Base/AIX returns a list of libraries to which you have access and that meet the criteria you specified on the command.

The message response records associated with the LISTLIBRARIES command are the LIBRARYLIST and RETURN records.

The syntax of the LISTLIBRARIES command is:

listlibraries

```
authority(w|r)
selection(a|c)
owner(library owning account);
```

`authority`

Indicates the user's access authority for the list of libraries requested.

w	Write or update access authority. This is the default.
r	Read access authority.

If SELECTION(C) is specified, this parameter is ignored.

`selection`

Indicates the level of library search.

a	A list of libraries with a specific owning account. This is the default.
c	A complete list of libraries.

`owner`

Indicates the owning account to use when SELECTION(A) is specified. The default is the user's account. If SELECTION(C) is specified, this parameter is ignored. Use 1 to 8 alphanumeric characters.

LISTLIBRARIES command example

```
listlibraries;
```

Results: Expedite Base/AIX returns a list of all libraries to which you have access.

LISTMEMBERS command

Use the LISTMEMBERS command to receive a list of members within an Information Exchange library.

The message response records associated with the LISTMEMBERS command are the MEMBERLIST and RETURN records.

The syntax of the LISTMEMBERS command is:

listmembers

`owner(`*`library owning account`*`)`

`library``(`*`library name`*`);`

`owner`

Indicates the library owning account. The default is the user's account. Use 1 to 8 alphanumeric characters.

`library`

Indicates the library containing the members to be listed. You must have read access to the library. This parameter is required. Use 1 to 8 alphanumeric characters.

LISTMEMBERS command example

```
listmembers library(mylib);
```

Results: Expedite Base/AIX returns a list of members within the library *`mylib`* to which you have access.

PURGE command

Use the PURGE command to delete a specific file from your Information Exchange mailbox. To authorize use of this command, the Service Administrator must use Information Exchange Administration Services to set the Use message purge command field to **y** in your Information Exchange profile.

The message response record associated with the PURGE command is the RETURN record.

The syntax of the PURGE command is:

purge

```
msgkey(message key);
```

msgkey

Indicates the 20-character message identifier of the file to be deleted. This message key can be obtained from the AVAILABLE record for this file in response to a QUERY command.

PURGE command example

```
purge msgkey(abc1de2fg34hijklm5n6);
```

Results: Expedite Base/AIX deletes the file having a message key of *abc1de2fg34hijklm5n6* from your mailbox.

PUTMEMBER command

Use the PUTMEMBER command to add a member to an existing Information Exchange library. The message response records associated with the PUTMEMBER command are the MEMBERPUT record and the RETURN record.

NOTE: You must have authority to update this library or you will receive a system error in your mailbox.

The following example shows the syntax of the PUTMEMBER command:

putmember

library(*library name)*

member(*member name*)

fileid(*file ID*)

owner(*library owner account*)

replace(*n*|*y*)

format(*n*|*y*) class(*class*)

ack(*blank*|*d*)

msgname(*message name*)

msgseqno(*message sequence number*)

datatype(*a*|*b*)

delimited(*n*|*y*)

verify(*n*|*y*)

description(*description*)

translate(*translate table*)

destfile(*destination file*)

destloc(*destination location*);

library

Indicates the name of the library to update. Use 1 to 8 alphanumeric characters.

member

Indicates the name of the member as it should appear in the library. Use 1 to 8 alphanumeric characters.

fileid

Indicates the name of the file you are sending. Use 1 to 128 alphanumeric characters.

owner

Indicates the account of the library owner. The default is your account ID. Use 1 to 8 alphanumeric characters.

replace

Indicates whether Expedite Base/AIX replaces a member with the same name as the file that you are putting into the library.

n — Do not replace the member with the same name as the file that you are putting into the library. This is the default.

NOTE: If you specify REPLACE(N), then if the member already exists, the data is sent but deleted, and you receive a system error message in your mailbox.

y — Replace the member with the same name as the file that you are putting into the library.

format

Indicates whether you want to send the data as a file or e-mail.

n — Send the data without e-mail formatting. This is the default.

y — Format the data as e-mail. This implies fixed 79-byte records. Expedite Base/AIX pads records with blanks. You cannot specify DATATYPE(B) or DELIMITED(Y) with this option.

class

Indicates the user class of the files you are sending. Use 1 to 8 alphanumeric characters.

ack

Indicates the type of acknowledgment you want to receive. Information Exchange puts these acknowledgments in your mailbox.

blank — No acknowledgment

d — Only delivery acknowledgments

For more information, refer to“Using acknowledgments” on page 268.

msgname

Indicates the name you assign the file as an identifier. Use 1 to 8 alphanumeric characters.

msgseqno

Indicates the control number you assign the file. Use 1 to 5 alphanumeric characters.

datatype

Indicates whether the data is text or binary.

a — Text data. This is the default.

b — Binary data.

delimited

Indicates whether data is delimited with newline characters.

y	Indicates the records in the file are delimited by newline characters. Carriage return characters will be inserted before the newlines to create CRLF delimiters. All ASCII text files are to be stored in Information Exchange with the conventional PC record delimiter CRLF. This way, if the file is received by a non-AIX, non-UNIX, or non-XENIX interface, it will be decipherable.
n	Indicates the records are not delimited by newline characters. Insertion of carriage returns will not be done.

The default value is **y** for text data and **n** for binary data.

verify

Indicates whether Expedite Base/AIX verifies that the library is defined and that you have update access before sending the file. You cannot verify a library that resides on another system.

n	Do not verify that the library is defined or that you have access. This is the default.
y	Verify that the library is defined and that you have update access before sending the file. If the library does not exist or if you do not have update access to it, the file is not sent and a WARNING record appears in the output file for this command.

When you use the VERIFY parameter on the PUTMEMBER command, you do not receive a charge for the verification request.

translate

Indicates the name of a translate table that overrides normal ASCII to EBCDIC translation. Use 1 to 8 alphanumeric characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the name of the file containing the translate table. If you do not use this parameter, Expedite Base/AIX uses the translate table specified in the TRANSMIT command. If no TRANSLATE was specified, Expedite Base/AIX will use the default Information Exchange translation table.

description

Provides a free-format description of the file. Use 1 to 79 alphanumeric characters. The description is only available to receiving interfaces that support the CDH. For more information on the CDH, see Appendix B, “Common data header.”

destfile

Indicates the file name to use in the common data header (CDH) as the original file name. If the receiver is using a workstation-based Expedite Base/AIX and specifies ORIGFILE (Y) on the RECEIVE or RECEIVEEDI command, Expedite Base/AIX uses this file name to store the date when it is received. If you specify a file name that is not valid on the receiver's system, AIX uses the file name in the FILEID parameter on the RECEIVE or RECEIVEEDI

command. By default, Expedite Base/AIX determines the original file name from the FILEID parameter on the SEND, SENDEDI, or PUTMEMBER command. Use 1 to 54 characters.

destloc

Indicates the file location to use in the common data header (CDH). When the file is received or the receiver's mailbox is queried, Expedite Base/AIX uses this value in the RECEIVE or AVAILABLE record in the SENDERLOG parameter. By default, Expedite Base/AIX uses the file location in the sender's system. Use 1 to 65 characters.

PUTMEMBER command example

The following is an example of the PUTMEMBER command:

```
putmember library(mylib) member(book2)
fileid(book.scr);
```

Results: This command puts file book.scr as member *book2* in library *mylib*.

QUERY command

Use the QUERY command to return a list of all files in your Information Exchange mailbox. The message response records associated with the QUERY command are the AVAILABLE record and the RETURN record. Expedite Base/AIX writes an AVAILABLE record for each file in your mailbox. For more information, “AVAILABLE record” on page 235.

The following example shows the syntax of the QUERY command:

query

```
cdh(y|n);
```

cdh

Indicates whether you want CDH information included in the response.

y	Include the CDH information in the response. This is the default.
n	Do not include the CDH information in the response.

QUERY command example

The following is an example of the QUERY command:

```
query;
```

Results: This command creates a list of available files in your mailbox. The list contains information from the CDH of each file.

RECEIVE command

Use the RECEIVE command to retrieve files from an Information Exchange mailbox. The message response records associated with the RECEIVE command are the RECEIVED record and the RETURN record.

The following example shows the syntax of the RECEIVE command:

receive

`alias(alias) aliasname(alias name)`

or

`sysid(system ID) account(account) userid(user ID)`

or

`account(account) userid(user ID)`

or

`listname(list name)`

or

`requeued(n|y)`
`fileid(file ID) format(n|y) class(class)`
`archiveid(archive ID) multfiles(n|y) origfile(n|y)`
`recordsize(record size) processlen(c|r|i)`
`autoedi(y|n) ediopt(y|n) delimited(n|y)`
`translate(translate table) removeof(n|y)`
`allfiles(y|n) nonedionly(n|y) msgkey(message key)`
`startdate(starting date) starttime(starting time)`
`enddate(ending date) endtime(ending time) timezone(l|g) wait(wait time);`

NOTE: If you are using supported data compression and receive compressed data, not all parameters are supported. See Appendix E, "Using data compression," for more information.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user from whom you are receiving data.

blank	An alias name was not used. This is the default.
g***xxx***	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o***xxx***	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p***xxx***	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in two ways:

- Using Information Exchange Administration Services (see the *Information Exchange Administration Services User's Guide*)
- Using the DEFINEALIAS command ("DEFINEALIAS command" on page 178.)

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user from whom you are receiving data. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of a user on another system from whom you are receiving data. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the account of an Information Exchange user from whom you are receiving data. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user from whom you are receiving data. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

listname

Indicates the name of a list of accounts and user IDs. Expedite Base/AIX uses this field to identify a list of users from whom you are receiving data. Use 1 to 8 alphanumeric characters.

requeued

Indicates whether Expedite Base/AIX receives only files that have been retrieved from archive.

n	Receive all files in your mailbox using the receive specifications in this command. This is the default value.
y	Receive files retrieved from archive only. If you specify **y**, you cannot specify an Information Exchange SYSID, ACCOUNT, USERID; ALIAS, or ALIASNAME in the RECEIVE command.

fileid

Indicates the name of the file where Expedite Base/AIX places the received data. Use 1 to 128 alphanumeric characters.

format

Indicates whether you want to receive the data as a file or in Expedite Base/AIX e-mail format. “Sending and receiving e-mail” on page 44 for more information.

Expedite Base/AIX ignores this option during EDI processing or record delimiter processing based on the CDH and other options.

n	Do not format the data as e-mail. This is the default value.
y	Format the data as Expedite Base/AIX e-mail. This implies fixed 79byte records.

class

Indicates the user class of the files to receive. You can limit the files you receive to files with this user class. Use 1 to 8 alphanumeric characters.

You can use a question mark as a wildcard for any character or characters. For example, to receive all files with a user class beginning with AB1, type **AB1?**

To receive all files with a user class ending with 999, type **?999**

If you specify *format(y)*, the default user class is *ffmsg001*, which is reserved for e-mail.

If you specify *format(n)*, the default is **blank**, which indicates all user classes.

archiveid

Indicates the archive reference identifier you want Information Exchange to assign to the files you receive. If your Information Exchange profile indicates that archiving is enabled, then Information Exchange will save the file in archive with this archive ID. You can use this archive ID on the ARCHIVEMOVE command later to place a copy of this file back in your mailbox. Use 1 to 8 alphanumeric characters.

multfiles

Determines whether the files you receive are put in separate files or are concatenated into one file.

n Concatenate all received files into a single file. This is the default value.

y Create a new separate file for the second and subsequent files. New files are named by adding a numbered extension starting with .002. The new extension will be added after any existing extension on the file name; any original extension will not be truncated. If more than 999 files are received, the extension becomes four digits: .1000, .1001, .1002, and so on. If more than 9999 files are received, the extension becomes five digits: .10000, .10001, .10002, and so on. If more than 99999 are received, the rest of the files are appended to the file name in FILEID with the extension *.ovf*.

For example, if you specify FILEID(*testmsg*) and three files are received, Expedite Base/AIX names the files as follows:

File 1 = testmsg

File 2 = testmsg.002

File 3 = testmsg.003

If you receive 1000 or more files, files 1001, 1002, and 1003 would be named as follows:

File 1001 = testmsg.1001

File 1002 = testmsg.1002

File 1003 = testmsg.1003

If you receive more than 99999 files, the data in the files after file 99999 is appended to a file with the extension *.ovf*.

origfile

Indicates whether you want to receive data into a file using the original name from the sending system. The original file name must be 54 characters or less.

n Receive all data into the file indicated by FILEID. value. This is the default

y Receive the file into the original file name if the original file name specified in the CDH sender file is valid for an AIX file system.

NOTE: For example, the receiver on Expedite, Expedite Base/AIX uses the sender's original file name, SENDERFILE from the CDH, to store the data received, but it does not use the original location, SENDERLOC, because the results may be undesirable for the receiver. Instead, Expedite Base/AIX uses the location (path). Base/DOS may want to receive a file named config.sys but undesirable results may occur if the original location, c:\, was used by Expedite Base/AIX because c:\config.sys is an important system file on most users' PCs. specified in the FILEID parameter on the RECEIVE or RECEIVEEDI command.

recordsize

Indicates whether Expedite Base/AIX breaks EDI data into fixed-length records by placing a newline in the data at specified intervals. This option is only active if EDIOPT is **n** and AUTOEDI is **y**.

000	Do not insert a newline in the file. This is the default value.
nnn	Insert a newline every *nnn* characters, where *nnn* is a number from 1 to 999.

NOTE: This parameter does not do anything for files sent without a CDH. For more information on the CDH, see Appendix B, "Common data header."

processlen

Controls the processing of length delimiters at the beginning of each record. Expedite Base/AIX uses this parameter only if the CDH indicates that the file is delimited by length delimiters at the beginning of each record.

c	Convert the length delimiters to CRLF delimiters. This is the default value.
r	Remove the length delimiters from the data.
i	Ignore the length delimiters in the data.

NOTE: This parameter does not do anything for files sent without a CDH. For more information on the CDH, see Appendix B, "Common data header."

autoedi

Indicates whether Expedite Base/AIX automatically performs EDI processing for files (as defined in Chapter 6, "Sending and receiving EDI data") if the CDH indicates that the files are EDI-formatted.

y	Automatically perform EDI processing if the CDH indicates that the file is EDI formatted.
n	Do not perform EDI processing for any of the files. This is the default value.

NOTE: This parameter does not do anything for files sent without a CDH. For more information on the CDH, see Appendix B, "Common data header."

ediopt

Indicates whether Expedite Base/AIX adds a newline delimiter at the end of EDI segments. Expedite Base/AIX ignores this option unless the CDH indicates that the received message is EDI data and AUTOEDI is set to **y**.

n	Do not add newline at the end of EDI segments.
y	Add newline at the end of EDI segments. This is the default value.

delimited

Indicates whether the data is delimited with CRLF (carriage-return line-feed, hex 0D0A) characters.

y	Indicates the records in the file are delimited by CRLF characters. All ASCII text files are stored in Information Exchange with the conventional PC record delimiter CRLF. This way, if the file is received by a nonAIX interface, it will be decipherable. The carriage return characters (CR) that occur before a newline (LF) will be deleted from the file when it is received to conform to the AIX line delimiter convention.
n	Indicates the records are not delimited by CRLF characters. The file will be received "as is."

The default value is **y** for ASCII data and **n** for binary data.

translate

Indicates the name of a translate table that overrides normal EBCDIC to ASCII translation. Use 1 to 8 alphanumeric characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the name of the file containing the translate table. If you do not use this parameter, Expedite Base/AIX uses the translate table specified in the TRANSMIT command. If no TRANSLATE was specified, Expedite Base/AIX uses the default Information Exchange translation table.

removeof

Indicates whether Expedite Base/AIX removes the EOF character from the end of a received file. This option is useful when you receive several files into a single file.

n	Do not remove the EOF character. This is the default.
y	Remove the EOF character if it is the last character of a received file.

allfiles

Indicates whether Expedite Base/AIX receives all files that match the RECEIVE specifications or just the first file in the Information Exchange mailbox that matches the RECEIVE specifications.

n	Receive only the first file that matches the RECEIVE specifications.
y	Receive all files that match the RECEIVE specifications. This is the default value.

nonedionly

Specifies that you receive only data other than EDI data.

n	Receive all files from your mailbox that satisfy your RECEIVE request. This is the default.
y	Receive only those files from your mailbox that match the RECEIVE specifications and are identified in the CDH as not having one of the EDI formats or have no CDH.

msgkey

Indicates a unique message key that you can use to receive a specific file from a mailbox. You can get this value from the AVAILABLE record in response to a QUERY command. Use 20 characters.

startdate

Indicates the starting date of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the date the file was sent to Information Exchange must fall within this date range.

Use the *YYMMDD* or *YYYYMMDD* format for the starting date. The default value is determined by Information Exchange and is currently **000102** (January 2, 1900).

If you limit the range to one day and an error occurs while sending the file, files could be left undetected in the mailbox. For example, assume that an error occurs while sending a file on June 11, 1998, and the file is not actually placed in the mailbox until 11:00 a.m. on June 12.

If you issue a RECEIVE command at 8:00 a.m. on June 12, 1998 with a range of 00:00:00 to 24:00:00 on June 11 specified, the file will not be received because it has not yet arrived in the mailbox. If you then issue another RECEIVE command at 8:00 a.m. on June 13, 1998 with a range of 00:00:00 to 24:00:00 on June 12 specified, the file will still not be received because its starting date of June 11 is outside of the specified range.

To avoid missing any files, you should either move the starting date earlier, or periodically check your mailbox.

starttime

Indicates the starting time of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the time the file was sent to Information Exchange must fall within this time range. The format is *HHMMSS*. The default value is determined by Information Exchange and is currently **000000** (00:00:00).

enddate

Indicates the ending date of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the date the file was sent to Information Exchange must fall within this date range. Use the *YYMMDD* or *YYYYMMDD* format for the ending date. The default value is determined by Information Exchange and is currently **420916** (September 16, 2042).

endtime

Indicates the ending time of a time range for files you want to receive from Information Exchange. For a file to qualify to be received, the time the file was sent to Information Exchange must fall within this time range. Use the *HHMMSS* format for the ending time. The default value is determined by Information Exchange and is currently **235347** (23:53:47).

NOTE: Information Exchange Administration Services displays the STARTTIME and ENDTIME as *HHMMSS*, but Information Exchange actually stores a more precise value. Therefore, you should specify a STARTTIME two seconds earlier and an ENDTIME two seconds later than the time shown in baseout.msg and in Information Exchange Administration Services.

timezone

Indicates the time zone reference in the STARTTIME and ENDTIME parameters.

l	Your local time, as specified on the TIMEZONE parameter of the IDENTIFY command. This is the default.
g	Greenwich mean time (GMT).

wait

Indicates the amount of time Expedite should wait for data to arrive in the mailbox. The format is *MMSSS* where *MM* is from 02 to 05 minutes and *SS* is from 00 to 59 seconds. The maximum allowed time is 0500 (5 minutes). The minimum time is 0200 (2 minutes). This parameter is supported only if you specify COMMTYPE(A) or (T) on the transmit command in basein.pro.

NOTES:

1. If you are using TCP/IP communications and the value specified for the TIMEOUT parameter on the TCPCOMM command is less than the value specified for the WAIT parameter, the value in TIMEOUT is used.
2. When a RECEIVE is specified with the WAIT parameter, only the first file that meets the criteria specified on the RECEIVE will be received. You need to specify an additional RECEIVE command without the WAIT parameter to receive any subsequent files.

RECEIVE command example

The following is an example of the RECEIVE command:

```
receive fileid(price.fil) class(prices)
recordsize(80) ediopt(n)
startdate(980701) starttime(000000)
enddate(981231) endtime(180000);
WAIT
```

Results: Files in your Information Exchange mailbox with a user class of *prices* are received in price.fil. If more than one such file exists in your mailbox, it is appended to price.fil. Expedite Base/AIX inserts a CRLF character after every 80 characters, and does not add newline delimiters at the end of EDI segments. Only files sent between 00:00:00 hours (your local time) July 1, 1998 and 18:00:00 hours December 31, 1998, are received from your mailbox.

RECEIVEEDI command

Use the RECEIVEEDI command to retrieve EDI-formatted files from an Information Exchange mailbox. Using this command does not guarantee that you receive only EDI data. For more information on receiving EDI data, “Receiving EDI data” on page 98.

You can use the CLASS parameter to make sure you are receiving EDI data. To do so, you and your trading partner must agree on a name for the EDI data. Your trading partner must specify that name in the user CLASS parameter when sending the data, and you must specify that name in the CLASS parameter on the RECEIVEEDI command. The following example shows a RECEIVEEDI command using the CLASS parameter, which can be used to receive all the files with the class *editest* from the mailbox:

```
receiveedi fileid(edidata.fil) class(editest);
```

You can also use the EDIONLY parameter to receive the data marked in the CDH as EDI data. If your trading partner sent the data with an interface that supports the DFORMAT field in the CDH, then the CDH will be marked properly.

The following example shows a RECEIVEEDI command using the EDIONLY parameter. You can use this parameter to receive all the mailbox files that are marked in the CDH as EDI data:

```
receiveedi fileid(edidata.fil) edionly(y);
```

If you receive a file whose CDH indicates that the file is not EDI data, Expedite Base/AIX receives the file in the format indicated by the CDH. If the data type is not indicated in the CDH, or no CDH is present, Expedite Base/AIX examines the data to determine the EDI data type. If the file is not recognizable as EDI data, Expedite Base/AIX receives the file without reformatting records. For more information on the CDH, see Appendix B, “Common data header.”

The message response records associated with the RECEIVEEDI command are the RECEIVED record and the RETURN record.

The following example shows the syntax of the RECEIVEEDI command:

receiveedi

alias(*alias*) aliasname(*alias name*)

or

sysid(*system ID*) account(*account*) userid(*user ID*)

or

account(*account*) userid(*user ID*)

or

listname(*list name*)

or

requeued(*n*|*y*)
fileid(*file ID*) class(*class*) archiveid(*archive ID*)
multfiles(*n*|*y*) origfile(*n*|*y*)
ediopt(*y*|*n*|*f*) recordsize(*record size*) translate(*translate table*)
allfiles(*y*|*n*) edionly(*n*|*y*) msgkey(*message key*)

```
startdate(starting date) starttime(starting time)
enddate(ending date) endtime(ending time) timezone(l|g) wait(wait time);
```

NOTE: If you are using supported data compression and receive compressed data, not all parameters are supported. See Appendix E, “Using data compression,” for more information.

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user from whom you are receiving data.

blank	An alias name was not used. This is the default.
g***xxx***	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o***xxx***	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p***xxx***	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in eithe of two ways:

- Using Information Exchange Administration Services (see the *Information Exchange Administration Services User’s Guide*)
- Using the DEFINEALIAS command (see “DEFINEALIAS command” on page 178)

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user from whom you are receiving data. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of a user on another system from whom you are receiving data. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the account of an Information Exchange user from whom you are receiving data. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user from whom you are receiving data. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

listname

Indicates the name of a list of accounts and user IDs. Expedite Base/AIX uses this field to identify a list of users from whom you are receiving data. Use 1 to 8 alphanumeric characters.

requeued

Indicates whether Expedite Base/AIX receives files retrieved from archive only.

n	Receive any file in your mailbox. This is the default value.
y	Receive files retrieved from archive only. If you specify **y**, you cannot specify an Information Exchange source ID such as ACCOUNT and USERID or ALIAS and ALIASNAME.

fileid

Indicates the name of the file where Expedite Base/AIX places the received data. Use 1 to 128 alphanumeric characters.

class

Indicates the user class of the files to receive. You can limit the files you receive to files with this user class.

You can use a question mark as a wildcard for any character or characters. For example, to receive all files whose user class begins with AB1, type **AB1?**

To receive all files with a user class ending with 999, type **?999**

The default value is blank, which indicates all user classes. It is recommended that you use the CLASS parameter when you receive files.

If the EDI data is sent with a SENDEDI command using the default user class, it arrives in your Information Exchange mailbox classified as follows:

#EE	EDIFACT data
#EU	UN/TDI data
#E2	X12 data
#EC	UCS data

To receive only the EDI files sent with the default user class, type **#E?**

archiveid

Indicates the archive reference identifier you want Expedite Base/AIX to assign to the files you receive. If your Information Exchange profile indicates that archiving is enabled, then Information Exchange will save the file in archive with this archive ID. You can use this archive ID on the ARCHIVEMOVE command later to place a copy of this file back in your mailbox. Use 1 to 8 alphanumeric characters.

multfiles

Determines whether the files you receive are put in separate files or are concatenated into one file.

n	Concatenate all received files into a single file. This is the default value.
y	Create a new separate file for the second and subsequent files. New files are named by adding a numbered extension starting with .002. The new extension will be added after any existing extension on the file name; any original extension will not be truncated. If more than 999 files are received, the extension becomes four digits: .1000, .1001, .1002, and so on. If more than 9999 files are received, the extension becomes five digits: .10000, .10001, .10002, and so on. If more than 99999 are received, the rest of the files are appended to the file name in FILEID with the extension **.***ovf*.

For example, if you specify FILEID(*testmsg*) and three files are received, Expedite Base/AIX names the files as follows:

File 1 = testmsg

File 2 = testmsg.002

File 3 = testmsg.003

If you receive 1000 or more files, files 1001, 1002, and 1003 would be named as follows:

File 1001 = testmsg.1001

File 1002 = testmsg.1002

File 1003 = testmsg.1003

If you receive more than 99999 files, the data in the files after file 99999 is appended to a file with the extension *.ovf*.

origfile

Indicates whether you want to receive data into a file using the original name from the sending system. The original file name must be no longer than 54 characters.

n	Receive all data into the file indicated by FILEID. This is the default value.
y	Receive the file into the original file name if the original file name specified in the CDH sender file is valid for an AIX file system. NOTE: Expedite Base/AIX uses the sender's original file name, SENDERFILE from the CDH, to store the data received, but it does not use the original location, SENDERLOC, because the results may be undesirable for the receiver. For example, the receiver on Expedite Base/DOS may want to receive a file named config.sys but undesirable results may occur if the original location, c:\, was used by Expedite Base/AIX because c:\config.sys is an important system file on most user's PCs. Instead, Expedite Base/AIX uses the location (path) specified in the FILEID parameter on the RECEIVE or RECEIVEEDI command.

ediopt

Indicates whether or not Expedite Base/AIX should split EDI data at the end of EDI segments.

y	Split records at the end of EDI segments. If the CDH indicates that the information received is not EDI data, this option is ignored. This is the default.
n	Do not split records at the end of EDI segments. If the CDH indicates that the information received is not EDI data, this option is ignored.
f	Attempt to split records at the end of EDI segments, regardless of what the CDH indicates. If the data is not valid CDH data, Expedite Base/AIX issues a warning.

recordsize

Indicates whether Expedite Base/AIX breaks EDI data into fixed-length records by placing newline characters in the data at specified intervals. This option is only active if EDIOPT is **n**.

000	Do not insert newline characters in the file. This is the default value.
nnn	Insert newline characters every *nnn* characters, where *nnn* is a number from 1 to 999.

NOTE: This parameter does not do anything for files sent without a CDH. For more information on the CDH, see Appendix B, “Common data header.”

translate

Indicates the name of a translate table that overrides normal EBCDIC to ASCII translation. Use 1 to 8 alphanumeric characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the name of the file containing the translate table. If you do not use this

parameter, Expedite Base/AIX uses the translate table specified in the TRANSMIT command. If no TRANSLATE was specified, Expedite Base/AIX uses the default Information Exchange translation table. The default is the translate table specified in your profile.

allfiles

Indicates whether Expedite Base/AIX receives all files that match the RECEIVEEDI specifications or just the first file in Information Exchange mailbox that matches the RECEIVEEDI specifications.

n	Receive only the first file that matches the RECEIVEEDI specifications.
y	Receive all files that match the RECEIVEEDI specifications. This is the default value.

ediionly

Specifies that you receive only EDI data.

n	Receive all messages from your mailbox that satisfy your RECEIVEEDI request. If the CDH indicates a file does not contain EDI data, Expedite Base/AIX receives the file, using the format indicated in the CDH. If a file does not have a CDH and is not recognizable as EDI data, Expedite Base/AIX receives the file without reformatting the records.
y	Receive only those messages from your mailbox that are identified in the CDH as having one of the EDI formats.

msgkey

Indicates a unique message key you can use to receive a specific file from the mailbox. You can get this value from the AVAILABLE record in response to a QUERY command. Use 20 characters.

startdate

Indicates the starting date of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the date the file was sent to Information Exchange must fall within this date range.

Use the *YYMMDD* or *YYYYMMDD* format for the starting date. The default value is determined by Information Exchange and is currently **000102** (January 2, 1900).

If you limit the range to one day and an error occurs while sending the file, files could be left undetected in the mailbox. For example, assume that an error occurs while sending a file on June 11, 1998, and the file is not actually placed in the mailbox until 11:00 a.m. on June 12.

If you issue a RECEIVEEDI command at 8:00 a.m. on June 12, 1998 with a range of 00:00:00 to 24:00:00 on June 11 specified, the file will not be received because it has not yet arrived in the mailbox. If you then issue another RECEIVEEDI command at 8:00 a.m. on June 13, 1998 with a range of 00:00:00 to 24:00:00 on June 12 specified, the file will still not be received because its starting date of June 11 is outside of the specified range.

To avoid missing any files, you should either move the starting date earlier, or periodically check your mailbox.

starttime

Indicates the starting time of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the time the file was sent to Information Exchange must fall within this time range. The format is *HHMMSS*. The default value is determined by Information Exchange and is currently **000000** (00:00:00).

enddate

Indicates the ending date of a time range for the files you want to receive from Information Exchange. For a file to qualify to be received, the date the file was sent to Information Exchange must fall within this date range. Use the *YYMMDD* or *YYYYMMDD* format for the ending date. The default value is determined by Information Exchange and is currently **420916** (September 16, 2042).

endtime

Indicates the ending time of a time range for files you want to receive from Information Exchange. For a file to qualify to be received, the time the file was sent to Information Exchange must fall within this time range. Use the *HHMMSS* format for the ending time. The default value is determined by Information Exchange and is currently 235347 (23:53:47).

NOTE: Information Exchange Administration Services displays the STARTTIME and ENDTIME as *HHMMSS*, but Information Exchange actually stores a more precise value. Therefore, you should specify a STARTTIME two seconds earlier and an ENDTIME two seconds later than the time shown in baseout.msg and in Information Exchange Administration Services.

timezone

Indicates the time zone reference in the STARTTIME and ENDTIME parameters.

l	Your local time, as specified on the TIMEZONE parameter of the IDENTIFY command. This is the default.
g	Greenwich mean time (GMT).

wait

Indicates the amount of time Expedite Base/AIX should wait for data to arrive in the mailbox. The format is *MMSS* where *MM* is from 02 to 05 minutes and *SS* is from 00 to 59 seconds. The maximum allowed time is 0500 (5 minutes). The minimum time is 0200 (2 minutes). This parameter is supported only if you specify *COMMTYPE(A)* or (T) on the transmit command in basein.pro.

NOTES:

1. If you are using TCP/IP communications and the value specified for the TIMEOUT parameter on the TCPCOMM command is less than the value specified for the WAIT parameter, the value in TIMEOUT is used.
2. When a RECEIVE is specified with the WAIT parameter, only the first file that meets the criteria specified on the RECEIVE will be received. You need to specify an additional RECEIVE command without the WAIT parameter to receive any subsequent files.

RECEIVEEDI command example

The following is an example of the RECEIVEEDI command:

```
receiveedi fileid(orders.fil) class(#e2)
ediopt(n) recordsize(80)
startdate(980701) starttime(00000)
enddate(981231) endtime(180000);
```

Results: Files with a user class of *#e2* are received in file orders.fil with a newline character added after every 80 characters. If more than one such file exists in the mailbox, Expedite Base/AIX appends it to orders.fil. Only files sent between 00:00:00 hours (your local time) July 1, 1998 and 18:00:00 hours December 31, 1998, are received from your mailbox.

SEND command

Use the SEND command to send a file to Information Exchange. The message response records associated with the SEND command are the SENT record and the RETURN record.

The following example shows the syntax of the SEND command:

send

alias(*alias*) **aliasname**(*alias name*)

or

sysid(*system ID*) **account**(*account*) **userid**(*user ID*)

or

account(*account*) **userid**(*user ID*)

or

listname(*list name*)

fileid(*file ID*)
format(*n*|*y*)
class(*class*) mode(*blank*|*t*) priority(*blank*|*i*|*p*)
charge(*1*|*2*|*3*|*4*|*5*|*6*) ack(*blank*|*a*|*b*|*c*|*d*|*e*|*f*|*r*) msgname(*msg name*)
msgseqno(*message sequence number*) datatype(*a*|*b*)
delimited(*n*|*y*) verify(*n*|*y*|*f*) description(*description*)
recfm(*f*|*v*) lrecl(*record length*) retain(*time*)
translate(*translate table*) compress(*n*|*y*|*t*)
destfile(*destination file*) destloc(*destination location*)
selectrcv(*f*|*n*|*blank*);

alias

Indicates the table type and table name of an alias table. Expedite Base/AIX uses this field in conjunction with the ALIASNAME parameter to identify the user to whom you are sending data.

blank	An alias name was not used. This is the default.
g***xxx***	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o***xxx***	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p***xxx***	Private alias table, where *xxx* identifies a 1- to 3-character table name.

If you specify an ALIAS parameter, you must specify an ALIASNAME parameter. Use 1 to 4 alphanumeric characters.

NOTE: You can create and maintain alias tables in two ways:

- Using Information Exchange Administration Services (see the Information Exchange Administration Services User's Guide)
- Using the DEFINEALIAS command ("DEFINEALIAS command" on page 178)

aliasname

Indicates an alias name in the alias table you specified in the ALIAS parameter. Expedite Base/AIX uses this field in conjunction with the ALIAS parameter to identify the user to whom you are sending data. If you specify an ALIASNAME parameter, you must specify an ALIAS parameter. Use 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of an Information Exchange user to whom you are sending data. You need the system ID only if you specify the ACCOUNT and USERID parameters for a user on another Information Exchange system. If you specify a SYSID parameter, you must specify the ACCOUNT and USERID parameters. Use 1 to 3 alphanumeric characters.

account

Indicates the account of an Information Exchange user to whom you are sending data. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters.

userid

Indicates the user ID of an Information Exchange user to whom you are sending data. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters.

listname

Indicates the name of a list of accounts and user IDs. Expedite Base/AIX uses this field to identify a list of users to whom you are sending data. Use 1 to 8 alphanumeric characters.

fileid

Indicates the name of the file you are sending. Use 1 to 128 alphanumeric characters.

format

Indicates whether you want to send the data as a file or in Expedite Base/AIX e-mail format. "Sending and receiving e-mail" on page 44 for more information.

n	Do not format the data as e-mail. This is the default value.
y	Format the data as Expedite Base/AIX e-mail. This implies fixed 79-byte records.

class

Indicates the user class of the files you are sending. A receiver can use this name to receive only files of this class. Use 1 to 8 alphanumeric characters. If you specify FORMAT(Y), this defaults to ffmsg001. Otherwise, it defaults to blank.

mode

Indicates whether the file is a test file or normal.

blank	Normal file. This is the default.
t	Testmode file.

priority

Indicates the class of delivery service for this file.

blank	Normal priority. This is the default.
i	Express delivery to those users who have continuous receive capability and are currently in session with Information Exchange. This file will be received before any other files with a lower priority. (Expedite Base/AIX does not support continuous receive capability.)
p	High priority.

charge

Indicates to Information Exchange how the sender wants to pay the file charges.

1	The receiver pays all charges.
2	The receiver pays all charges, if agreed to by the receiver. Otherwise, the sender and receiver split the charges.
3	The receiver pays all charges, if agreed to by the receiver. If not, the sender and receiver split the charges, if agreed to by the receiver. Otherwise, the sender pays all charges. This is the default value.
4	The sender and receiver split the charges, if agreed to by the receiver. Otherwise, the sender pays all charges.
5	Indicates the sender and receiver split the charges.
6	Indicates the sender pays all charges.

ack

Indicates what type of acknowledgment you want to receive. Information Exchange puts these acknowledgments in your mailbox.

blank	No acknowledgment.
a	Information Exchange creates only purge acknowledgments.
b	Information Exchange creates both receipt and delivery acknowledgments.
c	Information Exchange creates both receipt and purge acknowledgments.
d	Information Exchange creates only delivery acknowledgments.
e	Information Exchange creates either purge or delivery acknowledgments.
f	Information Exchange creates receipt acknowledgments and either purge or delivery acknowledgments.
r	Information Exchange creates only receipt acknowledgments.

For more information, refer to “Using acknowledgments” on page 268.

msgname

Indicates the name you assign the file as an identifier. Use 1 to 8 alphanumeric characters.

msgseqno

Indicates the control number you assign the file. Use 1 to 5 alphanumeric characters.

datatype

Indicates whether the data is text or binary. For more information on text and binary data, see Chapter 5, “Sending and receiving files.”

a	Text data. This is the default.
b	Binary data.

delimited

Indicates whether the data is delimited with newline characters.

y	Indicates the records in the file are delimited by newline characters. Carriage-return characters will be inserted before the newlines to create CRLF delimiters. All text files are to be stored in Information Exchange with the conventional PC record delimiter CRLF. This way, if the file is received by a non-AIX interface, it will be decipherable.
n	Indicates the records are not delimited by newline characters. Insertion of carriage returns will not be done.

The default value is **y** for text data and **n** for binary data.

verify

Indicates whether Expedite Base/AIX verifies that the receiver exists before sending the file.

n	Do not verify that the receiver exists. This is the default value.
y	Verify that the receiver exists before sending the data.
f	Verify that the receiver exists. If Information Exchange cannot verify whether the receiver exists (for example, if the receiver is on another Information Exchange system), send the data anyway.

If you request verification for an invalid destination, you incur an Information Exchange message charge for the verification. If you request verification for a valid destination or a destination that cannot be verified, you do not incur a message charge for the verification. See "Understanding validations, payment levels, and authorizations" on page 276 for detailed information on charges.

description

Provides a free-format description of the file. Use 1 to 79 alphanumeric characters. The description is only available to receiving interfaces that support the CDH. For more information on the CDH, see Appendix B, "Common data header."

recfm

Indicates the record format of the file. This parameter is primarily for documentation purposes. Expedite Base/AIX places this information in the CDH. Except for Expedite Base/VM, all Expedite programs ignore this parameter when receiving files. However, Expedite Base/AIX can use it to format records when receiving files.

f	File has a fixed record format.
v	File has a variable record format.

lrecl

Indicates the record length of the file. This parameter is primarily for documentation purposes. The record length field identifies record length across systems. Expedite Base/AIX places this information in the CDH. Except for Expedite Base/VM, all Expedite programs ignore this parameter when receiving files. However, Expedite Base/AIX can use it to format records when receiving files. Valid values are **1** to **65,535.** The default value is **blank**.

retain

Indicates the number of days Information Exchange keeps the file in the mailbox if no one receives it. Valid values are **blank** and **0** through **180**.

The maximum retention period and the default retention period can be different on different Information Exchange systems. These periods are system-dependent. The default retention period is determined when Information Exchange is installed. For installations in the U.S., the default retention period is 30 days. Contact your marketing representative for more information on these values.

If you specify **0** or **blank**, Information Exchange retains the file for the default retention period. If you specify a retention period that is longer than the maximum retention period for your system, Information Exchange retains the file for the default retention period.

`translate`

Indicates the name of a translate table that overrides normal EBCDIC to ASCII translation. Use 1 to 8 alphanumeric characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the name of the file containing the translate table. If you do not use this parameter, Expedite Base/AIX uses the translate table specified in the TRANSMIT command. If no TRANSLATE was specified, Expedite Base/AIX uses the default Information Exchange translate table.

`compress`

Indicates whether the specified file should be compressed.

n	Do not compress the specified file. This is the default.
y	Compress the specified file.
t	For each sender/receiver pair, use the setting of the COMPRESS parameter (*y* or *n*) indicated in the cplookup.tbl file.

You must have the supported data compression software installed if you specify `COMPRESS(Y)` or `COMPRESS(T)`.

Some of the parameters of the command may not apply when sending compressed data. See Appendix E, “Using data compression,” for more information.

`destfile`

Indicates the file name to use in the common data header (CDH) as the original file name. If the receiver is using a workstation-based Expedite Base/AIX and specifies ORIGFILE (Y) on the RECEIVE or RECEIVEEDI command, Expedite Base/AIX uses this file name to store the data when it is received. If you specify a file name that is not valid on the receiver's system, Expedite Base/AIX uses the file name in the FILEID parameter on the RECEIVE or RECEIVEEDI command. By default, Expedite Base/AIX determines the original file name from the FILEID parameter on the SEND, SENDEDI, or PUTMEMBER command. Use 1 to 54 characters.

`destloc`

Indicates the file location to use in the common data header (CDH). When the file is received or the receiver's mailbox is queried, Expedite Base/AIX uses this value in the RECEIVE or AVAILABLE record in the SENDERLOG parameter. By default, Expedite Base/AIX uses the file location in the sender's system. Use 1 to 65 characters.

selectrcv

Indicates force receive search criteria.

f	Identifies this file for selective receive only.
n	Turns off the selective-receive-only option. No special criteria is required to receive this file. The file can be received by a blanket receive.
blank	Turns off the selective-receive-only option. No special criteria is required to receive this file. The file can be received by a blanket receive. This is the default.

When f is selected in SELECTRCV, the file being sent is marked in Information Exchange and cannot be received by a blanket receive. The file can only be received if the receiver specifies one of the following:

- Sender ID (account ID and user ID)
- Message class
- Message key

SEND command example

The following is an example of the SEND command:

```
send fileid(test.fil) alias(ptb3) aliasname(mary) class(question);
```

Results: This command sends file test.fil to alias name *mary* in alias table *ptb3*. The file has a user class of question. The alias name mary must be defined in *ptb3* so that Information Exchange can resolve the account and user ID.

SENDEDI command

Use the SENDEDI command to send an EDI-formatted file to Information Exchange. The file can contain X12, UCS, EDIFACT, or UN/TDI data; or any combination of these. For more information on sending EDI data, **see Chapter 6, "Sending and receiving EDI data."**

The message response records associated with the SENDEDI command are the SENT and RETURN records.

The following example shows the syntax of the SENDEDI command:

sendedi

fileid(*file ID*) mode(*blank*|*t*)
priority(*blank*|*i*|*p*) charge(*3*|*1*|*2*|*4*|*5*|*6*) ack(***blank***|*a*|*b*|*c*|*d*|*e*|*f*|*r*)
msgname(*message name*) msgseqno(*message sequence number*) class(*class*)
verify(*n*|*y*|*f*|*c*|*g*) description(*description*) recfm(*f*|*v*)
lrecl(*record length*) retain(*time*) translate(*translate table*)
compress(*n*|*y*|*t*) destfile(*destination file*) destloc(*destination location*)
selectrcv(*f*|*n*|blank);

fileid

Indicates the name of the file you are sending. Use 1 to 128 alphanumeric characters.

mode

Indicates a test or normal file.

blank	Normal file. This is the default.
t	Test-mode file.

priority

Indicates the class of delivery service for the file.

blank	Normal priority. This is the default.
i	Express delivery to those users who have the continuous receive capability and are currently in session with Information Exchange. This file will be received before any other files with a lower priority. (Expedite Base/AIX does not support continuous receive capability.)
p	High priority.

charge

Indicates to Information Exchange how the sender wants to pay the file charges.

1	The receiver pays all charges.
2	The receiver pays all charges, if agreed to by the receiver. Otherwise, the sender and receiver split the charges.
3	The receiver pays all charges, if agreed to by the receiver. If not, the sender and receiver split the charges, if agreed to by the receiver. Otherwise, the sender pays all charges. This is the default value.
4	The sender and receiver split the charges, if agreed to by the receiver. Otherwise, the sender pays all charges.
5	Indicates the sender and receiver split the charges.
6	Indicates the sender pays all charges.

ack

Indicates what type of acknowledgment you want to receive. Information Exchange puts these acknowledgments in your mailbox.

blank	No acknowledgment.
a	Information Exchange creates only purge acknowledgments.
b	Information Exchange creates both receipt and delivery acknowledgments.
c	Information Exchange creates both receipt and purge acknowledgments.
d	Information Exchange creates only delivery acknowledgments.
e	Information Exchange creates either purge or delivery acknowledgments.
f	Information Exchange creates receipt acknowledgments and either purge or delivery acknowledgments.
r	Information Exchange creates only receipt acknowledgments.

For more information, refer to “Using acknowledgments” on page 268.

msgname

Indicates the name you assign the file as an identifier. Use 1 to 8 alphanumeric characters.

See “Providing a message name (MSGNAME)” on page 96 for more information on the default values of MSGNAME.

msgseqno

Indicates the control number you assign the file. Use 1 to 5 alphanumeric characters.

See “Providing a message sequence number (MSGSEQNO)” on page 96 for more information on the default values of MSGSEQNO.

class

Indicates the user class of the files you are sending. A receiver can use this name to receive only files of this class. Use 1 to 8 alphanumeric characters. If you do not specify the user class with either the CLASS parameter or the EDI envelope, the user class defaults as follows: See “Providing a user class (CLASS)” on page 97 for more information on CLASS assignment.

This EDI data type:	Uses this default user class:
X12	#E2
UCS	#EC
UN/TDI	#EU
EDIFACT	#EE

See “Providing a user class (CLASS)” on page 97 for more information on CLASS assignment.

verify

Indicates whether Expedite Base/AIX verifies that the receiver exists before sending the file.

n — Do not verify that the receiver exists. This is the default.

y — Verify that the receiver exists before sending the data. If the verification fails or the destination cannot be verified, the envelope is not sent. Furthermore, if there are multiple envelopes in the file, envelopes following the one with the error are not sent.

f — Verify that the receiver exists before sending the data. If Information Exchange cannot tell whether the receiver exists (for example, if the receiver is on another system), send the data anyway. If the verification fails for an envelope destined for the same Information Exchange system as the sender, the envelope is not sent and if there are multiple envelopes in the file, envelopes following the one with the error are not sent.

c — Verify that the receiver exists before sending the data. If the verification fails or the destination cannot be verified, the envelope is not sent. If there are multiple envelopes in the file, continue processing those envelopes following the one in error.

g — Verify that the receiver exists before sending the data. If Information Exchange cannot tell whether the receiver exists (for example, if the receiver is on another system), send the data anyway. If verification fails for an envelope destined for the same Information Exchange system as the sender, the envelope is not sent. If there are multiple envelopes in the file, continue processing those envelopes following the one in error.

NOTE: You are normally not charged to use the VERIFY parameter. However, in some circumstances you may incur a charge. “Understanding validations, payment levels, and authorizations with libraries” on page 271 for further information on charges.

description

Provides a free-format description of the file. The description is available only to receiving interfaces that support the CDH. For more information on the CDH, see Appendix B, "Common data header." Use 1 to 79 alphanumeric characters.

recfm

Indicates the record format of the file. This parameter is primarily for documentation purposes. Expedite Base/AIX places this information in the CDH. Except for Expedite Base/VM, all Expedite programs ignore this parameter when receiving files. However, Expedite Base/VM uses it to format records when receiving files.

f	File has a fixed record format.
v	File has a variable record format.

lrecl

Indicates the record length of the file. This parameter is primarily for documentation purposes. The record length field identifies record length across systems. Expedite Base/AIX places this information in the CDH. Except for expEDIte Base/VM, all Expedite programs ignore this parameter when receiving files. However, expEDIte Base/VM can use it to format records when receiving files. Valid values are **1** to **65535**. The default value is blank.

retain

Indicates the number of days Information Exchange keeps the file in the mailbox if no one receives it. Valid values are **blank** and **0** through **180**.

The maximum retention period and the default retention period can be different on different Information Exchange systems. These periods are system-dependent. The default retention period is determined when Information Exchange is installed. For installations in the U.S., the default retention period is 30 days. Contact your marketing representative for more information on these values.

If you specify **0** or **blank**, Information Exchange retains the file for the default retention period. If you specify a retention period that is longer than the maximum retention period for your system, Information Exchange retains the file for the default retention period.

translate

Indicates the name of a translate table that overrides normal EBCDIC to ASCII translation. Use 1 to 8 alphanumeric characters. Expedite Base/AIX appends the suffix .XLT to this value to produce the name of the file containing the translate table. If you do not use this parameter, Expedite Base/AIX uses the translate table specified in the TRANSMIT command. If no TRANSLATE was specified, Expedite Base/AIX use the default Information Exchange translation table.

compress

Indicates whether the specified file should be compressed.

n	Do not compress the specified file. This is the default.
y	Compress the specified file.
t	For each sender/receiver pair, use the setting of the `COMPRESS` parameter (*y* or *n*) indicated in the cplookup.tbl file.

You must have the supported data compression software installed if you specify `COMPRESS(Y)` or `COMPRESS(T)`.

Some of the parameters of the command may not apply when sending compressed data. See Appendix E, "Using data compression," for more information.

destfile

Indicates the file name to use in the common data header (CDH) as the original file name. If the receiver is using a workstation-based Expedite Base/AIX and specifies ORIGFILE (Y) on the RECEIVE or RECEIVEEDI command, Expedite Base/AIX uses this file name to store the date when it is received. If you specify a file name that is not valid on the receiver's system, Expedite Base/AIX uses the file name in the FILEID parameter on the RECEIVE or RECEIVEEDI command. By default, Expedite Base/AIX determines the original file name from the FILEID parameter on the SEND, SENDEDI, or PUTMEMBER command. Use 1 to 54 characters.

destloc

Indicates the file location to use in the common data header (CDH). When the file is received or the receiver's mailbox is queried, Expedite Base/AIX uses this value in the RECEIVE or AVAILABLE record in the SENDERLOG parameter. By default, Expedite Base/AIX uses the file location in the sender's system. Use 1 to 65 characters.

selectrcv

Indicates force receive search criteria.

f	Identifies this file for selective receive only.
n	Turns off the selective-receive-only option. No special criteria is required to receive this file. The file can be received by a blanket receive.
blank	Turns off the selective-receive-only option. No special criteria is required to receive this file. The file can be received by a blanket receive. This is the default.

- When **f** is selected in SELECTRCV, the file being sent is marked in Information Exchange and cannot be received by a blanket receive. The file can only be received if the receiver specifies one of the following:

- Sender ID (account ID and user ID)
- Message class
- Message key

SENDEDI command example

The following is an example of the SENDEDI command:

```
sendedi fileid(edi.fil) verify(c) retain(18.);
```

Results: This command sends the file edi.fil to Information Exchange. Expedite Base/AIX uses the EDI headers in the data to determine the destination or destinations. Before sending the file, Expedite Base/AIX verifies that the receiver exists. If the verification fails or the destination cannot be verified, Expedite Base/AIX does not send the envelope. Because no user class is specified in the SENDEDI command, Expedite Base/AIX will either use information from the EDI header as the user class or will assign a default user class depending on the EDI data type. “Providing a user class (CLASS)” on page 97 for a list of EDI data types and the associated default user classes. If no one receives the file within 180 days, Information Exchange will delete the file.

START command

Use the START command to begin an Information Exchange session. If Information Exchange successfully starts the session, Expedite Base/AIX returns a session start key in the RETURN response record. The message response record associated with the START command is the RETURN record.

The following example shows the syntax of the START command:

start

```
account(IE account) userid(IE user ID)
iepassword(IE password) niepassword(new IE password) check(y|n);
```

NOTE: If you specify AUTOSTART(Y) in basein.pro to tell Expedite Base/AIX to start the Information Exchange session automatically, you don't need to specify START in basein.msg. Specifying START in basein.msg results in an error if you use **y** as the AUTOSTART value in basein.pro.

account

Indicates the account of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the USERID parameter to identify the user. If you specify a USERID parameter, you must specify an ACCOUNT parameter. Use 1 to 8 alphanumeric characters. The default is the IEACCOUNT value of the IDENTIFY command.

userid

Indicates the user ID of an Information Exchange user. Expedite Base/AIX uses this field in conjunction with the ACCOUNT parameter to identify the user. If you specify an ACCOUNT parameter, you must specify a USERID parameter. Use 1 to 8 alphanumeric characters. The default is the IEUSERID value of the IDENTIFY command.

iepassword

Indicates the Information Exchange password. Use 1 to 8 alphanumeric characters. The default is the IEPASSWORD of the IDENTIFY command.

niepassword

Indicates the new Information Exchange password. If you specify this value, the Information Exchange password changes upon completion of the next Information Exchange session. If the Information Exchange session ends in an error, the password does not change. Use 1 to 8 alphanumeric characters.

NOTE: If you are an ESO user, your password must conform to ESO rules. See “Selecting the Extended Security Option” on page 163 for more information.

check

Indicates to Expedite that you only want to check the status of the previous session. If you specify CHECK on the START command, do not specify any other commands except the END command in the input file. Valid values are:

y	Check the status of the previous session.
n	Do not check the status of the previous session; start a session as usual. This is the default.

The following three parameters enable access to the Expedite Base/AIX GSKit environment using a Secure Sockets Layer (SSL) TCP/IP connection.

When you use the KEYRING* parameters on the START command and you want to run multiple Information Exchange sessions, set the TRANSMIT command parameters AUTOSTART and AUTOEND to **n**. See "TRANSMIT command" on page 156 for more information about using these parameters.

keyringfile

Indicates the file name of the keyring file. The keyring file requires a password. The maximum length is 256 characters.

NOTE: The KEYRINGFILE requires a password that you can supply either in a stash file or in the command as a parameter. You can use only one of these options. If you use the KEYRINGSTASHFILE parameter, you cannot use the KEYRINGPASSWORD option.

keyringstashfile

The file name of the stash file that stores the password for the KEYRINGFILE parameter. The maximum length is 256 characters. If you use this parameter, you do not need to specify the KEYRINGPASSWORD parameter.

keyringpassword

The password for the KEYRINGFILE. Maximum length is 16 characters. If you use this parameter, you do not need to specify the keyringstashfile parameter.

START command example

The following is an example of the START command:

```
start account(acct) userid(user01) iepassword(mypswd);
```

OR

```
start account(acct) userid(user01) iepassword(mypswd);
keyringfile(filename) keyringpassword(password);
```

OR

```
start account(acct) userid(user01) iepassword(mypswd);
keyringfile(filename) keyringstashfile(filename);
```

Results: This command starts an Information Exchange session manually for Information Exchange account *acct* and user ID *user01*. The password *mypswd* is used to log on to Information Exchange.

Chapter 9

Using Expedite Base/AIX message response records

To verify the completion or partial completion of message or profile commands, you need to understand the information in the response records. This chapter describes the response records and provides examples.

Working with message response records

When Expedite Base/AIX processes basein.msg, it echoes message commands, along with their response records and associated return codes, to baseout.msg.

The message response records are:

- AUTOEND, page 233

  This record indicates that an Information Exchange session ended automatically.

- AUTOSTART, page 234

  This record indicates that an Information Exchange session started automatically.

- AVAILABLE, page 235

  This record provides information about messages in your Information Exchange mailbox. Expedite Base/AIX writes this record as a result of the QUERY command.

- LIBRARYLIST, page 240

  This record provides the results of the LISTLIBRARIES command.

- MEMBERLIST, page 243

  This record provides the results of the LISTMEMBERS command.

- MEMBERPUT, page 245

  This record is a response to the PUTMEMBER command. It provides information about a member placed in the library.

- MOVED, page 246

  This record tells you how many files Information Exchange copied from archive as a result of an ARCHIVEMOVE command.

- NOTSENT, page 247

  This record provides information on the data that could not be sent when a destination verification failure occurs with the SENDEDI command.

- RECEIVED, page 249

  This record provides information on the data received with a RECEIVE or RECEIVEEDI command.

- RETURN, page 256

  This record indicates the completion of a command in basein.msg.

- SENT, page 257

  This record provides information on the data sent with a SEND or SENDEDI command.

- SESSIONEND, page 260

  This record indicates the final return code from Expedite Base/AIX. This is the last record in baseout.msg.

- WARNING, page 262

  This record indicates a minor problem that did not stop the command from finishing, but which should be noted.

The following sections provide detailed information on each of these records.

AUTOEND record

The AUTOEND record indicates that Expedite Base/AIX ended the Information Exchange session automatically.

The following example shows the format of the AUTOEND record:

```
AUTOEND;
```

AUTOSTART record

The AUTOSTART record indicates that Expedite Base/AIX started an Information Exchange session automatically.

The following example shows the format of the AUTOSTART record:

```
  AUTOSTART SESSIONKEY(session key);
  STARTED LASTSESS(0|1) RESPCODE(code)
SESSIONKEY(session key) IEVERSION(version) IERELEASE(release);
```

sessionkey

Indicates the unique identifier for this Information Exchange session. This is also used as the archive ID for the files that are archived and for which you did not specify an ARCHIVEID on the RECEIVE command. SESSIONKEY is 8 alphanumeric characters.

lastsess

Indicates the status of the previous session.

0	Indicates the last session was successful.
1	Indicates the last session was not successful.

respcode

Indicates the Information Exchange response code for the current session (not the previous session). These codes are interpreted for you by Expedite, so if it is not 0 and not 2, you will get a session end return code from Expedite indicating the problem. RESPCODE is 5 digits, padded on the left with zeros.

sessionkey

Indicates the unique identifier for this Information Exchange session. This is also used as the archive ID for the files that are archived and for which you did not specify an ARCHIVEID on the RECEIVE command. SESSIONKEY is 8 alphanumeric characters.

ieversion

Indicates the version of Information Exchange. Levels of Information Exchange are tracked as *V.R* where *V* is the version, a major enhancement in the service, and where *R* is the release, a minor enhancement. IEVERSION is two digits, padded on the left with zeros.

ierelease

Indicates the release of Information Exchange. Levels of Information Exchange are tracked as *V.R* where *V* is the version, a major enhancement in the service, and where *R* is the release, a minor enhancement. IERELEASE is two digits, padded on the left with zeros.

AVAILABLE record

The AVAILABLE record contains information describing a message. When you use the QUERY command, Expedite Base/AIX produces an AVAILABLE record for every message in the mailbox. All of the parameters shown here may not be included with every AVAILABLE record because Expedite Base/AIX does not write parameters with blank values and because some parameters come from the common data header (CDH). If you specify CDH(Y) on the QUERY command for a file that has no CDH, the CDH parameters contain default values.

The following example shows the format of the AVAILABLE record:

```
AVAILABLE
SYSID(system ID) ACCOUNT(account) USERID(user ID)
```

or

```
ACCOUNT(account) USERID(user ID)
MSGKEY(message key) CLASS(class) MODE(t) MSGDATE(message date)
MSGDATELONG(message date long format) MSGTIME(message time)
MSGNAME(message name)
MSGSEQNO(message sequence number) LENGTH(length)
SYSNAME(system name) SYSLEVEL(system level)
DATATYPE(a|b) EDITYPE(EDI type)
SENDERFILE(sender file) SENDERLOC(sender location) FILEDATE(file date)
FILEDATELONG(file date long format) FILETIME(file time)
RECFM(record format) RECLEN(record length) RECDLM(c|e|l|n|u)
DESCRIPTION(description) UNIQUEID(unique ID) CODEPAGE(code page)
SYSTYPE(01|10|12|15|17|20|21|30|31|40|44|80|90)
SYSVER(system version) TRANSLATE(translate table)
COMSW(compression software name) COMVER(compression software version)
COMFILE(name of compressed file);
```

Expedite Base/AIX does not display the following parameters if you specify CDH(N) using the QUERY command: DATATYPE, EDITYPE, RECDLM, DESCRIPTION, COMVER, COMFILE, SENDERFILE, SENDERLOC, FILEDATE, FILEDATELONG, FILETIME, RECFM, RECLEN, UNIQUEID, CODEPAGE, SYSTYPE, SYSVER, TRANSLATE, and COMSW.

sysid

Indicates the system ID of the user who sent the file. This parameter contains 1 to 3 alphanumeric characters.

account

Indicates the account of the user who sent the file. This parameter contains 1 to 8 alphanumeric characters.

userid

Indicates the user ID of the user who sent the file. This parameter contains 1 to 8 alphanumeric characters.

msgkey

Indicates a unique identifier assigned to the file by Information Exchange. You can use this value for the MSGKEY parameter in the RECEIVE or RECEIVEEDI command to receive only a specific file. This parameter contains 20 characters.

class

Indicates the user class of the data specified by the sender to identify the data. This parameter contains 1 to 8 alphanumeric characters.

mode

Indicates the network data class field for this data. The sender specifies this value in the SEND or SENDEDI command. If this is not a test-mode file, Expedite Base/AIX does not write this parameter.

t Indicates a test-mode file.

msgdate

Indicates the date Expedite Base/AIX placed the file in Information Exchange. The format is *YYMMDD*. This parameter contains 6 numeric characters.

msgdatelong

Indicates the date Expedite Base/AIX placed the file in Information Exchange. The format is *YYYYMMDD*. This parameter contains 8 numeric characters.

msgtime

Indicates the time Expedite Base/AIX placed the file in Information Exchange. The format is *HHMMSS*. This parameter contains 6 numeric characters.

msgname

Indicates the name of the file specified by the sender. This parameter contains 1 to 8 alphanumeric characters.

msgseqno

Indicates the sequence number assigned by the sender as a file control number for this data. This parameter contains 1 to 5 alphanumeric characters.

length

Indicates the length of the data in the Information Exchange mailbox. The length of the file received may be different from the length of the file sent because of reformatting. This parameter contains 1 to 10 characters.

sysname

Indicates the name of the interface that sent the data. This parameter contains 1 to 8 alphanumeric characters.

syslevel

Indicates the level of the system that sent the data. This parameter contains 1 to 4 alphanumeric characters.

datatype

Indicates whether the data is text or binary.

a	Text
b	Binary

editype

Indicates the type of EDI data available: X12, UCS, UNTDI, EDIFACT, or unformatted. This parameter contains 3 to 11 alphanumeric characters.

senderfile

Indicates the original file name of the file on the sender's system. This parameter contains 1 to 54 alphanumeric characters.

senderloc

If the sending system was a workstation, then this contains the directory where the file was stored on the sender's system. This parameter contains 1 to 65 alphanumeric characters.

filedate

Indicates the date that the file was created or last modified on the sender's system. The format is *YYMMDD*.

filedatelong

Indicates the date that the file was created or last modified on the sender's system. The format is *YYYYMMDD*.

filetime

Indicates the time that the file was created or last modified on the sender's system. The format is *HHMMSS*.

recfm

Indicates the record format of the file on the sender's system. If the record format is not appropriate for the sending machine, for example if the sending machine is a PC, the value is **????**. This parameter contains 1 to 4 alphanumeric characters.

reclen

Indicates the original record length of the file. This parameter contains 1 to 5 numeric characters.

recdlm

Indicates the method the sender used to delimit the records.

c	CRLF characters delimit the records.
e	EDI characters delimit the records.
l	A 2-byte length preceding each record delimits the records.
n	The records contain no delimiters, or the sender did not indicate the type of delimiters in the CDH.
u	Unknown delimiters.

description

Provides a free-format description of the data written by the sender. This parameter contains 1 to 79 alphanumeric characters.

uniqueid

Indicates the random ID assigned to the file by the sending interface. It can help you identify the file and also help you associate acknowledgments with the file. This parameter contains 8 alphanumeric characters.

codepage

Indicates the code page used by the sending system to determine the character representation of the data. This parameter contains 3 numeric characters.

systype

Indicates the type of system that sent the data. This parameter contains 2 hexadecimal digits. See the *Information Exchange Administration Services User's Guide* for relevant system types and codes.

sysver

Indicates the software version of the Expedite system sending the data. This parameter contains 1 to 3 numeric characters.

translate

Indicates the ASCII to EBCDIC translate table used to send this file to Information Exchange. This parameter contains 1 to 8 alphanumeric characters.

comsw

Indicates the name of the compression software package used to compress the file. This parameter contains 10 alphanumeric characters.

comver

Indicates the version of the compression software package used to compress the file. This parameter contains 1 to 5 alphanumeric characters.

comfile

Indicates the name of the compressed file. This parameter contains 1 to 54 alphanumeric characters.

LIBRARYLIST record

The LIBRARYLIST record returns information requested by the LISTLIBRARIES command. Each record contains information specific to that library. Only the information you're authorized to see about the library is returned. Parameters with blank values are not included. The format of the LIBRARYLIST record is:

```
LIBRARYLIST
OWNER(library owning account) OWNUSERID(owner's user ID)
LIBRARY(library name) MEMBERS(number of members)
DESCRIPTION(description) CREATEDATE(creation date)
CREATEDATELONG(creation date long format) CREATETIME(creation time)
UPDATEDBY(ID of last user) UPDATEDATE(date of last update)
UPDATEDATELONG(date of last update long format)
UPDATETIME(time of last update) WRITEAUTH(p|o|g|l)
WRITELIST(write distribution list) READAUTH(p|o|g|l)
READLIST(read distribution list) SEARCHABLE(y|n)
OWNERPAYS(y|n);
```

owner

Indicates the account that owns the library. This parameter contains 1 to 8 alphanumeric characters.

ownuserid

Indicates the user ID of the owner of the library. This parameter contains 1 to 8 alphanumeric characters.

library

Indicates the name of the library. This parameter contains 1 to 8 alphanumeric characters.

members

Indicates the number of members present in the library. This parameter contains 1 to 4 characters.

description

Provides a free-format description of the library. This parameter contains 1 to 79 alphanumeric characters.

createdate

Indicates the date the library was created in the format *YYMMDD*. Information Exchange adjusts the date to that of your local time zone. This parameter contains 6 numeric characters.

createdatelong

Indicates the date the library was created in the format *YYYYMMDD*. Information Exchange adjusts the date to that of your local time zone. This parameter contains 8 numeric characters.

createtime

Indicates the time the library was created. The format is *HHMMSS*. Information Exchange adjusts the time to that of your local time zone. This parameter contains 6 numeric characters.

updatedby

Indicates the account ID and user ID, separated by at least one blank, of the user who last redefined the library. This parameter contains 3 to 17 alphanumeric characters.

updatedate

Indicates the date the library was last redefined. The format is *YYMMDD*. Information Exchange corrects the date to that of your local time zone. This parameter contains 6 numeric characters.

updatedatelong

Indicates the date the library was last redefined. The format is *YYYYMMDD*. Information Exchange corrects the date to that of your local time zone. This parameter contains 8 numeric characters.

updatetime

Indicates the time the library was last defined. The format is *YYMMDD*. This parameter contains 6 numeric characters.

writeauth

Indicates the authority type for update access to the library.

p	Only the owner can update the library.
o	Only users within the same account can update the library.
g	Any user can update the library.
l	Any user in the list named in the WRITELIST parameter can update the library.

writelist

Indicates the name of a permanent distribution list that details the users who can update the library. This parameter contains 1 to 8 alphanumeric characters.

readauth

Indicates the authority type for read access to the library.

p	Only the owner can read the library.
o	Only users within the same account can read the library.
g	Any user can read the library.
l	Any users named in the READLIST parameter can read the library.

readlist

Indicates the name of a permanent distribution list that details the users who can use the library. This parameter contains 1 to 8 alphanumeric characters.

searchable

Contains **Y** if the library is searchable; **N** if it is not.

owner pays

Indicates whether the owner of a library is responsible for charges associated with transfer of a library member from the user's mail box to the user's system when the library member is retrieved. These are usually called receive side charges.

y	The owner of the library pays the receive side charges.
n	The receive side charges are charged to you.

No LIBRARYLIST or other response records are written to the message response file if no libraries are found that match the criteria you specified.

MEMBERLIST record

The MEMBERLIST record returns information requested by the LISTMEMBERS command. Each record contains information specific to that member. Parameters with blank values are not included. The format of the MEMBERLIST record is:

```
MEMBERLIST
MEMBER(member name) DESCRIPTION(description)
CREATEDBY(user's ID) CREATEDATE(creation date)
CREATEDATELONG(creation date long format) CREATETIME(creation time)
UPDATEDBY(ID of last user) UPDATEDATE(date of last update)
UPDATEDATELONG(date of last update long format)
UPDATETIME(time of last update) LENGTH(file length);
```

member

Indicates the name of a library member. This parameter contains 1 to 8 alphanumeric characters.

description

Provides a free format description of the member. This parameter contains 1 to 79 alphanumeric characters.

createdby

Indicates in a fixed format, the system ID, account ID, and user ID of the user who created the library member. The first four characters are the system ID (and may be blank). The next eight characters are the account ID. The last eight characters are the user ID.

createdate

Indicates the date the member was created. The format is *YYMMDD*.

createdatelong

Indicates the date the member was created. The format is *YYYYMMDD*.

createtime

Indicates the time the member was created. The format is *HHMMSS*. This parameter contains 6 numeric characters.

updatedby

Indicates in a fixed format the system ID, account ID, and user ID of the user who last updated the library member. The first four characters are the system ID (and may be blank). The next eight characters are the account ID. The last eight characters are the user ID. This parameter contains 3 to 17 alphanumeric characters.

updatedate

Indicates the date the member was last updated. The format is *YYMMDD*. Information Exchange adjusts the date to that of your local time zone. This parameter contains 6 numeric characters.

updatedatelong

Indicates the date the member was last updated. The format is *YYYYMMDD*. Information Exchange adjusts the time to that of your local time zone. This parameter contains 8 numeric characters.

updatetime

Indicates the time the member was last updated. The format is *HHMMSS*. Information Exchange adjusts the time to that of your local time zone. This parameter contains 6 numeric characters.

length

Indicates the length of the member in bytes.

No MEMBERLIST or other response records are written to the message response file if the user has no authority to access the library.

MEMBERPUT record

The MEMBERPUT record returns information about a member placed in an Information Exchange library. All of the parameters shown here may not be included in every MEMBERPUT record because Expedite Base/AIX does not write parameters with blank values.

The following example shows the format of the MEMBERPUT record:

```
MEMBERPUT
OWNER(owner) LIBRARY(library) MEMBER(member)
UNIQUEID(unique ID) LENGTH(length);
```

owner

Indicates the account that owns the library. This parameter contains 1 to 8 alphanumeric characters.

library

Indicates the name of the library. This parameter contains 1 to 8 alphanumeric characters.

member

Indicates the name of the member. This parameter contains 1 to 8 alphanumeric characters.

uniqueid

Indicates the random ID assigned by Expedite Base/AIX to identify the member. This parameter contains 8 alphanumeric characters.

length

Indicates the length of the file placed in the library. This parameter contains 1 to 8 numeric characters.

MOVED record

The MOVED record indicates how many files Information Exchange copied from short-term archive to your Information Exchange mailbox as a result of an ARCHIVEMOVE command.

The following example shows the format of the MOVED record:

```
MOVED
NUMBER(number);
```

number

Indicates the number of files copied from short-term archive to your Information Exchange mailbox. If this value is 0, no files in the archive match the ARCHIVEID you specified in the ARCHIVEMOVE command, or you already copied the files and they are in your mailbox. This parameter contains 1 to 5 numeric characters.

NOTSENT record

Expedite Base/AIX produces a NOTSENT record for every EDI envelope that could not be sent due to a destination verification failure. NOTSENT records are produced only when the VERIFY parameter of the SENDEDI command is set to *c* or *g*.

Because parameters with blank values do not print, Expedite Base/AIX might not include all the parameters listed below with each NOTSENT record.

The format of the NOTSENT record is:

```
NOTSENT
ALIAS(alias) ALIASNAME(alias name)
```

or

```
SYSID(system ID) ACCOUNT(account) USERID(user ID)
```

or

```
ACCOUNT(account)  USERID(user ID)
```

or

```
LISTNAME(list name)


EDITYPE(datatype) DESTINATION(destination) QUALIFIER(qualifier)
CONTROLNUM(control number) CLASS(class) MSGNAME(message name)
MSGSEQNO(message sequence no);
```

alias

Indicates the alias table type and table name of the Information Exchange destination.

g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to 3-character table name.

This parameter contains 1 to 4 characters.

aliasname

Indicates the alias name defined in the alias table. This parameter contains 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of the Information Exchange destination. This parameter contains 1 to 3 alphanumeric characters.

account

Indicates the account of the Information Exchange destination. This parameter contains 1 to 8 alphanumeric characters.

userid

Indicates the user ID of the Information Exchange destination. This parameter contains 1 to 8 alphanumeric characters.

listname

Indicates the name of a previously defined list of account and user IDs used as the Information Exchange destination. This parameter contains 1 to 8 alphanumeric characters.

editype

Indicates the type of EDI data that would have been sent, had the destination verification failure not occurred: X12, UCS, UNTDI, or EDIFACT. This parameter contains 1 to 7 alphanumeric characters.

destination

Indicates the destination specified in the EDI data. This parameter contains 1- to 35- alphanumeric characters.

qualifier

Indicates the EDI qualifier for the destination. This parameter contains 1 to 4 alphanumeric characters.

controlnum

Indicates the Interchange Control Number from the X12 or UCS data. For UN/TDI data, this is the SNRF element. For EDIFACT data, it is the data element *0020* (Interchange Control Reference). This parameter contains 1 to 14 alphanumeric characters.

class

Indicates the user class obtained from the EDI data, the CLASS parameter, or the default. This parameter contains 1 to 8 alphanumeric characters.

msgname

Indicates the message name obtained from the EDI data or the MSGNAME parameter. This parameter contains 1 to 8 alphanumeric characters.

msgseqno

Indicates the message sequence number obtained from the MSGSEQNO parameter or generated by Expedite Base/AIX. This parameter contains 1 to 5 alphanumeric characters.

RECEIVED record

The RECEIVED record contains information describing a file or EDI envelope. Expedite Base/AIX produces a RECEIVED record for every file or EDI envelope received from Information Exchange. All of the parameters shown here may not be included in every RECEIVED record because Expedite Base/AIX does not write parameters with blank values and because the sender may not have provided some parameters.

The following example shows the format of the RECEIVED record:

```
RECEIVED
ALIAS(alias) ALIASNAME(alias name)
```

or

```
SYSID(system ID) ACCOUNT(account) USERID(user ID)
```

or

```
ACCOUNT(account) USERID(user ID)
RECEIVER(receiver ID) RECVQUAL(receiver qualifier) SENDER(sender ID)
SENDQUAL(sender qualifier) CONTROLNUM(control number)
CLASS(class) MODE(mode) PRIORITY(a|i|p) CHARGE(1|5|6) ACK(d)
LENGTH(file length) FILEID(file ID) MSGDATE(message date)
MSGDATELONG(message date long format) MSGTIME(message time)
MSGSEQO(IE message number) MSGNAME(message name)
MSGSEQNO(msg sequence number) SESSIONKEY(session key) DELIMITED(l|e|n)
SYSNAME(system name) SYSLEVEL(system level)
STARTDATE(starting date) STARTTIME(starting time)
ENDDATE(ending date) ENDTIME(ending time) TIMEZONE(l|g)
DATATYPE(data type) EDITYPE(EDI type) SENDERFILE(sender file)
SENDERLOC(sender location) FILEDATE(file date)
FILEDATELONG(file date long format) FILETIME(file time)
RECFM(record format) RECLEN(record length)
RECDLM(c|e|l|n|u) DESCRIPTION(description) UNIQUEID(unique ID)
CODEPAGE(code page) SYSTYPE(01|10|12|15|17|20|21|30|31|40|44|80|90)
SYSVER(system version) TRANSLATE(translate table)
COMSW(compression software name) COMVER(compression software version)
COMFILE(name of compressed file) DCMPRC(decompression return code);
```

alias

Indicates the table type and table name of an alias table. This parameter contains 1 to 4 alphanumeric characters.

g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to 3-character table name.

aliasname

Indicates an alias name defined in the alias table. This parameter contains 1 to 16 alphanumeric characters.

sysid

Indicates the system ID of the user who sent the file. This parameter contains 1 to 3 alphanumeric characters.

account

Indicates the account of the user who sent the file. This parameter contains 1 to 8 alphanumeric characters.

userid

Indicates the user ID of the sender. This parameter contains 1 to 8 alphanumeric characters.

receiver

Indicates the receiver ID specified in the EDI data. This parameter contains 1 to 35 characters.

recvqual

Indicates the EDI qualifier for the receiver specified in the EDI data. This parameter contains 1 to 4 characters.

sender

Indicates the sender ID specified in the EDI data. This parameter contains 1 to35 characters.

sendqual

Indicates the EDI qualifier for the sender specified in the EDI data. This parameter contains 1 to 4 characters.

controlnum

Indicates the Interchange Control Number from the X12 or UCS data. For UN/TDI data, this is the SNRF element. For EDIFACT data, this is the element *0020* (Interchange Control Reference). This parameter contains 1 to 14 characters.

class

Indicates the user class of the data specified by the sender to identify the data. This parameter contains 1 to 8 alphanumeric characters.

mode

Indicates the network data class field for this data as specified by the sender. Expedite Base/AIX omits this parameter for normal mode files.

t	Indicates a test-mode file.

priority

Indicates the class of delivery service for this file. If this is a normal priority file, Expedite Base/AIX does not write this parameter.

a	Indicates a normal-priority file that was copied from archive.
i	Indicates express delivery to those users who have the continuous receive capability. This file was received before any other files with a lower priority. (Expedite Base/AIX does not support the continuous receive capability.)
p	Indicates high priority.

charge

Indicates how the file charges are paid.

1	Receiver pays all charges.
5	Sender and receiver split the charges.
6	Sender pays all charges.

ack

Indicates that the sender asked Information Exchange to send a delivery acknowledgment.

d	Indicates that Information Exchange returned a delivery acknowledgment to the sender.

length

Indicates the length of the file received. This parameter contains 1 to 9 numeric characters.

fileid

Indicates the name of the file in which Expedite Base/AIX placed the received data. This parameter contains 1 to 54 alphanumeric characters.

msgdate

Indicates the date the received data was placed into Information Exchange. The format of this parameter is *YYMMDD*. This parameter contains 6 numeric characters.

msgdatelong

Indicates the date received data was placed into Information Exchange. The format of this parameter is *YYYYMMDD*. This parameter contains 8 numeric characters.

msgtime

Indicates the time received data was placed into Information Exchange. The format of this parameter is *HHMMSS*. This parameter contains 6 numeric characters.

msgseqo

Indicates the unique number assigned to the data by Information Exchange. This parameter contains 1 to 6 numeric characters.

msgname

Indicates the name for the data specified by the sender. This parameter contains 1- to 8 alphanumeric characters.

msgseqno

Indicates the sequence number assigned by the sender as a file control number for this data. This parameter contains 1 to 5 alphanumeric characters.

sessionkey

Indicates the session access key that Expedite Base/AIX used when the file was received. This value is the ARCHIVEID for the file if you did not specify an ARCHIVEID parameter in the RECEIVE or RECEIVEEDI command. This parameter contains 1 to 8 alphanumeric characters.

delimited

Indicates the way Expedite Base/AIX actually processed record delimiters when the file was received.

l	Expedite Base/AIX added CLRF at the end of the records according to the 2-byte length delimiters at the beginning of each record.
e	Expedite Base/AIX processed the EDI delimiters according to the parameters you specified on the RECEIVE or RECEIVEEDI command for EDI data.
n	Expedite Base/AIX stored the data as it was received; if you specified the FORMAT option on the RECEIVE command, then Expedite Base/AIX formatted the data according to that option.

sysname

Indicates the name of the system that sent the data. This parameter contains 1 to 8 alphanumeric characters.

syslevel

Indicates the level of the system that sent the data. This parameter contains 1 to 8 alphanumeric characters.

startdate

Indicates the starting date of a time range for the files you received from Information Exchange. The format is YYYYMMDD or YYMMDD. This parameter contains 6 to 8 numeric characters.

starttime

Indicates the starting time of a time range for the files you received from Information Exchange. The format is *HHMMSS*. This parameter contains 6 numeric characters.

enddate

Indicates the ending date of a time range for the files you received from Information Exchange. The format is *YYYYMMDD* or *YYMMDD*. This parameter contains 6 to 8 numeric characters.

endtime

Indicates the ending time of a time range for files you received from Information Exchange. The format is *HHMMSS*. This parameter contains 6 numeric characters.

timezone

Indicates the time zone reference for the STARTTIME and ENDTIME parameters.

l	Local time, as specified on the TIMEZONE parameter of the IDENTIFY command.
g	Greenwich mean time (GMT).

This parameter contains 1 to 5 alphanumeric characters.

datatype

Indicates whether the data is text or binary.

a	Text
b	Binary

editype

Indicates the type of EDI data received: X12, UCS, UN/TDI, EDIFACT, or unformatted. This parameter contains 3 to 11 alphanumeric characters.

senderfile

Indicates the original file name of the file on the sender's system. This parameter contains 1 to 54 alphanumeric characters.

senderloc

If the sending system was a workstation, then this contains the directory where the file was stored on the sender's system. This parameter contains 1 to 65 alphanumeric characters.

filedate

Indicates the date that the file was created or last edited on the sender's system. The format is *YYMMDD*. This parameter contains 6 numeric characters.

filedatelong

Indicates the date that the file was created or last edited on the sender's system. The format is *YYYYMMDD*. The parameter contains 8 numeric characters.

filetime

Indicates the time that the file was created or last edited on the sender's system. The format is *HHMMSS*. This parameter contains 6 numeric characters.

recfm

Indicates the record format of the file on the sender's system. If the record format is not used by the sending machine (for example, if the sending machine is a workstation), the value is **????**. This parameter contains 1 to 4 alphanumeric characters.

reclen

Indicates the original record length of the file. This parameter contains 1 to 5 numeric characters.

recdlm

Indicates the method used to delimit the records.

c	CRLF characters delimit the records.
e	EDI characters delimit the records.
l	A 2-byte length preceding each record delimits the records.
n	The records contain no delimiters, or the sender did not indicate the type of delimiters on the CDH.
u	Unknown delimiters.

description

Provides a free-format description of the data written by the sender. This parameter contains 1 to 79 alphanumeric characters.

uniqueid

Indicates the random ID assigned to the data by the sending interface to help you identify the data. This parameter contains 8 alphanumeric characters.

codepage

Indicates the code page used by the sending system to determine the character representation of a given byte. This parameter contains 3 numeric characters.

systype

Indicates the type of system that sent the data. This parameter contains 2 hexadecimal digits. See the *Information Exchange Administration Services User's Guide* for relevant system types and codes.

sysver

Indicates the software version of the Expedite system sending the data. This parameter contains 1 to 3 numeric characters.

translate

Indicates the ASCII to EBCDIC translate table used when this file was sent to Information Exchange. This parameter contains 8 alphanumeric characters.

comsw

Indicates the name of the compression software package used to compress the file. This parameter contains 10 alphanumeric characters.

comver

Indicates the version of the compression software package used to compress the file. This parameter contains 1 to 5 alphanumeric characters.

comfile

Indicates the name of the compressed file. This parameter contains 1 to 54 alphanumeric characters.

dcmprc

Indicates the decompression return code. This parameter contains 1 to 5 alphanumeric characters.

RETURN record

The RETURN record indicates the completion of a command. A 0 value indicates that the command completed normally.

The following example shows the format of the RETURN record:

```
RETURN(return code) ERRDESC(error description)
ERRTEXT(error text) SESSIONKEY(session key);
```

return

Indicates completion of the Expedite Base/AIX command. If the return code is 0, the command completed successfully. If the return code is not 0, Expedite Base/AIX displays an error number along with ERRDESC and ERRTEXT records. This parameter contains 5 numeric characters.

errdesc

Provides a short description of an error. If the return code is 0, this parameter is not contained in baseout.msg. This parameter contains 1 to 76 alphanumeric characters.

errtext

Provides a detailed description of an error and may suggest steps to correct the problem. There may be multiple error text records. Each ERRTEXT record contains 1 to 76 alphanumeric characters.

sessionkey

Contains the session access key provided by Information Exchange upon a successful session start. The session access key is only provided after a START command. When the RETURN record contains a session access key, this is the only parameter included in the record. This parameter contains 1 to 8 alphanumeric characters.

SENT record

The SENT record returns information about the sent data that may not be apparent from the command parameters. Expedite Base/AIX creates one SENT record for each file or EDI envelope it sends. All of the parameters shown here may not be included in the SENT record because Expedite Base/AIX does not write parameters with blank values.

When the SENT record follows a SEND command, it includes only the UNIQUEID and the LENGTH parameters. When it follows a SENDEDI command, the SENT record returns all of the parameters shown, unless the parameter value is blank.

The following example shows the format of the SENT record:

```
SENT
ALIAS(alias) ALIASNAME(alias name)
```

or

```
SYSID(system ID) ACCOUNT(account) USERID(user ID)
```

or

```
ACCOUNT(account) USERID(user ID)
```

or

```
LISTNAME(list name)
UNIQUEID(unique ID) LENGTH(length)
EDITYPE(data type) DESTINATION(destination) QUALIFIER(qualifier)
CONTROLNUM(control number) CLASS(class) MSGNAME(message name)
MSGSEQNO(message sequence number);
```

alias

Indicates the alias table type and table name of the Information Exchange destination.This parameter contains 1 to 4 characters.

g*xxx*	Global alias table, where *xxx* identifies a 1- to 3-character table name.
o*xxx*	Organizational alias table, where *xxx* identifies a 1- to 3-character table name.
p*xxx*	Private alias table, where *xxx* identifies a 1- to 3-character table name.

aliasname

Indicates an alias name defined in the alias table. This parameter contains 1 to 16 alpha-numeric characters.

sysid

Indicates the system ID of the user to whom you sent data. This parameter contains 1- to 3-alphanumeric characters.

account

Indicates the account of the user to whom you sent data. This parameter contains 1 to 8 alphanumeric characters.

userid

Indicates the user ID of the user to whom you sent data. This parameter contains 1 to 8 alphanumeric characters.

listname

Indicates the name of a list of account and user IDs to whom you sent data. This parameter contains 1 to 8 alphanumeric characters.

uniqueid

Indicates the random ID assigned to the file by Expedite Base/AIX. It helps you identify the file and can be used to associate the sent file with any acknowledgments you may receive. The first 8 characters in the MSGDESCR field in Information Exchange acknowledgment contain the UNIQUEID. This parameter contains 8 alphanumeric characters.

length

Indicates the length of the file or EDI envelope. This parameter contains 1 to 9 numeric characters.

editype

Indicates the type of EDI data sent: X12, UCS, UN/TDI, or EDIFACT. This parameter contains 1 to 7 alphanumeric characters.

destination

Indicates the destination specified in the EDI data. This parameter contains 1 to 35 alphanumeric characters.

qualifier

Indicates the EDI qualifier for the destination. This parameter contains 1 to 4 alphanumeric characters.

controlnum

Indicates the Interchange Control Number from the X12 or UCS data. For UN/TDI data, this is the SNRF element. For EDIFACT data, it is the data element 0020 (Interchange Control Reference). This parameter contains 1 to 14 alphanumeric characters.

class

Indicates the user class specified in the SEND or SENDEDI command. If you used the SENDEDI command without this parameter, Expedite Base/AIX uses the default user class of the EDI data type for this parameter value. This parameter contains 1 to 8 alphanumeric characters.

msgname

Indicates the message name specified in the SEND or SENDEDI command. If you used the SENDEDI command without this parameter, Expedite Base/AIX uses the message name obtained from the EDI data type for this parameter value. This parameter contains 1- to 8-alphanumeric characters.

msgseqno

Indicates the message sequence number obtained from the MSGSEQNO parameter or assigned by Expedite Base/AIX. This parameter contains 1 to 5 alphanumeric characters.

SESSIONEND record

The SESSIONEND record is the last record in the response file. The SESSIONEND record indicates the completion of an entire basein.msg file. A 0 value indicates that all the commands in the file completed successfully.

The following example shows the format of the SESSIONEND record:

```
  SESSIONEND(session end) ERRDESC(error description)
ERRTEXT(error text);
```

NOTE: There is only one SESSIONEND record in a response file, even if there are multiple START and END commands.

sessionend

Indicates the return code for the entire message command file (basein.msg). A return code of 00000 indicates that processing of basein.msg completed successfully. If you receive a return code other than 00000, correct the error and restart Expedite Base/AIX. This parameter contains a 5-digit code.

errdesc

Provides a short description of an error. If the return code is 0, Expedite Base/AIX does not include this parameter. This parameter contains 1 to 76 alphanumeric characters.

errtext

Provides a detailed description of an error and may suggest steps to correct the problem. There may be multiple error text records. Each ERRTEXT record contains 1 to 76 alphanumeric characters.

STARTED record

This record provides information about the previous session. It is written to the output file as a result of a START or AUTOSTART command. The syntax of the STARTED record is as follows:

STARTED LASTSESS(*0|1*) RESPCODE(*code*) SESSIONKEY(*sesskey*)
IEVERSION(*version*) IERELEASE(*release*);

where:

lastsess

Indicates the status of the previous session.

0	Indicates the last session was successful.
1	Indicates the last session was not successful.

respcode

Indicates the Information Exchange response code for the current session (not the previous session). If RESPCODE is not 0 or 2, you will get a session end return code from Expedite indicating the problem. RESPCODE is 5 digits, padded on the left with zeros.

sessionkey

Indicates the unique identifier for this Information Exchange session. This is also used as the archive ID for the files that are archived and for which you did not specify an ARCHIVEID on the RECEIVE command. SESSIONKEY is 8 alphanumeric characters.

ieversion

Indicates the version of Information Exchange. Levels of Information Exchange are tracked as V.R where V is the version, a major enhancement in the service, and where *R* is the release, a minor enhancement. IEVERSION is two digits, padded on the left with zeros.

ierelease

Indicates the release of Information Exchange. Levels of Information Exchange are tracked as *V.R* where *V* is the version, a major enhancement in the service, and where *R* is the release, a minor enhancement. IERELEASE is two digits, padded on the left with zeros.

WARNING record

The WARNING record indicates a minor problem during the completion of a command.

The following example shows the format of the WARNING record:

```
WARNING(warning) ERRDESC(error description);
```

warning

Indicates the warning number. This parameter contains 5 numeric characters.

errdesc

Provides a short description of an error. This parameter contains 1 to 76 alphanumeric characters.

Chapter 10

Using additional features

You can use Expedite Base/AIX to compress and decompress data, retrieve audit data, query your mailbox, request acknowledgments, and work with libraries. You can also use it to archive files and validate addresses, payment levels, and authorizations. This chapter describes how to use these features and discusses how you can use command line parameters to implement some of the Expedite Base/AIX functions.

Compressing and decompressing data

If you have the appropriate compression and decompression software installed, you can request Expedite Base/AIX to compress data you are sending and decompress data you are receiving. Expedite Base/AIX provides a COMPRESS parameter on the SEND and SENDEDI commands to request compression. Fields in the RECEIVED record and in the CDH provide information on the decompressed data you receive. See Appendix E, “Using data compression,” for more information.

Using audit trails

Information Exchange provides an audit trail that you can retrieve using Expedite Base/AIX or view using Information Exchange Administration Services. For information on viewing audit data, see the *Information Exchange Administration Services User's Guide*. Audit trails are files that contain information about files you send and receive. Audit trails can include the following information:

- Name of the file sent or received
- Account and user ID of the person who received the file
- Account and user ID of the person who sent the file
- Date and time the file was sent or received
- User class of the file
- Current status of the file (for example, delivery date and time)

You can request either level 1, 2, or 3 audit trails. Level 1 audit trails contain basic information about your files. Level 2 audit trails contain more detailed information about your files. Level 3 audit trails contain the most detailed information about your files.

You can use the information in audit trails to see the status or final disposition of a file. For example, if you send an order electronically to a manufacturer on July 1 and do not receive the items as expected, you can request an audit trail of the files you sent to that manufacturer on July 1. The audit trail may show that the manufacturer never received the order and it is still in their mailbox. If the audit trail shows that they received the order, you may want to check with the manufacturer to find out if the order was ever filled.

You can also use audit trails to see how many files you sent or received over a specific time period. For example, you can request an audit trail that shows all of the files you sent over the last three weeks.

Retrieving audit trails

Use the AUDIT command to retrieve an audit trail from Information Exchange and place the information in your mailbox. When audit data is available, use the RECEIVE command to retrieve it. The source account the file comes from is **SYSTEM**, the user ID is **AUDITS**, and the user class is *#SAUDIT*. Information Exchange does not prepare a common data header (CDH) to accompany retrieved audit records.

Audits are not available immediately; you cannot successfully issue the RECEIVE command after the AUDIT command to receive the audit data. Audits are normally available during the next session. For more information on the AUDIT and RECEIVE commands, see "AUDIT command" on page 169 and "RECEIVE command" on 199.

The following steps show how you request and receive audit trails and what an audit trail looks like.

Step 1
The following example shows the basein.msg file you would use to create an audit file with information about all files received between July 1 and July 5 with a user class of ORDERS.

```
AUDIT STARTDATE(19980701) ENDDATE(19980705) CLASS(ORDERS);
```

The output file baseout.msg indicates a RETURN(00000) if the audit command was processed properly. The result would be a file placed in your mailbox with account **SYSTEM** and user ID **AUDITS**, with user class *#SAUDIT*.

Step 2
The next step is to receive the audit file from your mailbox. The following example shows 2 commands that you can use in basein.msg to receive the audit files.

```
RECEIVE FILEID(audits.fil) CLASS(#SAUDIT);
RECEIVE FILEID(audits.fil) ACCOUNT(*SYSTEM*) USERID(*AUDITS*);
```

If there was an audit file in your Information Exchange mailbox, the output file baseout.msg would have a RECEIVED record indicating the audit file was received.

Step 3
The following example shows the audit trail as it would appear in the file called audits.fil.

NOTE: When you receive files from your mailbox, the data is stored as a single record. The record length is 254 bytes for a level 1 audit trail, 326 bytes for a level 2 and for level 3. For this example the audit record has been split in order to display it on this page. For more information on audit record formats, see the following section.

```
ACCT  SENDER  1  ACCT  USERA  A69438B7.554CAE3F51RECEIVED
SORDERS            EXP/AIX    410 00000000200053473    980702123630
980702143701980702123703980702123704
```

Message audit record formats

Information Exchange provides three levels of audit record data. The length of the level 1 message audit record is 254 bytes and contains all fields up to and including AOEMSGID (field 30). The length of the level 2 message audit record is 326 bytes and contains AOSNDEDD (field 31) through AORCVCMT (field 42) in addition to AUSERID (field 1) through AOEMSGID (field 30). The length of the level 3 message audit record is 340 bytes and contains field 43 in addition to fields 142. All fields are in character format. Refer to *Information Exchange Administration Services Messages and Codes* for more information on audit record format and messages.

Querying a mailbox

Use the QUERY command to see a list of all the files in your Information Exchange mailbox. Expedite Base/AIX places AVAILABLE records in the response file in response to the QUERY command. It writes an AVAILABLE record for each file in your mailbox.

The information obtained in response to the QUERY command can be very useful for several reasons. The AVAILABLE records show you all of the files waiting to be received from your mailbox. Using this information, you can build RECEIVE or RECEIVEEDI commands to receive all of these files. In addition, the AVAILABLE records provide a MSGKEY for each of the files in the mailbox. You can use this information on a RECEIVE or RECEIVEEDI command to receive a specific file.

Example

This example shows how to use the QUERY command to make sure that you receive all files in your mailbox.

Company A receives e-mail and software updates from its trading partner. Both companies agree that the e-mail will have a user class of *FFMSG001* (this is the default for e-mail) and the software updates will have a user class of update. Company A builds an input file (basein.msg) with one RECEIVE command to receive all files with class *FFMSG001* into a file called e-mail.fil. The second RECEIVE command receives all files with class update into file software.fil.

On one occasion, the trading partner sent Company A a software update with the wrong user class, *updates*, instead of *update* Because Company A tries to receive using class *update*, it will not receive the file with class *updates*. They may never find out that the file is even in their mailbox.

If they issue a QUERY command in each session, they will see the file with class *updates* and know to receive it. The following examples show how to add the QUERY command to basein.msg and the resulting baseout.msg.

NOTE: The available record may have more information than is shown in this example.

The following is an example of the basein.msg file:

```
RECEIVE FILEID(email.fil) CLASS(FFMSG001);
RECEIVE FILEID(software.fil) CLASS(UPDATE);
QUERY CDH(N);
AUTOSTART SESSIONKEY(DDKL7749);
STARTED LASTSESS(0) RESPCODE(00000) SESSIONKEY(DDKL7749)
IEVERSION(04)
IERELEASE(05);
RETURN(00000);
```

```
RECEIVE FILEID(email.fil) CLASS(FFMSG001);
RECEIVED ACCOUNT(ACCT) USERID(PARTNER) CLASS(FFMSG001) CHARGE(5)
LENGTH(490) FILEID(email.fil) MSGDATE(980701) MSGDATELONG(19980701)
MSGTIME(113314) MSGSEQO(001950) SESSIONKEY(DDKL7749)
DELIMITED(N) SYSNAME(EB/AIX) SYSLEVEL(0450) TIMEZONE(L)
```

```
EDITYPE(UNFORMATTED) SENDERFILE(mail0a)
SENDERLOC(/home/email) FILEDATE(980630) FILEDATELONG(1998063.)
FILETIME(160340) RECFM(????) RECLEN(00000) RECDLM(C)
UNIQUEID(74662366) SYSTYPE(12) SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

RECEIVE FILEID(software.fil) CLASS(UPDATE);
RECEIVED ACCOUNT(ACCT) USERID(PARTNER) CLASS(UPDATE) CHARGE(5)
LENGTH(5781) FILEID(software.fil) MSGDATE(980701)
MSGDATELONG(19980701)
MSGTIME(113314) MSGSEQO(001950) SESSIONKEY(DDKL7749) DELIMITED(N)
SYSNAME(EB/AIX) SYSLEVEL(0450) TIMEZONE(L) DATATYPE(A)
EDITYPE(UNFORMATTED)
SENDERFILE(mail0a) SENDERLOC(/home/email) FILEDATE(980630)
FILEDATELONG(19980630) FILETIME(160340) RECFM(????)
RECLEN(00000) RECDLM(C) UNIQUEID(74662368) SYSTYPE(12)
SYSVER(1) TRANSLATE(IESTDTBL);
RETURN(00000);

QUERY CDH(N);
AVAILABLE ACCOUNT(ACCT) USERID(PARTNER) MSGKEY(1234567890123456789.)
CLASS(UPDATES) MSGDATE(980702) MSGTIME(081522) LENGTH(5000)
SYSTYPE(12) SYSVER(1);
RETURN(00000);

AUTOEND;
RETURN(00000);
SESSIONEND(00000);
```

This response to the QUERY command in this output file shows that there is one file available in the mailbox. The file is from account *acct* and user ID *partner*. It has a user class of *updates* and a message key of *12345678901234567890*. Based on this information, the user at Company A can build an input file to receive this file using a number of methods, including:

- The RECEIVE command can receive all files with a user class of *updates*.
- The RECEIVE command can specify a message key of *12345678901234567890* to receive the file. This is useful if there are multiple files in the mailbox from the same account and user ID, with the same user class, but you only want to receive one of them.

Using mailbox query with a panel-driven interface

If you have written a panel interface to Expedite Base/AIX, you may decide to allow users to query the mailbox and then receive selected individual files.

Your program would have to read the AVAILABLE records and save the appropriate information, including the unique message key. This key is displayed in the MSGKEY parameter of the AVAILABLE response record. Your program might present a panel with a list of the files available in the user's mailbox. The user could select individual files to receive. Based on these selections, your program could build RECEIVE or RECEIVEEDI commands in basein.msg that have the MSGKEY of the files they want to receive. For more information on receiving specific files, see "Receiving specific files" on page 71.

Using acknowledgments

You can request three types of Information Exchange acknowledgments using the ACK parameter in the SEND and SENDEDI commands:

receipt	Information Exchange generates a receipt acknowledgment when a file reaches the receiver's mailbox after a successful Expedite Base/AIX session.
delivery	Information Exchange generates a delivery acknowledgment when a destination user receives a file from the Information Exchange mailbox.
purge	Information Exchange generates a purge acknowledgment when a file is purged from the receiver's mailbox.

To retrieve these acknowledgments from your mailbox, use the RECEIVE command and specify an account ID of **SYSTEM** and a user ID of **ERRMSG**.

For information on acknowledgment formats, see *Information Exchange Messages and Formats*.

Working with libraries

A library is a facility of Information Exchange that allows data to be stored for an extended period of time. Unlike messages in a user's mailbox, information in a library is not deleted automatically after a certain amount of time or after all receivers have picked up the information. Some uses for libraries are:

- Product catalog information
- Technical specifications
- Problem descriptions
- Programs
- Newsletters
- Requests for quotations

Data is stored in a library in units called *library members*. For example, a product catalog library can consist of a separate member for each product. Expedite Base/AIX allows users to add members to a library as well as retrieve members from a library and put them in an Information Exchange mailbox.

To use libraries in Expedite Base/AIX

1. Create a library using Information Exchange Administration Services. The characteristics of a library that are specified when the library is created include:

Library name:

The name of the library that is unique within the account.

Owning account ID:

The account ID of the library owner.

Owning user ID:

The user ID of the library owner.

Library description:

A description that helps identify the library within a list of libraries.

Searchable indicator:

Determines whether or not the library members are searchable.

Owner willing to pay:

Specifies whether or not the owner is willing to pay for others to retrieve members or view the text of the members from the library.

Read and write authority:

Specifies who has the authority to list the members of a library, view the text of library members, search, retrieve, add, and delete library members.

2. Use the GETMEMBER and PUTMEMBER commands as appropriate. For more information, see “GETMEMBER command” on page 183 and “PUTMEMBER command” on 194.

3. Use the LISTLIBRARIES and LISTMEMBERS commands to identify libraries and members to which you have access. For more information, see “LISTLIBRARIES command” on page 191 and “LISTMEMBERS command” on 192.

For more detailed information about how to define and work with libraries, refer to the *Information Exchange Administration Services User's Guide*.

Adding and retrieving library members

Expedite Base/AIX provides the GETMEMBER and PUTMEMBER commands that enable you to work with libraries. Use the GETMEMBER command to retrieve a library member from an existing Information Exchange library and put it in your mailbox. The member is available in your mailbox as soon as the GETMEMBER command completes. Use the PUTMEMBER command to add a library member to an existing Information Exchange library. The following examples show how to add members to and retrieve members from an Information Exchange library.

Adding members to a library example

Vendor A builds standard applications for distribution to many businesses. When Vendor A develops software upgrades, it makes these upgrades available to the businesses by putting the code into an Information Exchange library. This way, it only needs to provide one copy of the new code, instead of having to mail diskettes to all of the businesses.

```
PUTMEMBER LIBRARY(NEWCODE) MEMBER(CODEFIX) FILEID(xyz123);
```

This command specifies that file xyz123 is to be added to library *newcode*. This library member will have the name *CODEFIX*.

After the member is put into the library, it is available for anyone who has read access to the library.

Retrieving members from a library example

To get a copy of the member described in the previous example, you must use the GETMEMBER command shown in the following example.

```
GETMEMBER LIBRARY(NEWCODE) MEMBER(CODEFIX) CLASS(UPGRADE);
```

This command tells Information Exchange to copy member *CODEFIX* from library *newcode* into this user's mailbox. The file in the mailbox will have a user message class of *upgrade*. After the file is in the mailbox, the user must issue a RECEIVE command to receive it from Information Exchange. The RECEIVE command might look as follows:

```
RECEIVE CLASS(UPGRADE) FILEID(xyz123) ORIGFILE(Y)
```

After you issue the RECEIVE command, the file is stored on the workstation using the original file name of *xyz123*.

Identifying libraries and library members

Expedite Base/AIX provides the LISTLIBRARIES and LISTMEMBERS commands to allow you to identify libraries and members to which you have access.

Use the LISTLIBRARIES command to identify libraries to which you have access. Expedite Base/AIX returns this information in a LIBRARYLIST record.

Use the LISTMEMBERS command to identify library members to which you have access. Expedite Base/AIX returns this information in a LIBRARYLIST record.

Identifying library members example

To obtain member information about the library described in the previous examples, use the following LISTMEMBERS command:

```
LISTMEMBERS LIBRARY(NEWCODE);
```

Expedite Base/AIX returns the description (if one exists) in the DESCRIPTION parameter of a MEMBERLIST record.

Identifying libraries example

To identify the libraries to which you have read access, use the following LISTLIBRARIES command:

```
LISTLIBRARIES AUTHORITY(R);
```

This command tells Information Exchange to determine which libraries you can read, and to return a description of each library. Expedite Base/AIX returns the descriptions (if they exist) in the DESCRIPTION parameter of a LIBRARYLIST record.

Using acknowledgments with libraries

If you request an acknowledgment with the GETMEMBER command, three types of acknowledgments are available:

receipt	Information Exchange creates a receipt acknowledgment when a member is placed in the receiver's mailbox.
delivery	Information Exchange creates a delivery acknowledgment when the member is successfully received from the receiver's mailbox.
purge	Information Exchange creates a purge acknowledgment when a member is deleted from the receiver's mailbox.

If the library owner is paying for the receive charges for the member, the library owner receives the acknowledgment. Otherwise, the individual who issued the GETMEMBER request receives the acknowledgment.

If you request an acknowledgment with the PUTMEMBER command, two types of acknowledgments are available:

delivery	Information Exchange creates a delivery acknowledgment when the member is placed in the library.

The acknowledgment is always sent to the individual that issued the PUTMEMBER command.

NOTE: The format of the acknowledgment when using libraries is the same as described in "Using acknowledgments" on 268.

Understanding validations, payment levels, and authorizations with libraries

As described previously in "Working with libraries," either a library owner or an administrator working on the owner's behalf creates a library using Information Exchange Administration Services. The person creating the library specifies the access authority level for each library user and specifies whether or not the library owner is willing to pay for others to retrieve members or view the text of members from the library. The following describes authority levels and charges for using Information Exchange libraries.

Understanding access authority levels

Authority levels can be either read or write access for each user of an Information Exchange library. Users who have read authority can look at library and member information, view the text of library members, search library members, and retrieve library members. Users who have write authority can also add, replace, and delete library members.

NOTE: Write access does not give a user authority to change library information or delete a library. Only the owner of a library, the owner's Information Exchange service administrator, or an alternate administrator for the owner can change library information or delete a library.

Understanding library charges

There are four types of library charges. You can incur charges for:

- Storing library members

  The library owner is charged a storage fee for the number of characters of data in library members.

- Adding or replacing a library member

  Each time you add or replace a library member you incur Information Exchange charges.

- Viewing the text of a library member

  Each time you use Information Exchange Administration Services to look at the text of a library member you incur Information Exchange charges.

- Retrieving a library member

  After you request that a library member be sent to your mailbox, an Information Exchange file containing the library member appears in your mailbox. When you receive the file from your mailbox, you incur Information Exchange charges.

Archiving and retrieving files

Information Exchange enables you to archive files when you receive them. This may be useful to you if you need a copy of a file in the near future. Information Exchange keeps a copy of your files for the number of days specified in your Information Exchange profile. Information Exchange does not keep the files in your Information Exchange mailbox. Refer to the *Information Exchange Administration Services User's Guide* for information about updating your Information Exchange profile to allow for archiving.

Archiving all files

To set up your Information Exchange profile so that Information Exchange archives every file you receive, use Information Exchange Administration Services to set FORCED ARCHIVE to *y* in your profile.

You can also specify the number of days that Information Exchange keeps the archived files. For example, if your profile indicates forced archiving for ten days, then each time you receive a file from your mailbox, Information Exchange keeps a copy of that file for ten days.

During those ten days, you can have Information Exchange put a copy of the archived file into your mailbox and you can receive it again. See “Retrieving files from the archive” on page 274. After the ten days have passed, Information Exchange deletes the archived file and you cannot receive it again.

Archiving selected files

Expedite Base/AIX and Information Exchange enable you to archive files on a file-by-file basis. For example, you may find it unnecessary to archive every file you receive. You can set up your profile so that you decide whether or not to archive files when you receive them.

To archive selected files, use Information Exchange Administration Services to set FORCED ARCHIVE to n in your profile. Then specify how long you want Information Exchange to keep copies of the files you archive.

When you set FORCED ARCHIVE to *n* and specify a number greater than zero for the number of days to keep archived files, Information Exchange archives files only if you use the ARCHIVEID parameter in the RECEIVE or RECEIVEEDI commands. For example, if you want to receive two files from your Information Exchange mailbox and you want to archive the first file, but not the second, make sure that FORCED ARCHIVE is set to n in your Information Exchange profile and that the number of days to keep files is greater than zero. Use the ARCHIVEID parameter on the first RECEIVE command, and omit the parameter on the second RECEIVE command.

When you use ARCHIVEID, Information Exchange keeps copies of the files you received for the number of days specified in your profile. Information Exchange uses the value specified for the ARCHIVEID parameter as the archive identifier. It uses the identifier to find the files you want to retrieve from the archive. Refer to “RECEIVE command” on page 199 and “RECEIVEEDI command” on 207.

NOTE: Information Exchange will not archive files you receive from your mailbox (even if you use ARCHIVEID) if the number of days for keeping archived files is set to zero in your Information Exchange profile.

Example 1

To archive every file that you receive for a maximum of ten days, use Information Exchange Administration Services to modify your Information Exchange profile as follows:

1. Set the field FORCED ARCHIVE to *y*.
2. Set the field NUMBER OF DAYS to *10*.

Results: When you receive a file, a copy of the file will be kept in the Information Exchange archive for ten days. If you specify an ARCHIVEID parameter in the RECEIVE command, then that value will be used as the archive identifier when the file is copied to the archive. If you do not specify an ARCHIVEID parameter, Information Exchange will assign an archive identifier automatically. You can refer to the “RECEIVE command” on page 199 to see what value was assigned.

Example 2
To selectively archive only certain files that you receive for a maximum of 15 days, do the following:

1. Use Information Exchange Administration Services to modify your Information Exchange profile as follows:

   a. Set the field FORCED ARCHIVE to *n*.

   b. Set the field NUMBER OF DAYS to *15*.

2. Use the ARCHIVEID parameter in the RECEIVE command for the files that you wish to archive. For example:

```
RECEIVE FILEID(file1.fil) ARCHIVEID(MYARCHID) CLASS(ARCHIVE);
RECEIVE FILEID(file2.fil) CLASS(NOARCH);
```

Results: File file1.fil will be received and a copy of the file will be kept in the Information Exchange archive for 15 days. File file2.fil will be received but there will be no copy of the file maintained in the Information Exchange archive.

Retrieving files from the archive

To retrieve a file from the Information Exchange archive, you must first move a copy of the archived file from the archive to your Information Exchange mailbox. After the file copy is in your mailbox, you can use the RECEIVE or RECEIVEEDI command to receive the file.

There are two ways to copy a file from the archive to your mailbox:

1. Use Information Exchange Administration Services. To learn how to use Information Exchange Administration Services, refer to the *Information Exchange Administration Services User's Guide*.

2. Use the Expedite Base/AIX ARCHIVEMOVE command. See "ARCHIVEMOVE command" on page 168 for information about the ARCHIVEMOVE command.

To use Expedite Base/AIX to copy an archived file, you specify the archive identifier for the file you want to copy to your mailbox by using the ARCHIVEID parameter on the ARCHIVEMOVE command. You can assign identifiers for files in one of two ways:

1. Use the ARCHIVEID parameter the first time you issue the RECEIVE or RECEIVEEDI command for the file.

2. Have Information Exchange assign identifiers automatically by setting FORCED ARCHIVE to y in your Information Exchange profile. In this case, Information Exchange assigns archive identifiers each time you issue RECEIVE and RECEIVEEDI commands without the ARCHIVEID parameter. The value assigned is shown in the SESSIONKEY parameter on the RECEIVED record in baseout.msg.

The ARCHIVEMOVE command tells Information Exchange to put a copy of the file with the specified archive identifier in your Information Exchange mailbox. For example, to place a copy of a file in the Information Exchange archive with the identifier of *MYARCHID* in your Information Exchange mailbox, specify the following command in the message command file basein.msg:

```
ARCHIVEMOVE ARCHIVEID(MYARCHID);
```

When Information Exchange processes this command, it places the file in your mailbox. To receive the file from your mailbox, issue the RECEIVE or RECEIVEEDI command. You can receive the file immediately after you issue the ARCHIVEMOVE command in basein.pro.

The response record for the ARCHIVEMOVE command is the MOVED record. Expedite Base/AIX places this record in baseout.msg and includes the number of files that were copied from the Information Exchange archive to the mailbox. If there were no files in the archive matching your request, the response record will be:

```
MOVED NUMBER(0);
```

Traveling user

The Traveling User feature allows you to access Information Exchange when traveling abroad, and can be used with any Expedite Base Version 4.5 product using asynchronous dial or 3270 emulation.

A Traveling User connect script, tuccnct.scr, is provided with Expedite Base/AIX to support Traveling User for use with asynchronous or 3270 communications.

To use the Traveling User Feature, you will need the following additional hardware:

- A modem cable compatible with the phone jacks in the country you plan to visit
- A voltage converter and plug adapter, if necessary

ATTENTION: Before dialing from a new location, ensure that the telephone line is a proper asynchronous data line and not a digital PBX line, which could damage your modem.

You will also need to make the following changes to Expedite Base/AIX:

1. Obtain the telephone number you will use at your destination, the baud rates supported by the telephone number, and your home ISO country code from your local Help Desk.
2. Update the tucnnct.scr file with your home ISO country code.A comment in the file marks the proper location, which is near the end of the file.
3. Update basein.pro file DIAL command:
    - Change the telephone number specified by the PHONE parameter to the telephone number provided to you for your destination. Include the international access code, country code, and city code if provided.
    - Verify that the telephone number you will use at your destination supports the data rate specified by the BAUDRATE parameter. If not, update the BAUDRATE to a supported value.

- If you need to use an escape sequence to access an outside line (such as dialing a 9 before the phone number in an office or hotel), add the ESCAPE parameter to the DIAL command. If you have an ESCAPE parameter and do not need it, change it to blanks. Do not delete the ESCAPE parameter.
- Change the CNNCTSCR parameter to the tucnnct.scr (the Traveling User Feature connect script that you will use at your destination).

4. Establish the connection by starting Expedite Base/AIX.

Understanding validations, payment levels, and authorizations

When you communicate with trading partners, you can reduce the cost of transmitting data and ensure delivery of data by verifying the following:

- The address you want to send a file to is a valid address.
- You can use a particular payment level with the intended destination.
- The intended destination has appropriate authorization levels to receive a file from your Information Exchange address.

If you are on the same system as the intended destination, you can use the VERIFY parameter in the SEND and SENDEDI commands to validate this information before you send a file. You can use the VERIFY parameter for distribution lists as well as for individual destinations. However, this parameter verifies only that the distribution list exists. It does not verify the existence of individual entries within the distribution list.

The following table shows which circumstances can result in a message charge:

If the destination system is:	And the destination:	Then a charge is:
On the same system	Can be verified	Not incurred
On the same system	Is invalid	Incurred
Another Information Exchange system	Cannot be verified	Not incurred

Using the VERIFY parameter, you can decide whether or not Expedite Base/AIX should send the data even if Information Exchange cannot verify the destination of a trading partner on another Information Exchange system.

If you are the sender, you can indicate how you want to pay file charges by using the CHARGE parameter in the SEND and SENDEDI commands. You can specify one of six payment options. For more information, see the *Information Exchange Charges Reference*.

For details on the VERIFY and CHARGE parameters, see “SEND command” on page 216 and “SENDEDI command” on 223.

Using command line parameters with the IEBASE command

Expedite Base/AIX provides command line parameters for some functions that you cannot easily implement using message commands. The command line parameters you can use with Expedite Base/AIX are:

- **-c**

  Using this parameter, you can verify that basein.pro and basein.msg are valid without running a session with Information Exchange. With the c command line parameter you can specify one of the optional parameters, c 1 or c 2, where:

1	Indicates that the syntax of the commands in basein.pro and basein.msg is to be checked.
2	Indicates that the syntax of the commands in basein.pro and basein.msg is to be checked, and in addition the existence of any files specified in the SEND, SENDEDI and PUTMEMBER commands is to be checked. This is the default.

  For example:

  ```
  iebase -c 1
  ```

  causes Expedite Base to check the syntax of the commands in basein.pro and basein.msg, write the results in the SESSIONEND record in the baseout.msg file, and then exit without attempting to process any commands or connect to Information Exchange.

  If the SESSIONEND record indicates a return code other than 00000, review the error message text and correct the problem. If you have specified checkpoint-level, file-level, or user-initiated data recovery, you will find more details about the error in tempout.msg. For more information, you can also specify y for the IOFILE parameter on the TRACE command in basein.pro to cause Expedite Base to produce a trace of the syntax parsing in the iebase.trc trace file.

  The internal profile, iebase.pro, will be modified during this check with the changes specified in basein.pro.

- **-b**

  This parameter indicates to Expedite Base/AIX that it is running in the background, and no attempts should be made to access the terminal. It is recommended that you use the **-p** parameter with this command line option.

- **-p<path>**

  With this parameter, you can specify a path. This path determines the directory for the message input files, message output files, the session file, the profile input file, the profile output file, and the trace files. The path is not used for the files sent and received or for the Expedite Base/AIX program files (such as display.scr or cnnct.scr). Use up to 128 characters. It is recommended that you use this parameter if you use the -b option.

- **-r**

  This parameter erases the information in the session file. It causes Expedite Base/AIX to reset the session with Information Exchange instead of restarting it at the last checkpoint.

- **-s**

  This parameter causes Expedite Base/AIX to return a code to the caller (or the operating system) that indicates whether there is a restart situation. It does not do any input or output. It only checks to see if a session file exists and returns the status immediately. An exit code of 0 indicates that there is no restart. Any other return code indicates that Expedite Base/AIX will attempt to restart

- **-v**

  This parameter causes Expedite Base/AIX to return the version number and date of the executable module, *iebase*. The format of the string that is returned is as follows:

  ```
  Version x.x, mm/dd/yy
  ```

  Where *x* is a digit from 09, mm is the two-digit month, *dd* is the two-digit day, and *yy* is the last two digits of the year.

  If you specify other command line parameters with the -v parameter, the other parameters will be ignored.

Type the command line parameters in lowercase. They follow the program name on the command line, as in the following example.

```
iebase -r -p /u/user1
```

Unattended operation

You can schedule a transmission for a later time using the AIX cron function, or you can run Expedite Base/AIX in the background with the & command line option. To do so, use the -b and -p parameters with the path (starting from the root, /, to the directory that contains basein.msg and basein.pro). The b switch tells Expedite Base/AIX not to access the terminal during operation. Also, all FILEID parameter values on commands in basein.msg must indicate the full path to the file starting from the root.

You must make sure that you have permission to use cron. See the *Expedite Base Programming Commands Quick Reference* for more information.

To use cron, create the crontab file as directed in the *Expedite Base Programming Commands Quick Reference*. For example, if you want the transmission to occur Sunday, June 11th, your crontab file should look like the following:

```
00 00 0 11 6 iebase -b -p /u/user1
```

Chapter 11

Communicating with users on different operating systems

Before you send files between Expedite products on different operating systems, consider the following questions:

- Is the receiving operating system the same type of system as the sending operating system?
- What is the record format of the file being sent?
- Is a binary file being sent?

This chapter addresses each of these questions and the actions you take depending on the answers. It also discusses a valuable aid to file transfer, the *common data header* (CDH).

NOTE: In this chapter, system refers to a particular type of operating system, rather than to a particular Information Exchange system.

Using the common data header

The common data header (CDH) contains data that Expedite Base/AIX uses to communicate detailed information about files and messages to other interfaces and to Information Exchange. The CDH provides details that enable the receiving interface to reconstruct a received message or file into its proper format. Refer to Appendix B, “Common data header,” to learn more about the CDH format.

When you are exchanging information with users on other types of operating systems, there are specific fields in the CDH that may be of interest to you. These fields are:

File name

This field contains the name of the file as it is stored on the sender’s operating system. The RECEIVE command can use this information to store the file on the receiving operating system using the same name.

However, you cannot take advantage of this feature if the naming convention used on the sending operating system is different than the receiving operating system. For example, if the sender's operating system is a host MVS operating system, and the original file name stored in this CDH field is sender.file.name, Expedite Base/AIX cannot store the file with this name because the file name is invalid.

Location of file

This field specifies the location of the file on the sender's operating system. The information stored in this field is specific to the type of operating system sending the data.

System type:	Field contents:
MVS	Device type and volume number
VM	Mini-disk label
AIX or UNIX	Data path
PC	Drive and directory

As with the file name field, you cannot take advantage of this feature if the sending operating system and receiving operating system are different types.

Record format

This field identifies the format of the data records. If you are sending data to an Expedite Base/VM user, you can specify the record format that the receiver should use when receiving your data.

Record length

This field identifies the length of the data records. If you are sending data to an Expedite Base/VM user, you can specify the record length that the receiver should use when receiving your data.

NOTE: The operating system uses the newline character (x'0A') to delimit lines of text in a file. Other workstation operating systems use the carriage return and line feed characters (x'0D0A') to delimit lines of text. To allow for consistency when sending data to users of other operating systems, Expedite Base/AIX inserts carriage return characters before the newline characters when sending data to Information Exchange, with two exceptions. Expedite Base/AIX will not insert carriage returns before newline characters in the following situations:

- You specify DELIMITED(N) in the SEND command.
- You specify DATATYPE(B) in the SEND command to indicate binary data.

When you receive data, Expedite Base/AIX removes carriage return characters from the data if:

- The CDH indicates the data is delimited by CRLF characters
- The data-type in the CDH is blank.
- You specify DELIMITED(Y) in the RECEIVE command.

Communicating with interfaces that do not support the CDH

Expedite Base/AIX can communicate with interfaces that do not support the CDH. However, there are restrictions in some features that are dependent on the CDH. These restrictions include:

- Versions of expEDIte/PC before Release 2 do not have an option to send binary data. All files sent by these releases are translated from ASCII to EBCDIC when sent to Information Exchange.
- Versions of expEDIte/PC before Release 2 cannot use alternate translate tables when receiving files.
- Expedite Base/AIX ignores the AUTOEDI and PROCESSLEN parameters in the RECEIVE command in files received from interfaces that do not support the CDH.
- Some of the information in the RECEIVED record (for example, the DESCRIPTION parameter) is not available for files received from interfaces that do not support the CDH.

Sending files to an ASCII operating system

You can send text files and binary files to an ASCII operating system. When you send a text file, Expedite Base/AIX translates the file from ASCII to EBCDIC as it sends the file to Information Exchange.

When you send a binary file, you can use the DATATYPE parameter in the SEND command to indicate the file contains binary data. If you specify DATATYPE(B) indicating binary data, Expedite Base/AIX does not translate the file when it sends it. When you send a binary file without specifying DATATYPE, Expedite Base/AIX translates the file from ASCII to EBCDIC even though the file does not contain text characters.

NOTE: The CDH indicates the type of data a file contains. If the receiving operating system is an ASCII operating system and it does not recognize the CDH, the ASCII operating system translates binary files from EBCDIC to ASCII and makes them unusable. Therefore, do not specify DATATYPE when you send files to versions of expEDIte/PC before Release 2.

Receiving files from an ASCII operating system

When you receive a file, Expedite Base/AIX checks the CDH to see if the file is EBCDIC or binary. If the file is EBCDIC, Expedite Base/AIX translates the file to ASCII as it receives it. If the translate table on your operating system matches the sender's, the file should be a perfect match of the original file. When the tables do not match, the file is damaged.

NOTE: To avoid problems with translation, use the Information Exchange default translate table. Expedite Base/AIX will use this table by default if you do not specify a translate table.

If the CDH indicates the file is binary, Expedite Base/AIX does not translate it because the sending operating system did not translate the original file. If the data type is unknown because there is no CDH, Expedite Base/AIX assumes the file is EBCDIC and translates it to ASCII. If the file is actually binary, the translation makes it unusable.

If your trading partner sends you a binary file using an interface that does not support the CDH, you can receive the file using the alternate translate table NOXLATE.XLT. The end result of this is that no translation is done when the file is received.

Sending files to an EBCDIC operating system

You can send text files and binary files to an EBCDIC operating system. When you send a text file, you must change the record structure of the data to that of the receiving operating system. Use the DELIMITED parameter in the SEND command to specify what delimiter type, if any, Expedite Base/AIX puts in the data and in the CDH for the receiving operating system. Expedite Base/AIX translates text files to EBCDIC when it sends them.

When you send a binary file to an EBCDIC operating system, there are several things you must consider when deciding how to send the file. If the receiver is going to use the file in the same format as that on your operating system, then you can specify DATATYPE(B) to indicate that the file is binary and no translation should be done on either the send or receive side.

If the receiver receives the data using a host Expedite product, and then downloads the data to their PC using an emulator, the data translation depends on what method the emulator uses. If the emulator uses IBM Personal Communications 4.2 or later translation, you may need to send the file using the DATATYPE(A) and TRANSLATE(IBM3270) parameters. This will result in an ASCII to EBCDIC translation using tables for the IBM Personal Communications 4.2 or later program. When your trading partner downloads the file from their host to their PC, the translation will result in a PC file identical to the one you sent. If your trading partner's emulator uses a different translation method, you may need to use an alternate translate table when sending the files.

In summary, you should consider how your trading partner receives and translates files in order to determine how to best send those files.

Receiving files from an EBCDIC operating system

When you receive a file with a CDH from an EBCDIC operating system, Expedite Base/AIX uses the record delimiters in the CDH to reconstruct the record format. Expedite Base/AIX translates these files from EBCDIC to ASCII. However, there is not a perfect match between EBCDIC and ASCII characters. Most characters translate properly, but some EBCDIC characters may not look correct on an ASCII operating system. Before you begin exchanging production data with a trading partner on an EBCDIC operating system, be sure to run tests to verify that ASCII to EBCDIC translation works as expected.

“Example 1” on page 283 provides more information about how host operating systems store data and how you use the LRECL and RECFM parameters to send files to host operating systems.

Using alternate translate tables

Previous sections of this chapter discussed how Expedite Base/AIX translates data from ASCII to EBCDIC when sending it to Information Exchange. When receiving data, Expedite Base/AIX translates from EBCDIC back to ASCII. To do this, Expedite Base/AIX uses a translate table, which specifies ASCII characters and the EBCDIC characters that they are translated to (and vice versa). Appendix D, “Information Exchange translate table,” shows the standard translation table.

When you are exchanging data with a trading partner, it is important that the same translate tables are used on the sending and receiving sides so that the data received is identical to the data sent. If both you and your trading partner are using Expedite Base/AIX products in workstation environments, there should be no problem. However, there are certain scenarios which might require the use of an alternate translate table.

Expedite Base/AIX provides two alternate translate tables.

- **IBM3270.XLT**

  This is the translate table used by the IBM eNetwork Personal Communications 4.2 program. As an example, suppose your trading partner uses a host product to receive the data from Information Exchange, and then downloads the data to a PC using IBM eNetwork Personal Communications 4.2. It might be best if you use the alternate translate table IBM3270.XLT when sending the data so that you and your trading partner are using the same translate tables.

- **NOXLATE.XLT**

  This is an alternate translate table. As the name implies, this table actually results in no translation of the data. This table is useful if you are receiving data that does not have a CDH with it, but you do not want the data translated. If there is no CDH, Expedite Base/AIX assumes it is EBCDIC text data and will use a translate table when receiving it. If you use the NOXLATE.XLT translate table, you will receive the data exactly as it was in your Information Exchange mailbox.

If neither the standard translate table or the alternate translate tables meet your needs, you can create your own. A sample C program called makexlt.c is provided with the Expedite Base/AIX product. Edit this file to change the translate characters and recompile it. Comments within makexlt.c provide directions to do this. If you create your own translate table, be sure to provide this table to your trading partners so that the translation on both the send and the receive sides is the same.

Learning more about sending and receiving files on different operating systems

This section provides examples for sending and receiving text and binary files to and from trading partners who use Expedite Base/AIX, IBM expEDIte/PC, and host operating systems. The host operating systems your trading partners use may or may not support the common data header (CDH). The following includes examples for both cases.

Example 1

Host operating systems store data with a specified logical record length (LRECL) and record format (RECFM). Data on workstations is not stored the same way. Records on a workstation are generally delimited by carriage-return and line-feed (CRLF) characters. This difference sometimes causes problems when you send files using a workstation product and the receiver is on a host operating system.

The Expedite Base/AIX SEND command allows you to indicate how the data should be structured on a host operating system. The parameters that allow you to do this are the LRECL and RECFM parameters.

For example, Company X uses Expedite Base/AIX to send files to Company Y. Company Y uses Expedite Base/VM to receive the data. The data is structured in 80character records. On the Company X workstation, this means the data is formatted with CRLF following each 80 characters of data. To ensure that the data is properly formatted when it is received by Expedite Base/VM, use the LRECL and RECFM parameters as follows:

```
SEND FILEID(aaaa.fil) ACCOUNT(ACCT) USERID(COMPANYY) LRECL(80)
RECFM(F);
```

The receiver can properly format the data by using the record format and record length information stored in the CDH. The receiver can receive the data using a fixed record format with an 80-character record length.

Example 2

The following example shows how you send text and binary files to trading partners who use the same release of Expedite Base/AIX. In this case, the trading partner's operating system can use the CDH to see how you sent the file. Remember that text files are translated from ASCII to EBCDIC when you send them to Information Exchange. When your trading partner receives the files, they are translated back to ASCII.

If you send files as binary, they are not translated when you send them to Information Exchange. In this case, they are not translated when your trading partner receives them. If you do not send files as binary, then translation takes place on both ends and your trading partner still receives the files exactly as you sent them.

In this example, it does not matter if you send the files as text or binary. Both of the following set of commands will work.

Sample 1

```
SEND FILEID(text.fil) ACCOUNT(ABCD) USERID(USER01)
CLASS(READABLE); SEND FILEID(binary.fil) ACCOUNT(ABCD)
USERID(USER01) CLASS(BINARY) DELIMITED(N);
```

Sample 2

```
SEND FILEID(text.fil) ACCOUNT(ABCD) USERID(USER01)
CLASS(READABLE) DATATYPE(B); SEND FILEID(binary.fil)
ACCOUNT(ABCD) USERID(USER01) CLASS(BINARY) DATATYPE(B);
```

NOTE: Specify DELIMITED(N) for binary files to prevent insertion of CR characters in the data.

Example 3

This example shows how you send text and binary files to trading partners who use an older version of expEDIte/PC that does not support the CDH. Since your trading partner's operating system cannot read the CDH, it assumes that all files must be translated from EBCDIC to ASCII when they are received. For this reason, you should not use the DATATYPE(B) option when you send binary files to this trading partner. You should send all text and binary files to Information Exchange as ASCII so that they will be translated to EBCDIC. When your trading partner receives the files, they are translated back to ASCII. Your trading partner receives the files just as you sent them.

In this example, you should send both files as text.

```
SEND FILEID(text.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(READABLE);
SEND FILEID(binary.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(BINARY)
DELIMITED(N);
```

Example 4

This example shows how you send files if your trading partner uses an Expedite product that supports the CDH on a host processor. In this case, your options are similar to those in Example 2. The difference is that host processors store data in EBCDIC format instead of ASCII format.

When you send text files to Information Exchange, they are translated to EBCDIC format. When your trading partner receives the files from Information Exchange, no translation is necessary since the data is already in the EBCDIC format. However, a perfect match between ASCII and EBCDIC data does not always occur.

If your trading partner is having problems reading the data you sent, you may have to use an alternate translate table. Refer to “Using alternate translate tables” on page 114 for more information.

When sending binary files to this user, you must consider how this user will receive and use the files in order to determine how to send them. Refer to “Sending files to an EBCDIC operating system” on page 282 for information about this. For this example, assume that the receiver is going to receive the file with expEDIte/MVS Host and will then download the file to a PC using eNetwork Personal Communications 4.2.

In this example, you should send the text file without the binary option so that it is translated from ASCII to EBCDIC and is readable on the host operating system. You should send the binary file using the alternate translate table equivalent to the IBM eNetwork Personal Communications 4.2 program. This way, the file on the receiver's PC will be the same as the file on your PC.

```
SEND FILEID(text.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(READABLE);
SEND FILEID(binary.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(BINARY)
TRANSLATE(IBM3270) DELIMITED(N);
```

Example 5

This example shows how to send files to a trading partner who uses a host product that does not support the CDH. In this case, your options are somewhat limited. The host processors store data in EBCDIC format instead of ASCII. When you send an ASCII file to Information Exchange, it is translated to EBCDIC format. When your trading partner receives the data from Information Exchange, no translation is necessary since the data is already in the EBCDIC format. However, a perfect match between ASCII and EBCDIC data does not always occur.

When sending binary files to this user, you must consider how this user will receive and use the files in order to determine how to send them. Refer to “Sending files to an EBCDIC operating system” on page 282 for information about this.

In this example, you should send the text file without the binary option so that it is translated from ASCII to EBCDIC and is readable on the host operating system. You should send the binary file using the binary option so that the file is not translated when it is sent to Information Exchange.

```
SEND FILEID(text.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(READABLE);
SEND FILEID(binary.fil) ACCOUNT(ABCD) USERID(USER01) CLASS(BINARY)
DATATYPE(B);
```

Example 6

This example shows how you receive text and binary files from a trading partner who is using the same release of Expedite Base/AIX. Since you both use Expedite Base/AIX, which supports the CDH, your operating system can use the CDH to see how your trading partner sent the files. When you issue the RECEIVE command, Expedite Base/AIX uses the CDH to determine whether the files need to be translated from EBCDIC to ASCII or just received without translation for binary files.

In this example, use the same RECEIVE command for both files. The text file will be properly translated from EBCDIC and the binary file will be received without translation.

```
RECEIVE FILEID(text.fil) CLASS(READABLE);
RECEIVE FILEID(binary.fil) CLASS(BINARY);
```

Example 7

This example shows how you receive text and binary files from a trading partner who uses an older version of expEDIte/PC that does not support the CDH. These versions of expEDIte/PC translate all files from ASCII to EBCDIC since there is no option to distinguish text files from binary files. In this case, Expedite Base/AIX will not be able to use the CDH to determine whether or not to translate a file. It will always assume EBCDIC and translate to ASCII when it receives the file. This should not be a problem because the data was translated on the sending side.

You can use the same RECEIVE command for both files.

```
RECEIVE FILEID(text.fil) CLASS(READABLE);
RECEIVE FILEID(binary.fil) CLASS(BINARY) DELIMITED(N);
```

NOTE: Use DELIMITED(N) so CR characters will not be removed from the binary data.

Example 8

This example shows how you receive files from a trading partner who uses a host Expedite product that supports the CDH. Since a CDH is sent with the files, Expedite Base/AIX can use the CDH to see how your trading partner sent the files. When you issue the RECEIVE command, Expedite Base/AIX uses the CDH to determine whether the files need to be translated from EBCDIC to ASCII or just received without translation for binary files.

In this example, use the same receive command for both files. The text file will be properly translated from EBCDIC and the binary file will be received without translation. Expedite will not try to remove CR characters from the binary file.

```
RECEIVE FILEID(text.fil) CLASS(READABLE);
RECEIVE FILEID(binary.fil) CLASS(BINARY);
```

If you are having problems using the data received from your trading partner, you may have to use an alternate translate table.

Example 9

This example shows how to receive files from a trading partner who uses a host product that does not support the CDH. Since no CDH is sent with the files, Expedite Base/AIX cannot determine if the original file was text or binary. In this case, Expedite Base/AIX always assumes EBCDIC and translates to ASCII when you receive the files. When receiving binary files from this user, you must consider where the files originated when determining how to receive them. Refer to “Sending files to an EBCDIC operating system” on page 282 for information about this. For this example, assume that the binary file in Information Exchange is in the binary format that you require on the workstation. In other words, you do not want any translation done while receiving the file. However, since there is no CDH with the file, Expedite Base/AIX assumes the file is in EBCDIC format and will attempt to translate the file to ASCII. You can get around this by specifying the NOXLATE.XLT translation table. The end result of this is that no translation is done when the file is received.

Following are the commands you can use to receive these files.

```
RECEIVE FILEID(text.fil) CLASS(READABLE);
RECEIVE FILEID(binary.fil) CLASS(BINARY) TRANSLATE(NOXLATE)
DELIMITED(N);
```

If you are having problems using the data received from your trading partner, you may have to use an alternate translate table.

Chapter 12

Displaying screen status

You can use Expedite Base/AIX to display a screen that includes a pictorial view of your workstation and the network (referred to as the *status picture*) as well as a session status box that displays information about your session. You can indicate whether you want both the picture and the status box displayed, only one of them displayed, or neither. In addition, you can modify the text of status messages and control the location and color of the status messages on the screen.

The picture and the session status box are treated independently. During a session, the picture is drawn on the screen first (if you chose to display a picture). Then the session status box is drawn under the picture.

This chapter explains how you display the status picture and status messages. It also discusses the display status script and describes the status events (commands) you can use in the file.

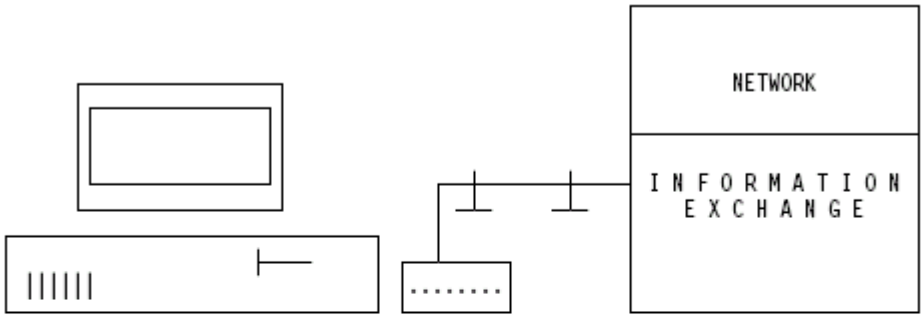
Overview of the session status screen

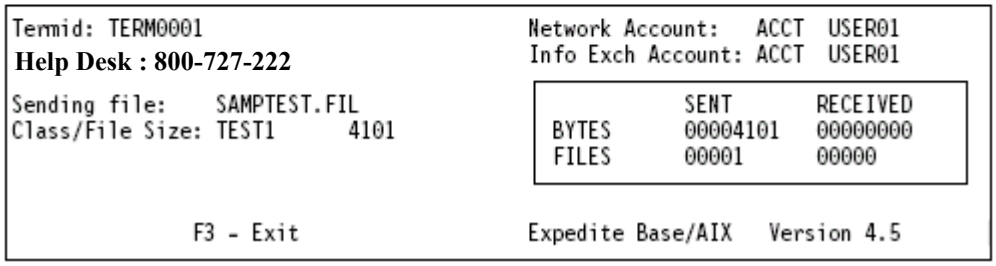
The Expedite Base/AIX session status screen is made up of the following two components:

- A picture containing a workstation, modem, the network, and Information Exchange as well as telephone lines and arrows to indicate transmission status
- A status box with session information including account ID, user ID, connection status, and file transfer status

The following shows the Expedite Base/AIX default session status screen, which includes the picture and the session status box. When the modem is responding, the modem lights appear on the screen. The modem lights disappear if the carrier is dropped. When you receive a successful connection, telephone lines appear from the modem to the network.

When the connection is established between the modem and the network, the lines appear between the telephone poles. If the connection is dropped, the lines disappear.





Following are the options you have for displaying the picture and session status box.

- You can display the default picture and session status information. To do this, specify PICTURE(Y) and STATUS(Y) on the SESSION command in basein.pro. This is the default.
- You can display the picture without the session status information. To do this, specify PICTURE(Y) and STATUS(N) on the SESSION command in basein.pro.
- You can display the session status box without the picture. To do this, specify PICTURE(N) and STATUS(Y) on the SESSION command in basein.pro.
- You can turn off the display of both the picture and the session status box by specifying PICTURE(N) and STATUS(N) on the SESSION command in basein.pro.

NOTE: Expedite Base/AIX does not provide the capability to create a customized picture.

Displaying session status messages

Expedite Base/AIX session status messages provide information such as:

- Connection and disconnection status
- The phone numbers Expedite Base/AIX uses to connect to the network
- The Help Desk phone number you can call for assistance if there are problems
- The terminal ID
- The account and user IDs. The data rate (modem speed) for dial communications
- Whether a network connection and IBM Global Services logon were established
- The transmit time and date for a delayed session
- File and message transfer status
- Error messages

You can either use the Expedite Base/AIX default status messages or modify the messages to meet your needs. The default status messages are stored in display.scr. To change the status messages, modify the display status script. For more information, see “Understanding command syntax” on page 26.

Using the display status script

The commands in the display status script (display.scr) are called status events. The syntax of the events in the display status script is similar to the syntax of basein.pro; the only exception is that commands are referred to as events in the display status script. Refer to “Using the Display Text action” on page 294 for a detailed description of this syntax. You can type event and parameter names in either uppercase or lowercase.

NOTE: You cannot specify blanks for any values in display.scr except on the TEXT parameter.

You may refer to the sample display.scr included with Expedite Base/AIX to see examples of any topics discussed in this chapter.

The following table provides a list of status events you can use in the display status script file. The table also shows:

- The point at which the events occur
- How often Expedite Base/AIX updates certain events
- Recommendations for usage

Status event:	Actions display when:
ARCHIVEMOVE	Expedite Base/AIX begins to process an ARCHIVEMOVE command.
AUDIT	Expedite Base/AIX begins to process an AUDIT command.
CANCEL	Expedite Base/AIX begins to process a CANCEL command.
CANWAITRCV	Action displayed when time expires for a wait on a RECEIVE or RECEIVEEDI command.
CHARSRCVD	Data is received from the network. It is updated approximately every 3500 characters received while processing a RECEIVE or RECEIVEEDI command.
CHARSSNT	Data is sent to the network. It is updated approximately every 3500 characters for each PUTMEMBER, SEND, and SENDEDI command.
CONNECTED	The connect script return code is 0 for dial communications, when the LU 6.2 connection is successful for SNA communications.
CONNECTING	Expedite Base/AIX is attempting a SNA connection on the RS/6000.
DEFINEALIAS	Expedite Base/AIX begins to process a DEFINEALIAS command.
DIALCYCLE	You specify CYCLE and WAIT on the DIAL command and Expedite Base/AIX cannot connect on the first cycle.
DIALING	Expedite Base/AIX begins to process the connect (dial) script for a dial session (not for manual dial or SNA communications on the RS/6000).
DISCONNECT	Expedite Base/AIX begins to process the disconnect script in dial communications, or begins to disconnect from the SNA communications.
END	Expedite Base/AIX sends the session end command to Information Exchange.
EXIT	You press the Exit key.
FILESRCVD	A file is received.
FILESSNT	A file is sent.
FIRST	Expedite Base/AIX finishes reading the display script after the picture is displayed.
GETMEMBER	Expedite Base/AIX begins to process a GETMEMBER command.
INLOGON	The logon to the network was successful. If you specified NINPASSWORD in your profile to change your GXS password, the password was changed.
LAST	The program ends.

Status event:	Actions display when:
LIST	Expedite Base/AIX begins to process a LIST command.
LISTLIBRARIES	Expedite Base/AIX begins to process a LISTLIBRARIES command.
LISTMEMBERS	Expedite Base/AIX begins to process a LISTMEMBERS command.
LOSTCONNECT	Expedite Base/AIX loses the connection with the network.
PICTURE	Expedite Base/AIX draws the picture on the screen. (Use this event to display text on the picture area of the display.)
PURGE	Expedite Base/AIX begins to process a PURGE command.
PUTMEMBER	Expedite Base/AIX begins to process a PUTMEMBER command.
QUERY	Expedite Base/AIX begins to process a QUERY command.
RECEIVE	Expedite Base/AIX begins to process a RECEIVE command.
RECIEVEEDI	Expedite Base/AIX begins to process a RECEIVEEDI command.
RESTART	The program begins and is restarting a previous session.
SEND	Expedite Base/AIX begins to process a SEND command.
SENDEDI	Expedite Base/AIX begins to process a SENDEDI command.
START	Expedite Base/AIX sends the SESSION START command to Information Exchange.
WAITRCV	Action displayed when a RECEIVE or RECEIVEEDI command that includes a wait value is received.
WECOMEMSG	Expedite Base/AIX receives the welcome message from IBM Global Services.

For each status event, you can specify a combination of parameters that cause something to be displayed on the screen when a status event occurs. These combinations of parameters on an event are called an action. There are four types of actions:

- Display text
- Draw a box
- Clear a line
- Clear the screen

If you do not specify an event in the display status script, then nothing is displayed when that event occurs. You can specify any action for any event, and you can specify as many actions for an event as you need by specifying the event multiple times in the display status script.

The order in which you specify the events does not matter; Expedite Base/AIX stores them at the beginning of the program and processes them when the event occurs.

The actions are processed in the order you specified them on the event. For example, to display a line of text and a box as the FIRST event, you could use the following lines in display.scr:

```
FIRST TEXT(F3 - Exit) ROW(24) COLUMN(2);
FIRST TOPLEFTCOL(1) TOPLEFTROW(1) TOPRIGHTCOL(8.) TOPRIGHTROW(7);
```

In this example, the exit key text will be displayed, then a box will be displayed around the top seven lines of the screen.

Events and actions are discussed in detail in the next sections.

Using the Display Text action

The following is an example of the syntax you use to display text on the screen:

```
EVENT    TEXT(text) ROW(row) COLUMN(col)
         FOREGROUND(color) BACKGROUND(color);
```

where:

text

This is the text that will display on the screen.

The maximum screen width supported by Expedite Base/AIX is 80 columns. The TEXT value can be up to 255 characters long (including variables), but Expedite Base/AIX truncates the text after substituting for variables if the text is too wide to display starting from the column you specified.

The value of the TEXT parameter is displayed exactly as you typed it. Trailing blanks will not be truncated from your text, so you can specify text that will overlay text displayed earlier if so desired. See the example below.

This is a required parameter.

row

This is the row on which the text should be displayed. Use a numeric value between 1 and 24. This is a required parameter.

column

This is the column at which the text should be displayed. Use a numeric value between 1 and 80. This is a required parameter.

foreground

This is the foreground color to be used. See “Using colors” on page 300 for a list of colors to use. The default is **white**.

background

This is the background color to be used. See “Using colors” on page 300 for a list of colors to use. The default is **black**.

Following is an example of several events with Display Text actions.

```
# This text will be displayed when Expedite begins to process
# the connect script.
DIALING TEXT(Dialing the Network          ) ROW(20) COLUMN(2);

# This text will be displayed when the network logon is successful.
INLOGON TEXT(Successful Network logon     ) ROW(20) COLUMN(2);

# This text will be displayed when Expedite processes a QUERY
command.
QUERY     TEXT(Checking the mailbox       ) ROW(20) COLUMN(2)
          BACKGROUND(BLUE) FOREGROUND(black) ;
```

Notice in this example the text parameter values are all the same size so that when a new one is written, it overwrites the previous one written to the same row and column on the screen.

Using variables in your text

You can use variables in TEXT parameters so that Expedite Base/AIX substitutes and displays the appropriate values. For example, you can display the number of bytes sent while Expedite Base/AIX is sending a file. Refer to the following table for this information.

Specify variable names with a% sign at the beginning and end, without spaces. For example, *%TIMENOW%* is correct, *% TIMENOW %* is not. Variables can be in uppercase or lowercase: *%LISTNAME%* is the same as *%listname%*.

Remember that if you specify text that is too long for a screen based on the column you specify, Expedite Base/AIX truncates the text to fit on the screen. Consider the fixed length of the value to be substituted for the variables you specify as part of the length of your text. Do not consider the length of the variable name itself. For example, if you specify

```
DIALING TEXT(Phone number: %PHONE%) ROW(17) COLUMN(42);
```

then since the length of the value for *%PHONE%* is 20, you have specified 34 characters to be displayed, 14 for *Phone number*: and 20 for the value of *%PHONE%*.

If you type a variable incorrectly, Expedite Base/AIX treats the variable as text and displays it on the screen. If you specify a variable that Expedite Base/AIX has not set yet, blanks appear in place of the variable on the screen. Expedite Base/AIX substitutes each variable with a fixed width on the screen. The following table lists the Expedite Base/AIX variables, their fixed widths, descriptions, and related events.

Variable:	Size of text substituted:	Description:	Events that support this variable:
ALIASTABLE	4	Name of the alias table from the DEFINEALIAS command.	DEFINEALIAS
ARCHIVEID	8	Archive ID from ARCHIVEMOVE command.	ARCHIVEMOVE
CANCELDEST	21	Mailbox from which you will cancel mail.	CANCEL
CHARSRCVDCNT	8	Characters received count.	CHARSRCVD
CHARSSNTCNT	8	Characters sent count.	CHARSSNT
CLASS	8	User class.	SEND, SENDEDI, RECEIVE, RECEIVEEDI, PUTMEMBER
CNNCTYPE	5	Data rate on connection or from profile, or SNA for SNA communications.	CONNECTED
DATE	8	The date for the delayed session.	DELAYSESS
DIALTIME	8	The time for the next dial attempt to occur when CYCLE and WAIT are specified in the profile.	DIALCYCLE

Variable:	Size of text substituted:	Description:	Events that support this variable:
EXITKEY	1	The exit key specified (or defaulted) in the profile. The default value is **3**.	Any event
FILENAME	14	File ID from commands.	SEND, SENDEDI, RECEIVE, RECEIVEEDI, PUTMEMBER
FILESIZE	6	Length of file to send.	SEND, SENDEDI, PUTMEMBER
FILESRCVDCNT	5	Files received count. Expedite Base/AIX updates for RECEIVE and RECEIVEEDI each time a file is received.	FILESRCVD
FILESSNTCNT	5	Files sent count. Expedite Base/AIX updates for PUTMEMBER, SEND, and SENDEDI each time a file is sent.	FILESSNT
HOTLINE	15	Help Desk phone number from welcome message.	WELCOMEMSG
IEACCOUNT	8	Information Exchange account.	START
IEUSERID	8	Information Exchange user ID.	START
INACCOUNT	8	IBM Global Services account.	Any event
INUSERID	8	IBM Global Services user ID.	Any event
LISTNAME	8	List name from LIST command.	LIST
MESSAGE	80	Message with final return code.	LAST
MEMBER	8	Member name for PUTMEMBER, GETMEMBER.	PUTMEMBER, GETMEMBER
PHONE	20	Phone number dialed.	DIALING
RETURNCODE	5	Final return code from Expedite Base/AIX.	LAST
SCRIPTNAME	14	Script name being processed (for dial communications	DIALING, DISCONNECTING
TERMID	8	Terminal ID from welcome message.	WELCOMEMSG
TIMENOW	8	HH:MM:SS (time now, for delayed session).	DIALCYCLE
VERSION	5	V.R.M - where V is version, R is release, M is modification.	Any event

Using the Clear Screen action

Expedite Base/AIX clears the screen at the beginning of the program (before the picture is drawn) and at the end of the program. Do not specify a CLEARSCREEN action in the display status script, unless it is on the last event, if you specified PICTURE(Y) on the SESSION command.

If you specify PICTURE(N) on the SESSION command, Expedite Base/AIX will not clear the screen at the end of the program. You can use a CLEARSCREEN action on any event in the display status script to clear the screen, although it is logical to use this action only on events FIRST and LAST.

Following is the syntax to use to specify an action to clear the screen:

```
EVENT CLEARSCREEN(Y) WAIT(seconds) ;
```

where:

clearscreen

Tells Expedite Base/AIX this is an action to clear the screen. The only valid value is **y**.

This is a required parameter.

wait

The number of seconds to wait before clearing the screen. Use 1 to 2 numeric characters. The default is **0**.

By default, Expedite Base/AIX display script uses a 3-second wait time on the LAST event to clear the screen. This allows the user a few seconds to view the final return code and message. You may want to use the WAIT parameter for the same reason if you display your own picture or status.

Using the Draw Box action

To create a box, you specify the top-left corner coordinates and the bottom-right corner coordinates on the screen where you want the box to be displayed.

The following is an example of the syntax you use for a status event to draw a box on the screen:

```
EVENT     TOPLEFTROW(top left row) TOPLEFTCOL(top left column)
          BOTRIGHTROW(bottom right row)
          BOTRIGHTCOL(bottom right column)
          FOREGROUND(color) BACKGROUND(color) ;
```

where:

topleftrow

The row of the top left corner of the box. Values from **1** to **24** are valid. Use 1 to 2 numeric characters. This is a required parameter.

topleftcol

The column of the top left corner of the box. Values from **1** to **80** are valid. Use 1 to 2 numeric characters. This is a required parameter.

botrightrow

The row of the bottom right corner of the box. Values from **1** to **24** are valid. Use 1 to 2 numeric characters. This is a required parameter.

botrightcol

The column of the bottom right corner of the box. Values from **1** to **80** are valid. Use 1 to 2 numeric characters. This is a required parameter.

foreground

This is the foreground color to be used. Refer to “Using colors” on page 300 for valid colors. The default is **white**.

background

This is the background color to be used. Refer to “Using colors” on page 300 for valid colors. The default is **black**.

Expedite Base/AIX will not stop you from writing text to the screen that will overwrite a box on the screen. You may need to experiment with the boxes and text, or calculate the length of the box and the enclosed text.

Using the Clear Line action

The following is an example of the syntax you use for an event that will clear a specified length on the screen:

```
EVENT COLOR(color)
      ROW(row) COLUMN(column)
      CLEARLENGTH(length to clear);
```

where:

color

This is the color you want to use. Refer to “Using colors,” below for valid colors. The default is **black**.

row

This is the row of the line that should be cleared. Use a numeric value between **1** and **24**. This is a required parameter.

column

This is the column of the line that should be cleared. Use a numeric value between **1** and **80**. This is a required parameter.

clearlength

This is the length to clear at the specified row and column. Use a numeric value between **1** and **80**. This is a required parameter.

Using colors

The following is a list of colors you can use for FOREGROUND:

Black	Grey
Blue	Light_blue
Green	Light_green
Cyan	Light_cyan
Red	Light_red
Magenta	Light_magenta
Brown	Yellow
White	Bright_white

The following is a list of colors you can use for COLOR or BACKGROUND:

Black	Red
Blue	Magenta
Green	Brown
Cyan	White

Expedite Base/AIX display script

The sample display script, display.scr, is provided with Expedite Base/AIX. You may wish to try running Expedite Base/AIX with this script to see how it works before making any modifications.

Chapter 13

Using the connectivity log and trace files

At times, you may need detailed information about Expedite Base/AIX processing. The connectivity log and trace files can provide this information. This chapter describes the connectivity log and trace files and provides examples.

Using the connectivity log

Expedite Base/AIX writes connectivity log data to the *cnnct.log* file. The log is written automatically each time you run the *iebase* program. You can review the connection activity by looking at the contents of this file.

The connectivity log shows what happens while Expedite Base/AIX is attempting to connect to the network. This log shows the data sent by Expedite Base/AIX to the modem and the data received by Expedite Base/AIX from the modem. Using this information, you can determine if a problem occurred while connecting to the network, and if it did, how you can correct it.

The connectivity log provides:

- Date and time the session started
- Communications port number and data rate used
- Commands issued to the modem
- Modem's response to the commands
- Phone number dialed
- Connection message received
- Status messages showing logon progress
- Final session return code
- Information about the use of old or new style connect scripts

For a description of the connectivity log, "Understanding the connectivity log" on 305.

Using asynchronous communication with a network communication gateway

The following shows a sample connectivity log using asynchronous communication and the network communication gateway, which is COMMTYPE(A) on the TRANSMIT command. Enclosed in parentheses to the far right of some of the modem lines are descriptions of those lines. They are not part of the connectivity log.

Expedite Base/AIX for the RISC System/6000 Version 4.5 started Wed Jul 01 14:40:03 1998

(14:40:03) Closing device	>Device: /dev/tty1	
(14:40:03) Opening device	>Device: /dev/tty1, Baud: 2400, Parity: 7	
(14:40:03) Closing port	>Port: 1	
(14:40:05) Opening port	>Port: 1, Baud: 19200, Parity: 8	
(14:40:06) Closing port	>Port: 1	
(14:40:07) Opening port	>Port: 1, Baud: 19200, Parity: 8	
(14:40:07) Send to modem	>AT&F	(Modem setup command)
(14:40:08) Receive from modem	>AT&F	(Modem response)
OK		
(14:40:08) Send to modem	>ATL1X1V1QO&C1&D2	(Modem setup command)
(14:40:08) Receive from modem	>ATL1X1V1QO&C1&D2	(Modem response)
OK		
(14:40:08) Send to modem	>ATDT9, 554-1101	(Dial command)
(14:40:08) Receive from modem	>ATDT9, 554-1101	
(14:40:26) Receive from modem	>	
CONNECT 19200/ARQ		(Connect response)
(14:40:26) Send to modem	>	(Carriage returns)
(14:40:27) Send to modem	>	
(14:40:27) Receive from modem	>	
(14:40:30) Receive from modem	>	
(14:40:30) Receive from modem	>	
(14:14:30) Receive from mode	->	
WELCOME TO THE IBM INFORMATION SERVICES		(Welcome message)
(14:40:30) Receive from modem	->	
SYSTEM: IBM.SM.1 TERMID: IBMPQADU 98/07/01 14:38:34		
(14:40:30) Receive from modem	->	
HELP DESK: 800-727-2222.		
(14:40:30) Receive from modem	->	
(14:40:30) Receive from modem	->	
ENTER “HELP” FOR LOGON ASSISTANCE.		
(14:40:30) Receive from modem	>	
(14:40:30) Receive from modem	>	
ENTER USERID ACCOUNT.		
(14:40:32) Receive from modem	>	

```
===>*
(14:40:32) * Status *          >Received Welcome Message successfully   (Status message)
(14:40:32) Send to modem       >*                                       (Batch logon)
EXP6

(14:40:32) Receive from modem  >TERMINAL TYPE = EXP6                    (Modeset command)
(14:40:32) Send to modem       >*                                       (End text)
(14:40:32) Receive from modem  >*
(14:40:32) * Status *          >Completed mode set successfully         (Status messages)
(14:40:36) * Status *          >Logged on to the network successfully
(14:40:43) Final return code   >00000                                   (Final return code)
(14:40:44) Send to modem       >+++                                     (Hang up)
(14:40:46) Send to modem       >ATH

(14:40:47) Receive from modem  >*
NO CARRIER
ATH

OK

(14:40:47) Closing port        >Port: 1
```

Using worldwide asynchronous communication

The following shows a sample connectivity log using worldwide asynchronous communication, which is COMMTYPE(W) on the TRANSMIT command. Enclosed in parentheses to the far right of some of the modem lines are descriptions of those lines. They are not part of the connectivity log.

Expedite Base/AIX for the RISC System/6000 Version 4.5 started Wed Jul 01 14:41:52 1998

(14:41:53) Closing device	>Device: /dev/tty1	
(14:41:53) Opening device	>Device: /dev/tty1, Baud: 2400, Parity: 7	
(14:41:53) Closing port	>Port: 1	
(14:41:54) Opening port	>Port: 1, Baud: 2400, Parity: 8	
(14:41:55) Closing port	>Port: 1	
(14:41:57) Opening port	>Port: 1, Baud: 2400, Parity: 8	
(14:41:57) Send to modem	>AT&F	(Modem setup command)
(14:41:57) Receive from modem	>AT&F	(Modem response)
OK		
(14:41:57) Send to modem	>ATL1X1V1Q0&C1&D2	
(14:41:57) Receive from modem	>ATL1X1V1Q0&C1&D2	
(14:41:57) Receive from modem	>	
OK		
(14:41:57) Send to modem	>ATDT9, 554-1101	(Dial command)
(14:41:57) Receive from modem	>ATDT9, 554-1101	
(14:42:12) Receive from modem	>CONNECT 2400/ARQ	(Connect response)
(14:42:12) Send to modem	>	
(14:42:13) Send to modem	>	
(14:42:13) Receive from modem	>	
(14:42:16) Receive from modem	>	
(14:42:16) Receive from modem	>	
(14:42:16) Receive from modem	>WELCOME TO THE IBM INFORMATION SERVICES.	(Welcome message)
(14:42:17) Receive from modem	>SYSTEM: IBMOSMO2 TERMID: IBMCQADA 98/07/01	
14:40:30		
(14:42:17) Receive from modem	>HELP DESK: 800-727-22220	
(14:42:17) Receive from modem	>	
(14:42:17) Receive from modem	>ENTER “HELP” FOR LOGON ASSISTANCE.	
(14:42:17) Receive from modem	>	
(14:42:17) Receive from modem	>ENTER USERID ACCOUNT.	
(14:42:19) Receive from modem	>===> *	
(14:42:19) * Status *	>Received Welcome Message successfully	(Status message)
(14:42:19) Send to modem	>/*LOGON ACCT,USERID,XXXXXX/*SELECT	(Batch logon)
INFOEXCH/*USERDATA BISYNCPC102410A E		

```
(14:42:23) Receive from modem  >SVM000800 LOGON!LOGOFF SUCCESSFUL EURASYN  (Logon response)
(14:42:23) * Status *          >Logged on to the network successfully
(14:42:34) Send to modem       >/*LOGOFF09A0B*
(14:42:35) Receive from modem  >*
(14:42:35) Receive from modem  >SVM00800 LOGON!LOGOFF SUCCESSFUL
(14:42:35) Final return code   >00000                                  (Final return code)
(14:42:36) Send to modem       >+++                                    (Hang up)
(14:42:38) Send to modem       >ATH
(14:42:40) Receive from modem  >
WELCOME TO THE IBM INFORMATION SERVICES.
SYSTEM: IBMOSM
NO CARRIER
ATH
OK

(14:42:40) Closing port                                        ->  Port: 1
*
```

Understanding the connectivity log

This section describes the major elements that make up a connectivity log.

Start of Expedite Base/AIX dial connect process

The day, date, and time of the session are on the first line of the connectivity log.

modem commands

The lines marked Send to modem are the commands that Expedite Base/AIX sends to the modem to be processed. These commands come from the modem script files. Most modem commands start with *AT*. Expedite Base/AIX uses the +++ to get the modem's attention.

modem responses

The modem responds to the commands Expedite Base/AIX issues. Some modems respond by echoing back the command, then delivering a response; other modems only deliver a response. The normal response is OK. If the response is ERROR, or some response other than OK, the modem is not properly handling the command. Some commands result in an error because the modem does not support them. However, a response of ERROR resulting from an unsupported command is not always cause for concern, because the command causing the error response might not be necessary for the type of modem you have.

dial command

After Expedite Base/AIX issues the modem setup (AT) commands, it sends the DIAL command. This begins with *ATD* and is followed by *T* or *P* (tone or pulse) and the telephone number. After the telephone number is dialed, Expedite Base/AIX waits for an answer. While the PC is waiting for the connection to be made, it issues a RECEIVE command every two seconds, for a maximum of 30 times. This means that the PC waits a total of 60 seconds for the connection.

connection response

When the connection is established, Expedite Base/AIX receives the CONNECT*xxxx* response, where *xxxx* is the data rate of the connection. You can see how long it took to establish the connection by comparing the time displayed on the line that shows when the DIAL command was first sent and the line that displays the connect statement.

carriage returns

After the connection is made, Expedite Base/AIX sends two carriage returns to the modem. On two lines the log shows Send to modem > with no information following it, because the log does not display carriage-return characters.

NOTE: If you are using worldwide asynchronous communication, you will not see the two carriage-return lines.

welcome message

Expedite Base/AIX receives the message “Welcome to the IBM Information Services or Welcome to the IBM Global Network.” The message includes the system ID, terminal ID, date, and time. The message also provides a telephone number for the Help Desk and instructions for getting logon assistance.

welcome message status

After the welcome message, the connectivity log shows the status Received Welcome Message successfully.

start-of-text character and MODESET command

In response to the welcome message status, Expedite Base/AIX sends a start-of-text character, ⓪, (x'02'), and the MODESET command (EXP1). The MODESET command describes the type of connection. EXP1 tells the network that it is an Expedite Base/AIX connection.

NOTE: If you are using worldwide asynchronous communication, you will not see the start-of-text character or the MODESET command. Instead, you will see the batch logon command.

MODESET command response

Expedite Base/AIX receives a response to the MODESET command (Terminal Type = EXP1). After this, Expedite Base/AIX establishes a successful session, and data is sent to and from Information Exchange. The connectivity log does not reflect this data.

NOTE: If you are using worldwide asynchronous communication, you will not see the MODESET command response. Instead, you will see a response to the batch logon.

end-of-text character

If you are using asynchronous communication, the connectivity log shows the end-of-text character that Expedite Base/AIX sends to the modem. This character is (x'03').

logon status

The connectivity log shows the status message *`Logged on to the network successfully`*. If you are using asynchronous communication, the log also shows the modeset status: *`Completed mode set successfully`*.

end of Expedite Base/AIX dial connect process

Expedite Base/AIX closes the port, hangs up (ATH), and issues the final return code.

Using the connectivity log for problem determination

The following table shows some modem symptoms, possible causes, and the actions to take for each symptom.

Symptom:	Questions/Actions:
Modem does not respond to any commands. The log shows Receive from modem with nothing following it.	Is the modem turned on? Is the device correct? Is the data rate correct?
Modem responds with *ERROR*.	Is this command supported by your modem? Check your modem manual. Is the syntax of the command correct?

Symptom:	Questions/Actions:
The DIAL command is issued, but no connect message is received from the modem, or a connect message is received, but the time elapsed since the DIAL command was issued is greater than 60 seconds.	Listen for the dial sounds. If the modem dials, but there is no dial tone: • Is the phone cable connected to the wall jack? • Is the phone cable connected to the modem? • Can you dial on this telephone line with a standard telephone? If there is a dial tone: • Did you dial the correct telephone number? • Did you hear an answer (modem signal)? • Did you specify an escape character if one is necessary? If there was a dial tone, but no answer, try an alternative telephone number. If the telephone answers, but the log shows that more than 60 seconds have elapsed before the connect message, change the wait time to greater than 60 seconds.
When the connection is made, the modem responds with something other than CONNECT.	Refer to the modem manual or manufacturer to determine how to change the modem response.
Modem responds with *BUFFER*, instead of *CONNECT*.	Refer to the modem manual or manufacturer to determine how to disable buffering. Add the DISABLE command to your modem control file or the modem initialization string. For more information, see Chapter 13, "Using modem scripts."
After the successful connection, the connectivity log does not show the response Welcome to the IBM Information Services or Welcome to the IBM Global Network.	Verify that you are using a valid telephone number to access Information Exchange.

Symptom:	Questions/Actions:
After the successful connection, the connectivity log shows unreadable data received from the modem.	Verify that you are using a valid telephone number to access Information Exchange. Check that you specified the correct modem type. If you are using asynchronous communication, unreadable data in the log could be caused by errorcorrecting protocol or data compression. If your modem supports an errorcorrecting protocol (such as MNP), disable the protocol. If it supports data compression, disable it. Refer to your modem manual or manufacturer for details. Then issue the command to disable MNP or data compression by modifying the modem control file or specifying the command in the modem initialization string. For more information, see “Using modem scripts” on page 321.

Using the trace files

Expedite Base/AIX writes trace data to the iebase.trc file automatically each time you run the *iebase* program. Expedite Base/AIX also writes trace data to the *s1dcl.trc* file if you specify LINK(Y) on the TRACE command. You can review the session activity by looking at the contents of these files.

The trace files provide detailed information about Expedite Base/AIX processing. The trace files can help you with Expedite Base/AIX problem determination.

NOTE: You should not write code to do automated processing based on the information in the trace files. The following sections describe the trace file parameters and provide examples.

The style of the trace includes a trace tag on each line, which generally looks like =TAGTEXT=> and ends with an arrow (<=). The tag text depends on which trace(s) you have selected. The tags are as follows:

Tag Text:	Description:	Trace:
=Message=>	General information	any
=Module=>	Base logic trace	BASE
=Input=>	Input file parser	IOFILE
=Output=>	Output file parser	IOFILE
=Data Sent=>	Data sent to Information Exchange	PROTOCOL
=Data Received=>	Data received from Information Exchange	PROTOCOL
=Display=>	Display status processing	DISPLAY
=Modem Send=>	Sent to modem	MODEM
=Modem Receive=>	Received from modem	MODEM

Tag Text:	Description:	Trace:
=CNNCT=>	Modem script parsing	CNNCT

Using trace file parameters

The seven Expedite Base/AIX traces are:

- Display trace—TRACE DISPLAY(Y);

  To see how Expedite Base/AIX processed the information in the display script (display.scr), specify DISPLAY(Y) in the TRACE command in basein.pro to turn the display trace on. This trace enables you to detect problems with the display script. If you are experiencing problems with the display, turn this trace on and run iebase again. Save this file and have it available when you contact the Help Desk.

- Modem Trace—TRACE MODEM(Y);

  The modem trace shows the data to and from the port so you can use it with the CNNCT trace if you want to debug your modifications to the modem script. If you have difficulty connecting to the network, you can use this trace to determine errors in the dial process. This information is also provided in the cnnct.log file. Refer to “Understanding the connectivity log” on page 305 to learn about the connectivity log.

- Modem script connect script trace—TRACE CNNCT(Y);

  This trace lists each of the commands processed in the modem script files as well as the command parameters and their assigned values. This trace can help you locate syntax and logic errors in a new or modified modem script file.

- Information Exchange protocol logic trace—TRACE PROTOCOL(Y);

  To see the flow of Information Exchange commands used by Expedite Base/AIX, use this parameter. The Help Desk uses this trace primarily for problem determination.

- Link level protocol logic trace—TRACE LINK(Y);

  To see the flow of data across the communication link during normal Expedite Base/AIX processing, use this parameter. The Help Desk uses this trace primarily for problem determination.

- Module flow logic trace—TRACE BASE(Y);

  To see the logic flow between the Expedite Base/AIX components, use this parameter. The Help Desk uses this trace primarily for problem determination.

- Command parser trace—TRACE IOFILE(Y);

  To debug problems with the basein.pro or basein.msg, use this parameter. For example, if you do not understand a return code received from the parsing of these files, you can use this trace to see how the file was parsed and to see what causes the return code.

With the exception of the link trace, Expedite Base/AIX places all trace data in iebase.trc. For link trace, it places the data in s1dcl.trc. When using TCP/IP communications, link trace data is placed in the file named link.trc. All traces are cumulative. Expedite Base/AIX does not erase any trace until the session following a successful session or a session in which you specify the RESET command line parameter.

All the trace file options default to **n** (no trace). To select trace options, you change the appropriate parameter to **y**.

NOTE: Activating too many traces at once can produce a confusing trace file. Request only the traces you need for debugging.

You can look at the trace file with a text editor or a file browse utility. However, if the end-of-file characters X'1A' appear in the trace data, your editor may not display an entire trace file.

NOTE: The trace file may contain messages indicating errors opening certain files. During its processing, Expedite Base/AIX works with multiple internal files. Not all of these files are necessary for its processing. If your trace shows an error opening a file, but processing continues normally, you need not be concerned about that message.

Learning from examples

The following sections shows examples of the following trace files:

- Modem trace
- Modem script command logic trace
- Display trace
- Connect script trace
- Display trace with error
- Modem script command logic trace with error
- Command parser trace

Modem trace example

The following is an example of a modem trace without errors.

```
=Message=>Expedite Base/AIX for the RISC System/6000 Version 4.5 started
Tue Sep 29 14:51:39 1998<-
=Modem Send=>^M
=Modem Receive=>^M<-
=Modem Receive=><-
=Modem Receive=><-
=Modem Send=>AT&F1^M<-
=Modem Receive=>AT&F1^M
OK
<-
=Modem Send=>ATDT9,877-1117^M<-
=Modem Receive=>ATDT9,877-1117^M<-
=Modem Receive=><-
=Modem Receive=><-
=Modem Receive=><-
=Modem Receive=><-
=Modem Receive=><-
=Modem Receive=><-
=Modem Receive=>^M^MCONNECT 14400/ABQ/V32/LAPM/U42BIS^M<-
=Modem Send=>^M<-
=Modem Send=>^M<-
=Modem Receive=>
<-
=Modem Receive=>^M^M ^M<-
=Modem Receive=>^M ^M<-
=Modem Receive=>^MWELCOME TO THE IBM INFORMATION SERVICES.^M<-
=Modem Receive=>^MSYSTEM: IBM.SW.3 TERMID: IBMT2TD4 98/.9/29 14:52:15^M<-
=Modem Receive=>^MHELP DESK: 800-727-2222.^M<-
=Modem Receive=>^M ^M<-
=Modem Receive=>^MENTER “HELP” FOR LOGON ASSISTANCE.^M<-
=Modem Receive=>^M ^M<-=Modem Receive=>^MENTER USERID ACCOUNT.^M<-
=Modem Receive=>^M===> ^Q<-
=Modem Send=>^A^[
EXP6^M<-
=Modem Receive=>TERMINAL TYPE = EXP6^M<-
=Modem Send=>^D<-
=Modem Receive=>^D<-
=Modem Send=>+++<-
=Modem Send=>ATH^M<-
=Modem Receive=>^E
OK
ATH^M
OK
<-
```

Modem script command logic trace example

The following is an example of a modem script command logic trace with an error.

=Message=>Expedite Base/AIX for the RISC System/6000 Version 4.5 started

Thu Oct 01 13:30:10 1998<-
=CNNCT=>Command >CLEARBUFFER<-, Value><-
=CNNCT=>Command >CLOSEPORT<-, Value><-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >OPNPORT<-, Value><-
=CNNCT=>Command >OPNPORT<-, Value><-
=Module=>modGetCmds rc(12055)<-
=Module=>ProcModScript rc(12055)<-
=Module=>DoAutoDial rc(12055)<-
=Module=>lognet rc(12055)<-
=Module=>connect rc(12055)<-

In this example, there is a syntax error in the modem script file. The command OPENPORT is misspelled as OPNPORT. The modem command processor ends with a return code 12055.

NOTE: Expedite Base/AIX places module flow information in the trace file once a nonzero code is returned. This information assists the Help Desk with problem determination.

Display trace example

The following is an example of a display trace file. The display trace is divided into two sections:

- The first section shows the information that Expedite Base/AIX reads from the display script. Expedite Base/AIX reads the entire script before displaying anything on the screen.
- The second section shows how Expedite Base/AIX uses the information from the display script during the connection process. Included is the actual data substituted in the variables specified in the script.

The following example shows part of *display.scr*.

```
PICTURE TEXT(IBM Global ) row(3) column(60);
PICTURE TEXT( Network ) row(4) column(60) ;
PICTURE TEXT(Information) row(7) column(60) ;
PICTURE TEXT( Exchange ) row(8) column(60) ;
#
FIRST TEXT(Expedite Base/AIX Version %VERSION%) ROW(23) COLUMN(42) ;
FIRST TEXT(%EXITKEY%  Exit) ROW(23) COLUMN(17);
FIRST TOPLEFTROW(15) TOPLEFTCOL(1) BOTRIGHTROW(24) BOTRIGHTCOL(80)
FOREGROUND(LIGHT_BLUE);
#
DIALING TEXT(Network Account: %INACCOUNT% %INUSERID%) ROW(16)
COLUMN(42);
DIALING TEXT(F1  Redial) ROW(23) COLUMN(2);
DIALING CLEARLENGTH(32) ROW(21) COLUMN(2);
```

```
DIALING TEXT(Dialing the Network ) ROW(20) COLUMN(2); DIALING TEXT(Phone
number: %PHONE% ) ROW(17) COLUMN(42);
#
CONNECTING TEXT(Connecting to the Network ) ROW(20) COLUMN(2);
#
CONNECTED TEXT(Successful %CNNCTYPE% Connection ) ROW(20) COLUMN(2);
```

The following example shows the first part of the display trace which shows what Expedite Base/AIX reads from the *display.scr* file. It shows each of the parameters and the values associated with those parameters.

```
=Message=>Expedite Base/AIX for the RISC System/6000 Version 4.5 started
Thu Oct1 13:46:4. 1998<-

=Display=>Command >TEXT<-, Value>IBM Global <-
=Display=>Command >ROW<-, Value>3<-
=Display=>Command >COLUMN<-, Value>60<-
=Display=>Command >TEXT<-, Value> Network <-
=Display=>Command >ROW<-, Value>4<-
=Display=>Command >COLUMN<-, Value>60<-
=Display=>Command >TEXT<-, Value>Information<-
=Display=>Command >ROW<-, Value>7<-
=Display=>Command >COLUMN<-, Value>60<-
=Display=>Command >TEXT<-, Value> Exchange <-
=Display=>Command >ROW<-, Value>8<-
=Display=>Command >COLUMN<-, Value>60<-
=Display=>Command >TEXT<-, Value>Expedite Base/AIX Version %VERSION%<-
=Display=>Command >ROW<-, Value>23<-
=Display=>Command >COLUMN<-, Value>42<-
=Display=>Command >TEXT<-, Value>%EXITKEY%  Exit<-
=Display=>Command >ROW<-, Value>23<-
=Display=>Command >COLUMN<-, Value>17<-
=Display=>Command >TOPLEFTROW<-, Value>15<-
=Display=>Command >TOPLEFTCOL<-, Value>1<-
=Display=>Command >BOTRIGHTROW<-, Value>24<-
=Display=>Command >BOTRIGHTCOL<-, Value>80<-
=Display=>Command >FOREGROUND<-, Value>LIGHT_BLUE<-
=Display=>Command >TEXT<-, Value>Network Account: %INACCOUNT%
%INUSERID%<-
=Display=>Command >ROW<-, Value>16<-
=Display=>Command >COLUMN<-, Value>42<-
=Display=>Command >TEXT<-, Value>F1  Redial<-
=Display=>Command >ROW<-, Value>23<-
=Display=>Command >COLUMN<-, Value>2<-
=Display=>Command >CLEARLENGTH<-, Value>32<-
=Display=>Command >ROW<-, Value>21<-
=Display=>Command >COLUMN<-, Value>2<-
=Display=>Command >TEXT<-, Value>Dialing the Network <-
=Display=>Command >ROW<-, Value>20<-
=Display=>Command >COLUMN<-, Value>2<-
=Display=>Command >TEXT<-, Value>Phone number: %PHONE% <-
```

```
=Display=>Command >ROW<-, Value>17<-
=Display=>Command >COLUMN<-, Value>42<-
=Display=>Command >TEXT<-, Value>Connecting to the Network <-
=Display=>Command >ROW<-, Value>20<-
=Display=>Command >COLUMN<-, Value>2<-
=Display=>Command >TEXT<-, Value>Successful %CNNCTYPE% Connection <-.
```

The following example shows the second part of the display trace which shows the values substituted during the connection. Note that foreground and background colors are designated by number. Refer to “Using colors” on page 300 to learn more about the colors you can use.

```
=Display=>R(3) C(6.) F(7) B(0) (IBM Global )<-
=Display=>R(4) C(6.) F(7) B(0) ( Network )<-
=Display=>R(7) C(6.) F(7) B(0) (Information)<-
=Display=>R(8) C(6.) F(7) B(0) ( Exchange )<-
=Display=>R(23) C(42) F(7) B(0) (Expedite Base/AIX Version 4.6)<-
=Display=>R(23) C(17) F(7) B(0) (F3  Exit)<-
=Display=>Box: TR(15) TC(1) BR(24) BC(80) F(9) B(0)<-
=Display=>R(16) C(42) F(7) B(0) (Network Account: ATAP IETESTD )<-
=Display=>R(23) C(2) F(7) B(0) (F1  Redial)<-
=Display=>Clear Line R(21) C(2) CL(0) L(32)<-
=Display=>R(20) C(2) F(7) B(0) (Dialing the Network)<-
=Display=>R(17) C(42) F(7) B(0) (Phone number: 9,8771117)<-
=Display=>R(20) C(2) F(7) B(0) (Successful 14400 Connection)<-
```

.
.
.

Connect script trace example

The following is an example of a connect script trace file:

```
=Message=>Expedite Base/AIX for the RISC System/6000 Version 4.5 started
Thu Oct 1 14:.2:16 1998<-

=CNNCT=>Command >CLEARBUFFER<-, Value><-
=CNNCT=>Command >CLOSEPORT<-, Value><-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >OPENPORT<-, Value><-
=CNNCT=>Command >BAUD<-, Value>%BAUD%<-
=CNNCT=>Command >BITS<-, Value>8<-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >CLOSEPORT<-, Value><-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >OPENPORT<-, Value><-
=CNNCT=>Command >BAUD<-, Value>%BAUD%<-
=CNNCT=>Command >BITS<-, Value>8<-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >STRING<-, Value>%RESET<-
=CNNCT=>Command >:CHECK_1<-, Value><-
=CNNCT=>Command >GETANSWER<-, Value><-
=CNNCT=>Command >TIMEOUT<-, Value>2<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>EQUAL<-
=CNNCT=>Command >TO<-, Value>ERROR<-
=CNNCT=>Command >GOTO<-, Value>OK_1<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>NOTEQUAL<-
=CNNCT=>Command >TO<-, Value>OK<-
=CNNCT=>Command >GOTO<-, Value>CHECK_1<-
=CNNCT=>Command >MAXREPEAT<-, Value>2<-
=CNNCT=>Command >:OK_1<-, Value><-
=CNNCT=>Command >CLEARBUFFER<-, Value><-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >STRING<-, Value>%INIT%<-
=CNNCT=>Command >:CHECK_2<-, Value><-
=CNNCT=>Command >GETANSWER<-, Value><-
=CNNCT=>Command >TIMEOUT<-, Value>2<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>EQUAL<-
=CNNCT=>Command >TO<-, Value>ERROR<-
=CNNCT=>Command >GOTO<-, Value>OK_2<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>NOTEQUAL<-
=CNNCT=>Command >TO<-, Value>OK<-
=CNNCT=>Command >GOTO<-, Value>CHECK_2<-
=CNNCT=>Command >MAXREPEAT<-, Value>05<-
=CNNCT=>Command >:OK_2<-, Value><-
```

```
=CNNCT=>Command >CLEARBUFFER<-, Value><-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >STRING<-, Value>ATD%PTYPE%%ESC%%PHONE%<-
=CNNCT=>Command >:CHECK_3<-, Value><-
=CNNCT=>Command >GETANSWER<-, Value><-
=CNNCT=>Command >TIMEOUT<-, Value>2<-
=CNNCT=>Command >MODE<-, Value>LINE<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>EQUAL<-
=CNNCT=>Command >TO<-, Value>BUSY<-
=CNNCT=>Command >GOTO<-, Value>EXIT_BAD<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>EQUAL<-
=CNNCT=>Command >TO<-, Value>NO CARRIER<-
=CNNCT=>Command >GOTO<-, Value>EXIT_BAD<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>NOTEQUAL<-
=CNNCT=>Command >TO<-, Value>CONNECT<-
=CNNCT=>Command >GOTO<-, Value>CHECK_3<-
=CNNCT=>Command >MAXREPEAT<-, Value>30<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>NOTEQUAL<-
=CNNCT=>Command >TO<-, Value>CONNECT<-
=CNNCT=>Command >GOTO<-, Value>EXIT_BAD<-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >CR<-, Value>Y<-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >CR<-, Value>Y<-
=CNNCT=>Command >GETANSWER<-, Value><-
=CNNCT=>Command >TIMEOUT<-, Value>.<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>EQUAL<-
=CNNCT=>Command >TO<-, Value>NO CARRIER<-
=CNNCT=>Command >GOTO<-, Value>EXIT_BAD<-
=CNNCT=>Command >GETVALUE<-, Value><-
=CNNCT=>Command >AFTER<-, Value>CONNECT <-
=CNNCT=>Command >INTO<-, Value>CNNCTBAUD<-
=CNNCT=>Command >GO<-, Value><-
=CNNCT=>Command >TO<-, Value>EXIT_OK<-
=CNNCT=>Command >:EXIT_BAD<-, Value><-
=CNNCT=>Command >RETURN<-, Value><-
=CNNCT=>Command >CODE<-, Value>12130<-
=CNNCT=>Command >:EXIT_OK<-, Value><-
=CNNCT=>Command >RETURN<-, Value><-
=CNNCT=>Command >CODE<-, Value>.<-
=CNNCT=>Return code set to 00000.<-
=CNNCT=>Command >:HANGUP<-, Value><-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
```

```
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >STRING<-, Value>+++<-
=CNNCT=>Command >CR<-, Value>N<-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>2<-
=CNNCT=>Command >SAY<-, Value><-
=CNNCT=>Command >STRING<-, Value>ATH<-
=CNNCT=>Command >WAIT<-, Value><-
=CNNCT=>Command >SECONDS<-, Value>1<-
=CNNCT=>Command >GETANSWER<-, Value><-
=CNNCT=>Command >TIMEOUT<-, Value>2<-
=CNNCT=>Command >IFANSWER<-, Value><-
=CNNCT=>Command >IS<-, Value>NOTEQUAL<-
=CNNCT=>Command >TO<-, Value>OK<-
=CNNCT=>Command >GOTO<-, Value>HANGUP<-
=CNNCT=>Command >MAXREPEAT<-, Value>2<-
=CNNCT=>Command >CLOSEPORT<-, Value><-
=CNNCT=>Command >RETURN<-, Value><-
=CNNCT=>Command >CODE<-, Value>.<-
=CNNCT=>Return code set to 00000.<-
```

Display trace with error example

The following is an example of a display trace showing an error in the display script. In this example, the foreground color BLUE was misspelled as VLUE.

NOTE: This example does not show the entire trace; only the last part of the trace illustrating the error.

```
=Display=>Command >ROW<-, Value>23<-
=Display=>Command >COLUMN<-, Value>42<-
=Display=>Command >TEXT<-, Value>%EXITKEY%  Exit<-
=Display=>Command >ROW<-, Value>23<-
=Display=>Command >COLUMN<-, Value>17<-
=Display=>Command >TOPLEFTROW<-, Value>15<-
=Display=>Command >TOPLEFTCOL<-, Value>1<-
=Display=>Command >BOTRIGHTROW<-, Value>24<-
=Display=>Command >BOTRIGHTCOL<-, Value>80<-
=Display=>Command >FOREGROUND<-, Value>VLUE<-
=Module=>ValEvent rc(12211)<-
=Module=>readmsg rc(12211)<-
```

Modem script command logic trace with error example

The following is an example of a modem script command logic trace showing an error in the connect script. In this example, an invalid modem command receives an ERROR response from the modem. If you are having problems connecting to the IBM Global Network, you may need to correct invalid modem commands.

NOTE: This example does not show the entire trace; only the part of the trace illustrating the error.

```
=Message=>Expedite Base/AIX Version 4.2 started  Wed Mar 3. .2:55:.3 1994<-
=Modem Send=>AT !A<-
=Modem Receive=>
AT !A<-
=Modem Receive=>
ERROR
<-
=Modem Send=>ATL1X1V1Q.&C1&D2<-
=Modem Receive=>ATL1X1V1Q.&C1&D2<-
=Modem Receive=>
OK
<-
=Modem Send=>ATDT9,8766601<-
=Modem Receive=>ATDT9,8766601<-
```

Command parser trace example

The following is an example of a command parser trace showing an error in basein.pro. In this example, the user forgot to put a right parenthesis at the end of the PRODUCT parameter value INFOEXCH.

```
=Message=>Expedite Base/AIX for the RISC System/6000 Version 4.5 started  Thu Oct 1
15:21:59 1998<-
=Input=>Command >IDENTIFY<-, Value><-
=Input=>Command >INACCOUNT<-, Value>acct<-
=Input=>Command >INUSERID<-, Value>user01<-
=Input=>Command >INPASSWORD<-, Value>XXXXXXXX<-
=Input=>Command >IEACCOUNT<-, Value>acct<-
=Input=>Command >IEUSERID<-, Value>ieuser1<-
=Input=>Command >IEPASSWORD<-, Value>XXXXXXXX<-
=Input=>Command >PRODUCT<-, Value>infoexch ;DIAL PHONE1(877-1117)
ESCAPE(9,) DIALCOUNT1(2) BAUDRATE(38400) DEVI
CEA(/dev/tty.) DBAUDRATEA(38400) INITSCR( ) RESETSCR( ) <-
=Module=>parscmd rc(14030)<-
=Module=>proiden rc(14030)<-
=Output=>RETURN(14030) ERRDESC(Parameter value too long.) <-
```

Using the link trace file

The link level protocol logic trace (TRACE LINK(Y)) shows the flow of data across the communication link during Expedite Base/AIX processing. The Help Desk primarily uses this trace for problem determination. Expedite Base/AIX places this trace into the file named s1dcl.trc. When using TCP/IP communications, link trace data is placed in the file named link.trc.

The data in the link trace is in binary format. You must provide this trace to the Help Desk by either mailing a diskette, or by sending the file to Information Exchange using another Expedite system.

Chapter 14

Using modem scripts

The first step in asynchronous communication is to set up a modem. Expedite Base/AIX provides a modem setup program to assist you. This chapter describes the setup program and provides information about modem scripts and commands.

Creating modem scripts

You use modem scripts to control your Expedite Base/AIX modem processing. The command syntax for modem scripts is the same syntax you use for the input and output files (basein.msg and basein.pro) with the addition of support for labels. For information on the command syntax, refer to “Understanding command syntax” on page 26.

NOTE: A modem script cannot contain more than 100 commands.

There are five types of modem scripts:

modem initialization

This script provides commands to the modem to initialize it for connectivity.

connect script

Expedite Base/AIX runs this script each time it attempts to dial. The default name for the connect script is cnnct.scr.

disconnect script

Expedite Base/AIX runs this script each time it completes dialing. The default name for the disconnect script is discnnct.scr.

reset script

This script provides commands to the modem to reset the settings to some other required configuration.

welcome message script

This script can be used to define the search strings for a Service Manager screen that has been customized. The name of the file must be welcmsg.scr. This file is generally not needed in the United States.

The following sections describe how you create modem scripts.

Using labels in modem scripts

You use labels in modem scripts to control the flow of processing. Labels must have a colon (:) as the first character, and must be followed by 1 to 12 characters. The labels you create can consist of uppercase or lowercase characters. When Expedite Base/AIX reads the labels, it converts them to uppercase. The characters you use in each label must be unique, whether they are uppercase or lowercase.

Using variables in modem scripts

You can use variables in modem scripts to enable users with different systems to use the same script. For example, instead of specifying the phone number in a script, specify the %PHONE% variable. Expedite Base/AIX will replace %PHONE% with a phone number that you specified in your profile. By using variables in modem scripts, other users can use the script without modifying it first.

The following table shows a list of variables you can use in modem scripts and the commands you would most likely use the variables with. However, you are not limited to using the variables with the commands listed in the recommended usage column.

Variable:	Description:	Recommended Usage:
%ESC%	Telephone escape sequence	SAY
%PHONE%	Telephone number	SAY
%PTYPE%	Telephone type (**t**one or **p**ulse)	SAY
%BAUD%	Speed to open the port	OPENPORT and IFANSWER
%CNNCTBAUD%	Connection data rate	IFVALUE, GETVALUE
%HOTLINE%	service manager screen	GETVALUE
%TERMID%	Define the string after which the terminal will be found on the server manager screen	GETVALUE
%LOGON%	Define the string to signal the logon prompt on the service manager screen	GETVALUE
%xHH% (see note 1)	Hex characters	Any parameter that accepts variables
%NETINIT%	Secondary network setup parameters	SAY
%NETPW% (see note 2)	Secondary network password	SAY
%NETADDR%	Secondary network address	SAY
%INIT%	The modem initialization string	SAY

Variable:	Description:	Recommended Usage:
%RESET%	The modem reset string	SAY
%USER1%	The user-defined USER1 value	Any parameter that accepts variables
%USER2%	The user-defined USER2 value	Any parameter that accepts variables
%USER3%	The user-defined USER3 value	Any parameter that accepts variables

NOTES:

1. You can specify multiple hex characters in a string by using multiple occurrences of xHH inside the % variable delimiters:

   %xHHxHHxHHxHH%
   %x.Dx.Ax.4%

2. If you specify ENCRYPT(Y) on the IDENTIFY command in the profile, you must encrypt the NETPW value. Use 167 as the encryption key value.

The percent signs around a variable name tell Expedite Base/AIX to substitute the VALUE of the variable. For example, in the command OPENPORT BAUD(%BAUD%); the string %BAUD% will be replaced by the data rate specified in your profile.

If you are storing a value with the variable, you do not use percent signs on the variable name. In the following example,

```
GETVALUE AFTER(CONNECT) INTO(CNNCTBAUD);
```

The GETVALUE command get the string after the word CONNECT and assigns this value to the variable CNNCTBAUD.

Using modem script commands

This section contains information on the commands and parameters you can use in modem scripts. The script commands along with the page numbers on which they are discussed are:

- CLEARBUFFER, page 324

  Use this command to clear the buffer that stores data received by the GETANSWER command if you do not want new data appended to data already in the buffer.

- CLOSEPORT, page 324

  Use this command to close the port specified in the profile.

- GETANSWER, page 325

  Use this command to read the data from the async port into a buffer.

- GETVALUE, page 325

  Use this command to search for a string in the buffer of data received by the GETANSWER command and copy data after that string into a variable.

- GO, page 326

  Use this command to jump to a labeled statement.

- IFANSWER, page 326

  Use this command to compare a given value with the contents of the buffer containing the results of the GETANSWER command, and then branch accordingly.

- IFVALUE, page 327

  Use this command to check the data found by the GETVALUE command, which was stored in a variable, and compare that value to a string, and branch accordingly.

- OPENPORT, page 328

  Use this command to open the port specified in the profile.

- RETURN, page 329

  Use this command to specify a code value to return to Expedite Base/AIX.

- SAY, page 329

  Use this command to send a string of characters to the async port.

- SETLINE, page 330

  Use this command to set the data bits, parity, and stop bits when connecting through APBX or similar, or for another application.

- SETPACING, page 331

  Use this command to set a pacing value in tenths of seconds to be used on SAY commands.

- WAIT, page 332

  Use this command to have Expedite Base/AIX wait for a specified time before proceeding to the next command.

The following sections provide detailed information on each of these commands.

CLEARBUFFER command

Use the CLEARBUFFER command to clear the buffer that stores data received from the async port by a GETANSWER command. For example, to have the buffer contain only the data from the most recent GETANSWER command, issue a CLEARBUFFER command before issuing the GETANSWER command. Otherwise, the new data received with each GETANSWER command will be appended to existing data. There are no parameters associated with CLEARBUFFER.

CLOSEPORT command

Use the CLOSEPORT command to close the port specified in the profile. There are no parameters associated with CLOSEPORT.

GETANSWER command

Use the GETANSWER command to read the data from the async port into a buffer.

Valid parameters in the GETANSWER command are:

timeout

The length of time, in whole seconds, to use as a time-out when receiving data from the async port. Valid values are 1 to 99. The default is 2 seconds.

mode

The mode to receive date, which can be:

async	Expedite Base/AIX receives all the data on the line for the length of time specified in the TIMEOUT parameter into the buffer. ASYNC is the default for MODE.
line	Expedite Base/AIX receives all the data from the async port up to the next carriage return (ASCII X'OD') into the buffer.

In either case, GETANSWER terminates if the async port is quiet for the length of time specified for TIMEOUT. Expedite Base/AIX adds the data received up to that point to the buffer.

The maximum length that the buffer can receive is 1K.

For example, the following command will wait up to 5 seconds for data coming from the async port. The data will be stored in a buffer. Use the IFANSWER command to read the contents of the buffer.

```
GetAnswer Timeout(5);
```

GETVALUE command

Use the GETVALUE command to search for a string in the buffer of data received and copy data after that string into a variable. GETVALUE will copy data up to the next blank.

For example, when a modem establishes a successful connection, it returns a message that might look as follows.

```
CONNECT 2400
```

You may be interested to know the data rate at which the connection was made. To do this, use the GETVALUE command to get the value after the word CONNECT and store it in the variable CNNCTBAUD.

```
GetValue After(CONNECT ) Into(CNNCTBAUD);
```

NOTE: The variable name CNNCTBAUD is used without the percent signs since you are putting a value into the variable instead of asking Expedite Base/AIX to make a substitution.

GO command

Use the GO command to jump to a labeled statement. Valid parameters for the GO command are:

to

This is the label on the line that you want to go to. The label can be up to 12 characters. This is a required parameter.

maxrepeat

This is the maximum number of times that you want to execute this command. Valid values are **1** to **99**. The default is **99** times.

NOTE: Using this command you may accidentally create a for-loop with no end. Expedite Base/AIX will not detect logic errors of this kind in your script. Therefore, only use this command to jump to a labeled RETURN statement. To make a for-loop, use the GETANSWERIFANSWER commands. See the example with the IFANSWER command on 138.

The following example shows how to use the GO command to jump to a labeled RETURN statement.

```
Go to(exit_ok);
...
...
# Normal return - continue processing
:exit_OK
Return code(0);
```

IFANSWER command

Use the IFANSWER command to compare a given value against the contents of the buffer containing the results of the GETANSWER command. The comparisons that Expedite Base/AIX makes when you issue an IFANSWER command are case sensitive. The search is for the exact value you specify for the TO parameter in the IFANSWER command.

Valid parameters in the IFANSWER command are:

is — You can specify either **equal** or **notequal** to create a conditional statement. This parameter is required.

- equal — The statement is true if Expedite Base/AIX finds the value you specify for the TO parameter in the buffer.
- notequal — The statement is true if Expedite Base/AIX does not find the value you specify for the TO parameter in the buffer.

to

Expedite Base/AIX searches the buffer for the value you specify for this parameter when you issue an IFANSWER command. Remember, Expedite Base/AIX searches for the value exactly as it appears here. The search is case sensitive. You can use up to 30 characters. This parameter is required.

maxrepeat

This is the maximum number of times that you want to execute this command. Use MAXREPEAT to create a for-loop in the script. Valid values are **1** to **99**. The default is **99** times.

goto

This is the label on the line that you want to go to if the comparison you specified for the IFANSWER command is true. This parameter is required. The label name can be up to 12 characters.

For example, the following command will check to see if the data in the buffer contains the word, ERROR, and will branch to the label, EXIT, if this is true.

```
IfAnswer is(equal) to(CONNECT) goto(FOUND) maxrepeat(4);
```

For more information, see “Example 2” on page 333.

IFVALUE command

Use this command to branch to a label in the script, based on the current value of the %CNNCTBAUD% variable. (Use the GETVALUE command to set the %CNNCTBAUD% variable.) Valid parameters for the IFVALUE command are:

of

Specifies the variable in which Expedite Base/AIX stored the value retrieved with the GETVALUE command. The only valid variable is %CNNCTBAUD%. This parameter is required. Maximum length is 11 characters.

is Specifies he type of comparison to make. This parameter is required. Maximum length is 8 characters

equal	The statement is true if the value of the extracted string specified by the OF parameter is the same as the value specified by the TO parameter.
notequal	The statement is true if the value of the extracted string specified by the OF parameter is not the same as the value specified by the TO parameter.

to

The character string to be compared with the value of the variable specified in the OF parameter. This parameter is required. Maximum length is 30 characters.

goto

This is the label on the line that you want to go to if the comparison you specified for the IFVALUE command is true. The label name can be up to 12 characters. This parameter is required.

maxrepeat

This is the maximum number of times that you want to execute this command. Valid values are **1** to **99**. The default is **99** times.

For example, use the following IFVALUE command to check the value of the data rate from the CONNECT message from the modem, and redial if it is not 2400 bps.

```
IfValue of(%CNNCTBAUD%) Is(NotEqual) To(2400) goto(REDIAL);
```

OPENPORT command

Use this command to open the port specified in the profile. Valid parameters in the OPENPORT command are:

baud

This is the data rate at which you want the port to open. The default data rate is 2400.

It is best to let Expedite Base/AIX select the data rate based on the rates associated with the phone number and the modem by specifying the %BAUD% variable as the value for the BAUD parameter, or by omitting the BAUD parameter from the command. If you need to specify a data rate, you can use the following rates:

- **300**
- **1200**
- **2400**
- **4800**
- **9600**
- **19200**
- **38400**

bits

This number specifies the bit rate (number of data bits) at which you want the port to open.

7 data bits, 1 stop bit, even parity. This is the default.
8 data bits, 1 stop bit, no parity.

Although Expedite Base/AIX changes bit rates automatically as needed, you can use the BITS parameter to specify a bit rate other than the default rate for some modems.

For example, the following command will open the port at a data rate of 9600 bps, using 8 bits no parity.

```
OpenPort baud(9600) bits(8);
```

Following is the recommended OPENPORT command which will use the data rate specified in the profile.

```
OpenPort baud(%BAUD%) bits(8);
```

RETURN command

Use the RETURN command to return a value to Expedite Base/AIX. No matter where you put the RETURN command in the script, Expedite Base/AIX stops processing the script and returns to Expedite Base/AIX with the value you specify for the CODE parameter.

NOTE: If you do not specify a RETURN command in a modem script, Expedite Base/AIX will end with a 12010 return code before the modem script commands are processed.

The valid parameter in the RETURN command is:

code

This is the value you want to return to Expedite Base/AIX. Specify 1 to 5 numeric characters. Use the value **12130** to have Expedite Base/AIX attempt a redial, if the redial count is not exceeded. Use the value **12998** to stop Expedite Base/AIX. A value of **0** indicates that the connection is successful.

If you use any other value, Expedite Base/AIX will end with the specified value.

You should not specify any return code documented in this book, other than **0**, **12130**, and **12998**, because it may cause Expedite Base/AIX to take an action that you do not expect. Any other return code will simply cause Expedite Base/AIX to exit with that return code.

This is a required parameter.

For example, the following command will exit the modem script processing with a return code of 12130, requesting a redial.

```
Return code(12130);
```

SAY command

Use this command to send a string of characters to the modem or network. You can tell Expedite Base/AIX to pace the string, which means that every character sent is followed by a pause. You specify the duration of the pause with the SETPACING command. The default pace value is one tenth of a second if you do not use the SETPACING command.

Valid parameters in the SAY command are:

string

These are the characters you want to send to the modem.

To send only a carriage return, use CR(Y) and omit the STRING parameter.

The STRING parameter is required on the SAY command if you do not use the CR parameter.

The maximum length is 80 characters, including variables.

cr

This parameter indicates whether or not you want a carriage return sent after the string. Valid values are:

y	Indicates that you want a carraiage return sent after the string. This is the default.
n	Indicates that you do not want a carriage return sent after the string. The CR parameter is required in the SAY command if you do not use the STRING parameter. Usually, if you are sending the modem attention string (+++), you do not want to send a carriage return to the modem.

paced

Indicates whether or not you want the string to be sent with a delay between each character. You specify the length of the delay with the SETPACING command if you want some other value besides the default one tenth second delay.

n	Indicates that you do not want the data sent with a delay between characters. This is the default.
y	Indicates that you want the data sent with a specified delay between each character.

For example, the following command will send the modem initialization string to the modem. You can specify a modem initialization string using the MODEMINIT parameter on the DIAL command. This string is substituted in the variable %INIT%.

```
Say string(%INIT%) paced(y);
```

Expedite Base/AIX will send the string to the modem with a one tenth second wait in between each character. Expedite Base/AIX will also send a carriage return to the modem after the string is sent.

SETLINE command

Use this command to leave the asynchronous line in a particular state when Expedite Base/AIX has completed or when connecting through a phone switch.

Also, use this command when you need to leave the asynchronous line in a particular state when Expedite has completed or when connecting through a phone switch. You can use SETLINE in the disconnect or reset modem script files to set the line for an application other than Expedite. You can also use it in the INIT or CNNCT scripts to connect through a PBX.

When Expedite Base/AIX communicates with the network, it sets the line to either 7bit-even-1 or 8bit-none-1, depending on the type of communication used. However, if you set the line to anything other than 7bit-even-1 in the connect script, set it back to 7bit-even-1 with another SETLINE command before the dial command is issued in the connect script, or Expedite Base/AIX cannot connect to the network.

Valid parameters in the SETLINE command are:

databits	Indicates 7- or 8-bit communications. The default is **8**
parity	Indicates even, odd, or no parity. The default is **none**.
stopbits	Indicates 1 or 2 stop bits. The default is **1**.

SETPACING command

Use this command to set a pacing value in tenths of seconds to be used on SAY commands. To use pacing on a SAY command, you must use the PACED parameter in the SAY command. You need only specify the SETPACING command once at the beginning of a script, it will be valid for the entire script.

The valid parameter in the SETPACING command is:

tenths

The length of time, in tenths of a second, to set the pacing value. Valid values are **0** to **99** seconds.

If you do not specify TENTHS, the default value will be one tenth of a second.

For example, the following command will set the pacing value to 5 tenths of a second. This pacing value is used if you specify PACED(y) on the SAY command.

```
SetPacing tenths(5);
```

SETWWASYNC command

Use this command to indicate the timeouts and line errors that Expedite Base/AIX should allow when communicating via worldwide asynchronous communication. If you are dialing directly to the network, you should have no problems using the default (NORMAL) setting. However, if you are dialing a secondary network or if you are experiencing poor line quality, you may need to use the LONG option to increase the timeout and line errors that Expedite Base/AIX will accept before terminating the session.

Specify this command in your modem initialization script file if you are using one. Otherwise, specify the command in the modem connect script.

The valid parameters on the SETWWASYNC command are:

timeout

This parameter indicates the type of timeout and error rate level that Expedite Base/AIX will use.

normal	Indicates that Expedite Base/AIX will use a default level of timeouts and acceptable error level when communicating. NORMAL is the default value for the TIMEOUT parameter.
long	Indicates that Expedite Base/AIX should wait longer and be more tolerant of line errors during communication. If you are using a secondary network to access the network, then use this value. Another reason to specify LONG is if you are not using a secondary network, but you see that Expedite Base/AIX must redial frequently due to poor line conditions.

WAIT command

Use this command to have Expedite Base/AIX wait for a specified time before proceeding to the next command. The valid parameter for the WAIT command is:

seconds

The length of time, in whole seconds, that you want Expedite Base/AIX to suspend activity. Valid values are **1** to **99** seconds. The default is **1** second.

For example, the following command will cause Expedite Base/AIX to wait three seconds before proceeding to the next command:

```
Wait seconds(3);
```

Sample modem scripts

This section provides examples on how to:

- Use conditional branching in modem scripts
- Create a for-loop in modem scripts
- Check the value of a variable in a script
- Use the welcome message script

Example 1

The following is an example of how to create a statement that compares a given value against the receive buffer (containing the results of the GETANSWER command) and then branch accordingly. In this example, the script ends with a return code 0 if Expedite Base/AIX received "OK" or 12130 if Expedite Base/AIX received "ERROR." Otherwise, Expedite Base/AIX continues processing.

```
    ClearBuffer;
    GetAnswer Timeout(5);
    IfAnswer is(Equal) to(OK) goto(Exit_OK);
    IfAnswer is(Equal) to(ERROR) goto(Exit_BAD);
    ...
    ...
:Exit_BAD              # Bad connection redial
   Return code(12130);
:Exit_OK               # No problems connected to Service Manager
   Return code (0);
```

Example 2

To create a for-loop in the script, use the GETANSWER-IFANSWER commands with a MAXREPEAT as follows:

```
# Send the 'reset to factory config' command and check for OK or
ERROR
Say String(AT &F);
:check_1
GetAnswer Timeout(2);
IfAnswer is(Equal) to(ERROR) goto(ok_1);
IfAnswer is(NotEqual) to(OK) goto(check_1) MaxRepeat(2);
:ok_1
```

This example shows how to read data from the modem for two seconds, and then check if the modem responded with ERROR or OK. If the modem did not respond with either ERROR or OK after the two seconds, then Expedite Base/AIX will repeat the read (GETANSWER) and compare (IFANSWER). This statement will be executed a maximum of two times.

Notice that the SAY command is outside the for-loop, which is between the two labels *check_1* and *ok_1*. If you create a for-loop that includes a SAY command, you may not get the results you expected if the modem is slow to respond. When the string is repeated to the modem during the loop execution, the modem will try to respond to each request. The modem's response to the first request will cause Expedite Base/AIX to execute the next command. The next response from the modem will actually be for the same SAY command executed earlier during the for-loop, but Expedite Base/AIX will interpret it as the response to the next GETANSWER command.

Example 3

The following example shows how to obtain the data rate at which the modems are actually connected and exit if the data rate is not what it is supposed to be. Use the GETVALUE command with IFVALUE in the connect script as follows. Note that this is not a complete script.

```
# Now send the dial command to the modem

Say string(ATD%ptype%%esc%%phone%);

# Wait for the CONNECT from the modem for 6. seconds (30 repeats * 2
seconds)
:check_3

GetAnswer Timeout(2);
IfAnswer is(NotEqual) to(CONNECT) goto(check_3) MaxRepeat(30);
IfAnswer is(NotEqual) to(CONNECT) goto(NoConnect);

# Since we just got the connect message, search the buffer for the
connect
# data rate and store it into CNNCTBAUD. First make sure we got the
# whole string back from the modem by issuing a GetAnswer command
# with a timeout of 0 seconds.

GetAnswer timeout(0);
GetValue after(CONNECT ) into(CNNCTBAUD);

# If we didn't connect at the highest data rate we can support,
# exit with a redial request.

IfValue of(%CNNCTBAUD%) is(NotEqual) to(%BAUD%) goto(NoConnect);

# Return code 12130 will cause Expedite to redial
:NoConnect
Return code(12130);
```

Example 4

The following example shows how to define the search strings for the Help Desk phone number and the Term ID if the Service Manager screen has been customized. You may include these commands in welcmsg.scr, or, if you are using dial access to the network and specified an INITSCR parameter on the DIAL command you can include them in your initialization script or dial connect script.

```
GetValue After(RING: ) Into(HOTLINE);
GetValue After(TERMINAL: ) Into(TERMID);
```

Example 5

The following is the modem script included as a sample with Expedite Base/AIX, which you can use as an example of the modem script commands described in this chapter.

```
# Clear the receive buffer

ClearBuffer ;

# Close and open the port

ClosePort ;
OpenPort baud(%BAUD%) Bits(8);

#Send the 'reset to factory config' command and check for OK or ERROR

Say String(%RESET%);

:check_1

GetAnswer Timeout(2);
IfAnswer is(Equal) to(ERROR) goto(ok_1);
IfAnswer is(NotEqual) to(OK) goto(check_1) MaxRepeat(2);

# Send the modem initialization string

:ok_1

ClearBuffer ;
Say String(%INIT%) ;

:check_2

GetAnswer Timeout(2);
IfAnswer is(Equal) to(ERROR) goto(ok_2);
IfAnswer is(NotEqual) to(OK) goto(check_2) MaxRepeat(.5);

# Now send the dial command to the modem
:ok_2
ClearBuffer ;
Say String(ATD%PTYPE%%ESC%%PHONE%);

# Wait for the CONNECT from the modem for 6.0. seconds (30 * 2)

:check_3

# Use line mode to receive data up to the Carriage Return. This way
# we are sure to get the data rate with the CONNECT message from the modem.

GetAnswer Timeout(2) Mode(LINE);
IfAnswer is(NotEqual) to(CONNECT) goto(check_3) maxrepeat(30);
IfAnswer is(NotEqual) to(CONNECT) goto(exit_bad);

# Send characters necessary for autobps for the BSC dial

Say CR(Y);
Wait Seconds(1);
Say CR(Y);

# Get the connect data rate and put it in CNNCTBAUD to be displayed
# using the display script.
# We do this here to make sure we received all characters in the
# data rate (instead of CONNECT 240)
GetAnswer Timeout(0);
GetValue after(CONNECT ) into(CNNCTBAUD);

Go to(exit_ok);

# Error return no CONNECT message
```

```
:exit_bad
Return code(12130);

# Normal return continue processing

:exit_OK
Return code(0);
```

Using modem initialization and reset scripts

By default, Expedite Base/AIX uses the information in the connect script when attempting to establish a connection to the network, and the disconnect script when disconnecting. These scripts are run each time you attempt a connection.

If you use other software with your modem that requires a different setup than Expedite Base/AIX, you may want to use initialization and reset scripts to set up the modem only once for the Expedite Base/AIX session and reset for the other software at the end of the Expedite Base/AIX session. To do this, create an initialization script with the commands to change the modem setup to what Expedite Base/AIX needs, create a reset script to change the modem setup to what the other software needs, and specify the names of the scripts on the DIAL command in the INITSCR and RESETSCR parameters. You may also specify special initialization and reset scripts for particular modems by specifying INITSCR*x* or RESETSCRx where *x* matches the *x* in the DEVICE*x* name for the modem.

NOTE: Initialization and reset scripts are customized for the modem. Connect and disconnect scripts are customized for the phone number dialed.

If you use an initialization script, you may want to remove any initialization commands from the connect script. There are several things you can do to accommodate this:

1. Modify the connect script that you are currently using, removing any initialization commands
2. Create a new connect script without initialization commands and specify the name of the script in the CNNCTSCR parameter on the DIAL command

If you have multiple modems, but only one of them requires a special initialization script, you may run into problems if you remove the initialization commands from the connect script. The modems that do not use initialization scripts will not be properly initialized. In this case you should set up two initialization scripts: one with specialized commands for one modem, and another for the remaining modems. This second script will simply have the initialization commands taken from the default connect script. Another alternative is to create a single initialization script for the special modem, but not modify the connect script.

It is not necessary to change the disconnect script if you are using the default script that was provided with Expedite Base/AIX.

Following is an example of a DIAL command showing how to use initialization, reset, and connect scripts.

```
DIAL   PHONE1(1234567)
       DEVICEA(/dev/tty0) DBAUDRATEA(9600)
       DDEVICEB(/dev/tty1) DBAUDRATEB(9600)
       DEVICEC(/dev/tty2) DBAUDRATEC(2400)
       DEVICED(/dev/tty3) DBAUDRATED(9600)
       INITSCR(geninit.scr)
       CNNCTSCR(gencnnct.scr)
       INITSCRD(init3.scr)
       RESETSCRD(reset3.scr)
       ;
```

This example shows how to set up for 4 modems on the system, where you need special initialization commands for */dev/tty3* only. This device is specified in parameter DEVICED. When this device is used, Expedite Base/AIX will run the initialization script *init3.scr* (specified in parameter INITSCRD) and reset script *reset3.scr* (specified in parameter RESETSCRD). The commands in these scripts have commands specific for this device.

When any of the other devices are used, Expedite Base/AIX will run the initialization script *geninit.scr* (specified in parameter INITSCR). There is no reset script to process for these other devices. Since these devices do not require special initialization commands, the script *geninit.scr* has the default initialization commands provided in the default connect script.

It is not necessary to change the disconnect script if you are using the default script that was provided with Expedite Base/AIX.

The following is an example of an initialization script:

```
# Clear the receive buffer
ClearBuffer ;

# Close and open the port

ClosePort ;
OpenPort baud(%BAUD%) Bits(7);

#Send the 'reset to factory config' command and check for OK or ERROR

Say String(AT &F); :check_1

GetAnswer Timeout(02);
IfAnswer is(Equal) to(ERROR) goto(ok_1);
IfAnswer is(NotEqual) to(OK) goto(check_1) MaxRepeat(02);

# Send the modem initialization string

:ok_1

ClearBuffer ;
Say String(%INIT%) ;

:check_2
```

```
GetAnswer Timeout(02);
IfAnswer is(Equal) to(ERROR) goto(ok_2);
IfAnswer is(NotEqual) to(OK) goto(check_2) MaxRepeat(05);

# Exit after the modem is initialized. Always return 0.
:ok_2
Return code(0);
```

The following example is the corresponding connect script, with the initialization commands removed:

```
# Close and open the port.
# If init script closed the port,
# Expedite will know how to handle this.
# If we are on a redial,
# we may need to close the port.

ClosePort ;
OpenPort baud(%BAUD%) Bits(7);

# Send the dial command to the modem.

ClearBuffer ;
Say String(ATD%PTYPE%%ESC%%PHONE%);

# Wait for the CONNECT from the modem for 60 seconds (30 * 2)

:check_3

GetAnswer Timeout(02);
IfAnswer is(NotEqual) to(CONNECT) goto(check_3) maxrepeat(30);
IfAnswer is(NotEqual) to(CONNECT) goto(exit_bad);

# Send characters necessary for autobps for the BSC dial

Say CR(Y);
Wait Seconds(1);
Say CR(Y);

# Get the connect data rate and put it in CNNCTBAUD to be displayed
# using the display script.
# We do this here to make sure we received all characters in the
# data rate (instead of CONNECT 240)
GetAnswer Timeout(0);
GetValue after(CONNECT ) into(CNNCTBAUD);

Go to(exit_ok);

# Error return  no CONNECT message

:exit_bad
Return code(12130);

# Normal return  continue processing

:exit_OK
Return code(0);
```

Finally, here is an example of a reset script.

```
# Clear the receive buffer
ClearBuffer ;

# Close and open the port

ClosePort ;
OpenPort baud(%BAUD%) Bits(7);
```

```
# Send the settings for the other software.

Say String(AT &W1&C2);

:check_1

GetAnswer Timeout(02);

IfAnswer is(Equal) to(ERROR) goto(ok_1);
IfAnswer is(NotEqual) to(OK) goto(check_1) MaxRepeat(02);

:ok_1

ClosePort ;
Return code(0);
```

Using a customized Service Manager logon screen

If the Network Services Manager logon screen has been customized in your country for your local language or for other reasons, then Expedite Base/AIX will need to know how to find the Help Desk hotline number and the terminal ID. You provide the information about the customized Service Manager logon screen in a script called welcmsg.scr.

In the United States and some other countries, Expedite Base/AIX searches for the default string ASSISTANCE: on the screen to find the hotline number, for TERMID: to find the terminal ID for that session, and for ==> to find the logon prompt. If these strings are different in your country, specify the strings Expedite Base/AIX should search in welcmsg.scr.

If you are using 3270 emulator communications, Expedite Base/AIX looks for the welcmsg.scr file and if it does not exist, assumes that the default search strings are acceptable. See “Creating modem scripts” on page 131 for more information about welcmsg.scr.

The format of welcmsg.scr is consistent with the other free-format commandstyle input files. The only command that you should use in welcmsg.scr is GETVALUE. See “GETVALUE command” on page 136 for more information about the GETVALUE command. The possible values for GETVALUE are:

hotline	Text that Expedite Base/AIX should search for to find the Help Desk phone number. The default is **ASSISTANCE:**
termid	Text that Expedite Base/AIX should search for to find the terminal ID. The default is **TERMID:**
logon	Text that Expedite Base/AIX should search for to find the logon prompt. The defacult is **===>**

The following is an example of a welcmesg.scr file:

```
# Find the help desk phone number.

GetValue After(RING: ) Into(HOTLINE);

# Find the termid.

GetValue After(TERMID: ) Into(TERMID);
```

Chapter 15

Using SNA LU 6.2 communications

You need to perform certain install and setup activities before you install LU 6.2 for your RISC System/6000 system.

Installation prerequisites

Prior to installing LU 6.2 for the RISC System/6000, the necessary hardware and device interfaces for a leased line must be configured and Communication Server 4.2 and the related PTFs must be installed. You must have AIX Version 5.1 Base Operating System.

Installing LU 6.2

To install and set up Expedite Base/AIX for the IBM RISC System/6000 (for the RISC System/6000 only) for LU 6.2 communications, do the following:

1. Add Data Link Control (DLC) for SDLC.
2. Import the *expconn.sna* file into SNA Services.
3. Update the Communication Server configuration with your own information.
4. Verify the Communication Server configuration.
5. Start Communication Server.

The steps to complete these tasks are described in the following sections.

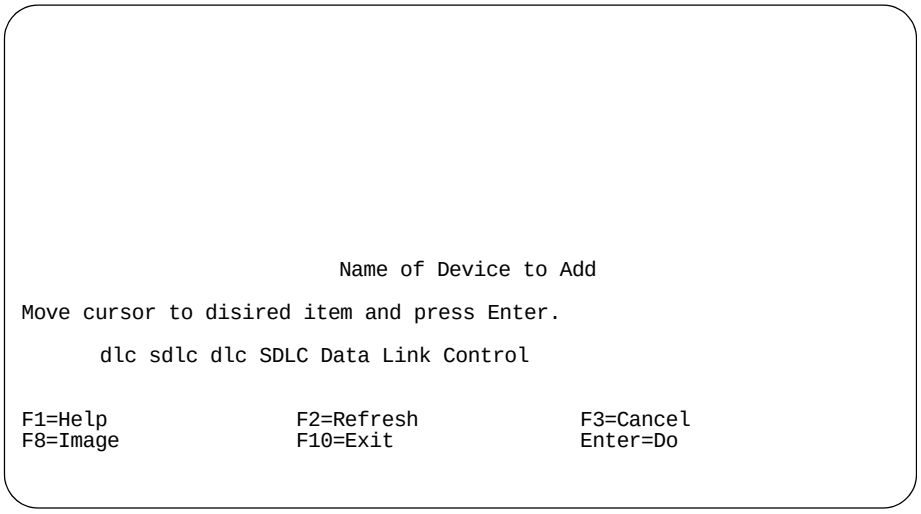
NOTE: The function keys displayed on your panels may be different than those shown in these instructions.

Adding DLC for SDLC

To add DLC for SDLC, do the following:

1. Type **smit cmddlc_sdlc_mk**

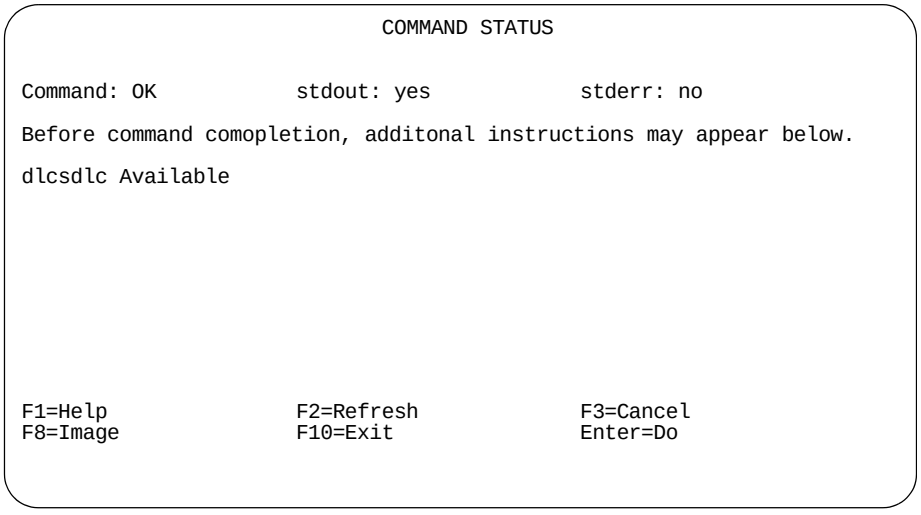
   The system displays the Name of Device to Add popup.



2. Select **dlc sdlc dlc SDLC Data Link Control** and press Enter.

   The system displays the Command Status panel.

   Verify that OK is returned for the addition of SDLC Data Link Control.



3. Continue to press F3 until you have exited from *smit*; or press F10.

Importing the expconn.sna file into Communication Server

To import the expconn.sna file, do the following:

1. Type **smit importsna**

   The system displays the Import Configuration Profiles panel.

```
                        Import Configuration Profiles

Type or select values in entry fields.
Press Enter AFTER make all desired changes.
                                                          [Entry Fields]

* File from which to                                      []
  REPLACE duplicate profiles?                             yes
  Security information source file                        []



F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

2. In the File from which to import field, type **expconn**.

NOTE: Be sure to include the full path name to indicate where the files are located for both fields.

The *experr* file will be created when this step is completed. It will contain any errors generated from importing the expconn configuration files.

The system displays the Command Status panel.

```
                                COMMAND STATUS

Command: OK            stdout: no             stderr: no

Before command completion, additinal instructions may appear below.









F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

3. Continue to press F3 until you have exited from *smit*.

Updating any configuration values needed in Communication Server

To update any configuration values needed in Communication Server, do the following:

1. Type **smit _sna232linkch**

   The system displays the EIA232D SNA DLC Profile Name panel.

```
                    EIA232D SNA DLC Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.


                                              [Entry Fields]
* Profile name                                []




F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

2. Press F4 to view the list of profiles names

   The system displays the Profile name pop-up.

```
                    EIA232D SNA DLC Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.


                                                  [Entry Fields]
* Profile name                                    []            +

                               PROFILE name

   Move cursor to desired item and press Enter.

   DLINKDEFAULT
   expconn

F1 F1=Help              F2=Refresh            F3=Cancel
F5 F8=Image             F10=Exit              Enter=Do
F9
```

3. Select **expconn** and press Enter.

The system displays the Change / Show SDLC EIA232D SNA DLC Profile panel.

```
                Change / Show SDLC EIA232D SNA DLC Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                          [Entry Fields]
   Current profile name                        expconn
   New profile name                            []
   Data link device name                       [mpq.]           +
   Force disconnect timeout (1600 seconds)     [120]            #
   Userdefined maximum IField size?            no               +
     If yes, Max. IField size (2654096)        [265]            #
   Link type                                   point-to-point   +
   Max. num of active link stations (1-255)    [1]              #
     Number reserved for inbound activation    [0]              #
     Number reserved for outbound activation   [0]              #
   Serial encoding                             nrz              +
   Request to send (RTS)                       controlled       +
   DTR control                                 DTR              + [
   MORE...25]


F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

4. Update the Data link device name field if needed (based upon where your multiprotocol adapter is installed), and press Enter.

NOTE: For LU 6.2 communication outside the U.S., the serial ENCODING parameter may need to be changed to NZRI.

The system displays the Command Status panel after the update has been made with the Command field as OK.

```
                              COMMAND STATUS

Command: OK               stdout: no              stderr: no

Before command completion, additional instructions may appear below.









F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

5. Continue to press F3 until you have exited *smit*.

6. Type **smit _snalocalu6ch**

The system displays the LU 6.2 Local LU Profile Name panel.

```
                        LU 6.2 Local LU Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.
                                                    [Entry Fields]
\ Profile name                                      []                +









F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

7. Press F4.

   The system displays the Profile name pop-up.

```
                        EIA232D SNA DLC Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.


                                                    [Entry Fields]
* Profile name                                      []             +

    ┌──────────────────────────────────────────────────────────────┐
    │                          Profile name +                      │
    │                                                              │
    │ Move cursor to desired item and press Enter.                 │
    │                                                              │
    │    LDEFAULT +                                                │
    │    expconn +                                                 │
F1  │                                                              │
F5  │ F1=Help               F2=Refresh            F3=Cancel +      │
F9  │ F8=Image              F1ð=Exit              Enter=Do +       │
    └──────────────────────────────────────────────────────────────┘
```

8. Select **expconn** and press Enter.

The system displays the Change / Show SDLC EIA232D SNA DLC Profile panel.

```
                Change / Show SDLC EIA232D SNA DLC Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
Current profile name                              expconn
New profile name                                  []
* Local LU name                                   [IBMX1ES3]
Local LU alias                                    [IBMX1ES3]
Local LU is dependent?                            yes           1
     If yes,
       Local LU address (1-255)                   [4]           #
       System services control point
         (SSCP) ID (*,0-65535)                    [*]
       Link Station Profile name                  [expconn]     +
Conversation Security Access List Profile name    []            +
Recovery resource manager (RRM) enabled?          no            +


Comments                                          []

F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
```

9. Update the Local LU Name and Local LU Address fields to the values provided by your systems engineer and press Enter.

   The system displays the Command Status panel after the update has been made with the Command field as OK.

```
                              COMMAND STATUS

Command: OK               stdout: no              stderr: no

Before command completion, additional instructions may appear below.









F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

10. Continue to press F3 until you have exited smit.

11. Type **smit _snasidech**

The system displays the LU 6.2 Side Information Profile Name panel.

```
                  LU 6.2 Side Information Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.
                                                  [Entry Fields]
* Profile name                                    []               +









F1=Help                 F2=Refresh      F3=Cancel        F4=List
F5=Reset                F6=Commad       F7=Edit          F8=Image
F9=Shell                F10=Exit        Enter=Do
```

12. Press F4.

    The system displays the Profile name pop-up.

```
                  LU 6.2 Side Information Profile Name

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.


                                                  [Entry Fields]
* Profile name                                    []           +

                           Profile name +

   Move cursor to desired item and press Enter.

      CDEFAULT
      expconn

F1 F1=Help                 F2=Refresh             F3=Cancel +
F5 F8=Image                F1ð=Exit               Enter=Do +
F9
```

13. Select **expconn** and press Enter.

The system displays the Change / Show LU 6.2 Side Information Profile panel.

```
              Change / Show LU 6.2 Side Information Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
Current profile name                              expconn
New profile name                                  []
Local LU name or Control Point alias              [IBMX1ES3]
Provide only one of the following:                [IBMX1ES3]    +
     Partner LU alias                             []
     Fully qualified partner LU name              [IBMIN, IBMORELY]
 Modem name                                       [LU62]
Remote transaction program name RTPN)             [RCON]
RTPN in hexadecimal?                              no


Comments                                          []

F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

14. Update the fully qualified partner LU name field to the value provided by your systems engineer and press Enter.

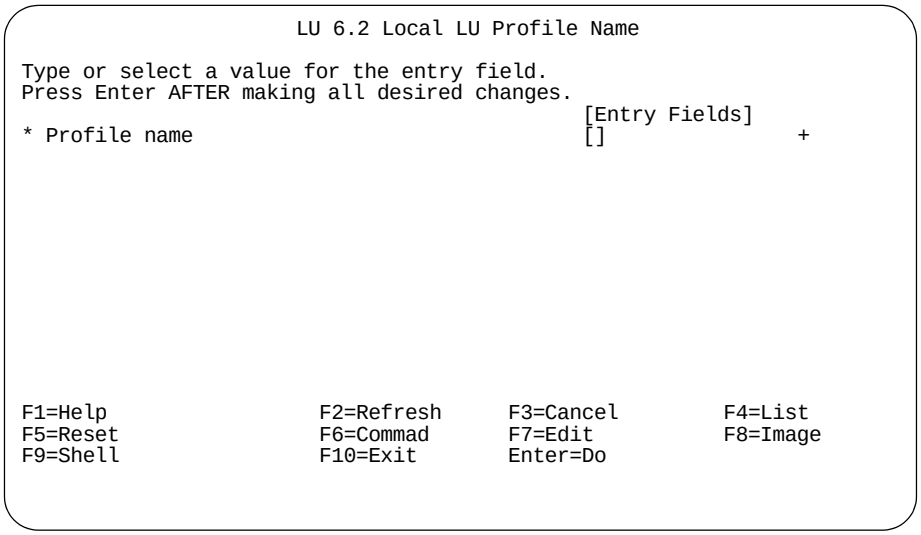
    The system displays the Command Status panel after the update has been made with the Command field as OK.

```
                              COMMAND STATUS

Command: OK               stdout: no              stderr: no

Before command completion, additional instructions may appear below.









F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

15. Continue to press F3 until you have exited *smit*.

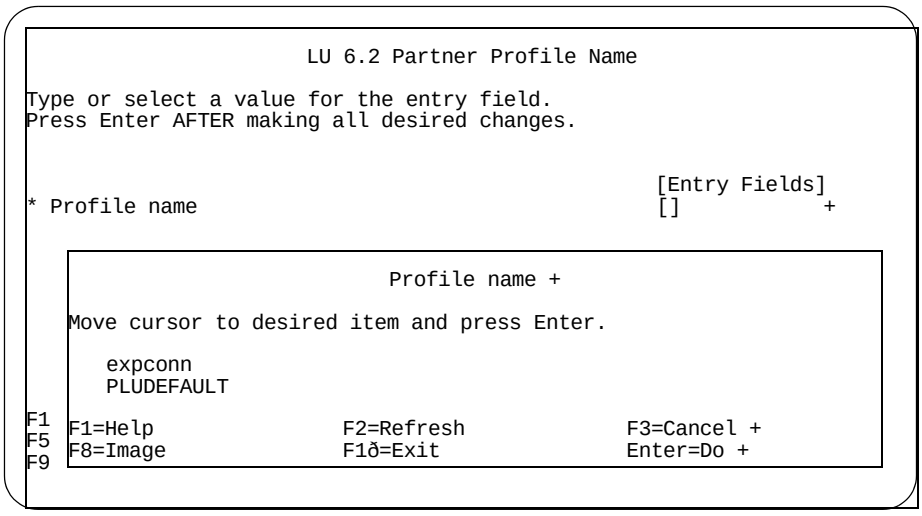
16. Type **smit _snapartch**

The system displays the LU 6.2 Local LU Profile Name panel.



17. Press F4.

    The system displays the Profile name pop-up.



18. Select **expconn** and press Enter.

The system displays the Change / Show LU 6.2 Partner LU Profile panel.

```
                  Change / Show LU 6.2 Partner Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
   Current profile name                           expconn
   New profile name                               []
*  Fully qualified partner LU name                [IBMIN, IBMORELY]
Partner LU alias                                  []
Parallel sessions supported?                      no            +
Session security supported?                       no            +
Conversation security level                       none          +

Comments                                          []




F1=Help                 F2=Refresh      F3=Cancel         F4=List
F5=Reset                F6=Commad       F7=Edit           F8=Image
F9=Shell                F10=Exit        Enter=Do
```

19. Update the Fully qualified partner LU name field to the value provided by your systems engineer and press Enter.

    The system displays the Command Status panel after the update has been made with the Command field as OK.

20. Continue to press F3 until you have exited *smit*.

Verifying the SNA configuration

To verify the SNA configuration in Communication Server, do the following:

1. Type **smit verifysna**
2. Verify that the system displays the Command Status panel after the command has been processed with OK returned for the Command field.

NOTE: If OK is not returned, then either the hardware is not properly configured, or the updates made to the Communication Server configuration are not correct. Verify the hardware configuration and the changes made during the previous section, then re-run the steps in this section.

```
                                COMMAND STATUS

Command: OK                stdout: no              stderr: no

Before command completion, additional instructions may appear below.









F1=Help                 F2=Refresh      F3=Cancel       F4=List
F5=Reset                F6=Commad       F7=Edit         F8=Image
F9=Shell                F10=Exit        Enter=Do
```

Starting SNA

To start SNA, do the following:

1. Type **smit _snastart**
2. Verify that the system displays the Command Status panel after the command is processed with OK returned for the Command field and a Subsystem PID specified.

NOTE: The Subsystem PID number will be different each time SNA is started. If OK is not returned, then either the hardware is not properly configured, or the updates made to the Communication Server configuration are not correct. Verify the hardware configuration and the changes made during the previous sections, then re-run the steps in this section.

```
                                COMMAND STATUS

Command: OK               stdout: no              stderr: no

Before command completion, additional instructions may appear below.

ð513-ð59 The SNA Subsystem has been started. Subsystem PID is 9715.









F1=Help                 F2=Refresh      F3=Cancel       F4=List
F5=Reset                F6=Commad       F7=Edit         F8=Image
F9=Shell                F10=Exit        Enter=Do
```

3. Continue to press F3 until you have exited *smit*.

Chapter 16

Using TCP/IP communications

With TCP/IP communication, Expedite Base/AIX for the IBM RISC System/6000 can connect to the network once a TCP/IP connection is established. You establish the TCP/IP connection prior to starting Expedite Base/AIX for the IBM RISC System/6000.

For TCP/IP communication, the COMMITDATA default is 141000 and the MSGSIZE default is 47000, both of which are higher than the defaults for other COMMTYPEs. The link trace information is found in link.trc.

Preparing for TCP/IP communication

Before you begin TCP/IP communication, follow these steps:

1. Configure the TCP/IP stack.
2. Update the TRANSMIT command and optionally add the TCPCOMM command in basein.pro.

NOTE: Your Information Exchange user ID and your IBM Global Network ID must be registered for TCP/IP communication.

Updating the TRANSMIT command

To indicate that you are using TCP/IP communication, specify the COMMTYPE parameter with the value T on the TRANSMIT command in basein.pro. The following is an example of the updated TRANSMIT command for TCP/IP:

```
TRANSMIT COMMTYPE(T);
```

If you do not specify T for the COMMTYPE, Expedite Base/AIX for the IBM RISC System/6000 will use the default value a, which indicates asynchronous communications.

NOTE: If you are using TCP/IP communication, do not specify the DIAL command in basein.pro.

Updating the TCPCOMM command

You can include the TCPCOMM command in basein.pro with the TIMEOUT parameter specified to override the Expedite Base/AIX defaults for the inactivity timeout.

The following is an example of the TCPCOMM command that shows the parameters you use to change the inactivity timeout defaults for TCP/IP:

```
TCPCOMM TIMEOUT(5);
```

Updating hostname.fil

Expedite Base/AIX comes with a file called hostname.fil. This file contains the addresses for all regions that can be used to communicate with the network using TCP/IP. The pound signs/hash marks (#) which prefix each address indicate that the line is “commented out.” To use TCP/IP, you must decide which address you want to use. Then, remove the # character preceding that address so that it is no longer commented out.

If you wish to change this address at any time, simply edit the file hostname.fil. Type a # character at the beginning of the original address so that Expedite Base/AIX no longer uses it. Then, remove the # character preceding the new address that you want to use and save the file. Expedite Base/AIX will now use the new address you selected.

Appendix A

Expedite Base/AIX messages and codes

This appendix describes the Expedite Base/AIX messages and codes. They are divided into the following categories:

- Expedite Base/AIX completion codes (0 - 114), page 358
- Expedite Base/AIX return codes

  This section includes the following codes:

  - Message command file syntax errors (02014 - 04988), page 360
  - Profile command file syntax errors (05018 - 06020), page 388
  - Network errors (11801 - 11999), page 399
  - Modem script syntax errors (12010 - 12130), page 406
  - Display status script syntax errors (12210 - 12300), page 413
  - Communication device driver errors (13030 - 13094), page 417
  - Parser errors (14000 - 15042), page 419
  - Destination verification errors (16020 - 16060), page 421
  - EDI errors (17102 - 19017), page 422
  - General environment errors (20365 - 23610), page 431
  - Session start and end errors (24000 - 24610), page 437
  - PF key exit errors (25000), page 440
  - Internal communications errors (26401 - 26999), page 440
  - Old message.fil errors (27010), page 445
  - Session errors (28000 - 29999), page 445
  - SSL communication errors (30006 - 30120), page 451
  - Unexpected and program interrupt errors (31000 - 32000), page 455

For information on messages generated by Information Exchange, see *Information Exchange Messages and Formats*, GC34-2324.

Expedite Base/AIX completion codes

The following are Expedite Base/AIX completion codes. The program returns these completion codes via the operating system exit code.

0000 Session completed successfully.

Explanation: Expedite Base/AIX completed successfully.

User Response: No user action is required.

0001 Expedite is in restart mode.

Explanation: The -s command line parameter indicates that Expedite Base/AIX is in a restart mode. The next time you run iebase, Expedite Base/AIX will attempt to continue where the last session ended.

User Response: There is no action necessary if you wish to restart where the last session ended. You may wish to check the output files, baseout.pro, baseout.msg, and tempout.msg, to see if there is a problem which requires a fix.

0101 Invalid command line parameter.

Explanation: Valid command line parameters are -b, -s, -r, -v, and -p< pathname>. This return code is passed to the operating system.

User Response: Correct the command line parameter and try again.

0102 Control path is too long.

Explanation: The path specified by the -p command line parameter is too long. The maximum length is 128 characters. This return code is passed to the operating system.

User Response: Correct the control path and try again.

0103 Control path does not exist.

Explanation: The path specified by the -p command line parameter does not exist. This return code is passed to the operating system.

User Response: Correct the control path and try again.

0104 Profile syntax error.

Explanation: Profile syntax error.

User Response: Look at the return code specified in the profile response file, baseout.msg, for a description of the error. Correct the problem and restart the program.

0110 Unable to complete the session due to a network problem.

Explanation: Expedite Base/AIX was unable to complete a session because Information Exchange was not available, or a timeout occurred while Expedite Base/AIX was waiting for a response.

User Response: Wait and try the program later. You can use the CYCLE and WAIT parameters on the DIAL command to automatically retry a connection at periodic intervals. If the problem persists, contact the Help Desk.

0111 Session was unsuccessful.

Explanation: The previous session was unsuccessful due to a problem that Expedite Base/AIX encountered.

User Response: Look at the return codes specified in the response files for a description of the error. Correct the problem and restart the program.

0112 Session completed with warnings.

Explanation: The session completed, but warnings were generated for one or more commands. It is possible that not all of the commands were processed successfully.

User Response: Look at the return codes specified in the response files for a description of the warnings/errors. If necessary, correct the problems and restart the program.

0113 Session reset recommended.

Explanation: Expedite Base/AIX encountered an error condition which could not be resolved by restarting the session. You must reset the session before using Expedite Base/AIX again.

User Response: Look at the return codes specified in the response files for a description of the error. Correct the problem and reset the session. Refer to the product documentation for information about resetting the session.

0114 Session ended in restart mode.

Explanation: An error has occurred which can possibly be resolved by reconnecting to the network. Expedite Base/AIX attempts to reconnect automatically, but has excluded the reconnect count.

User Response: Run the program again. If the problem persists, check the session return codes and correct the error.

Message command file syntax errors

This system describes the return codes for message command file syntax errors.

2014 ALIAS invalid on LIST command.

Explanation: You specified an invalid value for an ALIAS parameter in the LIST command. The first character of the ALIAS is the destination table type and must be blank, g, o, or p. The last three characters of the ALIAS are the destination table ID and must be one to three alphanumeric characters. If the first character is blank, then the last three characters of the ALIAS parameter must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2032 LISTNAME missing on LIST command.

Explanation: The LISTNAME parameter in the LIST command is missing or blank. A LISTNAME must be specified.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LIST command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2062 FUNCTION missing or invalid on LIST command.

Explanation: The FUNCTION parameter in the LIST command is missing, or you have specified an invalid value. The FUNCTION value must be a, d, e, or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LIST command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2064 Invalid entries on LIST command.

Explanation: Either entries were specified in the LIST command with an 'e' for FUNCTION parameter, or no entries were specified for one of the other FUNCTION values.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LIST command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2066 LISTTYPE invalid on LIST command.

Explanation: The LISTTYPE value in the LIST command is invalid. The LISTTYPE value must be t, p, a, or g.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2098 Invalid command sequence on LIST command.

Explanation: A LISTNAME, FUNCTION, or LISTTYPE parameter was specified more than once on the LIST command. You can only specify these parameters once in each LIST command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2099 Incomplete destination on LIST command.

Explanation: The list ended or a new destination was started before a previous destination was finished. This is usually caused by a missing parameter somewhere in the list. List entries are made up of either an ACCOUNT and USERID, or an ALIAS and ALIASNAME. Therefore, you must specify each list entry component next to its counterpart. For example, an ACCOUNT parameter should be entered next to a USERID parameter. If you specify the SYSID parameter, you must specify it either before the ACCOUNT and USERID parameters to which it belongs or between them. It cannot follow the ACCOUNT and USERID parameters for that SYSID.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the LIST command in the message command file, basein.msg, and retry the program.

2201 Both KEYRINGSTASHFILE and KEYRINGPASSWORD parameters specified.

Explanation: You specified both the KEYRINGTSTASHFILE and the KEYRINGPASSWORD parameters on the START command.

User Response: When the file specified in the KEYRINGFILE parameter was created, it was created to have either a password or to have the password stored in a stash file. If the password was stored in a stash file, then specify the name of the stash file in the KEYRINGSTASHFILE parameter. Otherwise, specify the password in the KEYRINGPASSWORD parameter. Correct the START command by specifying only the KEYRINGSTASHFILE parameter or the KEYRINGPASSWORD parameter, and then retry the program.

2202 KEYRINGFILE must be specified.

Explanation: The KEYRINGFILE names was not specified. This is a required parameter.

User Response: When making an SSL connection, there must be a KDB file that contains the certificate that allows the connection. This filename is specified in the KEYRINGFILE parameter. This parameter is required to enable SSL. Correct the START command by specifying the filename of the KEYRINGFILE parameter, and then retry the program.

2204 No password provided for KEYRINGFILE.

Explanation: The key database or key ring specified in the KEYRINGFILE parameter requires a password.

User Response: Provide either the correct password for the KEYRINGPASSWORD parameter or KEYRINGSTASHFILE that is associated with the key database that is specified in the KEYRINGFILE parameter, and then retry the program.

2402 No destination on CANCEL command.

Explanation: You must specify a destination for this command. You can use ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2404 Multiple destinations on CANCEL command.

Explanation: You can specify only one destination in the CANCEL command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2406 Insufficient destination on CANCEL command.

Explanation: You must specify a complete destination for this command. You can use ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2414 ALIAS invalid on CANCEL command.

Explanation: You specified an invalid value for the ALIAS parameter in the CANCEL command. The first character of the ALIAS is the destination table type and must be blank, g, o, or p. The last three characters of the ALIAS are the destination table ID and must be one to three alphanumeric characters. If the first character is blank, then the last three characters of the ALIAS must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2444 PRIORITY invalid on CANCEL command.

Explanation: You specified an invalid value for the PRIORITY parameter in the CANCEL command. The value of the PRIORITY parameter must be blank or p.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2450 ACK invalid on CANCEL command.

Explanation: You specified an invalid value for the ACK parameter in the CANCEL command. The value of the ACK parameter must be blank, h, or t.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2464 STARTDATE invalid on CANCEL command.

Explanation: You specified an invalid value for the STARTDATE parameter in the CANCEL command. The format must be YYMMDD, where YY is the last two digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2466 STARTTIME invalid on CANCEL command.

Explanation: You specified an invalid value for the STARTTIME parameter in the CANCEL command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2468 ENDDATE invalid on CANCEL command.

Explanation: You specified an invalid value for the ENDDATE parameter in the CANCEL command. The format must be YYMMDD, where YY is the last two-digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2470 ENDTIME invalid on CANCEL command.

Explanation: You specified an invalid value for the ENDTIME parameter in the CANCEL command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2472 TIMEZONE invalid on CANCEL command.

Explanation: You specified an invalid value for the TIMEZONE parameter in the CANCEL command. TIMEZONE must be either g or l.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program.

2474 End date/time is before start date/time.

Explanation: You specified an end date and time in the CANCEL command that is before the start date and time. If you specified either date field with a two-digit year, a sliding window technique was used to decide the century indicator which is used when comparing the start and end dates.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which CANCEL command produced the error. Correct the CANCEL command in the message command file, basein.msg, and retry the program. The sliding window technique is described in the readme file.

2604 Multiple destinations on AUDIT command.

Explanation: You can specify only one destination in the AUDIT command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2606 Incomplete destination on AUDIT command.

Explanation: You specified an incomplete destination in the AUDIT command. If you specify a destination, you must use ACCOUNT and USERID; SYSID, ACCOUNT and USERID; or ALIAS and ALIASNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2614 ALIAS invalid on AUDIT command.

Explanation: You specified an invalid value for the ALIAS parameter in the AUDIT command. The first character of the ALIAS is the destination table type and must be blank, g, o, or p. The last three characters of the ALIAS are the destination table ID and must be one to three alphanumeric characters. If the first character is blank, then the last three characters of the ALIAS must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2664 STARTDATE invalid on AUDIT command.

Explanation: You specified an invalid value for the STARTDATE parameter in the AUDIT command. The format must be YYMMDD, where YY is the last two digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2668 ENDDATE invalid on AUDIT command.

Explanation: You specified an invalid value for the ENDDATE parameter in the AUDIT command. The format must be YYMMDD, where YY is the last two digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2670 ALTACCT invalid on AUDIT command.

Explanation: You specified ALTACCT and did not specify ALTUSERID or you specified ALTACCT and LEVEL 1 or 2. If you specify ALTACCT you must specify ALTUSERID. ALTACCT is only supported for LEVEL 3 or higher.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2672 TIMEZONE invalid on AUDIT command.

Explanation: You specified an invalid value in the TIMEZONE parameter on the AUDIT command. TIMEZONE must be either g or l.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2674 STATUS invalid on AUDIT command.

Explanation: You specified an invalid value for the STATUS parameter in the AUDIT command. STATUS must be blank, u, p, or d.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2676 MSGTYPE invalid on AUDIT command.

Explanation: You specified an invalid value for the MSGTYPE parameter in the AUDIT command. MSGTYPE must be s, r, or b.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2678 End date is before start date.

Explanation: You specified an end date in the AUDIT command that is before the start date. If you specified either date field as a two-digit year, a sliding window technique was used to decide the century indicator which is used when comparing the start and end dates.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program. The sliding window technique is described in the readme file.

2684 LEVEL invalid on AUDIT command.

Explanation: You specified an invalid value for the LEVEL parameter in the AUDIT command. LEVEL must be 1, 2 or 3.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which AUDIT command produced the error. Correct the AUDIT command in the message command file, basein.msg, and retry the program.

2802 No destination specified on SEND command.

Explanation: You must specify a destination in the SEND command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2804 Multiple destinations on SEND command.

Explanation: You specified multiple destinations in a SEND command. You can only specify one destination.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2806 Incomplete destination on SEND command.

Explanation: You specified an incomplete destination in the SEND command. The destination must be an ACCOUNT and USERID; SYSID, ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2814 ALIAS invalid on SEND command.

Explanation: You specified an invalid value for the ALIAS parameter in the SEND command. The first character of the ALIAS is the destination table type and must be blank, g, o, or p. The last three characters of the ALIAS are the destination table ID and must be of the ALIAS must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2830 No FILEID specified on SEND command.

Explanation: You must specify a FILEID in the SEND command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2834 FILEID invalid on SEND command.

Explanation: You specified an invalid value for the FILEID parameter on the SEND command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2836 FORMAT invalid on SEND command.

Explanation: You specified an invalid value for the FORMAT parameter in the SEND command. FORMAT must be y or n. You cannot specify 'Y' for FORMAT with 'B' for DATATYPE or 'Y' for DELIMITED.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2844 PRIORITY invalid on SEND command.

Explanation: You specified an invalid value for the PRIORITY parameter in the SEND command. PRIORITY must be blank, i, or p.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2846 MODE invalid on SEND command.

Explanation: You specified an invalid value for the MODE parameter in the SEND command. MODE must be blank or t.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2848 CHARGE invalid on SEND command.

Explanation: You specified an invalid value for the CHARGE parameter in the SEND command. CHARGE must be a numeric character from 1 to 6.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2850 ACK invalid on SEND command.

Explanation: You specified an invalid value for the ACK parameter in the SEND command.

ACK must be blank, a, b, c, d, e, f, or r.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2872 VERIFY invalid on SEND command.

Explanation: You specified an invalid value for the VERIFY parameter in the SEND command. VERIFY must be y, n, or f.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2874 RETAIN invalid on SEND command.

Explanation: You specified an invalid value for the RETAIN parameter in the SEND command. RETAIN must be a numeric value from 0 to 180.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2876 DATATYPE invalid on SEND command.

Explanation: You specified an invalid value for the DATATYPE parameter in the SEND command. DATATYPE must be a or b.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2878 DELIMIT invalid on SEND command

Explanation: You specified an invalid value for the DELIMITED parameter in the SEND command. DELIMITED must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2894 RECFM invalid on SEND command.

Explanation: You specified an invalid value for the RECFM parameter in the SEND command. RECFM must be f or v.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2895 LRECL invalid on SEND command.

Explanation: You specified an invalid value for the LRECL parameter in the SEND command. LRECL must be 1 to 5 numeric characters, between 1 and 65535.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2898 COMPRESS invalid on SEND command.

Explanation: You specified an invalid value for the COMPRESS parameter in the SEND command. COMPRESS must be y, n, or t.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

2899 SELECTRCV invalid on SEND command.

Explanation: You specified an invalid value for the SELECTRCV parameter in the SEND command. SELECTRCV must be f, n, or blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SEND command produced the error. Correct the SEND command in the message command file, basein.msg, and retry the program.

3030 No FILEID specified on SENDEDI command.

Explanation: You must specify a FILEID in the SENDEDI command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3034 FILEID invalid on SENDEDI command.

Explanation: You specified an invalid value for the FILEID parameter on the SENDEDI command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3044 PRIORITY invalid on SENDEDI command.

Explanation: You specified an invalid value for the PRIORITY parameter in the SENDEDI command. PRIORITY must be blank, i, or p.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3046 MODE invalid on SENDEDI command.

Explanation: You specified an invalid value for the MODE parameter in the SENDEDI command. MODE must be blank or t.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3048 CHARGE invalid on SENDEDI command.

Explanation: You specified an invalid value for the CHARGE parameter in the SENDEDI command. CHARGE must be a numeric character from 1 to 6.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3050 ACK invalid on SENDEDI command.

Explanation: You specified an invalid value for the ACK parameter in the SENDEDI command. ACK must be blank, a, b, c, d, e, f, or r.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3072 VERIFY invalid on SENDEDI command.

Explanation: You specified an invalid value for the VERIFY parameter in the SENDEDI command. VERIFY must be y, c, f, g, or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3074 RETAIN invalid on SENDEDI command.

Explanation: You specified an invalid value for the RETAIN parameter in the SENDEDI command. RETAIN must be a numeric value from 0 to 180.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3094 RECFM invalid on SENDEDI command.

Explanation: You specified an invalid value for the RECFM parameter in the SENDEDI command. RECFM must be f or v.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3095 LRECL invalid on SENDEDI command.

Explanation: You specified an invalid value for the LRECL parameter in the SENDEDI command. LRECL must be 1 to 5 numeric characters, between 1 and 65535.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3098 COMPRESS invalid on SENDEDI command.

Explanation: You specified an invalid value for the COMPRESS parameter in the SENDEDI command. COMPRESS must be y, t, or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3099 SELECTRCV invalid on SENDEDI command.

Explanation: You specified an invalid value for the SELECTRCV parameter in the SENDEDI command. SELECTRCV must be f, n, or blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which SENDEDI command produced the error. Correct the SENDEDI command in the message command file, basein.msg, and retry the program.

3204 Multiple sources on RECEIVE command.

Explanation: You specified multiple sources in the RECEIVE command. You can specify only one source.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3206 Incomplete source specified on RECEIVE command.

Explanation: You specified an incomplete source for the RECEIVE command. If you specify a source, you must use ACCOUNT and USERID; SYSID, ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3214 ALIAS invalid on RECEIVE command.

Explanation: You specified an invalid value for the ALIAS parameter in the RECEIVE command. The first character of the ALIAS parameter is the destination table type, and it must be blank, g, o, or p. The last three characters of the ALIAS parameter are the destination table ID, and they must be 1 to 3 alphanumeric characters. If the first character is blank, the last three characters of the ALIAS parameter must also be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3230 No FILEID specified on RECEIVE command.

Explanation: You must specify a FILEID in the RECEIVE command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3234 FILEID invalid on RECEIVE command.

Explanation: You specified an invalid value for the FILEID parameter on the RECEIVE command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3236 FORMAT invalid on RECEIVE command.

Explanation: You specified an invalid value for the FORMAT parameter in the RECEIVE command. FORMAT must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3238 DELIMITED invalid on RECEIVE command.

Explanation: You specified an invalid value for the DELIMITED parameter in the RECEIVE command. DELIMITED must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3240 MULTFILES invalid on RECEIVE command.

Explanation: You specified an invalid value for the MULTFILES parameter on the RECEIVE command. MULTFILES must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3242 ALLFILES invalid on RECEIVE command.

Explanation: You specified an invalid value for the ALLFILES parameter in the RECEIVE command. ALLFILES must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3246 EDIOPT invalid on RECEIVE command.

Explanation: You specified an invalid value for the EDIOPT parameter in the RECEIVE command. EDIOPT must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3248 ORIGFILE invalid on RECEIVE command.

Explanation: You specified an invalid value for the ORIGFILE parameter on the RECEIVE command. ORIGFILE must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3250 PROCESSLEN invalid on RECEIVE command.

Explanation: You specified an invalid value for the PROCESSLEN parameter on the RECEIVE command. PROCESSLEN must be c, i, or r.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3252 RECORDSIZE invalid on RECEIVE command.

Explanation: You specified an invalid value for the RECORDSIZE parameter on the RECEIVE command. RECORDSIZE must be a numeric value from 1 to 999.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3254 REMOVEOF invalid on RECEIVE command.

Explanation: You specified an invalid value for the REMOVEOF parameter on the RECEIVE command. REMOVEOF must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3256 REQUEUED invalid on RECEIVE command.

Explanation: You specified an invalid value for the REQUEUED parameter in the RECEIVE command. REQUEUED must be y or n. If you specify a source, REQUEUED cannot be y.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3258 AUTOEDI invalid on RECEIVE command.

Explanation: You specified an invalid value for the AUTOEDI parameter in the RECEIVE command. AUTOEDI must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3266 MSGKEY invalid on RECEIVE command.

Explanation: You specified an invalid value for the MSGKEY parameter in the RECEIVE command. MSGKEY must be 20 hex characters.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3270 STARTDATE invalid on RECEIVE command.

Explanation: You specified an invalid value for the STARTDATE parameter in the RECEIVE command. The format must be YYMMDD, where YY is the last two digits of the year, MM is the two-digit month, and DD is the two digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3271 STARTTIME invalid on RECEIVE command.

Explanation: You specified an invalid value for the STARTTIME parameter in the RECEIVE command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3272 ENDDATE invalid on RECEIVE command.

Explanation: You specified an invalid value for the ENDDATE parameter in the RECEIVE command. The format must be YYMMDD, where YY is the last two-digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3273 ENDTIME invalid on RECEIVE command.

Explanation: You specified an invalid value for the ENDTTIME parameter in the RECEIVE command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3274 TIMEZONE invalid on RECEIVE command.

Explanation: You specified an invalid value for the TIMEZONE parameter in the RECEIVE command. TIMEZONE must be g or l.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3275 End date/time is before start date/time.

Explanation: You specified an end date and time in the RECEIVE command that is before the start date and time. If you specified either date field with a two-digit year, a sliding window technique was used to decide the century indicator which is used when comparing the start and end dates.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program. The sliding window technique is described in the readme file.

3284 NONEDIONLY invalid on RECEIVE command.

Explanation: You specified an invalid value for the NONEDIONLY parameter in the RECEIVE command. NONEDIONLY must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3285 WAIT invalid on RECEIVE command.

Explanation: You specified an invalid value for the WAIT parameter on the RECEIVE command. WAIT must be four numeric characters. The first two represent minutes from 02 to 05, and the last two represent seconds from 00 to 59. The total time specified cannot be more than 5 minutes. WAIT is not valid unless COMMTYPE is A or T on the TRANSMIT command in basein.pro.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVE command produced the error. Correct the RECEIVE command in the message command file, basein.msg, and retry the program.

3404 Multiple sources on RECEIVEEDI command.

Explanation: You specified multiple sources in the RECEIVEEDI command. You can specify only one source.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3406 Incomplete source on RECEIVEEDI command.

Explanation: You specified an incomplete source in the RECEIVEEDI command. If you specify a source, you must use ACCOUNT and USERID; SYSID, ACCOUNT and USERID; ALIAS and ALIASNAME; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3414 ALIAS invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the ALIAS parameter in the RECEIVEEDI command. The first character of the ALIAS parameter is the destination table type, and it must be blank, g, o, or p. The last three characters of the ALIAS parameter are the destination table ID, and they must be one to three alphanumeric characters. If the first character is blank, then the last three characters of the ALIAS must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3430 No FILEID specified on RECEIVEEDI command.

Explanation: You must specify a FILEID parameter in the RECEIVEEDI command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3434 FILEID invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the FILEID parameter on the RECEIVEEDI command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3440 MULTFILES invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the MULTFILES parameter meter on the RECEIVEEDI command. MULTFILES must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3442 ALLFILES invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the ALLFILES parameter meter in the RECEIVEEDI command. ALLFILES must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3446 EDIOPT invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the EDIOPT parameter in the RECEIVEEDI command. EDIOPT must be y, n or f.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3448 ORIGFILE invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the ORIGFILE parameter on the RECEIVEEDI command. ORIGFILE must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3452 RECORDSIZE invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the RECORDSIZE parameter on the RECEIVEEDI command. RECORDSIZE must be a value from 000 to 999.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3456 REQUEUED invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the REQUEUED parameter in the RECEIVEEDI command. REQUEUED must be y or n. If you specify a source, REQUEUED cannot be y.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3466 MSGKEY invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the MSGKEY parameter in the RECEIVEEDI command. MSGKEY must be 20 hex characters.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the message command file, basein.msg, and retry the program.

3470 STARTDATE invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the STARTDATE parameter in the RECEIVEEDI command. The format must be YYMMDD, where YY is the last two digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3471 STARTTIME invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the STARTTIME parameter in the RECEIVEEDI command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3472 ENDDATE invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the ENDDATE parameter in the RECEIVEEDI command. The format must be YYMMDD, where YY is the last two-digits of the year, MM is the two-digit month, and DD is the two-digit day or YYYYMMDD, where YYYY is a four-digit year.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3473 ENDTIME invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the ENDTTIME parameter in the RECEIVEEDI command. The format must be HHMMSS, where HH is the two-digit hour, MM is the two-digit minutes, and SS is the two-digit seconds.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3474 TIMEZONE invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the TIMEZONE parameter in the RECEIVEEDI command. TIMEZONE must be either g or l.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3475 End date/time is before start date/time.

Explanation: You specified an end date and time in the RECEIVEEDI command that is before the start date and time. If you specified either date field with a two-digit year, a sliding window technique was used to decide the century indicator which is used when comparing the start and end dates.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program. The sliding window technique is described in the readme.

3484 EDIONLY invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the EDIONLY parameter in the RECEIVEEDI command. EDIONLY must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3485 WAIT invalid on RECEIVEEDI command.

Explanation: You specified an invalid value for the WAIT parameter on the RECEIVEEDI command. WAIT must be four numeric characters. The first two represent minutes from 02 to 05, and the last two represent seconds from 00 to 59. The total time specified cannot be more than 5 minutes. WAIT is not valid unless COMMTYPE is A or T on the TRANSMIT command in basein.pro.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which RECEIVEEDI command produced the error. Correct the RECEIVEEDI command in the message command file, basein.msg, and retry the program.

3600 Requested a session start when already in session.

Explanation: You requested that Expedite Base do a session start when it was already in an Information Exchange session. If you use multiple START commands in a command file, you must end the previous Information Exchange session before you start the next one.

System Action: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which command produced the error. Correct the message command file, basein.msg, and retry the program.

3610 START command invalid with automatic session start.

Explanation: You specified the START command in the command file, but your profile, basein.pro, indicates that the system should start the session automatically. If the session starts automatically, the START command is invalid.

User Response: Specify 'N' for AUTOSTART on the TRANSMIT command in the profile command file, basein.pro, if you do not want the session started automatically. Otherwise, remove the START command from the message command file, basein.msg, and retry the program.

3620 END command invalid with automatic session end.

Explanation: You specified the END command in the command file, but your profile indicates that the session should end automatically. If the system ends the session automatically, the END command is invalid.

User Response: Specify 'N' for AUTOEND on the TRANSMIT command in the profile command file, basein.pro, if you do not want the session ended automatically. Otherwise, remove the END command from the message command file, basein.msg, and retry the program.

3630 Session not started before first command.

Explanation: You specified a command in the message command file, basein.msg, other than START, but were not currently in session.

User Response: Add a START command to the message command file, basein.msg, or specify 'Y' for AUTOSTART on the TRANSMIT command in the profile command file, basein.pro, and retry the program.

3640 No END command or automatic end for session.

Explanation: If you specify 'N' for AUTOEND on the TRANSMIT command, you must specify an END command in the command file.

User Response: Add an END command in the message command file, basein.msg, or specify 'Y' for AUTOEND on the TRANSMIT command in the profile command file, basein.pro. Retry the program.

3860 CDH invalid on QUERY command.

Explanation: You specified an invalid value for the CDH parameter in the QUERY command. CDH must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which QUERY command produced the error. Correct the QUERY command in the message command file, basein.msg, and retry the program.

4060 Missing ARCHIVEID on ARCHIVEMOVE command.

Explanation: You did not specify an ARCHIVEID value on the ARCHIVEMOVE command. This is a required parameter.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which ARCHIVEMOVE command produced the error. Correct the ARCHIVEMOVE command in the message command file, basein.msg, and retry the program.

4110 MSGKEY missing on PURGE command.

Explanation: You did not specify the MSGKEY parameter on the PURGE command. This is a required parameter.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PURGE command produced the error. Correct the PURGE command in the message command file, basein.msg, and retry the program. The message key can be found on the AVAILABLE record in response to a QUERY command.

4502 Missing destination on DEFINEALIAS command.

Explanation: You did not specify a destination on the DEFINEALIAS command. A destination consists of an ACCOUNT and USERID; SYSID, ACCOUNT, and USERID; or ALIAS and ALIASNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4514 Invalid table type on DEFINEALIAS command.

Explanation: You specified an invalid value for an ALIAS parameter in the DEFINEALIAS command. The first character of the ALIAS parameter is the destination table type, and it must be blank, g, o, or p. The last three characters of the ALIAS parameter are the destination table ID, and they must be 1 to 3 alphanumeric characters. If the first character is blank, then the last three characters of the ALIAS must be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4532 Invalid table name on DEFINEALIAS command.

Explanation: The ALIASTABLE parameter in the DEFINEALIAS command is missing, blank, or invalid. An ALIASTABLE name must be specified. The first character of the ALIASTABLE parameter is the destination table type, and it must be g, o, or p. The last three characters of the ALIAS parameter are the destination table ID, and they must be 1 to 3 alphanumeric characters.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4562 FUNCTION invalid on DEFINEALIAS command.

Explanation: The value for the FUNCTION parameter in the DEFINEALIAS command is invalid. The FUNCTION parameter value must be a, c, d, e, or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4564 Invalid use of FUNCTION parameter on DEFINEALIAS command.

Explanation: Either entries were specified in the DEFINEALIAS command with 'E' for FUNCTION, or no entries were specified for one of the other FUNCTION values.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4566 AUTHORITY invalid on DEFINEALIAS command.

Explanation: Use of the AUTHORITY parameter in the DEFINEALIAS command is invalid or you specified an invalid value for the AUTHORITY parameter. You may use the AUTHORITY parameter only if you are defining a new alias table, where FUNCTION is set to 'N'. Valid values for the AUTHORITY parameter are p, a, or g. If you are defining a private alias table or an organizational alias table, the first character of the alias table name is P or O, then you cannot specify 'G' for AUTHORITY.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4598 Invalid parameter duplication on DEFINEALIAS command.

Explanation: An ALIASTABLE, FUNCTION, or AUTHORITY parameter is duplicated. These parameters can be specified only once in each DEFINEALIAS command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4599 Incomplete destination on DEFINEALIAS command.

Explanation: A DEFINEALIAS command ended before a destination was completed, or a new type of destination started before the previous destination entry was completed. This is usually caused by a missing parameter somewhere in a destination of a DEFINEALIAS command. Destination entries are made up of either an ACCOUNT and USERID; SYSID, ACCOUNT, and USERID; or an ALIAS and ALIASNAME. Therefore, you must specify each destination component next to its counterpart. For example, an ACCOUNT parameter should be entered next to a USERID parameter. If you specify a SYSID parameter, you must specify it next to its associated ACCOUNT and USERID parameters. An ALIAS parameter must have an associated ALIASNAME parameter. A SYSID next to an ALIASNAME is invalid.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which DEFINEALIAS command produced the error. Correct the DEFINEALIAS command in the message command file, basein.msg, and retry the program.

4630 FILEID missing on PUTMEMBER command.

Explanation: You must specify a FILEID in the PUTMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4634 FILEID invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the FILEID parameter on the PUTMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4636 FORMAT invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the FORMAT parameter in the PUTMEMBER command. FORMAT must be y or n. You cannot specify 'Y' for FORMAT with 'B' for DATATYPE or 'Y' for DELIMITED.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4650 ACK invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the ACK parameter in the PUTMEMBER command. ACK must be blank or d.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4672 VERIFY invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the VERIFY parameter in the PUTMEMBER command. VERIFY must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4676 DATATYPE invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the DATATYPE parameter in the PUTMEMBER command. DATATYPE must be a or b.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4678 DELIMIT invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the DELIMITED parameter in the PUTMEMBER command. DELIMITED must be y or n.

User Response: Check the response file or response work file to determine which command produced the error. Correct the command file and retry the program.

4686 MEMBER missing on PUTMEMBER command.

Explanation: You must specify a MEMBER name on the PUTMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4688 LIBRARY missing on PUTMEMBER command.

Explanation: You must specify a LIBRARY name on the PUTMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4694 REPLACE invalid on PUTMEMBER command.

Explanation: You specified an invalid value for the REPLACE parameter in the PUTMEMBER command. REPLACE must be y or n.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which PUTMEMBER command produced the error. Correct the PUTMEMBER command in the message command file, basein.msg, and retry the program.

4704 Multiple destinations on GETMEMBER command.

Explanation: You specified multiple destinations in a GETMEMBER command. You can specify only one destination.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4706 Partial destinations on GETMEMBER command.

Explanation: The destination must be ALIAS and ALIASNAME; ACCOUNT and USERID; SYSID, ACCOUNT and USERID; or LISTNAME.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4714 Invalid ALIAS on GETMEMBER command.

Explanation: You specified an invalid value for an ALIAS parameter in the GETMEMBER command. The first character of the ALIAS parameter is the destination table type, and it must be blank, g, o, or p. The last three characters of the ALIAS parameter are the destination table ID, and they must be 1 to 3 alphanumeric characters. If the first character is blank, the last three characters of the ALIAS parameter must also be blank.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4748 CHARGE invalid on GETMEMBER command.

Explanation: You specified an invalid value for the CHARGE parameter in the GETMEMBER command. CHARGE must be one of the following: 1, 3, 5, or 6.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4750 ACK invalid on GETMEMBER command.

Explanation: You specified an invalid value for the ACK parameter in the GETMEMBER command. ACK must be blank, a, b, c, d, e, f, or r.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4774 RETAIN invalid on GETMEMBER command.

Explanation: You specified an invalid value for the RETAIN parameter in the GETMEMBER command. RETAIN must be a numeric value from 0 to 180.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4786 MEMBER name missing on GETMEMBER command.

Explanation: You must specify a MEMBER name on the GETMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4788 LIBRARY name missing on GETMEMBER command.

Explanation: You must specify a LIBRARY name on the GETMEMBER command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which GETMEMBER command produced the error. Correct the GETMEMBER command in the message command file, basein.msg, and retry the program.

4866 AUTHORITY invalid on LISTLIBRARIES command.

Explanation: You specified an invalid value for the AUTHORITY parameter in the LISTLIBRARIES command. AUTHORITY must be r or w.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LISTLIBRARIES command produced the error. Correct the LISTLIBRARIES command in the message command file, basein.msg, and retry the program.

4868 SELECTION invalid on LISTLIBRARIES command.

Explanation: You specified an invalid value for the SELECTION in the LISTLIBRARIES command. SELECTION must be a or c.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LISTLIBRARIES command produced the error. Correct the LISTLIBRARIES in the message command file, basein.msg, and retry the program.

4988 LIBRARY name missing on LISTMEMBERS command.

Explanation: You must specify a LIBRARY name on the LISTMEMBERS command.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which LISTMEMBERS command produced the error. Correct the LISTMEMBERS command in the message command file, basein.msg, and retry the program.

Profile command file syntax errors

This section describes the return codes for profile command file syntax errors.

5018 INACCOUNT missing on IDENTIFY command.

Explanation: You must specify a network account in your profile.

User Response: Specify the account using the INACCOUNT parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5020 INUSERID missing on IDENTIFY command.

Explanation: You must specify a network user ID in your profile.

User Response: Specify the user ID using the INUSERID parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5022 INPASSWORD missing on IDENTIFY command.

Explanation: You must specify a network password in your profile.

User Response: Specify the password using the INPASSWORD parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5024 IEACCOUNT missing on IDENTIFY command.

Explanation: You must specify an Information Exchange account in your profile when Expedite Base starts the session automatically.

User Response: Specify the account using the IEACCOUNT parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5026 IEUSERID missing on IDENTIFY command.

Explanation: You must specify an Information Exchange user ID in your profile when Expedite Base starts the session automatically.

User Response: Specify the user ID using the IEUSERID parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5028 IEPASSWORD missing on IDENTIFY command.

Explanation: You must specify an Information Exchange password in your profile when Expedite Base starts the session automatically.

User Response: Specify the password using the IEPASSWORD parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5029 Both KEYRINGSTASHFILE and KEYRINGPASSWORD parameters specified.

Explanation: You specified both the KEYRINGSTASHFILE and the KEYRINGPASSWORD parameters on the IDENTIFY command. This is not allowed.

User Response: When the file that you specified in the KEYRINGFILE parameter was created, it was created to have either a password or to have the password stored in a stash file. If the password was stored in a stash file, then specify the name of the stash file in the KEYRINGSTASHFILE parameter. Otherwise, specify the password in the KEYRINGPASSWORD parameter. Correct the IDENTIFY command by specifying only the KEYRINGSTASHFILE parameter or the KEYRINGPASSWORD parameter, and then retry the program.

5030 KEYRINGFILE must be specified.

Explanation: The KEYRINGFILE name was not specified. This is a required parameter.

User Response: When making an SSL connection, there must be a KDB file that contains the certificate that allows the connection. This filename is specified in the KEYRINGFILE parameter and is a required parameter to enable SSL. Correct the IDENTIFY command by specifying the filename in the KEYRINGFILE parameter, and then retry the program.

5031 KEYRINGFILE parameter invalid.

Explanation: You cannot specify a value in the KEYRINGFILE parameter on the IDENTIFY command when the AUTOSTART parameter is set to Y on the TRANSMIT command.

User Response: Remove the following parameters from the IDENTIFY command if they exist.

- KEYRINGFILE
- KEYRINGPASSWORD
- KEYRINGSTASHFILE

As an alternative, you can set the AUTOSTART parameter to Y on the TRANSMIT command and remove the START command in the basein.pro file, and then retry the program.

5032 No password provided for KEYRINGFILE.

Explanation: The key database or key ring specified in the KEYRINGFILE parameter requires a password.

User Response: Provide either the correct KEYRINGPASSWORD parameter or KEYRINGSTASHFILE associated with the key database specified in the KEYRINGFILE parameter and retry the program.

5072 TIMEZONE invalid on IDENTIFY command.

Explanation: You specified an invalid value for the TIMEZONE parameter in the IDENTIFY command. TIMEZONE must be 1 to 5 alphanumeric characters. Valid values are ast, ahd, ahs, bst, cdt, cst, ead, edt, emt, est, gmt, jst, mdt, mst, pdt, pst, wed, wes, ydt, and yst. You can also specify the time zone as hours and minutes east or west of Greenwich Mean Time, for example, w0400 or e0400.

User Response: Correct the TIMEZONE parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5074 ENCRYPT invalid on IDENTIFY command.

Explanation: You specified an invalid value for the ENCRYPT parameter in the IDENTIFY command. ENCRYPT must be y or n.

User Response: Correct the ENCRYPT parameter in the IDENTIFY command in the profile command file, basein.pro. Retry the program.

5272 No phone number found in profile, or dialcount is 0.

Explanation: You are attempting to use asynchronous communication and you either didn't specify a PHONEn parameter on the DIAL command or all DIALCOUNT parameters are set to 0.

User Response: Specify a phone number on the DIAL command in the profile command file, basein.pro, and verify that at least one phone number has a DIALCOUNT greater than 0. Retry the program.

5274 DIALCOUNTn invalid on DIAL command.

Explanation: You specified an invalid DIALCOUNTn parameter on the DIAL command. DIALCOUNTn must be a single numeric character from 0 to 9.

User Response: Correct the DIALCOUNTn parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5276 BAUDRATEn invalid on DIAL command.

Explanation: You specified an invalid BAUDRATEn parameter on the DIAL command. BAUDRATEn must be 300, 1200, 2400, 4800, 9600, 19200, or 38400.

User Response: Correct the BAUDRATEn parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5278 ACCESS invalid on DIAL command.

Explanation: You specified an invalid ACCESS parameter on the DIAL command. ACCESS must be d or s.

User Response: Correct the ACCESS parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5280 No DEVICEx specified on DIAL command.

Explanation: You must specify a DEVICEx parameter on the DIAL command.

User Response: Specify a DEVICEx parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5282 DBAUDRATEx invalid on DIAL command.

Explanation: You specified an invalid DBAUDRATEx parameter on the DIAL command. DBAUDRATEx must be 300, 1200, 2400, 4800, 9600, 19200, or 38400.

User Response: Correct the DBAUDRATEx parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5284 CYCLE invalid on DIAL command.

Explanation: You specified an invalid CYCLE parameter on the DIAL command. CYCLE must be one numeric character from 0 to 9.

User Response: Correct the CYCLE parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5286 WAIT invalid on DIAL command.

Explanation: You specified an invalid WAIT parameter on the DIAL command. The WAIT parameter has the format HHMM where HH is number of hours and MM is number of minutes. You must specify four numeric characters from 0000 to 9959.

User Response: Correct the WAIT parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5288 PHONETYPE invalid on DIAL command.

Explanation: You specified an invalid PHONETYPE parameter on the DIAL command. PHONETYPE must be t or p.

User Response: Correct the PHONETYPE parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5290 MODEMTYPE M invalid on DIAL command.

Explanation: You specified an invalid MODEMTYPE parameter on the DIAL command. MODEMTYPE cannot be M.

User Response: Correct the MODEMTYPE parameter on the DIAL command in the profile command file, basein.pro. Retry the program.

5292 Modem script conflict, MODEMTYPE and INITSCR/RESETSCR specified.

Explanation: You have indicated the use of both old style scripts and new style scripts. You have specified a MODEMTYPE parameter on the DIAL command which indicates the use of old style modem scripts. In addition, you have specified INITSCR or RESETSCR. If you want to use old style scripts, you cannot also use initialization or reset scripts.

User Response: If you wish to use the old style modem scripts, do not specify INITSCR or RESETSCR. If you wish to use initialization or reset scripts, specify the MODEMTYPE parameter with a blank value on the DIAL command in the profile command file, basein.pro. Retry the program.

5293 Modem script conflict, MODEMTYPE and CNNCTSCR/DISCNNCTSCR specified.

Explanation: You have indicated the use of both old style scripts and new style scripts. You have specified a MODEMTYPE parameter on the DIAL command which indicates the use of old style modem scripts. In addition, you have specified CNNCTSCR or DISCNNCTSCR. If you want to use old style scripts, you cannot use CNNCTSCR or DISCNNCTSCR.

User Response: If you wish to use the old style modem scripts, do not specify CNNCTSCR or DISCNNCTSCR. If you wish to use CNNCTSCR and DISCNNCTSCR, specify the MODEMTYPE parameter with a blank value on the DIAL command in the profile command file, basein.pro. Retry the program.

5294 Modem script conflict, new scripts indicated, ACCESS specified.

Explanation: You have indicated the use of both old style scripts and new style scripts. You have specified an ACCESSn parameter on the DIAL command which indicates the use of old style modem scripts. In addition, you have specified one or more of CNNCTSCR, DISCNNCTSCR, INITSCR, or RESETSCR.

User Response: If you wish to use the old style modem scripts, do not specify any of CNNCTSCR, DISCNNCTSCR, INITSCR, or RESETSCR. If you wish to use these scripts, specify the ACCESS parameter with a blank value on the DIAL command in the profile command file, basein.pro. Retry the program.

5295 Modem script conflict, MODEMTYPE not blank and no old scripts.

Explanation: You have indicated the use of old style modem scripts by using the MODEMTYPE parameter on the DIAL command. However, Expedite Base could not find the associated modem script file.

User Response: If you wish to use the old style modem scripts, verify that the old style modem script file exists in the directory where Expedite Base is installed. If Expedite Base is not installed in the default installation directory, then make sure you specified the directory where it is installed on the IEPATH parameter on the SESSION command, or that you specified a symbolic link from the default directory to the installation directory. The old style modem script filenames have the format xcnnct.fil and xdcnnct.fil where x is the character specified in the MODEMTYPE parameter. If you do not wish to use the old style modem scripts, then specify the MODEMTYPE parameter with a blank value on the DIAL command in basein.pro. The MODEMTYPE may be stored in the internal profile, IEBASE.PRO, from a previous release of Expedite Base. Specifying a blank value for the MODEMTYPE parameter on the DIAL command will remove the old value. Make sure you erase or rename the old modem control files. Retry the program.

5296 DCLVERSION invalid on DIAL or LANDIAL command.

Explanation: You specified an invalid DCLVERSION parameter on the DIAL or LANDIAL command. DCLVERSION must be one numeric character, either a 1 to use old DCL block size, or a 2 to use the new DCL block size.

User Response: Correct the DCLVERSION parameter on the DIAL or LANDIAL command in the profile command file, basein.pro. Retry the program.

5472 EXITKEY invalid on SESSION command.

Explanation: You specified an invalid EXITKEY parameter on the SESSION command. EXITKEY must be a numeric character from 2 to 10.

User Response: Correct the EXITKEY parameter on the SESSION command in the profile command file, basein.pro. Retry the program.

5474 PICTURE invalid on SESSION command.

Explanation: You specified an invalid PICTURE parameter on the SESSION command. PICTURE must be y or n.

User Response: Correct the PICTURE parameter on the SESSION command in the profile command file, basein.pro. Retry the program.

5476 STATUS invalid on SESSION command.

Explanation: You specified an invalid value for the STATUS parameter on the SESSION command. STATUS must be y or n.

User Response: Correct the STATUS parameter in the SESSION command in the profile command file, basein.pro. Retry the program.

5478 IEPATH invalid on SESSION command.

Explanation: You specified an invalid IEPATH parameter on the SESSION command. IEPATH must specify a valid path up to 128 characters in length.

User Response: Correct the IEPATH parameter on the SESSION command in the profile command file, basein.pro. Retry the program.

5480 OVERWRITE invalid on SESSION command.

Explanation: You specified an invalid value for the OVERWRITE parameter on the SESSION command. OVERWRITE must be y or n.

User Response: Correct the OVERWRITE parameter in the SESSION command in the profile command file, basein.pro. Retry the program.

5672 CNNCT invalid on TRACE command.

Explanation: You specified an invalid value for the CNNCT parameter on the TRACE command. CNNCT must be y or n.

User Response: Correct the CNNCT parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5674 DISPLAY invalid on TRACE command.

Explanation: You specified an invalid value for the DISPLAY parameter in the TRACE command. DISPLAY must be y or n.

User Response: Correct the DISPLAY parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5676 MODEM invalid on TRACE command.

Explanation: You specified an invalid value for the MODEM parameter in the TRACE command. MODEM must be y or n.

User Response: Correct the MODEM parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5678 PROTOCOL invalid on TRACE command.

Explanation: You specified an invalid value for the PROTOCOL parameter in the TRACE command. PROTOCOL must be y or n.

User Response: Correct the PROTOCOL parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5680 LINK invalid on TRACE command.

Explanation: You specified an invalid value for the LINK parameter in the TRACE command. LINK must be y or n.

User Response: Correct the LINK parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5682 BASE invalid on TRACE command.

Explanation: You specified an invalid value for the BASE parameter in the TRACE command. BASE must be y or n.

User Response: Correct the BASE parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5684 IOFILE invalid on TRACE command.

Explanation: You specified an invalid value for the IOFILE parameter in the TRACE command. IOFILE must be y or n.

User Response: Correct the IOFILE parameter in the TRACE command in the profile command file, basein.pro. Retry the program.

5872 AUTOSTART invalid on TRANSMIT command.

Explanation: You specified an invalid value for the AUTOSTART parameter in the TRANSMIT command. AUTOSTART must be y or n.

User Response: Correct the AUTOSTART parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5874 RECONNECT invalid on TRANSMIT command.

Explanation: You specified an invalid value for the RECONNECT parameter in the TRANSMIT command. RECONNECT must be one numeric character from 0 to 9.

User Response: Correct the RECONNECT parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5876 AUTOEND invalid on TRANSMIT command.

Explanation: You specified an invalid value for the AUTOEND parameter in the TRANSMIT command. AUTOEND must be y or n.

User Response: Correct the AUTOEND parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5878 COMMITDATA invalid on TRANSMIT command.

Explanation: You specified an invalid value for the COMMITDATA parameter in the TRANSMIT command. COMMITDATA must be a numeric value from 1000 to 37000, or 1000 to 9999999 for the RISC System/6000. COMMITDATA must be greater than or equal to the MSGSIZE parameter value.

User Response: Correct the COMMITDATA parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5880 BLOCKSIZE invalid on TRANSMIT command.

Explanation: You specified an invalid value for the BLOCKSIZE parameter in the TRANSMIT command. BLOCKSIZE must be 256 to 3500 for U.S. asynchronous communication, and 256 to 1024 for worldwide asynchronous communication.

User Response: Correct the BLOCKSIZE parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5882 COMMTYPE invalid on TRANSMIT command.

Explanation: You specified an invalid value for the COMMTYPE parameter in the TRANSMIT command. COMMTYPE must be a, s, t, or w.

User Response: Correct the COMMTYPE parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5884 MAXMSGS invalid on TRANSMIT command.

Explanation: You specified an invalid value for the MAXMSGS parameter in the TRANSMIT command. MAXMSGS must be a numeric value from 1 to 10.

User Response: Correct the MAXMSGS parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5886 DELAYTIME invalid on TRANSMIT command.

Explanation: You specified an invalid value for the DELAYTIME parameter in the TRANSMIT command. DELAYTIME must be 6 numeric characters with the format HHMMSS where HH is hours, MM is minutes, SS is seconds.

User Response: Correct the DELAYTIME parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5888 DELAYDATE invalid on TRANSMIT command.

Explanation: You specified an invalid value for the DELAYDATE parameter meter in the TRANSMIT command. DELAYDATE must be 6 numeric characters with the format YYMMDD where YY is year, MM is month, DD is day.

User Response: Correct the DELAYDATE parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5892 RECOVERY invalid on TRANSMIT command

Explanation: You specified an invalid value for the RECOVERY parameter in the TRANSMIT command. RECOVERY must be c, f, u, or s.

User Response: Correct the RECOVERY parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

5894 Session-level recovery not valid with session in progress.

Explanation: You specified session-level recovery, but the session file indicates that a check-point session is in progress.

User Response: Continue the present session using previous recovery level. If you want to use session-level recovery without completing the present session, either specify the r command line parameter with the IEBASE command, or remove the session file, session.fil.

ATTENTION: If you reset the session using the r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

5896 MSGSIZE invalid on TRANSMIT command.

Explanation: You specified an invalid value for the MSGSIZE parameter in the TRANSMIT command. MSGSIZE must be a numeric value from 1000 to 37000, or 1000 to 9999999 for the RISC System/6000.

User Response: Correct the MSGSIZE parameter in the TRANSMIT command in the profile command file, basein.pro. Retry the program.

6020 A phone number was specified with S for COMMTYPE on TRANSMIT command.

Explanation: A phone number cannot be specified with SNA LU 6.2 communications.

User Response: If you want to use SNA communications over a leased line, remove the DIAL command from basein.pro and erase iebase.pro. If you want to use dial access, change the COMMTYPE parameter on the TRANSMIT command to either a or w.

6200 ENABLESSL invalid on SSL command.

Explanation: You specified an invalid value for the ENABLESSL parameter in the SSL command. ENABLESSL must be a y or n.

User Response: Correct the ENABLESSL parameter, and then retry the program.

6201 CIPHERSUITES invalid on SSL command.

Explanation: You specified an invalid value for the CIPHERSUITES parameter in the SSL command. The value of CIPHERSUITES is a string consisting of 1 or more 2-character values. The valid characters are 0 through 9 or a through f.

The values for CIPHERSUITES are determined when Expedite Base with SSL support is installed on your system with an established default value. You should never change this value unless requested to do so by GXS Community Support.

User Response: Correct the CIPHERSUITES parameter in the SSL command in the profile command file, basein.pro, and then retry the program.

6202 SSLVERSION invalid on SSL command.

Explanation: You specified an invalid value for the SSLVERSION parameter in the SSL command. The valid values for SSLVERSION are 3031, 30**,** or 31.

The 3031 value is the default value for this parameter. You should never change this value unless requested to do so by GXS Community Support.

User Response: In the profile command file, inpro, correct the SSLVERSION parameter in the SSL command, and then retry the program.

6204 SSL only valid with TCP/IP communication types.

Explanation: You specified Y for the ENABLESSL parameter, but the communication type that is specified is not TCP/IP.

User Response: Set your communication type to TCP/IP leased line or dial, and then retry the program.

Network errors

This section describes the return codes for network errors.

11801 Your modem did not respond to any commands issued by Expedite Base.

Explanation: In attempting to connect to the network, Expedite Base issued commands to your modem but did not receive any response.

User Response: Use the following checklist to identify the problem.

- Check to see that the baud rate and the communications device specified in the profile command file, basein.pro, match that of your modem.
- If using an external modem, check to see that the modem is turned on and the modem cable is securely connected.
- Check your modem manual to verify that the modem configuration is set to be Hayescompatible.
- If using an external modem, test with a different modem cable.
- Test with another modem.
- Make sure you specified the correct fully qualified path of the device you are using on the DIAL command in the profile command file.

11802 Expedite Base was unable to establish a successful phone connection.

Explanation: Your modem responded to commands issued by Expedite Base but was unable to establish a successful telephone connection.

User Response: The modem script files that are included with Expedite Base expect to receive a response containing CONNECT, for example, CONNECT 2400, from the modem. If this connect response is not received from the modem then the return code set in the modem command file is 12130, which signals Expedite Base to redial. If the redial count has reached the maximum specified by the DIALCOUNTn parameters and the CYCLE parameter on the DIAL command, then Expedite Base will exit with a 11802 return code. Use the following checklist to identify the problem.

- If you did not hear the modem dial, make sure the device is disabled and ready for dial out.
- If you heard the modem dial but DID NOT hear a dial tone:
  - Make sure the phone cable is securely connected to the wall jack.
  - Make sure the phone cable is securely connected to the line jack in the back of the modem.
  - Verify that your phone line is active. You can do this by trying to dial out on this line with a standard telephone.
  - If you heard the modem dial and DID hear a dial tone:
  - Try an alternate phone number.
  - Verify that your modem responds with a CONNECT xxxx message when a successful connection is made, where xxxx is the baudrate of the connection. You can verify this by looking at the file CNNCT.LOG.
  - Test with another modem.

11803 Expedite Base was unable to log on to the IBM Global Network.

Explanation: Expedite Base established a successful telephone connection, but could not log on to the network.

User Response: Use the following checklist to identify the problem.

- Make sure the BAUDRATEN parameter in basein.pro is at least equal to the data rate of the connection. If the specified BAUDRATEN value is sufficient, check your modem initialization string, MODEMINIT parameter.
- Verify that the supported data rate for the phone number you are dialing is not less than the BAUDRATEN parameter specified in basein.pro.
- Try an alternate phone number.
- If your modem has an error-correcting protocol such as MNP, and you are using a network communications gateway, make sure the error-correcting protocol is enabled. You can use the MODEMINIT parameter of the DIAL command in the profile file, basein.pro, to issue the modem commands necessary to enable error correction.
- To connect to Information Exchange using Expedite Base, you must set up your modem to use hardware flow control and a fixed serial port rate. You can use error correction and data compression if the number you are dialing supports it. Otherwise, use normal mode. Error correction only enhances performance if the data you are sending is not already compressed or encrypted.
- Test with another modem.

11804 Mode Set error received during logon process.

Explanation: While attempting to log on to the network, Expedite Base received a mode set error.

User Response: Use the following checklist to identify the problem.

- Retry the transmission.
- Attempt to transmit using an alternate phone number.
- If you are dialing a network communications gateway, be sure to specify 'A' for COMMTYPE on the TRANSMIT command in basein.pro.
- If the problem persists, contact the Help Desk.

11811 Invalid request.

Explanation: There was an error attempting to log on to the network.

User Response: Contact the Help Desk.

11821 Account number missing or invalid.

Explanation: The network did not recognize the INACCOUNT that you specified in your profile.

User Response: Correct the INACCOUNT parameter on the IDENTIFY command in basein.pro and retry the program. If the problem persists, contact the Help Desk.

11822 Invalid project number.

Explanation: There was an error attempting to log on to the network.

User Response: Contact the Help Desk.

11823 User ID missing or invalid.

Explanation: The network did not recognize the INUSERID that you specified in your profile.

User Response: Correct the INUSERID parameter on the IDENTIFY command in basein.pro and retry the program. If the problem persists, contact the Help Desk.

11824 Password missing or invalid.

Explanation: The password specified in the INPASSWORD parameter is not correct.

User Response: Correct the INPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program. If you just changed your password, be sure to update the IDENTIFY command in basein.pro to reflect the new password. Also, if you specify 'y' for encrypt, make sure the password is encrypted properly.

11825 Unexpected error processing new password.

Explanation: Expedite Base encountered an unexpected error while processing the password.

User Response: Contact the Help Desk.

11826 Product missing or invalid.

Explanation: The product specified by the PRODUCT parameter is not correct.

User Response: Correct the PRODUCT parameter of the IDENTIFY command in basein.pro and retry the program. If the problem persists, contact the Help Desk.

11827 Verify password invalid.

Explanation: There was an error attempting to log on to the network.

User Response: Contact the Help Desk.

11828 New/verify passwords not equal.

Explanation: There was an error attempting to log on to the network.

User Response: Contact the Help Desk.

11831 Profile not found.

Explanation: The network was not able to find a network profile for the INACCOUNT and INUSERID specified.

User Response: Make sure the INACCOUNT and INUSERID fields in basein.pro are correct and retry the program. If the problem persists, contact the Help Desk.

11836 Product not on profile.

Explanation: The product specified by the PRODUCT parameter is not on your network profile.

User Response: Make sure the PRODUCT parameter of the IDENTIFY command in basein.pro is correct and retry the program. If the problem persists, contact the Help Desk.

11841 User ID already logged on.

Explanation: There are too many session logons with the same user ID.

User Response: Make sure that you end at least one other session for this user ID. If the problem persists, contact the Help Desk.

11842 User ID not logged on.

Explanation: There was an error attempting to log on to the network.

User Response: User Response: Contact the Help Desk.

11851 User ID not defined to RACF.

Explanation: There was an error attempting to log on to the network.

User Response: Contact the Help Desk.

11852 Password not authorized.

Explanation: The password specified in the INPASSWORD parameter is not correct.

User Response: Correct the INPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program. If you just changed your password, be sure to update the IDENTIFY command in basein.pro to reflect the new password. Also, if you specified 'y' for encrypt, make sure the password is encrypted properly. If the problem persists, contact the Help Desk.

11853 Password expired.

Explanation: Your INPASSWORD has expired and you must change it.

User Response: Enter a new password in the NINPASSWORD parameter of the IDENTIFY command in basein.pro. Retry the program. Be sure to update basein.pro once the password has been changed.

11854 New password invalid.

Explanation: The new password specified in the NINPASSWORD parameter is invalid. The password you specified may violate password rules or may be the same as one of your previous three passwords.

User Response: Enter a valid password in the NINPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program. If you just changed your password, be sure to update the IDENTIFY command in basein.pro to reflect the new password. If you specify 'y' for encrypt, make sure the password is encrypted properly. If the problem persists, contact the Help Desk.

11855 User access revoked.

Explanation: Your user ID access was revoked.

User Response: Contact the Help Desk.

11856 Password must be extended.

Explanation: Your password must be extended but it does not contain any extended password characters. The following characters: & ! : " . ? and the right and left parentheses are valid extended password characters. At least one of these characters must be in your network password to make it extended.

User Response: Enter a valid password in the INPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program.

11857 New password must be extended.

Explanation: Your new password must be extended, but it does not contain any extended password characters. The following characters: & ! : " . ? and the right and left parentheses are valid extended password characters. At least one of these characters must be in your network password to make it extended.

User Response: Enter a valid password in the NINPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program.

11858 Dial access authorization failed.

Explanation: Your user ID is not defined to have dial access to the network.

User Response: Contact the Help Desk.

11859 Password contains an invalid character.

Explanation: You specified an invalid character in the INPASSWORD parameter. See “Using accounts, user IDs, and passwords” on page 2 for network password rules.

User Response: Enter valid password in the INPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program. If you specify 'Y' for encrypt, make sure the password is encrypted properly. If the problem persists, contact the Help Desk.

11860 New password contains an invalid character.

Explanation: You specified an invalid character in the NINPASSWORD parameter. See the section “Using account IDs, user IDs, and passwords” in Chapter 1, “Understanding Expedite Base,” for network password rules.

User Response: Enter a valid password in the NINPASSWORD parameter of the IDENTIFY command in basein.pro and retry the program. If you specify 'Y' for ENCRYPT, make sure the password is encrypted properly. If the problem persists, contact the Help Desk.

11862 Product access denied.

Explanation: You do not have access to the product you specified.

User Response: Make sure the PRODUCT parameter of the IDENTIFY command in basein.pro is correct and retry the program. If the problem persists, contact the Help Desk.

11863 Product not available.

Explanation: You may not start a session with Information Exchange using Expedite Base at this time.

User Response: Wait and retry the program later. If the problem persists, contact the Help Desk.

11864 Account/userid invalid for this terminal.

Explanation: You cannot use this phone number with your account and user ID.

User Response: Contact the Help Desk to get a valid phone number.

11865 Invalid new password entered.

Explanation: The NINPASSWORD you specified is invalid. It is a reserved word and is not allowable as a network password.

User Response: Enter another password in the NINPASSWORD parameter of the IDENTIFY command in basein.pro and retry. If the problem persists, contact the Help Desk.

11866 Too many logons with same user ID.

Explanation: There are too many session logons with the same user ID.

User Response: Make sure that you end at least one other session for this user ID. If the problem persists, contact the Help Desk.

11867 User has reached logon maximum.

Explanation: There are too many session logons with the same user ID.

User Response: Make sure that you end at least one other session for this user ID. If the problem persists, contact the Help Desk.

11868 Error logging on to IBM Global Network.

Explanation: There was an error attempting to log on to the network.

User Response: Try the program again. If the problem persists, contact the Help Desk

11869 Product access denied from this connection type.

Explanation: You cannot use the specified product with a dial connection.

User Response: Make sure the PRODUCT parameter of the IDENTIFY command in basein.pro is correct and retry the program. If the problem persists, contact the Help Desk.

11870 Incomplete user ID definition.

Explanation: There was an error attempting to log on to the network.

User Response: Try the program again. If the problem persists, contact the Help Desk.

11871 Error logging on to IBM Global Network.

Explanation: There was an error attempting to log on to the network.

User Response: Try the program again. If the problem persists, contact the Help Desk.

11997 Batch logon response invalid for worldwide async.

Explanation: There was an error attempting to log on to the network.

User Response: Try the program again. If the problem persists, contact the Help Desk.

11998 Batch logon response invalid.

Explanation: There was an error attempting to log on to the network.

User Response: Make sure you have specified the correct product name in the PRODUCT parameter on the DIAL command. If you omit the PRODUCT parameter or leave it blank, it defaults to INFOEXCH. Do not change the default unless instructed to do so by network personnel.

11999 Batch logon request failed.

Explanation: There was an error attempting to log on to the network.

User Response: Try the program again. If the problem persists, contact the Help Desk.

Modem script syntax errors

This section describes the return codes for modem script syntax errors.

12010 No return code set for the command file.

Explanation: There was no return code set while processing the modem command file. Therefore, Expedite Base does not know whether the dial procedure was successful.

User Response: Modify the modem command file to make sure that the logic in the command allows the RETURN command to be processed or add the RETURN command to the modem command if it is missing. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'Y' for CNNCT and 'Y' for MODEM on the TRACE command in basein.pro to turn on trace for the modem command processing. Correct the modem command file and retry the program. See Chapter 10, “Using modem control files and secondary network control files,” for more information.

12011 CODE missing on RETURN command in modem script file.

Explanation: The CODE parameter in the RETURN command is missing or blank. A CODE must be specified.

User Response: Correct the RETURN command and run the program again.

12012 Invalid parameter on GETVALUE command in modem script file.

Explanation: The variable named in the INTO parameter on the GETVALUE command is not recognized.

User Response: You specified an invalid variable name for the INTO parameter on the GETVALUE command. The supported variable is CNNCTBAUD. Correct the GETVALUE command and run the program again.

12013 BAUD invalid on OPENPORT command in modem script file.

Explanation: You specified an invalid value for the BAUD parameter in the OPENPORT command.

User Response: Correct the OPENPORT command and run the program again.

12014 BITS invalid on OPENPORT command in modem script file.

Explanation: You specified an invalid value for the BITS parameter in the OPENPORT command.

User Response: Correct the OPENPORT command and run the program again.

12015 CR invalid on SAY command in modem script file.

Explanation: You specified an invalid value for the CR parameter in the SAY command.

User Response: Correct the SAY command and run the program again.

12016 CODE too large on RETURN command in modem script file.

Explanation: The CODE parameter in the RETURN command is too large.

User Response: Correct the RETURN command in the modem script file and run the program again.

12017 STRING missing on SAY command in modem script file.

Explanation: The STRING parameter in the SAY command is missing or blank. A STRING must be specified if CR is not specified on the SAY command.

User Response: Correct the SAY command in the modem script file and run the program again.

12018 PACED invalid on SAY command in modem script file.

Explanation: You specified an invalid value for the PACED parameter in the SAY command.

User Response: Correct the SAY command and run the program again.

12019 TIMEOUT invalid on GETANSWER command in modem script file.

Explanation: You specified an invalid value for the TIMEOUT parameter in the GETANSWER command.

User Response: Correct the GETANSWER command and run the program again.

12020 Line number too long in modem script file.

Explanation: A line number in the old style modem command file is too long. The maximum value of a line number is 998.

User Response: Correct the file and retry the program.

12021 MODE invalid on GETANSWER command in modem script file.

Explanation: You specified an invalid value for the MODE parameter in the GETANSWER command.

User Response: Correct the GETANSWER command and run the program again.

12022 REPEAT invalid on IFANSWER command in modem script file.

Explanation: You specified an invalid value for the REPEAT parameter in the IFANSWER command.

User Response: Correct the IFANSWER command and run the program again.

12024 GOTO invalid on IFANSWER command in modem script file.

Explanation: You specified an invalid value for the GOTO parameter in the IFANSWER command.

User Response: Correct the IFANSWER command and run the program again.

12025 TO missing on GO command in modem script file.

Explanation: The TO parameter in the GO command is missing or blank. A TO parameter must be specified.

User Response: Correct the GO command in the modem script file and run the program again.

12026 OF missing on IFVALUE command in modem script file.

The OF parameter in the IFVALUE command is missing or blank. An OF parameter must be specified.

User Response: Correct the IFVALUE command in the modem script file and run the program again.

12027 IS missing on IFVALUE command in modem script file.

Explanation: The IS parameter in the IFVALUE command is missing or blank. An IS parameter must be specified.

User Response: Correct the IFVALUE command in the modem script file and run the program again.

12028 TO missing on IFVALUE command in modem script file.

Explanation: The TO parameter in the IFVALUE command is missing or blank. A TO parameter must be specified.

User Response: Correct the IFVALUE command in the modem script file and run the program again.

12029 GOTO missing on IFVALUE command in modem script file.

Explanation: The GOTO parameter in the IFVALUE command is missing or blank. A GOTO parameter must be specified.

User Response: Correct the IFVALUE command in the modem script file and run the program again.

12030 Non-numeric character in line number in command file.

Explanation: A line number in the old style modem command file contains non-numeric characters.

User Response: Correct the file and retry the program.

12031 OF invalid on IFVALUE command in modem script file.

Explanation: You specified an invalid value for the OF parameter in the IFVALUE command.

User Response: Correct the IFVALUE command and run the program again.

12032 Modem script file not found.

Explanation: Expedite Base could not find the script file to process.

User Response: Verify that the script file exists in the install directory or the directory specified in the IEPATH parameter on the SESSION command in the profile command file, basein.pro.

12033 Duplicate label in modem script file.

Explanation: The script file had the same label specified more than once.

User Response: Correct the label names so that each is unique. Retry the program.

12034 Label undefined in modem script file.

Explanation: A command in the script file referenced a label that doesn't exist.

User Response: Verify that all referenced label names in the script file exist and retry the program.

12035 TYPE invalid on SETWWASYNC command in modem script file.

Explanation: You specified an invalid value for the TYPE parameter in the SETWWASYNC command.

User Response: Correct the SETWWASYNC command and run the program again.

12036 IS invalid on IFANSWER command in modem script file.

Explanation: You specified an invalid value for the IS parameter in the IFANSWER command.

User Response: Correct the IFANSWER command and run the program again.

12037 GOTO missing on IFANSWER command in modem script file.

Explanation: The GOTO parameter in the IFANSWER command is missing or blank. A GOTO parameter must be specified.

User Response: Correct the IFANSWER command and run the program again.

12038 Invalid hex string specified in modem script file.

Explanation: You specified an invalid hex string in the script file. Hex strings must be in the format %xHHxHH%, where H represents a hex character. Be sure to pair hex characters together.

User Response: Correct the hex string in the modem script file and run the program again.

12044 Too many commands specified in the modem script file.

Explanation: You have specified too many commands in your modem script file. The maximum number of commands allowed is 100.

User Response: Modify the modem script file so that you do not exceed the maximum number of commands and retry the program.

12045 MAXREPEAT invalid on IFVALUE command in modem script file.

Explanation: You specified an invalid value for the MAXREPEAT parameter in the IFVALUE command.

User Response: Correct the IFVALUE command and run the program again.

12047 DATABITS invalid on SETLINE command in modem script file.

Explanation: You specified an invalid value for the DATABITS parameter in the SETPARITY command. Valid values are 7 and 8.

User Response: Correct the SETLINE command and run the program again.

12048 STOPBITS invalid on SETLINE command in modem script file.

Explanation: You specified an invalid value for the STOPBITS parameter in the SETLINE command. Valid values are 1 and 2.

User Response: Correct the SETLINE command and run the program again.

12049 PARITY invalid on SETLINE command in modem script file.

Explanation: You specified an invalid value for the PARITY parameter in the SETPARITY command. Valid values are EVEN, ODD, and NONE.

User Response: Correct the SETLINE command and run the program again.

12050 Syntax error in command file.

Explanation: There is a syntax error in the old style modem command file.

User Response: Check the command lines that you changed or added for accuracy. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'Y' for CNNCT on the TRACE command in IEBASE.PRO to turn on the trace for the modem command processor. Correct the error and retry the program.

12055 Command not recognized in modem script file.

Explanation: You specified an invalid command in the script file.

User Response: Determine which command is in error. One possible cause is that you misspelled the command. Correct the problem and retry the program.

12060 Too many parameters on command line.

Explanation: You specified too many parameters or too much text on one modem command line in an old style modem script. The maximum number of parameters allowed is 10.

User Response: Correct the command line and retry the program.

12070 Label too long in modem script file.

Explanation: You specified a label in the script file that is too long. Labels can be up to 12 characters in length.

User Response: Correct the label and retry the program.

12081 Null character found in modem script file.

Explanation: There is a null character in the script file. Null characters are not permitted in the script files.

User Response: Correct the script file and retry the program.

12083 Command or parameter name too long in modem script file.

Explanation: A command or parameter in the script file is invalid because it is too long.

User Response: Determine which parameter is in error and retry the program.

12084 Command too long in modem script file.

Explanation: A parameter value in the script file is invalid because it is too long. You may have omitted the closing parenthesis from a value.

User Response: Determine which command is in error and retry the program.

12085 Command expected but not found in modem script file.

Explanation: The modem script processor was expecting either a command or the end of file and found something unexpected. It is possible that you specified a parameter and value before a command.

User Response: Determine where the error occurred. Correct the error and retry the program.

12086 Parameter/value or semicolon expected but not found.

Explanation: There is a syntax error in the script file. Either a semicolon or a parameter and value was expected but was not found. This is usually caused by omitting the semicolon from a command, omitting the value of a parameter, or leaving space between the parameter name and value.

User Response: Determine which command produced the error. Correct the error and retry the program.

12091 Parameter value too long in modem script file.

Explanation: A parameter value in the script file is invalid because it is longer than the maximum length permitted for the parameter. This is sometimes caused by unbalanced parentheses.

User Response: Determine which parameter is in error. Correct the error and retry the program.

12092 Duplicate parameter found in modem script file.

Explanation: A parameter in the script file was specified more than once for the command.

User Response: Determine which parameter is in error. Correct the problem and retry the program.

12093 Invalid parameter found in modem script file.

Explanation: You specified an invalid parameter in the script file.

User Response: Determine which parameter is in error. Correct the problem and retry the program.

12110 Attempted to receive data before the device is open.

Explanation: There is not an OPENPORT command before you try to receive data. The port must be opened before you can receive any data.

User Response: Correct the modem command file and retry the program. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'y' for cnnct on the TRACE command in IEBASE.PRO to turn on the trace for the modem script processor.

12120 Attempted to send data before the device is open.

Explanation: There is not an OPENPORT command before you try to receive data. The port must be opened before you can send any data.

User Response: Correct the modem command file and retry the program. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'y' for cnnct on the TRACE command in IEBASE.PRO to turn on the trace for the modem script processor.

12130 Redial requested from modem script file.

Explanation: An error was encountered and the modem script set the return code to cause the modem to redial.

User Response: No action required. If the modem script logic requests a redial and the maximum number of redials has been exceeded, Expedite Base will return the 11802 return code. Otherwise, Expedite Base will try to execute the dial connect script again.

Display status script syntax errors

This section describes the return codes for display status script syntax errors.

12210 Event not recognized in display script file.

Explanation: You specified an invalid event in the script file.

User Response: Determine which event is in error. A possible cause of this error is a misspelled event. Correct the problem and retry the program.

12211 Invalid foreground color specified in display script file.

Explanation: You specified an invalid FOREGROUND parameter value in the script file. See documentation for a list of valid foreground colors.

User Response: Correct the error in the script file and retry the program.

12212 Invalid background color specified in display script file.

Explanation: You specified an invalid BACKGROUND parameter value in the script file. See documentation for a list of valid background colors.

User Response: Correct the error in the script file and retry the program.

12213 Invalid row specified in display script file.

Explanation: You specified an invalid ROW parameter value in the script file. ROW must be 1 to 24.

User Response: Correct the error in the script file and retry the program.

12214 Invalid column specified in display script file.

Explanation: You specified an invalid COLUMN parameter value in the script file. COLUMN must be 1 to 80.

User Response: Correct the error in the script file and retry the program.

12215 Cannot determine the action on event in display script file.

Explanation: The display script file consists of events followed by parameters which specify the action to perform for the event, such as CLEARLENGTH to clear text or TEXT to display text. Your display script has an event with a missing parameter or incomplete set of parameters. Expedite Base cannot determine what needs to be done. A generic example of this problem would be to have an EVENT statement with only the ROW and COLUMN parameters specified. With only ROW and COLUMN specified the statement does not indicate what needs to be done for this event.

User Response: Correct the error in the display script file and retry the program.

12216 Invalid top left row specified in display script file.

Explanation: You specified an invalid TOPLEFTROW parameter value in the script file. TOPLEFTROW is used to draw a box and must be 1 to 24.

User Response: Correct the error in the script file and retry the program.

12217 Invalid top left column specified in display script file.

Explanation: You specified an invalid TOPLEFTCOL parameter value in the script file. TOPLEFTCOL is used to draw a box and must be 1 to 80.

User Response: Correct the error in the script file and retry the program.

12218 Invalid bottom right row specified in display script file.

Explanation: You specified an invalid BOTRIGHTROW parameter value in the script file. BOTRIGHTROW is used to draw a box and must be 1 to 24. BOTRIGHTROW must be greater than TOPLEFTROW.

User Response: Correct the error in the script file and retry the program.

12219 Invalid bottom right column specified in display script file.

Explanation: You specified an invalid BOTRIGHTCOL parameter value in the script file. BOTRIGHTCOL is used to draw a box and must be 1 to 80. BOTRIGHTCOL must be greater than TOPLEFTCOL.

User Response: Correct the error in the script file and retry the program.

12220 Invalid clear length specified in display script file.

Explanation: You specified an invalid CLEARLENGTH parameter value in the script file. CLEARLENGTH must be 1 to 80.

User Response: Correct the error in the script file and retry the program.

12221 Invalid color specified in display script file.

Explanation: You specified an invalid COLOR parameter value in the script file.

User Response: Correct the error in the script file and retry the program.

12222 Invalid clear screen specified in display script file.

Explanation: You specified an invalid CLEARSCREEN parameter value in the script file. CLEARSCREEN must be Y.

User Response: Correct the error in the script file and retry the program.

12223 Invalid wait time specified in display script file.

Explanation: You specified an invalid WAIT parameter value in the script file. WAIT must be 0 to 9 seconds.

User Response: Correct the error in the script file and retry the program.

12281 Null character found in display script file.

Explanation: There is a null character in the script file. Null characters are not permitted in the script files.

User Response: Correct the script file and retry the program.

12283 Command or parameter name too long in display script file.

Explanation: A command or parameter in the script file is invalid because it is too long.

User Response: Determine which parameter is in error and retry the program.

12284 Command parameter value too long in display script file.

Explanation: A parameter value in the script file is invalid because it is too long. You may have omitted the closing parenthesis from a value.

User Response: Determine which parameter is in error and retry the program.

12285 Unexpected command or parameter found in display script file.

Explanation: The display script processor was expecting either a command or the end of file and found something unexpected. It is possible that you specified a parameter and value before a command.

User Response: Determine where the error occurred. Correct the error and retry the program.

12286 Parm/value or semicolon expected, not found in display script file.

Explanation: There is a syntax error in the script file. Either a semicolon or a parameter and value was expected but was not found. This is usually caused by omitting the semicolon from a command, omitting the value of a parameter, or leaving space between the parameter name and value.

User Response: Determine which command produced the error. Correct the error and retry the program.

12291 Parameter value too long in display script file.

Explanation: A parameter value in the script file is invalid because it is longer than the maximum length permitted for the parameter. This is sometimes caused by unbalanced parentheses.

User Response: Determine which parameter is in error. Correct the error and retry the program.

12292 Duplicate parameter found in display script file.

Explanation: A parameter in the script file was specified more than once for the command.

User Response: Determine which parameter is in error. Correct the problem and retry the program.

12293 Invalid parameter found in display script file.

Explanation: You specified an invalid parameter in the script file.

User Response: Determine which parameter is in error. Correct the problem and retry the program.

12300 Invalid combination of parameters on event in display script file.

Explanation: You specified an invalid combination of parameters for a particular event.

User Response: Determine which event is in error. Correct the problem and retry the program.

Communication device errors

This section describes the return codes for communication device driver errors.

13030 Error occurred while trying to access specified device.

Explanation: Expedite Base encountered an error while trying to access the specified device.

User Response: Use the following checklist to identify the problem.

- Check that the permissions are specified so that Expedite Base has read/write access to the lock directory and the device.
- Verify that the device does exist.
- Verify that the correct fully qualified path name of the device was specified on the DEVICEx parameter of the DIAL command.
- Check the lock file directory for a file named LCK.nnnnn, where nnnnn is the name of the device you specified. This file will contain the process ID of the program that wrote the lock file. If that process is no longer running, it is it is safe to erase the lock file.
- Verify that the port is disabled.
- Once you have completed the preceding steps, retry the program.

13040 Invalid baud rate.

Explanation: The specified baud rate is invalid. Valid baud rates are 300, 1200, 2400, 4800, 9600, 19200, or 38400.

User Response: Check that you specified the appropriate baud rate for your modem. Retry the program.

13050 Invalid parity.

Explanation: The parity specified in the modem script file is invalid. Valid parity values are 7 and 8.

User Response: Check that you specified the appropriate parity for your modem. Retry the program.

13070 Error occurred while trying to lock device.

Explanation: Expedite Base encountered an error while trying to lock the device.

User Response: Check to see if the device is in use by another process. If it is, wait until it becomes available. Retry the program.

13080 Error occurred while trying to acquire device.

Explanation: Expedite Base is unable to acquire the device. There may be a configuration problem.

User Response: Make sure the device is installed in the AIX operating system using the 'devices' command, AIX PS/2, or 'SMIT' command, AIX RISC System/6000.

13090 Error occurred while trying to configure device.

Explanation: Expedite Base is unable to configure the device.

User Response: Make sure the device is installed in the AIX operating system using the 'devices' command, AIX PS/2, or 'SMIT' command, AIX RISC System/6000.

13092 Could not get process ID.

Explanation: Expedite Base could not get the process ID. There may be a configuration problem.

User Response: Make sure the device is installed in the AIX operating system using the 'devices' command, AIX PS/2, or 'smit' command, AIX RISC System/6000.

13094 Could not set process ID.

Explanation: Expedite Base could not set the process ID. There may be a configuration problem.

User Response: Make sure the device is installed in the AIX operating system using the 'devices' command, AIX PS/2, or 'SMIT' command, AIX RISC System/6000.

Parser errors

This section describes the return codes for parser errors.

14000 Null character in input file.

Explanation: There is a null character in either the profile command file, basein.pro, or the message command file, basein.msg. Null characters are not permitted in basein.msg or basein.pro.

User Response: Check the message response file, baseout.msg, the profile response file, baseout.pro, or the response work file, tempout.msg, to determine which command produced the error. Correct the command file and retry the program.

14020 Command or parameter name too long.

Explanation: A command or parameter name in the command file is invalid because it is too long.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file tempout.msg, to determine which command produced the error. Correct the command file and retry the program.

14030 Parameter value too long.

Explanation: A parameter value in the command file is invalid because it is longer than the maximum length permitted for the parameter. This is sometimes caused by unbalanced parentheses.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

14040 Command expected but not found.

Explanation: You did not specify an expected command in the command file. It is possible that you specified a parameter before a command.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

14050 Parameter and value or semicolon expected but not found.

Explanation: There is a syntax error in the profile command file, basein.pro, or message command file, basein.msg. Either a semicolon or a parameter and value was expected but was not found. This is usually caused by omitting the semicolon from a command, omitting the value of a parameter, or leaving space between the parameter name and value.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

15010 Parameter value too long.

Explanation: The parameter value for one of the commands in the command file is longer than the maximum allowed for that parameter.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

15020 Duplicate parameter found.

Explanation: The same parameter was specified more than once in a command.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

15030 Invalid parameter found.

Explanation: You specified an invalid parameter in the command file.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

15040 Command not recognized.

Explanation: You specified an unrecognized command in the command file.

User Response: Check the message response file, baseout.msg, profile response file, baseout.pro, or response work file, tempout.msg, to determine which command produced the error. Correct the appropriate command file and retry the program.

15042 COMMIT command only valid with user-initiated recovery.

Explanation: You specified a COMMIT command with a RECOVERY parameter on the TRANSMIT command specified as c, s, or f. COMMIT commands are processed with U for RECOVERY only.

User Response: Update the RECOVERY parameter on the TRANSMIT command to u, or remove all COMMIT commands from the message command file, basein.msg. Retry the program.

ATTENTION: If you reset the session using the -r parameter, you will no longer be able to continue the previous session. If it is necessary to reset the session, modify the message command file, basein.msg, deleting any commands that have already been processed. Failure to do this may result in some data being lost or duplicated.

Destination verification errors

This section describes the return codes for errors that occur when Information Exchange verifies a destination.

16020 The destination specified is not a valid IE destination.

Explanation: The destination specified in the SEND or SENDEDI command does not exist. The data was not sent.

User Response: Correct the destination in the SEND command, EDI data, EDI destination table, or EDI qualifier table. Retry the command.

16030 IE destination is blocked by trading partner list or payment levels.

Explanation: The destination in the SEND or SENDEDI command exists. However, the message cannot be sent because it is blocked by the payment level specified or by the trading partner list. The data was not sent.

User Response: Make sure the Information Exchange destination and payment levels are correct. Check with your service administrator to change a trading partner list or payment levels. Retry the command.

16040 IE was unable to verify the destination immediately.

Explanation: The destination in the SEND or SENDEDI command could not be verified immediately because it is on another Information Exchange system. The data was not sent.

User Response: Retry the command using either an f, g or n value for the VERIFY parameter. Information Exchange cannot immediately verify a destination on another system.

16050 No update access to library.

Explanation: You specified 'Y' for VERIFY on the PUTMEMBER command and either the library does not exist or you do not have update access to the library. The data was not sent.

User Response: Use Information Exchange Administration Services to verify that the library you are trying to update exists and that you have update authority for it. Retry the command.

16052 File with specified message key does not exist.

Explanation: The message key you specified does not match any of the files in your mailbox.

User Response: Verify you specified the correct message key. The message key is shown on the AVAILABLE record in response to a QUERY command. Correct the message key and retry the command.

16054 File to purge is being received.

Explanation: The message key you specified is for a file that is in the process of being received, and it cannot be purged.

User Response: If you do not want to receive the file, then discontinue the receive process and reset the Information Exchange session. Retry the command.

16056 Information Exchange profile does not allow files to be purged.

Explanation: Your Information Exchange profile does not allow purging of files from your mailbox.

User Response: If you want to be able to purge files from your mailbox, use Information Exchange Administration Services or ask your Service Administrator to change your profile to allow this action. Retry the command.

16060 Unable to retrieve library member.

Explanation: Information Exchange could not retrieve the library member because either the library does not exist, you do not have read access to the library, or the destination you specified is invalid.

User Response: Receive the system error message from your mailbox. The system error message explains why the GETMEMBER failed. Correct the problem and retry the command.

EDI errors

This section describes the return codes for EDI errors.

17102 Invalid X12 header in file.

Explanation: The X12 header in the data you attempted to send is invalid. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 ISA header that caused the error in the envelope, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17106 Missing X12 destination in file.

Explanation: The X12 header in the data you attempted to send does not contain an X12 receiver ID. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 ISA header that caused the error in the envelope, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17108 Invalid X12 destination in file.

Explanation: The X12 header in the data you attempted to send contains an X12 receiver ID that could not be resolved. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI qualifier table, the EDI destination table, or the X12 ISA header that caused the error in the envelope. Retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17110 Invalid X12 binary or encrypted segment in file.

Explanation: The X12 data contains an invalid binary or security segment. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 data and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17112 Invalid X12 binary or encrypted length in file.

Explanation: The length element in an X12 binary or security segment is invalid. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 data and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17114 X12 control number is missing or invalid.

Explanation: The X12 header in the data you attempted to send does not contain an interchange control number, or the interchange control number is not numeric. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 ISA header that caused the error in the envelope and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17116 X12 control number error.

Explanation: The X12 control number in the IEA is missing or does not match the control number in the ISA. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 data that caused the error and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17118 Missing IEA in X12 data.

Explanation: The X12 IEA segment was not found in the data. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the X12 data that caused the error and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17202 Invalid UCS header in file.

Explanation: The UCS BG header in the data you attempted to send is invalid. The segment terminator must be hex 15. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the UCS BG header that caused the error and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17206 Missing UCS destination in file.

Explanation: The UCS header in the data you attempted to send does not contain a UCS receiver ID. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the UCS BG header that caused the error and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17208 Invalid UCS destination in file.

Explanation: The UCS BG header in the data you attempted to send contains a UCS receiver ID that could not be resolved. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI qualifier table, the EDI destination table, or the UCS BG header that caused the error, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17302 Invalid EDIFACT header in file.

Explanation: The EDIFACT UNB header in the data you attempted to send is invalid. The segment terminator was not found. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDIFACT UNB header in the envelope that caused the error and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17306 Missing EDIFACT destination in file.

Explanation: The EDIFACT UNB header in the data you attempted to send does not contain an EDIFACT receiver ID. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDIFACT UNB header that caused the error, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17308 Invalid EDIFACT destination in file.

Explanation: The EDIFACT UNB header in the data you attempted to send contains an EDIFACT receiver ID that could not be resolved. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI qualifier table, the EDI destination table, or the EDIFACT UNB header in the envelope that caused the error. Retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17316 EDIFACT control number error.

Explanation: The EDIFACT control number in the UNZ is missing or does not match the control number in the UNB. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDIFACT data in the envelope that caused the error and retry the command. If there were multiple envelopes in this file, check the response file for SENT records to determine which envelopes in the file were sent.

17318 Missing UNZ in EDIFACT data.

Explanation: A UNA or UNB was found before the preceding EDIFACT envelope ended. This envelope and envelopes following it in the file were not sent.

User Response: Correct the EDIFACT data that caused the error and retry the command. If there were multiple envelopes in this file, check the response file for SENT records to determine which envelopes in the file were sent.

17402 Invalid UN/TDI header in file.

Explanation: The UN/TDI STX header in the data you attempted to send is invalid. The segment terminator was not found. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the UN/TDI STX header that caused the error, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17406 Missing UN/TDI destination in file.

Explanation: The UN/TDI STX header in the data you attempted to send does not contain a UN/TDI receiver ID. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the UN/TDI STX header that caused the error, and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

17408 Invalid UN/TDI destination in file.

Explanation: The UN/TDI STX header in the data you attempted to send contains a UN/TDI receiver ID that could not be resolved. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI qualifier table, the EDI destination table, or the UN/TDI STX header in the envelope that caused the error. Retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

18010 Null character found in EDI table.

Explanation: You specified a null character in the EDI destination table or the EDI qualifier table specified.

User Response: Correct the table and retry the command.

18020 EDI table has an invalid parameter.

Explanation: The parameter name in the EDI destination table or EDI qualifier table is not a valid parameter name or does not have an associated value.

User Response: Correct the table and retry the command.

18030 EDI table has an invalid parameter value.

Explanation: The parameter value in the EDI destination table or EDI qualifier table is longer than the maximum length allowed.

User Response: Correct the table and retry the command.

18040 EDI table contains a duplicate parameter.

Explanation: You specified the same parameter more than once for an entry in the EDI destination table or EDI qualifier table. This may be caused by a missing semicolon between entries.

User Response: Correct the table and retry the program.

18110 End of file found before end of EDI envelope in file.

Explanation: Expedite Base encountered the end of the file before the end of the EDI envelope. This envelope was not sent.

User Response: Correct the EDI data and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

18120 Unable to determine EDI type in file.

Explanation: Expedite Base could not determine the type of EDI data you attempted to send. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI data and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

18130 Error processing EDI data in file.

Explanation: There was an error processing the EDI data you attempted to send. This envelope and the envelopes following it in the file were not sent.

User Response: Correct the EDI data and retry the command. If there were multiple envelopes in this file, check the message response file, baseout.msg, for SENT records to determine which envelopes in the file were sent.

18210 Qualifier table contains an invalid alias.

Explanation: You specified an invalid value for the default alias in the qualifier table. The first character of the ALIAS is the destination table type and must be blank, g, o, or p. The last three characters of the ALIAS are the destination table ID and must be one to three alphanumeric characters.

User Response: Correct the EDI destination table and retry the program.

18230 Invalid IE destination for entry.

Explanation: You specified an invalid Information Exchange destination for the EDI destination in the EDI destination table. This indicates that the destination is missing, incomplete, contains more than one destination type, or uses an invalid alias type.

User Response: Correct the EDI qualifier table and retry the program.

18250 End of file found before end of destination in file.

Explanation: Expedite Base encountered an end-of-file in the middle of either a qualifier table entry or EDI destination table entry. Make sure the last table entry ends with a semicolon.

User Response: Correct the EDI qualifier table and retry the program.

18300 Invalid EDI data received.

Explanation: The data received with the RECEIVEEDI command was not valid EDI data. Once Expedite Base determines that the data was not valid EDI data, the rest of the data in the message is not reformatted. Therefore the records may not be separated as you want them.

User Response: No response is needed. This is only an informational message.

19003 TIMEOUT invalid on TCPCOMM command.

Explanation: You specified an invalid value for the TIMEOUT parameter on the TCPCOMM command. TIMEOUT must be a numeric character from 2 to 10.

User Response: Correct the TIMEOUT parameter on the TCPCOMM command in the profile command file, basein.pro. Retry the program.

19005 TCP/IP control file, hostname.fil, not found.

Explanation: Expedite Base could not find the TCP/IP control file, hostname.fil. You must have a TCP/IP control file, hostname.fil, if you specified COMMTYPE T or C on the TRANSMIT command.

User Response: Verify the TCP/IP control file, hostname.fil, exists in the directory where Expedite Base is running. If the problem still persists, contact the Help Desk.

19006 Unable to create socket.

Explanation: Expedite Base was unable to create a socket for communication with Information Exchange.

User Response: Verify that TCP/IP has been configured properly on your system. Try to restart Expedite Base. If the problem still persists, contact the Help Desk.

19007 Unable to connect to Information Exchange.

Explanation: Expedite Base was not able to connect with any of the host names or addresses specified in the TCP/IP control file, hostname.fil. Information Exchange, a route to Information Exchange, or the Domain Name Server may be temporarily unavailable.

User Response: Verify that host names or host addresses and port numbers specified in hostname.fil are valid. Retry the program. If the problem persists, contact the Help Desk.

19008 Invalid host name, host address, or port number.

Explanation: Expedite Base detected that a host name, host address, or port number specified in the TCP/IP control file, hostname.fil, is invalid.

User Response: Verify that each entry in the TCP/IP control file, hostname.fil, has either a valid host name or address and a valid port number. Retry the program. If the problem persists, contact the Help Desk.

19009 Unable to resolve host name.

Explanation: Expedite Base attempted to resolve the host name specified in the TCP/IP control file, hostname.fil, but was unsuccessful. The host name may be invalid, or Information Exchange or the Domain Name Server may be temporarily unavailable.

User Response: Verify the host name specified in the TCP/IP control file, hostname.fil, is valid. Retry the program. If the problem persists, contact the Help Desk.

19011 Unable to connect to host.

Explanation: Expedite Base was unable to connect to Information Exchange. Either your user ID is not enabled for TCP/IP or Information Exchange is temporarily unavailable.

User Response: If you have never connected to Information Exchange using TCP/IP with this user ID, your user ID may not be enabled for TCP/IP. If this problem persists, contact the Help Desk. If you have connected to Information Exchange using TCP/IP with this user ID, verify that the host name or address and the port number specified in hostname.fil are valid. Retry the program. If the problem persists, contact the Help Desk.

19012 TCP/IP subsystem is not running.

Explanation: TCP/IP subsystem on your system is not running.

User Response: Verify that TCP/IP has been installed, configured, and started properly. Retry the program. If the problem persists, contact the Help Desk.

19014 Invalid Information Exchange account and/or user ID specified.

Explanation: You specified an invalid account and user ID. Information Exchange does not recognize the account ID or user ID specified in the START command or the IDENTIFY command.

User Response: If a START command was used with ACCOUNT and USERID parameters, make sure they are correct. If the IEACCOUNT and IEUSERID are taken from the profile, make sure you specified them correctly in the IDENTIFY command in the profile command file, basein.pro. If the problem continues, contact the Help Desk.

19015 Invalid Information Exchange password specified.

Explanation: Your Information Exchange password is incorrect.

User Response: Correct the IEPASSWORD in the IDENTIFY command in the profile command file, basein.pro, or in the START command in the message command file, basein.msg, and retry the program. If you just changed your password, make sure you update the IDENTIFY or START command to reflect the new password. Retry the program. If the problem continues, contact the Help Desk.

19016 Unable to receive from host.

Explanation: Expedite Base was unable to receive data from Information Exchange. User Response:

User Response: Verify that the link has not been dropped and your Information Exchange ID is not being used by someone else. Also, verify that the value specified for TIMEOUT parameter on the TCPCOMM command in the profile command file, basein.pro, is greater than the value specified for the WAIT parameter on the RECEIVE or RECEIVEEDI command in the message command file, basein.msg. Retry the program. If the problem persists, contact the Help Desk.

19017 Unable to send to host.

Explanation: Expedite Base was unable to send data to Information Exchange.

User Response: Verify that the link has not been dropped, your Information Exchange ID is not being used by someone else, and the inactivity timeout has not been reached. Retry the program. If the problem persists, contact the Help Desk.

19031 Invalid data received.

Explanation: Expedite Base received invalid data during session initialization.

User Response: Check to make sure that you have used the correct IP address for a non-SSL connection and retry the program.

General environment errors

This section describes the return codes for general environment errors.

20365 Not enough memory.

Explanation: Expedite Base is unable to get the memory it needs.

User Response: Make sure there is adequate memory and retry the program. If the problem persists, contact the Help Desk.

20410 Profile not found.

Explanation: Expedite Base could not find the profile information. You must have a profile command file, basein.pro.

User Response: Create the profile command file, basein.pro, containing all of the required information. Verify that basein.pro exists in the current directory or in the path specified by the -p command line parameter. Make sure the file permissions allow Expedite Base to access the file in read mode. Retry the program.

20611 Error opening input file.

Explanation: Expedite Base could not open the message command file, basein.msg.

User Response: Verify that the input file, basein.msg, exists in the current directory or in the directory specified by the -p command line parameter. Make sure the file permissions allow Expedite Base to access the file in read mode. Retry the program.

20620 Unable to restart at checkpoint due to error in BASEIN.MSG.

Explanation: Commands in the message command file, basein.msg, that were processed before the last checkpoint have been changed. Expedite Base is unable to continue the current session at the latest checkpoint. Commands in basein.msg that have already been processed, echoed to baseout.msg, should not be changed if you want to restart the session at the last checkpoint.

User Response: Reset the session using the -r command line parameter on the IEBASE command. Before starting the next session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

21000 Expedite Base encountered an unexpected condition.

Explanation: Expedite Base encountered an unexpected condition during execution.

User Response: Retry the program. If the problem persists, contact the Help Desk. You will be asked to FAX or send the trace, iebase.trc, from the failed session to the network for problem determination.

21013 Error reading EDI table.

Explanation: Expedite Base could not read the EDI qualifier table file or EDI destination table file.

User Response: Make sure there are no input or output problems on your workstation. Check that the permissions on the file are set so that Expedite Base can access the file in the desired mode. Retry the program.

21410 Display status script file missing.

Explanation: Expedite Base could not find the display status script file.

User Response: Verify that the display status script file, display.scr, is in the default install directory or in the directory specified in the IEPATH parameter of the SESSION command in the profile command file, basein.pro. Make sure the file permissions are set so that Expedite Base can access the file in read mode.

21606 Receiver not found in look up table.

Explanation: You specified 'T' for the COMPRESS parameter, which indicates you want the look-up-table to be consulted before your data is sent compressed, but the receiver was not in the look-up-table; therefore, the data was not compressed before sending. The COMPRESS parameter in your baseout.msg will be set to 'W' for warning.

User Response: Check the message response file, baseout.msg, or response work file, tempout.msg, to determine which receiver was not found in the table. Either specify 'Y' for COMPRESS or add the receiver to your look-up-table, and retry the program.

21810 File operation failed on file.

Explanation: Expedite Base could not access a file in the requested mode.

User Response: Check your disk and make sure there are no I/O problems. Check the file permissions to make sure Expedite Base has access to the file. Retry the program. If the problem persists, contact the Help Desk. You will be asked to FAX or send the trace file, iebase.trc, to the network for problem determination.

21950 Received files table damaged.

Explanation: An internal file used by Expedite Base to keep track of the files received, rcvofset.fil, was damaged.

User Response: Reset the session using the -r command line parameter on the IEBASE command. Before starting the next session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

22010 Error opening translate table file.

Explanation: Expedite Base could not open the translate table you specified.

User Response: Verify that the translate table file exists in the default install directory or in the directory specified in the IEPATH parameter of the SESSION command in basein.pro. The translate table name must have the extension XLT. If the translate table is in the correct directory, check your disk to make sure there are no I/O problems. Make sure the file permissions are set so that Expedite Base can access the file in the desired mode. Retry the program.

22020 Invalid translate table file.

Explanation: The translate table file specified is invalid.

User Response: Verify that the format of the table is correct and retry the program.

22240 Error looking up message.

Explanation: An error occurred while looking up an error message. This is caused by an error in the error message description file, ERRORMSG.FIL, or the extended error text file, ERRORTXT.FIL.

User Response: If you modified either the error message description file, ERRORMSG.FIL, or the extended error text file, ERRORTXT.FIL, retry the program with the original files. If the problem continues, contact the Help Desk.

22411 Error opening receive file.

Explanation: Expedite Base could not open the file for the data being received.

User Response: Make sure that you specified a valid FILEID for the RECEIVE or RECEIVEEDI command. Also, check the receive file for errors. Make sure the file permissions are set so that Expedite Base can access the file in the desired mode. Retry the program.

22416 Unable to access directory for receive file.

Explanation: Expedite Base could not access or create the directory specified on the FILEID parameter of the RECEIVE or RECEIVEEDI command.

User Response: Check the temporary response file, tempout.msg, to determine which command produced the error. If the drive and directory on the FILEID parameter is valid, make sure that there is enough disk space and there are no input or output problems. Correct the input file if necessary and retry the program.

22430 Free-format message error.

Explanation: Expedite Base could not convert the received data to a free-format message. The data was written without any reformatting.

User Response: No response is needed. This is only an informational message.

22440 Length delimiters in data invalid.

Explanation: The common data header or DELIMITED parameter indicated that the data contained two-byte length delimiters to separate records, but the lengths indicated by the delimiters did not match the length of the data. The length delimiters were processed, but the records may not be separated as you want them.

User Response: No response is needed. This is only an information message.

22450 Translate table in Common Data Header invalid.

Explanation: The common data header indicated a translate table that was not found or was invalid on your system. The translate table from your profile was used instead.

User Response: Check your file to verify that the data was translated correctly. If it wasn't, you need to use another translate table. Check with your trading partner to see what translate table was used when the file was sent.

22610 Send or Putmember file not found.

Explanation: Expedite Base could not open the file indicated in the SEND or PUTMEMBER command.

User Response: Check that the FILEID parameter on the SEND or PUTMEMBER command is specified correctly and that the file exists. Make sure the file permissions are set so that Expedite Base can access the file in read mode. Retry the program.

22615 Send or Putmember file was empty.

Explanation: The file indicated in the SEND or PUTMEMBER command was empty.

User Response: Correct the message command file, basein.msg, or send or putmember file and retry the command.

23410 EDI send file not found.

Explanation: Expedite Base could not open the file indicated in the SENDEDI command.

User Response: Check that the file name is specified correctly on the command, and that the file exists in the directory specified. Make sure the file permissions are set so that Expedite Base can access the file in the desired mode. Check your disk to make sure there are no I/O problems. Retry the command.

23415 EDI send file was empty.

Explanation: The file indicated in the SENDEDI command did not contain any EDI data. It was empty or contained only blanks.

User Response: Correct the message command file, basein.msg, or EDI send file and retry the command.

23500 No libraries found to list.

Explanation: There were no libraries found for the parameters specified on the LISTLIBRARIES command.

User Response: Check parameters specified in the LISTLIBRARIES command. If the AUTHORITY, SELECTION, and/or OWNER parameters are incorrect, correct them and retry the command.

23502 Owning account for libraries invalid.

Explanation: The owning account ID specified by the OWNER parameter of the LISTLIBRARIES command is not recognized by Information Exchange.

User Response: Check the owning account ID specified in the LISTLIBRARIES command is correct. If not, correct the OWNER parameter and retry the command.

23504 Library does not contain any members.

Explanation: The library specified in the LISTMEMBERS command does not contain any members. The command was not processed.

User Response: Check the library specified in the LISTMEMBERS command. Correct the LIBRARY parameter and try again.

23506 Library does not exist.

Explanation: The library specified in the LISTMEMBERS command does not exist on Information Exchange.

User Response: Check the library specified in the LISTMEMBERS command. If it is correct, use Information Exchange Administration Services to verify that the library exists.

23508 Library owning account invalid.

Explanation: The library owning account specified in the OWNER parameter of the LISTMEMBERS command is not recognized by Information Exchange.

User Response: Check the library owning account specified in the LISTMEMBERS command. Correct the OWNER parameter and retry the command.

23510 Read access not permitted for library.

Explanation: The ACCOUNT or USERID does not have read access to the library specified in the LISTMEMBERS command.

User Response: Use Information Exchange Administration Services to verify that you have read access to the library. Retry the command.

23610 Unexpected error in asynchronous communications.

Explanation: Expedite Base encountered an unexpected error during asynchronous communications.

User Response: Try the program again. If the problem persists, contact the Help Desk.

Session start and end errors

This section describes the return codes for session start and end errors.

24000 Error in restart processing.

Explanation: Expedite Base was not able to restart the session with Information Exchange. The session file, session.fil, may be damaged.

User Response: Reset the session using the -r command line parameter on the IEBASE command. Also, make sure there is not another user using this user ID. If the problem persists, contact the Help Desk. Before starting the next session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

24020 Restart and original recovery levels are not equal.

Explanation: The restart level differs from the original recovery level. The session has not started. Either your session file, session.fil, is damaged or another user is using this user ID.

User Response: Reset the session using the -r command line parameter on the IEBASE command. Also, make sure there is not another user using this user ID. If the problem persists, contact the Help Desk. Before starting the next session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

24100 Session and Information Exchange checkpoints do not match.

Explanation: In a session using checkpoint-level recovery, the check- point numbers for the send or receive side of the session do not match the values Information Exchange recorded. Your session file, session.fil, may be damaged.

User Response: Reset the session using the -r command line parameter on the IEBASE command. Also, make sure there is not another user using this user ID. If the problem persists, contact the Help Desk. Before starting the next session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

24200 Invalid time zone.

Explanation: You specified an invalid time zone.

User Response: Correct the TIMEZONE parameter in the IDENTIFY command in the profile command file, basein.pro, and retry the program. If the problem persists, contact the Help Desk.

24210 Invalid maximum segments.

Explanation: Expedite Base used an invalid value when trying to start the session with Information Exchange.

User Response: Contact the Help Desk.

24220 Invalid reset field.

Explanation: Expedite Base used an invalid value for a field when trying to start the session with Information Exchange.

User Response: Contact the Help Desk.

24230 Invalid field on session start.

Explanation: Expedite Base used an invalid value for a field when trying to start the session with Information Exchange.

User Response: Contact the Help Desk.

24270 Incorrect Information Exchange password.

Explanation: Your Information Exchange password is incorrect.

User Response: Correct the IEPASSWORD in the IDENTIFY command in the profile command file, basein.pro, or in the START command in the message command file, basein.msg, and retry the program. If you just changed your password, make sure you update the IDENTIFY or START command to reflect the new password.

24280 Invalid Information Exchange user ID.

Explanation: You specified an invalid user ID. Information Exchange does not recognize the account ID or user ID specified in the START command or the profile.

User Response: If a START command was used with ACCOUNT and USERID parameters, make sure they are correct. If the IEACCOUNT and IEUSERID are taken from the profile, make sure you specified them correctly in the IDENTIFY command in the profile command file, basein.pro. If the problem continues, contact the Help Desk.

24290 Invalid new Information Exchange password.

Explanation: You are an Extended Security Option user, and you specified an invalid new Information Exchange password.

User Response: Correct the NIEPASSWORD parameter in the IDENTIFY command in the profile command file, basein.pro, or in the START command in the message command file, basein.msg, and retry the program. ESO passwords have special requirements. Refer to the product documentation for information about these requirements.

24300 Invalid password. Userid was revoked.

Explanation: You are an ESO user and sent three successive session starts to Information Exchange with incorrect passwords. The Information Exchange user ID has been revoked.

User Response: Contact your service administrator to request that your password be reset using Information Exchange Administration Services. Resetting the password resumes the user ID.

24310 New password is required.

Explanation: You are an ESO user and did not specify a new password. If the Information Exchange password for an ESO user is the same as the Information Exchange user ID, the ESO user must specify a new password.

User Response: Use the NIEPASSWORD parameter of the IDENTIFY command in the profile command file, basein.pro, or of the START command in the message command file, basein.msg, to change the password.

24320 Error starting session with Information Exchange.

Explanation: There was an error trying to start the session with Information Exchange.

User Response: Contact the Help Desk.

24600 Information Exchange did not return a valid session end response.

Explanation: The Information Exchange session may not have ended.

User Response: Try to restart the session. If the problem persists, contact the Help Desk.

24610 Information Exchange did not end the session due to an error.

Explanation: Information Exchange indicated that there was an error, and the session could not end properly.

User Response: Try to restart the session. If the problem persists, contact the Help Desk.

PF key exit errors

This section describes the return code for PF key exit errors.

25000 Exit key pressed.

Explanation: You pressed the defined exit key, EXITKEY on the SESSION command in basein.pro, and processing is not complete.

User Response: If you want to continue the interrupted processing, call IEBASE again. If you do not want to continue, you can reset the session using the -r command line parameter on the IEBASE command. Before resetting the session, review the message response file, baseout.msg, to see which commands were processed successfully. Remove these commands from the message command file, basein.msg, so they are not processed again.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

Comm-Press error messages

For Comm-Press error messages, see “Error messages and return codes for data compression” on page E-6.

Internal communications errors

This section describes the return code for internal communications errors.

26401 Expedite Base encountered corrupted data.

Explanation: Expedite Base data control layer detected the data sent or received was corrupted. Expedite Base will end the session, redial and try again for as many times as allowed by the RECONNECT parameter on the TRANSMIT command in the profile command file, basein.pro. The default is 5 times. If this error is returned again then Expedite Base could not recover from the problem by redialing. This error is often caused by a hardware problem on the workstation side.

User Response: Try another modem. Try another system. If the problem persists, contact the Help Desk.

26402 A timeout occurred while Expedite Base was waiting to receive data.

Explanation: Expedite Base timed out while waiting to receive data from IE. Expedite Base will end the session, redial and try again for as many times as allowed by the RECONNECT parameter on the TRANSMIT command in the profile command file, basein.pro. The default is 5 times. If this error is returned again then Expedite Base could not recover from the problem by redialing.

User Response: Wait and try the transmission again later. If the problem persists, contact the Help Desk.

26403 Expedite Base encountered an error processing data from IE.

Explanation: The data received should have ended with the chaining character C or L but did not. Expedite Base will end the session, redial and try again for as many times as allowed by the RECONNECT parameter on the TRANSMIT command in the profile command file, basein.pro. The default is 5 times. If this error is returned again then Expedite Base could not recover from the problem by redialing.

User Response: Wait and try the transmission again later. If the problem persists, contact the Help Desk.

26407 Expedite Base encountered corrupted data while receiving a file.

Explanation: Expedite Base data control layer detected the data received was corrupted. Expedite Base will end the session, redial and try again for as many times as allowed by the RECONNECT parameter on the TRANSMIT command in the profile command file, basein.pro. The default is 5 times. If this error is returned again then Expedite Base could not recover from the problem by redialing. This may indicate a hardware problem on the workstation side.

User Response: Try another modem. Try another system. If the problem persists, contact the Help Desk.

26410 Expedite Base timed out while sending or receiving data from IE.

Explanation: Expedite Base was unable to continue communicating with the protocol converter. Expedite Base will end the session, redial and try again for as many times as allowed by the RECONNECT parameter on the TRANSMIT command in the profile command file, basein.pro. The default is 5 times. If this error is returned again then Expedite Base could not recover from the problem by redialing.

User Response: Wait and try the transmission again later. If the problem persists, contact the Help Desk.

26411 Expedite Base ran out of buffer space while encoding data to send.

Explanation: Expedite Base encodes 8-bit binary data to be sent over the 7-bit data connection. In the process, Expedite Base ran out of buffer space.

User Response: Try the transmission again. If the problem persists, contact the Help Desk.

26412 Expedite Base ran out of buffer space while decoding data received.

Explanation: Expedite Base decodes the 8-bit binary data received over the 7-bit data connection. In the process, Expedite Base ran out of buffer space.

User Response: Try the transmission again. If the problem persists, contact the Help Desk.

26805 Lost carrier.

Explanation: The carrier was lost during data transmission.

User Response: Retry the program. If the problem persists, contact the Help Desk.

26806 DCL error.

Explanation: A DCL error has occurred.

User Response: Retry the program. If the problem reoccurs, contact the Help Desk.

26810 Program error.

Explanation: A program error has occurred.

User Response: Retry the program. If the problem reoccurs, contact the Help Desk.

26897 Incorrect segment length on received data.

Explanation: The segment length, assigned by Information Exchange, was incorrect. This occurs for one of two reasons. Either DCL received data that was not in the expected format, or there was an SDIERR, which is an error from Information Exchange. These error messages are not sent with the length values.

- One possible cause of this problem is that you have specified the wrong product name in the PRODUCT parameter of the IDENTIFY command in basein.pro.
- A second possible cause is that the Service Manager profile has the wrong LU name defined for the product INFOEXCH. If the problem persists, contact Help Desk to check the Service Manager profile for the correct LU name.
- A third cause can be the modem echoing the DCL frames back to the workstation instead of sending them over the asynchronous line. Because either a hardware problem with the modem or async adapter, or due to the fact that the modem is in command state. Some modems return to command state when the line is dropped.

Retry the program. Try the system with another communications program. If this program fails, have the modem and workstation tested for hardware problems. If the other program works, contact the Help Desk.

User Response: If the steps itemized above do not stop the problem from occurring, contact the Help Desk.

26899 Missing chaining indicator.

Explanation: The record chaining indicator was missing from the data.

User Response: Follow the steps described for 26897.

26986 DCL received an EOT when not expecting one.

Explanation: DCL received an end-of-transmission, EOT, character when one was not expected. You may see this if the modem goes into command state and begins to echo characters back to the workstation. DCL must be in a particular state and send an EOT character to the modem. If the modem echoes the EOT back to the workstation, this error will occur. Or, the modem is online but is echoing frames back to the workstation instead of sending the frame to the network communications gateway.

User Response: Retry the program. If it continues to fail, try another modem. If the new modem works, have the old modem checked for hardware problems. If the new modem fails with the same results, have the async adapter on the workstation checked or replaced. If the problem persists, contact the Help Desk.

26987 DCL received a BID when not expecting one.

Explanation: DCL received an inquiry, BID, character when one was not expected. You may see this if the modem goes into command state and begins to echo characters back to the workstation. DCL must be in a particular state and send a BID character to the modem. If the modem echoes the BID back to the workstation, this error will occur.

User Response: Retry the program. If it continues to fail, try another modem. If the new modem works, have the old modem checked for hardware problems. If the new modem fails with the same results, have the async adapter on the workstation checked or replaced. If the problem persists, contact the Help Desk.

26988 The network communications gateway has prematurely ended the session.

Explanation: Expedite Base did not expect the network communications gateway to end the session.

User Response: Retry the program. If the problem persists, contact to the Help Desk.

26989 An internal buffer has been overrun.

Explanation: Expedite Base receives data from the modem into an internal buffer. If the data received exceeds the size of the buffer, then this error will occur. Expedite Base and the BSC dial asynchronous protocols or the network communications gateway precautions programmed to prevent this from happening.

User Response: Retry the program. If the problem persists, contact the Help Desk.

26990 A series of NAKs has occurred.

Explanation: DCL has encountered a situation where 10 frames of data with CRC characters attached have been sent to the network communications gateway and it indicates the CRCs are not correct by responding with a NAK (negative acknowledgment). This is almost always caused by hardware problems. The frame of data is altered by the modem or async adapter and the CRCs do not match.

User Response: Make sure the modem is set in transparency mode so that it is not trying to interpret characters in the data stream. Retry the program. If it continues to fail, try another modem.

26991 A series of NAKs has occurred.

Explanation: DCL has encountered a situation where 10 frames of data with CRC characters attached have been received from the network communications gateway and the DCL indicates the CRCs are not correct by responding with a NAK (negative acknowledgment). This would be caused by a hardware problem. The frame of data may have been altered by the modem or async adapter, and the CRCs do not match.

User Response: Make sure the modem is set in transparency mode so that it is not trying to interpret characters in the data stream. Retry the program. If it continues to fail, try another modem. If the new modem works, have the old modem checked for hardware problems. If the new modem fails with the same results, have the async adapter on the workstation checked or replaced. If the problem persists, contact the Help Desk.

26992 DCL received data without an ETX or ETB control character.

Explanation: DCL was expecting to receive a frame of data with the ETX or ETB, End of Text or End of Text Block, control character at the end. This is caused by a modem hardware problem in most cases. The modem may be echoing partial frames of data back to the workstation.

User Response: Retry the program. If it continues to fail, try another modem. If the new modem works, have the old modem checked for hardware problems. If the new modem fails with the same results, have the asynch adapter on the workstation checked or replaced. If the problem persists contact the Help Desk.

26993 DCL did not receive a response when one was expected.

Explanation: DCL was expecting a response but didn't receive one.

User Response: Retry the program. If the problem persists, contact the Help Desk.

26994 DCL has encountered a BID-WAK loop.

Explanation: DCL sends BIDs to the network communications gateway, but receives only Wait Acknowledgements, WAKs. This usually occurs when the user increased the transmission block size above 1024.

User Response: Try the default block size of 1024. If the problem persists, contact the Help Desk.

26995 DCL has encountered a BID-BID loop.

Explanation: DCL gets a BID response to the BID it sends. This can be caused by a modem which has returned to command state and is echoing the characters back to the workstation. Or, it can be a hardware problem.

User Response: Retry the program. If it continues to fail, try another modem. If the new modem works, have the old modem checked for hardware problems. If the new modem fails with the same results, have the asynch adapter on the workstation checked or replaced. If the problem persists contact the Help Desk.

26996 Timed-out while waiting for a response.

Explanation: DCL timed-out while waiting for a response from the network communications gateway. This can occur if the line is dropped and the operating system does not return the lost-carrier condition to the program.

User Response: Retry the program. If the problem persists, it may be that the Asynchronous Relay is down. Try the program again in about 30 minutes. If the problem still occurs, contact the Help Desk.

26997 DCL communications error encountered with network communications gateway.

Explanation: DCL received an unexpected control character, or no response at all when one was expected.

User Response: Retry the program. If it continues to fail, try another modem. If the problem still persists, contact the Help Desk.

26998 DCL error.

Explanation: DCL was inserting transparency characters in the data and the length was 0.

User Response: Contact the Help Desk.

26999 DCL error.

Explanation: A DCL error has occurred.

User Response: Retry the program. If the problem reoccurs, contact the Help Desk.

Old message.fil errors

This section describes the return codes for old message.fil errors.

27010 Error in the draw picture and display status command file.

Explanation: There is an error in the draw picture and display status command file.

User Response: Check the command lines you changed or added for accuracy. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'Y' for DISPLAY on the TRACE command in basein.pro to turn on the trace for the draw picture and display status command processor. Correct the error and retry the program.

Session errors

This section describes the return codes for session errors.

28000 Warnings generated for the command.

Explanation: This return code indicates that warning messages were generated during the command, but the command was able to complete.

User Response: Check the WARNING records in the message response file, baseout.msg, for details.

28010 IE session completed normally but not all requests were processed.

Explanation: The Information Exchange session completed normally, but not all of the requests in the message command file, basein.msg, processed, or some requests generated warnings.

User Response: Check the message response file, baseout.msg, to see which requests did not process normally. Correct those requests in a new basein.msg and retry the program.

28020 IE session completed normally but an error occurred during disconnect.

Explanation: All of the commands in the message command file completed. While Expedite Base was disconnecting from the network, it encountered an error. Most likely, there is a problem with one of the script files.

User Response: The SESSIONEND record in baseout.msg will be followed by a WARNING record, ERRDESC record, and ERRTEXT records explaining what the error was and where it occurred. Correct the error before running IEBASE again.

28100 QUERY response indicates warning.

Explanation: Information Exchange found one or more errors in the QUERY command but was still able to process the command. Expedite Base may not write AVAILABLE records for some or all of the messages in your mailbox.

User Response: Check the error messages in your mailbox to see what caused the error. If needed, correct the error and retry the command.

28120 GETMEMBER response indicates a warning.

Explanation: The response from Information Exchange to the GETMEMBER command shows that there was a warning while processing the command.

User Response: Make sure the library and member that you are trying to retrieve exist, and that you have access to them. If there is a system error message in your mailbox, retrieve it or view it via Information Exchange Administration Services to determine what caused the error. Correct the problem and try the command again.

28140 AUDIT response indicates warning.

Explanation: Information Exchange found one or more errors in the AUDIT command but was still able to process the command. However, the audit records in your mailbox may be different than requested.

User Response: Check the error messages in your mailbox to see what caused the error. If the audit file in the mailbox does not meet your requirements, correct the error and retry the program.

28141 AUDIT response indicates error.

Explanation: Information Exchange found one or more errors in the AUDIT command and was unable to process the command.

User Response: Check the error messages in your mailbox to see what caused the error. Correct the error and retry the command. No audit file has been placed in your mailbox.

28160 An unexpected error occurred while processing a VERIFY command.

Explanation: An unexpected error occurred while processing a VERIFY command. Because Expedite Base was unable to use the results of the VERIFY command, it proceeded to send the data to Information Exchange.

User Response: Check your mailbox for system error messages that may have been generated if your send request was invalid. You can also check the mailbox for acknowledgments, if you requested any, to verify whether the data was sent or not.

28170 LISTLIBRARIES response indicates a warning.

Explanation: Information Exchange found one or more errors in the LISTLIBRARIES command but was still able to process the command. Expedite Base may not write LIBRARYLIST records for some or all of the libraries you have access to.

User Response: Check the error messages in your mailbox to see what caused the error. If needed, correct the error and retry the command.

28171 There are more files to be received.

Explanation: The session ended successfully, but there are more files in the mailbox to be received.

User Response: Process the data already received and run Expedite Base again to receive the additional data. Refer to Chapter 5, “Sending and receiving files,” and Chapter 6, “Sending and receiving EDI data,” for more information on session-level recovery. If you switch to checkpoint, user-, or file-level recovery, you will be able to receive all of the files in your mailbox in a single session without encountering the 28171 return code.

28175 LISTMEMBERS response indicates a warning.

Explanation: Information Exchange found one or more errors in the LISTMEMBERS command but was still able to process the command. Expedite Base may not write MEMBERLIST records for some or all of the libraries you have access to.

User Response: Check the error messages in your mailbox to see what caused the error. If needed, correct the error and retry the command.

28180 PURGE response indicates an error.

Explanation: An unexpected error occurred while trying to process the PURGE command.

User Response: Retry the program. If the problem persists, contact the Help Desk.

28190 Invalid common data header received.

Explanation: Expedite Base received an invalid Common Data Header. The data was received and processed as if no CDH was received.

User Response: Verify that the file was received as expected. You may wish to inform the sender that the interface sent an invalid CDH. If the sending interface was an Expedite Base product, contact the Help Desk.

28200 COMMIT command only valid after a SEND, SENDEDI or PUTMEMBER command.

Explanation: COMMIT command only valid after a SEND, SENDEDI or PUTMEMBER has been specified.

User Response: A COMMIT command was specified but there was no SEND, SENDEDI or PUTMEMBER command preceding it. The COMMIT command did not initiate a commit.

29960 The open connection failed for SNA LU 6.2 communications.

Explanation: Communication Server attempted to open a connection, but could not do so successfully.

User Response: Check the following items to verify they are correct:

- Verify 'S' for COMMTYPE is specified for LU 6.2.
- Verify CONNECTION name is valid and defined in Communication Server.
- Verify SNA is started.
- Verify the modem is configured properly and turned on.
- Verify DataLink Device Name, under EIA232 Physical Link in SNA Services, is valid.
- Verify the DLC for SDLC is active.

29961 Could not load SNA.

Explanation: Attempt to load SNA dynamic library failed.

User Response: Check the following items to verify they are correct:

- Verify you have installed Communication Server.
- Verify the file lu62shrd.o is in the same directory where you installed Expedite Base.
- Verify that if you have installed Expedite Base in a directory other than the default, you have added that directory to LIBPATH environment variable. For example, if you installed Expedite Base in /usr/local/expedite you must define LIBPATH as follows before you start Expedite Base:

```
export LIBPATH=/lib:/usr/lib:/usr/local/expedite:
```

29970 Send data failed for SNA LU 6.2 communications.

Explanation: Communication Server tried to send data but failed.

User Response: Retry the program. If it still fails, check the following items:

- Verify the SNA is still active. If not, restart it.
- Make sure the leased line hardware is working.
- If you have never sent data before, review the installation instructions and make sure you have configured Communication Server correctly to work with Expedite Base, and that you have verified the SNA configuration.

29975 Receive data failed for SNA LU 6.2 communications.

Explanation: Communication Server tried to receive data but failed.

User Response: Retry the program. If it still fails, check the following items:

- Verify the SNA is still active. If not, restart it.
- Make sure the leased line hardware is working.
- If you have never received data before, review the installation instructions and make sure you have configured Communication Server correctly to work with Expedite Base, and that you have verified the SNA configuration.

29980 The allocate failed for SNA LU 6.2 communications.

Explanation: Communication Server attempted to allocate a conversation with the Partner LU, but was unable to do so.

User Response: Check the LU 6.2 Partner LU Name in Communication Server to verify it specifies the correct Partner LU Name. If you are not sure which Partner LU name to use, contact the Help Desk.

29998 Modem command processor asked to stop.

Explanation: The modem script processor encountered some error and the return code was set within the modem script file to stop IEBASE processing.

User Response: Determine why the modem script processor file asked to stop. The trace file, iebase.trc, may be helpful in determining the problem. Specify 'Y' for CNNCT on the TRACE command in basein.pro to turn on the trace for the modem script processor. Correct the error and retry the program.

29999 Session end response failure.

Explanation: Expedite did not receive a session end response from Information Exchange, or a communication failure occurred upon receiving the session end response.

User Response: Follow these steps after a session that fails with 29999 to see if the previous session has completed.

1. Specify AUTOSTART(N), AUTOEND(N) and RECOVERY(S) on your TRANSMIT command in the Expedite Base profile.
2. Create an input file containing START and END records. An example follows.

```
START CHECK(Y);
END;
```

Do not specify any other commands in the input file if you specify CHECK(Y) on the session start command.

3. Run Expedite Base. No data will be transferred in the above example and you will not be charged for this inquiry.

4. Examine the output file to check the LASTSESS parameter value on the STARTED record.

LASTSESS(0)	Indicates that the previous session was successful. No further recovery is required.
LASTSESS(1)	Indicates that the previous session was not successful.

If Expedite Base reported the 29999 Session End return code for a session, you should switch to checkpoint, file-level or user-level recovery instead of session-level recovery for future sessions with a similar number of commands.

SSL communication errors

30006 Invalid common name on returned certificate.

Explanation: The certificate returned during the SSL handshake contains an invalid common name.

User Response: The common name that was returned has been written to the LINKTRC, if you requested to create a LINKTRC. To generate a LINK trace, specify LINK(Y) in the TRACE command in the basein.pro file. Rerun the session and send the resulting LINK trace to the Help Desk to determine the cause of the error.

30007 Un-trusted certificate received.

Explanation: The certificate that was returned during the SSL handshake matches a certificate listed in the certs.fil file.

User Response: Contact the Help Desk for instructions about how to proceed.

30008 User not allowed through the Secure Front End (SFE) Gateway.

Explanation: The user has no access privileges to the Secure Front End Gateway.

User Response: Ensure that you specified the correct IP address to communicate with the Secure Front End Gateway. If the IP address that you specified is correct, contact the Help Desk for instructions about how to proceed.

30009 Account/Userid does not match certificate's account/userid.

Explanation: The user information that is found in the X.509 certificate does not match the Information Exchange system ID, account ID, and user ID that was specified on the IDENTIFY or START command.

User Response: Make sure the information that you gave when you requested the certificate matches the Information Exchange system ID, account ID, and user ID that you specified on the IDENTIFY or START command. Correct the error, and then retry the program.

30010 Invalid Certificate.

Explanation: The certificate is not valid for use on this Secure Front End (SFE) Gateway.

User Response: Make sure the certificate that was specified is one that is generated by the GXS PKI Profile Server. Correct the error, and then retry the program.

30011 Unable to negotiate security specifications.

Explanation: The SSL negotiation did not complete with a satisfactory protection level for the Secure Front End (SFE) Gateway.

User Response: Contact the Help Desk for instructions on how to proceed.

30012 Connection temporarily refused.

Explanation: The connection to the Secure Front End (SFE) Gateway has been temporarily refused.

User Response: Try again later. If you continue to experience problems, contact the Help Desk for instructions on how to proceed.

30013 SSL setup failed

Explanation: The SSL (Secure Sockets Layer) initialization process failed.

User Response: The SSL session could not be started. If you enabled the link trace, you can find additional information about the error within the link trace in the iebase.trc file. If you did not enable the link trace, LINK(Y) in the TRACE command, run the session again, and send the trace file to the Help Desk for instructions on how to proceed.

30014 Invalid keyring file

Explanation: The file name that you specified in the KEYRINGFILE parameter of the IDENTIFY or START command was not found or the file could be opened.

User Response: Verify that you typed the correct file name in the KEYRINGFILE parameter and that the file permission is set so that the file is readable.

30015 Invalid KEYRINGSTASHFILE

Explanation: The file name that you specified in the KEYRINGSTATSHFILE parameter of the IDENTIFY or START command was not found or the file could not be opened.

User Response: Verify that you typed the correct filename in the KEYRINGSTASHFILE parameter and that the file permission is set so that the file is readable.

30100 SSL API Error: gsk_environment_open function call

Explanation: The SSL API encountered an error on the gsk_environment_open function call.

User Response: Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30101 SSL API Error: KEYRINGFILE error

Explanation: The SSL API reported a problem with the KEYRINGFILE parameter.

User Response: Check that the KEYRINGFILE parameter is present and correct. Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30102 SSL API Error: KEYRINGFILE error

Explanation: The SSL API encountered an error with the KEYRINGFILE parameter.

User Response: Check that the KEYRINGFILE parameter is present and correct. Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30103 SSL API Error: KEYRINGPASSWORD or KEYRINGSTASHFILE error

Explanation: The SSL API reported a problem with the KEYRINGPASSWORD or the KEYRINGSTASHFILE parameters.

User Response: Check that the KEYRINGPASSWORD or the KEYRINGSTASHFILE parameter is present and correct. Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30104 SSL API Error: Certificate error

Explanation: The SSL API reports a problem with the certificate.

User Response: Check the certificate, turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30105 SSL API Error: CIPHERSUITES error

Explanation: The SSL API encountered a parameter error with the CIPHERSUILTES parameter.

User Response: Check the CIPHERSUILTES parameter to make sure it is correct, turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30106 SSL API Error: SSLVERSION error

Explanation: The SSL API encountered an error with the SSLVERSION parameter.

User Response: Check the SSLVERSION parameter to make sure it is correct, turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30107 SSL API Error: AUTHORIZATION error

Explanation: The SSL API encountered a certificate authorization error.

User Response: Check the certificate to make sure it is correct, turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30108 SSL API Error: Session-type error

Explanation: The SSL API encountered a session-type error.

User Response: Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30109 SSL API Error: gsk_environment_init error

Explanation: The SSL API encountered an error on the gsk_environment_init function call.

User Response: Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30110 SSL API Error: gsk_secure_socket_init error

Explanation: The SSL API encountered an error on the gsk_secure_socket_init function call.

User Response: Turn on the LINK trace and try the program again. If the problem persists, contact a Customer Care representative .

30111 SSL API Error: gsk_secure_soc_read error

Explanation: The SSL API encountered an error on the gsk_secure_soc_read function call.

User Response: Turn on the LINK trace, and then try the program again. If the problem persists, contact the Help Desk.

30119 SSL API Error: Unable to open libgsk6ssl.so

Explanation: The shared object file for the GSKit library, libgsk6ssl.so, could not be opened.

User Response: Ensure that the GSKit is installed properly. Review the GSKit installation instructions (Step 6 on page 11) and the Readme file.

30120 SSL API Error: Unable to locate function in libgsk6ssl.so

Explanation: One or more of the functions in the shared object file, libgsk6ssl.so, could not be located.

User Response: Ensure that the GSKit is installed properly and that the GSKit version number is the version that is supplied by Expedite Base/AIX for RISC System/6000 Version 4 Release 6.

Unexpected and program interrupt errors

This section describes the return codes for program logic errors, commit errors, and unexpected program interrupt errors.

31000 Unexpected condition found.

Explanation: Expedite Base has encountered an unexpected condition.

User Response: Contact the Help Desk. You will be asked to FAX or send your trace file, iebase.trc, to the network for problem determination.

31360 Error indication received from Information Exchange.

Explanation: Expedite Base received an unexpected error from Information Exchange.

User Response: Wait and retry the program later. Make sure there is not another user using this user ID. Make sure you have specified the correct product name in the PRODUCT parameter on the DIAL command. If you omit the PRODUCT parameter, or leave it blank, it defaults to INFOEXCH. Do not change the default unless instructed to do so by GXS Community Support. Also, check to see that you are not trying to send more than 1000 files using session-level recovery. If the problem persists, contact the Help Desk.

31400 Program interrupted.

Explanation: A control-break or re-IPL command interrupted Expedite Base. This error will not appear in your output file but may appear on the display as the restart return code.

User Response: No response needed. This is only display information.

31810 File operation failed on file.

Explanation: A checkpoint recovery file was damaged.

User Response: Check your disk and make sure there are no I/O problems. Check the file permissions to make sure Expedite Base for UNIX has access to the file. Reset the session and retry the program. If the problem persists, contact the Help Desk. You will be asked to FAX or send the trace file, iebase.trc, to the network for problem determination.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

32000 Expedite Base encountered an unexpected condition.

Explanation: Expedite Base has encountered an unexpected condition and is unable to continue.

User Response: Reset the session and retry the program. If the problem persists, contact the Help Desk. You will be asked to FAX or send the trace file, iebase.trc, to the GXS Community Support for problem determination.

ATTENTION: If you reset the session using the -r command line parameter, you will no longer be able to continue the previous session. Failure to modify the message command file, basein.msg, before resetting the session may result in some data being lost or duplicated.

Appendix B

Common data header

Information Exchange interfaces can use a common data header (CDH) to communicate detailed information about files and messages to other interfaces and to Information Exchange. Expedite Base/AIX builds a CDH for every file sent, so you do not have to build a CDH yourself. Expedite Base/AIX also recognizes CDHs received from other interfaces.

The CDH provides details, such as file name, carriage-return, and line-feed options, that let the receiving interface reconstruct a received message into its original format. It also makes more information available to the recipient of the file.

The information in the CDH is presented to you as parameters in the RECEIVED or AVAILABLE records. See *Information Exchange Messages and Formats* for more information on the CDH

Appendix C

Reserved file names and user classes

The following sections contain Expedite Base/AIX reserved file names and user classes.

Reserved file names for PATH

The following are Expedite Base/AIX reserved file names that reside in a directory specified in the PATH statement.

File name:	File description
iebase	The Expedite Base/AIX main executable file.
iebaser	The renamed iebase file, which provides all the functions that iebase provided in the previous version of Expedite Base/AIX.
iebasepr	The file that checks the COMPRESS() parameter, and verifies that the appropriate compression software exists.
iebasepo	The file that checks the message response file, baseout.msg, for compressed files, and verifies that the appropriate decompression software exists.
lu62shrd.o	This file is the SNA dynamic library required for LU 6.2 communications.

NOTE: If you are using supported data compression, see Appendix E, “Using data compression,” for additional reserved names.

Reserved file names for -p parameter

The following are Expedite Base/AIX reserved file names contained in the directory specified in the -p command line parameter.

File name:	File description:
basein.msg	The message command file.
baseout.msg	The message response file.
tempout.msg	The temporary message response file.
basein.pro	The profile command file.
baseout.pro	The profile response file.
ediwork.fil	The file Expedite Base/AIX uses to keep track of which EDI envelopes are sent and which are not sent when using SENDEDI with VERIFY(C) or VERIFY(G) specified.
iebase.pro	The internal profile in which Expedite Base/AIX stores profile command values.
iebase.trc	The trace file in which Expedite Base/AIX places all trace information other than the LINK trace.
rcvfiles.fil	The file containing the names of all files Expedite Base/AIX receives during an Information Exchange session.
rcvofset.fil	The file Expedite Base/AIX uses to track files received since the last checkpoint.
session.fil	The control file Expedite Base/AIX uses to restart an Information Exchange session.
s1dcl.trc	The trace file in which Expedite Base/AIX places LINK trace information.

Reserved file names for IEPATH parameter

The following are Expedite Base/AIX reserved file names contained in the directory specified by the IEPATH parameter in the SESSION profile command.

File name:	File description:
cnnct.scr	The script file Expedite Base/AIX uses to connect to the network.
direct.fil	The file Expedite Base/AIX uses to connect to a customized Service Manager logon screen. This file is used only if you have migrated from expEDIte/AIX Base Release 3 and you are using the old style connect files.
discnnct.scr	The script file Expedite Base/AIX uses to disconnect from the network.
display.scr	The script file Expedite Base/AIX uses to display the transmit picture and status information.
ecnnct.scr	A sample connect script for use in Austria, Belgium, Finland, France, Germany, Israel, Italy, Netherlands, and Sweden.
edcnnct.scr	A sample disconnect script for use in Austria, Belgium, Finland, France, Germany, Israel, Italy, Netherlands, and Sweden.
errormsg.cmp	The file where Comm-Press programs place error messages.
errormsg.fil	The file containing Expedite Base/AIX error message text.
errortxt.fil	The file containing explanations and user response information for all error messages.
IBM3270.XLT	A translate table that performs ASCII EBCDIC translation that matches the IBM eNetwork Personal Communications 4.2 program.
NOXLATE.XLT	A translate table that provides no translation.
QUALTBL.TBL	The file Expedite Base/AIX uses to determine which destination table to use to translate EDI addresses to Information Exchange addresses.
scnnct.scr	A sample connect script for use in Switzerland and Slovenia.
sdcnnct.scr	A sample disconnect script for use in Switzerland and Slovenia.
second.fil	The file Expedite Base/AIX uses to connect to a secondary network. This file is used only if you have migrated from expEDIte/AIX Base Release 3 and you are using the old style connect files.
tracemsg.fil	The file Expedite Base/AIX uses to change the language of trace messages.
ucnnct.scr	A sample connect script for use in Denmark, Norway, South Africa, Spain, and the United Kingdom.
udcnnct.scr	A sample disconnect script for use in Denmark, Norway, South Africa, Spain, and the United Kingdom.
welc3270.scr	The script file Expedite Base/AIX uses to set search strings during logon for 3270 emulation.
xcnnct.fil	The file containing modem dial and connect instructions.
xdcnnct.fil	The file containing modem disconnect instructions.

Reserved user classes

The following are Expedite Base/AIX reserved user classes.

ffmsg001	The user class for a free-format message.
file0001	The user class for file inquiries used with expEDIte/PC. File inquiries are not created with Expedite Base/AIX.
#ec	The default user class for UCS data.
#ee	The default user class for EDIFACT data.
#eu	The default user class for UN/TDI data.
#e2	The default user class for X12 data.

Appendix D

Information Exchange translate table

This appendix provides the Information Exchange ASCII-to-EBCDIC and EBCDIC-to-ASCII translate tables.

ASCII to EBCDIC

When ASCII data is sent through Expedite Base/AIX, it is translated into EBCDIC using this table and stored in Information Exchange as EBCDIC characters.

ASCII:	EBCDIC:	CHARACTER:
00	00	NUL
01	01	SOH
02	02	STX
03	03	ETX
04	04	EOT (Disconnect)
05	05	ENQ (Terminate block)
06	06	ACK (Terminate block)
07	07	BEL
08	08	BS
09	09	HT
0A	0A	LF
0B	0B	VT
0C	0C	FF
0D	0D	CR (Line end)
0E	0E	SO
0F	0F	SI
10	10	DLE

ASCII:	EBCDIC:	CHARACTER:
11	11	DC1/XON
12	12	DC2
13	13	DC3/XOFF (Terminate)
14	14	DC4
15	15	NAK (NACK)
16	16	SYN (SYNCH)
17	17	ETB
18	18	CAN
19	19	EM (ENDBLOCK)
1A	1A	SUB
1B	1B	ESC
1C	1C	FS (HEXBEGIN)
1D	1D	GS (REPEAT)
1E	1E	RS
1F	1F	US (HEXEND)
20	40	(blank)
21	5A	!
22	7F	“
23	7B	#
24	5B	$
25	6C	%
26	50	&
27	7D	'
28	4D	(
29	5D	)
2A	5C	*
2B	4E	+
2C	6B	,
2D	60	

EBCDIC to ASCII

When EBCDIC data is received through Expedite Base/AIX, it is translated into ASCII using this table.

EBCDIC:	ASCII:	CHARACTER:
00	00	NUL
01	01	SOH
02	02	STX
03	03	ETX
04	04	EOT (Disconnect)
05	05	ENQ (Terminate block)
06	06	ACK (Terminate block)
07	07	BEL
08	08	BS
09	09	HT
0A	0A	LF
OB	OB	VT
OC	OC	FF
OD	OD	CR (Line end)
0E	0E	S0
0F	0F	SI
10	10	DLE
11	11	DC1/XON
12	12	DC2
13	13	DC3/X0FF (Terminate)
14	14	DC4
15	15	NAK (NACK)
16	16	SYN (SYNCH)
17	17	ETB
18	18	CAN
19	19	EM (ENDBLOCK)
1A	1A	SUB
1B	1B	ESC
1C	1C	FS (HEXBEGIN)
1D	1D	GS (REPEAT)
1E	1E	RS
1F	1F	US (HEXEND)
20	80	
21	81	

EBCDIC:	ASCII:	CHARACTER:
22	82	
23	83	
24	84	
25	8E	
26	87	
27	8F	
28	88	
29	89	
2A	8A	
2B	8B	
2C	8C	
2D	8D	
2E	86	
2F	7F	DEL (Terminate block)
30	90	
31	91	
32	97	
33	93	
34	94	
35	95	
36	96	
37	9C	
38	98	
39	99	
3A	9A	
3B	9B	
3C	9D	
3D	85	
3E	9E	
3F	92	
40	20	SP / Blank
41	A0	
42	A1	
43	A2	
44	A3	
45	A4	
46	A5	

EBCDIC:	ASCII:	CHARACTER:
47	A6	
48	A7	
49	A8	
4A	D5	
4B	2E	.
4C	3C	<
4D	28	(
4E	2B	+
4F	7C	|
50	26	&
51	A9	
52	AA	
53	AB	
54	AC	
55	AD	
56	AE	
57	AF	
58	B0	
59	B1	
5A	21	!
5B	24	$
5C	2A	*
5D	29	)
5E	3B	;
5F	5E	¬ CIRCUMFLEX / NOT
60	2D	HYPHEN / MINUS
61	2F	/
62	B2	
63	B3	
64	B4	
65	B5	
66	B6	
67	B7	
68	B8	
69	B9	
6A	E5	
6B	2C	,

EBCDIC:	ASCII:	CHARACTER:
6C	25	%
6D	5F	_ UNDERSCORE
6E	3E	>
6F	3f	?
70	BA	
71	BB	
72	BC	
72	BD	
74	BE	
75	BF	
76	C0	
77	C1	
78	C2	
79	60	‘ GRAVE ACCENT
7A	3A	:
7B	23	#
7C	40	@
7D	27	‘
7E	3D	=
7F	22	“
80	C3	
81	61	a
82	62	b
83	63	c
84	64	d
85	65	e
86	66	f
87	67	g
88	68	h
89	69	i
8A	C4	
8B	C5	
8C	C6	
8D	C7	
8E	C8	
8F	C9	
90	CA	

EBCDIC:	ASCII:	CHARACTER:
91	6A	j
92	6B	k
93	6C	l
94	6D	m
95	6E	n
96	6F	o
97	70	p
98	71	q
99	72	r
9A	CB	
9B	CC	
9C	CD	
9D	CE	
9E	CF	
9F	D0	
A0	D1	
A1	7E	-
A2	73	s
A3	74	t
A4	75	u
A5	76	v
A6	77	w
A7	78	x
A8	79	y
A9	7A	z
AA	D2	
AB	D3	
AC	D4	
AD	5B	OPEN BRACKET
AE	D6	
AF	D7	
B0	D8	
B1	D9	
B2	DA	
B3	DB	
B4	DC	
B5	DD	

EBCDIC:	ASCII:	CHARACTER:
B6	DE	
B7	DF	
B8	EO	
B9	E1	
BA	E2	
BB	E3	
BC	E4	
BD	5D	
BE	E6	
BF	E7	
CO	7B	{ OPEN BRACE
C1	41	A <UPPERCASE>
C2	42	B
C3	43	C
C4	44	D
C5	45	E
C6	46	F
C7	47	G
C8	48	H
C9	49	I
CA	E8	
CB	E9	
CC	EA	
CD	EB	
CE	EC	
CF	ED	
D0	7D	} CLOSE BRACE
D1	4A	J
D2	4B	K
D3	4C	L
D4	4D	M
D5	4E	N
D6	4F	0
D7	50	P
D8	51	Q
D9	52	R
DA	EE	

EBCDIC:	ASCII:	CHARACTER:
DB	EF	
DC	FO	
DD	F1	
DE	F2	
DF	F3	
E0	5C	\ REVERSE SLASH
E1	9F	
E2	53	S
E3	54	T
E4	55	U
E5	56	V
E6	57	W
E7	58	X
E8	59	Y
E9	5A	Z
EA	F4	
EB	F5	
EC	F6	
ED	F7	
EE	F8	
EF	F9	
F0	30	0
F1	31	1
F2	32	2
F3	33	3
F4	34	4
F5	35	5
F6	36	6
F7	37	7
F8	38	8
F9	39	9
FA	FA	
FB	FB	
FC	FC	
FD	FD	
FE	FE	
FF	FF	

Appendix E

Using data compression

Expedite Base/AIX provides integrated data compression and decompression through the Comm-Press product, which may not be available in all countries.

Compression reduces the size of the files that are transmitted through Information Exchange. Significant savings in network charges and transmission time are possible when using the data compression supplied by Comm-Press, Inc.

In the United States, you can contact your GXS sales representative for Comm-Press product ordering information.

When you use data compression, some Expedite Base/AIX parameters are impacted, and session restarts may have to be handled differently. These considerations are described in this appendix.

NOTE: When using COMPRESS(Y) or COMPRESS(T), both the sender and receiver of compressed data must have the licensed Comm-Press product in order to compress and decompress the data.

Understanding the Comm-Press files used with Expedite Base/AIX

The *iebaser*, *iebase*, *iebasepr*, and *iebasepo* programs and the errormsg.cmp error message file are shipped as part of the basic Expedite Base/AIX product. These files are described in "Understanding Expedite Base/AIX files" on page 2-7.

The following files are also provided when you use the Comm-Press product with Expedite Base/AIX. These are reserved file names, as described in Appendix C, "Reserved file names and user classes."

- *inmsgp*

  This compression program reads the basein.msg file, compresses the appropriate files based on the COMPRESS(Y) and COMPRESS(T) parameters, and builds a baseinc.msg file reflecting the location of the compressed files for transmission. This file must be in the same subdirectory as the other executable Expedite Base/AIX files.

- *outmsgp*

  This decompression program reads the baseout.msg file after the *iebaser* program has received the files, then decompresses any received compressed files. This file must be in the same subdirectory as the other executable Expedite Base/AIX files.

- CPLOOKUP.TBL

  This is a sample compression lookup table that is used when COMPRESS(T) is specified. It indicates whether compression should be performed for a particular sender/receiver pair. This file is one of those affected by the IEPATH parameter of the SESSION command (see "Reserved file names for IEPATH parameter" on page 461).

These files are distributed in *tar* format on the Comm-Press installation diskette. To install the Comm-Press files, change to the subdirectory where you previously installed Expedite Base/AIX and issue the following tar command:

```
tar xvf /dev/fd0
```

The following temporary files are created when using Expedite Base/AIX to compress files:

- baseinc.msg

  The *inmsgp* program builds this file as files are compressed. The baseinc.msg file contains all the entries found in basein.msg, plus the name and location of the corresponding compressed files. The *iebaser* program reads baseinc.msg to determine what commands to process. The baseinc.msg file is deleted upon successful completion of Expedite processing. However, if you are using a data recovery method other than session-level recovery and your Information Exchange session does not complete successfully, baseinc.msg is not deleted, and must remain unchanged for successful recovery processing.

- baseinr.msg

  This file is used in recovery situations to synchronize basein.msg and baseinc.msg. It is deleted after *inmsgp* processing ends successfully.

- baseoutr.msg

  This file is used during *outmsgp* processing. It is deleted when *outmsgp* processing ends successfully.

- baseoutc.msg

  When a data compression error condition occurs during *inmsgp* and *outmsgp* processing, baseoutc.msg contains the error message text.

- iecomp.trc

  This file is created when BASE(Y) is specified on the TRACE command.

- ˜IEn and ˜CMn files

  These temporary files have names that begin with the characters ˜IE or ˜CM and are created in the AIX temporary directory.

Normally, the temporary files are deleted as part of successful Expedite processing. However, when your Information Exchange session does not complete successfully, the temporary files remain. You should not delete, move, or rename these files. You should instead complete the Information Exchange session or allow RESET processing to remove the temporary files.

Compressing files with COMPRESS(Y)

When COMPRESS(Y) is included on the SEND or SENDEDI command and *inmsgp* is available, the specified file is compressed prior to transmission. If the file to be sent contains multiple EDI envelopes, all envelopes are sent compressed. (The EDI header itself is not compressed.) Additional parameters appear in the baseout.msg file along with the echoed SEND command:

```
COMSW(COMM-PRESS) COMVER(3ð1) COMFILE(/˜IEn)
```

where COMSW refers to the software providing the compression, COMVER refers to the version and release of that software, and COMFILE refers to the directory and E-2 Expedite Base/AIX for RISC System/6000 Programming Guide file name for the compressed file to be sent. This information is also contained in the CDH, to allow proper decompression at the receiving locations.

Compressing files with COMPRESS(T)

When COMPRESS(T) is included on the SEND or SENDEDI command, the specified file is compressed prior to transmission only if the SENDER, RECEIVER, and COMPRESS parameters are listed in the CPLOOKUP.TBL file. This file defines a series of paired receivers and senders, and for each pair indicates whether compression should be performed.

Each entry in the compression lookup table must follow this format:

```
sender(sender) receiver(receiver) compress(y|n);
```

sender

Indicates the account and user ID or EDI source of a sender.

receiver

Indicates the account and user ID, alias and alias name, list name, or EDI destination of a receiver.

compress

Indicates whether compression should be performed for this sender/receiver pair.

y	Compress the data for this sender/receiver pair.
n	Do not compress the data for this sender/receiver pair.

The following is an example of a compression lookup table:

```
SENDER(acct1 userð1) RECEIVER(acct1 userð2) COMPRESS(y);
SENDER(acct1 userð1) RECEIVER(aliasð1 aliasð2) COMPRESS(y);
SENDER(acct1 userð1) RECEIVER(acct1 userð3) COMPRESS(y);
SENDER(acct1 userð1) RECEIVER(listnameð2) COMPRESS(y);
```

For each SEND command, the *inmsgp* program identifies the sender from a START command in basein.msg or from an IDENTIFY command if AUTOSTART(Y) is specified in basein.pro. The *inmsgp* program identifies the receiver from the ACCT and USERID, ALIAS and ALIASNAME, or LISTNAME parameters of the SEND command.

When entering values for the SENDER and RECEIVER parameters, the ACCT or ALIAS must be 7 characters long (padded with blanks, if necessary). The USERID or ALIASNAME must begin in the next character position.

For the SENDEDI command, *inmsgp* identifies the sender and receiver from the EDI header. The *inmsgp* program looks for a corresponding entry in the CPLOOKUP.TBL file. The Comm-Press product supports X12, UCS, EDIFACT and UN/TDI EDI formats. The SENDER and RECEIVER entries for EDI data must match exactly what appears in the appropriate field of the EDI header. The *inmsgp* program examines only the EDI header to resolve sender/receiver pairs. Refer to Chapter 6, “Sending and receiving EDI data,” for a description of which EDI header fields are used to identify the EDI source and destination.

The COMPRESS(T) parameter and the CPLOOKUP.TBL file allow you to control what gets compressed, based on the receiver. The CPLOOKUP.TBL file, like the basein.pro and basein.msg files, can be edited and modified when the iebase program is not running.

Decompressing received compressed files

When compressed files are received, baseout.msg contains new parameters used by the *outmsgp* program to process the received compressed files. The outmsgp program reads the baseout.msg file and decompresses the data. The following new parameters are found in the baseout.msg file for each compressed file received:

```
COMPRESS(Y) COMSW(COMM-PRESS) COMVER(301) COMFILE(xxx)
DCMPRC(00000)
```

All of the parameters except DCMPRC are obtained from the CDH. When the received file is not compressed, none of the data compression parameters listed above appear in the baseout.msg entry.

The *outmsgp* program copies the baseout.pro file to the baseoutr.msg file, then processes and copies each response record back into baseout.msg. When a RECEIVED response record indicates compressed data, *outmsgp* decompresses the data in the received file into the ˜IEn file. Any uncompressed messages contained in the received file are also copied to the ˜IEn file. After successful decompression, the received file is deleted, and the temporary file is renamed to the received file name.

The DCMPRC parameter in the RECEIVED response record provides a Comm-Press return code to the user, so that successful decompression processing can be verified or an existing error condition reported to the user. The DCMPRC parameter value of 'ðððð' is the desired result of processing incoming compressed data. If DCMPRC is not 'ðððð', refer to the related error messages in the baseoutc.msg file and the error descriptions at the end of this appendix to determine the appropriate action. Error messages are also written to the baseoutc.msg file.

Expedite Base/AIX considerations when using COMPRESS(Y) or COMPRESS(T)

Most of the command parameters, when used with the COMPRESS parameter, are supported as documented in this publication. However, the following command parameters function differently when they are used with the COMPRESS(Y) and COMPRESS(T) parameters.

DATATYPE(A|B) on SEND commands

The *inmsgp* program uses the DATATYPE parameter during the compression process to determine whether ASCII or EBCDIC translation should be performed. After compression, the data is sent with DATATYPE(B).

DELIMITED(N|Y) on SEND commands

The *inmsgp* program uses the DELIMITED parameter during the compression process to determine whether the data is delimited with line-feed characters. After compression, the data is sent with DELIMITED(N).

TRANSLATE(translate table) on SEND, SENDEDI, RECEIVE, and RECEIVEEDI commands

This parameter is not supported with COMPRESS(Y) or COMPRESS(T) and results in an error if specified on the SEND command (it is ignored on the RECEIVE command). Data is always translated from ASCII to EBCDIC during the compression process using the standard Expedite/Base translate table.

RECORDSIZE(record size) on RECEIVE and RECEIVEEDI commands

Expedite Base/AIX ignores this option for compressed EDI data.

PROCESSLEN(C|R|I) on RECEIVE commands

Expedite Base/AIX ignores this option when receiving compressed data. If the data was compressed as DELIMITED(Y), then line-feed delimiters are inserted in the data in their original locations.

FORMAT(Y)

The FORMAT(Y) parameter is not supported. If FORMAT(Y) is specified on a SEND or SENDEDI command, an error is returned. If FORMAT(Y) is specified on a RECEIVE or RECEIVEEDI command, the parameter is ignored.

EDIOPT(F)

The EDIOPT(F) parameter is not supported.

The Comm-Press programs read the basein.pro file to determine the values for the RECOVERY, TRACE, and IEPATH parameters. For this reason, basein.pro must exist for successful compression and decompression processing.

For EDI data sent using the SENDEDI command, the character used as the segment terminator must not be in the range X'E0' to X'FF'. Also, line feed characters that appear within segments are not removed. Instead, these characters are compressed as part of the data.

Restart and recovery considerations with Comm-Press

The logic used in restart and recovery situations, described earlier in this publication, applies to restart and recovery situations where some or all of the files sent and received are compressed.

Because *inmsgp* processes the basein.msg file before Expedite Base/AIX processing takes place, *inmsgp* must determine whether a restart situation exists before it does any processing. If the baseinc.msg file exists, *inmsgp* assumes that a restart is required.

The *inmsgp* program restarts by comparing the basein.msg and baseout.msg files. All commands that were echoed to the baseout.msg file are left unchanged in the baseinc.msg file and are not reprocessed. Remaining commands in the basein.msg file are processed (that is, any requested compression is performed) and copied to the baseinc.msg file.

Because of the way *inmsgp* uses the baseinc.msg file to determine restart status, you should be careful to avoid using an old baseinc.msg file that might contain data. Unpredictable results would occur. You may want to use the -r run-time parameter each time you run the iebase program, except for known restart situations. Specifying the -r parameter causes all basein.msg commands to be processed.

Restarting the session when using the COMPRESS parameter: In restart situations, you cannot change certain Expedite Base/AIX files, such as basein.msg and baseout.msg. You also cannot change the baseinc.msg file. The program uses the baseinc.msg file when it processes compressed files.

Restarting the session after modifying basein.msg: As noted earlier in this publication, you can modify entries in basein.msg that are in error and restart the session. However, you cannot modify basein.msg entries that have already been echoed to the baseout.msg file.

Resetting the session: When you reset the session, the baseinc.msg file is erased.

Restarting the session for outmsgp processing: When the DCMPRC parameter indicates an error condition, you may be able to correct the error condition and restart *outmsgp* processing to decompress received files. To restart just the decompression processing, specify the **-d** parameter with *iebase* on the operating system command line:

```
iebase -d
```

Decompressing files independently of Expedite Base/AIX: In some situations, you may want to manually decompress compressed files you received from Information Exchange. Instead of decompressing the files through Expedite Base/AIX, you can invoke the *decompx* program directly:

```
find . -name filename.ext -print | decompx output.fil
```

If the input file contains compressed EDI data, enter the following command:

```
find . -name filename.ext -print | decompx output.fil edi
```

The *decompx* program is included on the Comm-Press installation diskette. See the *Comm-Press User's Guide* for more information.

Error messages and return codes for data compression

The *inmsgp* and *outmsgp* programs write error messages to baseoutc.msg for error conditions that require user intervention. The programs set the error level to 115. The *outmsgp* program also updates the DCMPRC field in the RECEIVED response record to indicate the results of decompressing that received message.

The programs issue error messages when a condition occurs that prevents further processing. When *inmsgp* and *outmsgp* encounter an error condition, they write an appropriate error message to the baseoutc.msg file.

Some decompression errors do not prevent further processing, and only result in a non-zero value being placed in the DCMPRC field of the appropriate RECEIVED response record. For example, assume that a compressed message is corrupted during transmission. The cyclic-redundancy check (CRC) would fail, resulting in a DCMPRC return code of 26015. Processing would continue with the next RECEIVED response record. And, if no other session errors occurred, the SESSIONEND return code would be changed to 28010, indicating that not all commands were processed successfully.

26001 Error allocating memory.

Explanation: A request for storage failed.

User Response: Close some applications and try again. See Restart and Recovery Considerations with Comm-Press for more information.

26003 Invalid use of COMPRESS parameter.

Explanation: COMPRESS(Y) or COMPRESS(T) was specified on a SEND or SENDEDI command, but the Comm-Press data compression software was not found.

User Response: Contact your marketing representative to acquire the Comm-Press data compression software.

26004 Unable to decompress received files.

Explanation: Compressed files were received, but the Comm-Press data compression software was not found.

User Response: Contact your marketing representative to acquire the Comm-Press data compression software.

26005 Invalid value for COMPRESS parameter.

Explanation: An invalid value was specified for the COMPRESS parameter.

User Response: Valid values are 'E', 'N', 'Y', 'T', and 'V'. Consult the Expedite Base and Comm-Press User Guides for instructions regarding the use of the COMPRESS parameter.

26006 FORMAT parameter not valid with COMPRESS(Y).

Explanation: The FORMAT and COMPRESS parameters were both specified on a SEND command.

User Response: FORMAT is not supported when using compression. Remove one of the parameters.

26007 Cannot compress for reserved message class %s.

Explanation: An invalid message class was specified.

User Response: The message class is a reserved class for use by the STEDI system. These files cannot be compressed.

26008 TRANSLATE parameter not valid with COMPRESS(Y).

Explanation: The TRANSLATE and COMPRESS parameters were both specified on a SEND or SENDEDI command.

User Response: TRANSLATE is not supported when using compression. Remove one of the parameters.

26009 Compressed segment not found in file.

Explanation: The RECEIVED response record indicates compressed data was received; however, no compressed data was found in the received file.

User Response: The file has probably been corrupted during transmission. Receive the file again.

26010 Error creating temporary file name.

Explanation: An error occurred creating a temporary file name.

User Response: Either the TMP environment variable is set to a non-existent directory, or the directory is full. Verify the setting or delete the temporary files in the directory.

26012 Trial period has expired.

Explanation: The Comm-Press trial period has expired.

User Response: Contact your marketing representative to obtain a permanent software licence.

26015 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26017 Restricted license violation.

Explanation: You have a restricted version of the software that is limited to use with a certain trading partner.

User Response: Contact your marketing representative to obtain an unrestricted license for the Comm-Press data compression software.

26018 Invalid EDI envelope.

Explanation: The EDI header or trailer does not conform to the EDI standard, or a premature end-of-file condition was encountered on input. This error can also be caused by an invalid segment terminator in the EDI header.

User Response: Verify valid EDI headers and trailers are present. The segment terminator must be less than X'EO'.

26020 Error opening input file: %s.

Explanation: An error occurred opening the file for compression or decompression.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26021 Error reading input file: %s.

Explanation: An error occurred reading the file during compression or decompression.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26024 Error positioning input file %s.

Explanation: An error occurred positioning the file during compression or decompression.

User Response: Examine the following message to determine the cause of the error.

26030 Error opening output file: %s.

Explanation: An error occurred opening the file for compression or decompression.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26032 Error writing output file: %s.

Explanation: An error occurred writing the file during compression or decompression.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26033 Error closing output file: %s.

Explanation: An error occurred closing the file after compression or decompression.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26040 Error opening work file.

Explanation: A file error occurred.

User Response: Examine the operating system error message to determine the cause of the error. Correct the problem and restart the session.

26041 Error reading work file.

Explanation: A file error occurred.

User Response: Examine the operating system error message to determine the cause of the error. Correct the problem and restart the session.

26042 Error writing work file.

Explanation: A file error occurred.

User Response: Examine the operating system error message to determine the cause of the error. Correct the problem and restart the session.

26050 Error opening 'encrypt.key' file.

Explanation: An error occurred opening the file containing the encryption key.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26051 Error reading 'encrypt.key' file.

Explanation: An error occurred reading the file containing the encryption key.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26055 Error decrypting input file: %s.

Explanation: The received file did not decrypt successfully.

User Response: This is most likely due to an incorrect decryption key specified in the 'encrypt.key' file. Correct the key and rerun IEBASE. See Restart and Recovery Considerations with Comm-Press for more information.

26060 Error opening file: %s.

Explanation: An error occurred opening the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26061 Error reading file: %s.

Explanation: An error occurred reading the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26062 Error writing file: %s.

Explanation: An error occurred writing the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

Explanation: An error occurred closing the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26064 Error positioning file: %s.

Explanation: An error occurred positioning the file during compression or decompression.

User Response: Examine the following message to determine the cause of the error.

26066 EOF reached before end of command.

Explanation: End-of-file was reached while processing a SEND or SENDEDI command.

User Response: The command is not terminated by a semicolon. Correct the command and rerun IEBASE.

26067 Error removing file: %s.

Explanation: An error occurred deleting the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26068 Error renaming file: %s.

Explanation: An error occurred renaming the indicated file.

User Response: Examine the following message to determine the cause of the error. See Restart and Recovery Considerations with Comm-Press for more information.

26091 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26092 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26093 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26094 Compressed segment in error.

User Response: The error checking routine indicated the compressed data was corrupted. User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26095 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26096 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26097 Compressed segment in error.

Explanation: The error checking routine indicated the compressed data was corrupted.

User Response: The data was probably corrupted during transmission. The data has been removed from the received file and must be received again.

26098 Invalid Comm-Press version.

Explanation: The received file was compressed with a newer version of the Comm-Press software.

User Response: Contact your marketing representative to acquire the latest version of the Comm-Press data compression software.

Glossary

This glossary defines words as they are used in this book. It includes terms and definitions from the IBM Dictionary of Computing (New York: McGraw-Hill, 1994). If you are looking for a term and cannot find it here, see the Dictionary of Computing for additional definitions.

A

account. A set of users who work for the same company.

account name. The name assigned to a group of users.

acknowledgment. A response from Information Exchange that tells you whether files were delivered, received, purged, or various combinations of the three.

address. A user's account name and user identification (ID) that Information Exchange uses to route files.

Advanced Interactive Executive (AIX). An operating system that serves as an interface between users and Information Exchange.

AIX. Advanced Interactive Executive.

alias name. An alternate name used in place of an account and user ID.

alias table. A nickname file kept in Information Exchange.

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks.

American National Standard Code for Information. Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

American National Standards Institute (ANSI). An organization consisting of procedures, consumers, and general interest groups, that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

ANSI. American National Standards Institute.

ANSI X12. A data standard used by many industries for EDI and supported by IBM's EDI services.

API. Application Program Interface.

Application Program Interface (API). A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

archive. A copy of one or more files or a copy of a database that is saved for future reference or for recovery purposes in case the original data is damaged or lost.

ASCII. American National Standard Code for Information Interchange.

asynchronous. A protocol that permits a communication device to operate in an unsynchronized and unpredictable manner, much like a human conversation; used for modems and low-speed ASCII terminals (PCs).

attribute. A property or characteristic of one or more entities; for example, length, value, color, or intensity.

audit trail. Data, in the form of a logical path linking a sequence of events, used for tracing the transactions that have affected the contents of a record.

B

binary. Pertaining to files, such as an executable computer program, that contain machine instructions that a person cannot read or enter from a computer keyboard.

C

carriage-return and line-feed characters (CRLF). A word processing formatting control that moves the printing or display point to the first position of the next line.

CDH. Common data header.

centralized alias table. Permanent tables that reside in Information Exchange and contain a centralized list of addresses. You can put a listing of your trading partners' addresses in this table instead of maintaining destination tables in multiple locations. A centralized alias table enables Information Exchange to resolve destinations because it contains a list of EDI destinations paired with Information Exchange destinations. Expedite Base/AIX searches this table for an EDI destination and then uses the corresponding Information Exchange destination as the actual address.

certificate. A digital document that binds a public key to the identity of the certificate owner; thereby, enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority (CA).

certificate authority (CA). A digital document that binds a public key to the identity of the certificate owner; thereby, enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority (CA).

character. A letter, digit, or other symbol that is used as part of the organization, control, or representation of data.

checkpoint-level recovery. A method of restart and recovery within Expedite Base/AIX. A point where information about the status of a job can be recovered so that the job step can be restarted later.

command. A request from a terminal for the performance of an operation or the execution of a particular program.

command file. A file that contains Expedite Base/AIX commands. There are two Expedite Base/AIX command files, profile command and message command. Place commands pertaining to your profile in basein.pro (profile command file). Place commands pertaining to the transfer of files or information in basein.msg (message command file).

command line. On a display screen, a display line in which only commands can be entered.

commit. The point at which a file is either delivered, canceled, or purged. When a session fails, all uncommitted files are lost.

common data header (CDH). A set of control information about a file. Expedite Base/AIX builds a CDH for every file sent. The CDH information is sent with the file to Information Exchange. When the file is received by the trading partner, the receiving interface can use the information in the CDH.

compression. The process of eliminating gaps, empty fields, and redundant data to shorten the length of files.

CRLF. Carriage-return and line-feed characters.

D

default value. A value assumed when no value has been specified.

delivery acknowledgment. An acknowledgment that Information Exchange generates when a destination user receives a file from an Information Exchange mailbox.

dial connection. A connection between a terminal and a telecommunications device over a switched line, initiated by using a dial or push-button telephone.

distribution list. A list of the addresses of users with whom a certain user communicates. It is used to send files to several people at one time instead of having to send the same files many times.

E

EBCDIC. Extended binary-coded decimal interchange code.

EDI. Electronic data interchange.

EDI destination table. A list of EDI destinations paired with Information Exchange destinations used by Expedite Base/AIX.

EDI envelope. A group of EDI transactions with a single destination address.

EDIFACT. Electronic data interchange for administration, commerce, and transport.

electronic data interchange (EDI). The exchange of data and documents between different users according to standardized rules.

Electronic Data Interchange for Administration, Commerce, and Transport (EDIFACT). An EDI standard for the fields of administration, commerce, and transportation.

electronic mail (e-mail). Correspondence in the form of files transmitted between user terminals over a computer network.

electronic mailbox. Synonym for mailbox.

e-mail. Electronic mail.

emulator. A combination of programming techniques and special machine features that permits a computing system to execute programs written for a different system.

ESO. Extended Security Option.

extended binary-coded decimal interchange code. (EBCDIC). A coded character set consisting of 8-bit coded characters.

Extended Security Option (ESO). An option you can specify in your profile for stricter password security.

extended security users. Users with stricter security requirements, such as levels of password protection.

F

field. An area of a panel reserved for data of a certain type or length.

file. A named set of records stored or processed as a unit.

file-level recovery. A method of restart and recovery within Expedite Base/AIX; checkpoints are taken for each file sent and received.

G

global alias. An alias that can be used by any Information Exchange user on a particular system.

global alias table. (1) A system-wide alias table. (2) An alternative name table set up within a system.

H

host system. The controlling or highest level system in a data communication configuration; for example, a System/38 is the host system for the workstations connected to it.

I

Information Exchange. A commerce engine of IBM Application Services - EDI Services for e-business that users can use to send and receive information electronically. (2) A continuously running CICS application on a managed network that stores and forwards information to trading partners.

Information Exchange Administration Services. An online, panel-driven product that the Information Exchange Service Administrator uses to perform administrative tasks for Information Exchange.

Information Exchange Service Administrator. The person who coordinates the use of Information Exchange in a company.

L

leased lines. A connection between systems or devices that does not have to be made by dialing.

library. A place to store information for an extended period of time. A library consists of a collection of files called library members.

library member. A named collection of records or statements in a library.

M

mailbox. A file that holds the electronic mail. Synonymous with electronic mailbox.

managed network. A worldwide communications network infrastructure, such as those provided by AT&T Global Network Services or the Advanced Network eXchange (ANX). Managed networks are also commonly referred to as value-added networks (VANs).

member. See library member.

message command. A command that pertains to the transferring of files or data. Place message commands in basein.msg (message command file).

message command file. A file in which you place commands that pertain to the transferring of files or data. In Expedite Base/AIX, this file is basein.msg.

message group. A collection of messages that is treated as a single entity by Information Exchange; for example, a file of records to be printed as a single report.

message response file. A file that contains the Expedite Base/AIX and Information Exchange replies to certain message commands. In Expedite Base/AIX, this file is baseout.msg. Expedite Base/AIX generates this file after it processes the message command file (basein.msg). The message response file contains return codes such as error messages and completion codes.

modem. A device that converts digital data from a computer to an analog signal that can be transmitted on a telecommunication line, and converts the analog signal received to data for the computer.

O

organizational alias. (1) An alias that can be used by any user in an account. (2) A company-wide alias table.

organizational alias table. An alias table set up within an account on Information Exchange.

P

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (3) Data passed between programs or procedures.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it.

path. The subdirectory in which a file is located. permanent distribution list. A distribution list that is stored permanently in Information Exchange.

private alias. An alias that can be used only by the user who created it.

private alias table. An alias table set up for an individual user.

profile command. A command that pertains to profile specific information, such as your password, user ID, and account. Place profile commands in basein.pro

profile command file. A file in which you place profile commands pertaining to profile specific information. In Expedite Base/AIX, this file is basein.pro. profile response file. A file that contains the Expedite Base/AIX and Information Exchange replies to certain profile commands. In Expedite Base/AIX, this

file is baseout.pro. Expedite Base/AIX generates this file after processing the profile command file (basein.pro). The profile response file contains return codes such as error messages or completion codes.

purge acknowledgment. An acknowledgment Information Exchange generates when a file is purged from the receiver's mailbox.

Q

qualifier table. A list of EDI data types (X12, UCS, EDIFACT, or UN/TDI) paired with the ID qualifier for a particular type of data (for example, 01 for an X12 DUNS number).

R

receipt acknowledgment. An acknowledgment Information Exchange generates when a file reaches the receiver's mailbox after a successful Expedite Base/AIX session.

reserved files. Files that enable Expedite Base/AIX to perform various session tasks. Expedite Base/AIX uses these files to track and store session information, provide optional session information, and control session function.

reset. To start a session at the beginning of a command file when the session ends in error and you do not want Expedite Base/AIX to continue it.

response file. A file that contains the Expedite Base/AIX and Information Exchange replies to certain commands. Expedite Base/AIX generates three response files: profile response, message response, and response work. In response to profile commands, Expedite Base/AIX generates baseout.pro (profile response file). In response to message commands, Expedite Base/AIX generates baseout.msg (message response file). In response to commands processed since the last Information Exchange checkpoint, Expedite Base/AIX generates tempout.msg (response work file). Response files contain return codes, such as error messages or completion codes.

response work file. A file that contains response information processed since the last Information Exchange checkpoint. In Expedite Base/AIX, this file is tempout.msg.

restart. To resume a session at the last checkpoint, when the session ends in error and you want Expedite Base/AIX to continue it.

S

Secure Socket Layer . A secure method of communicating by way of the Internet.

service administrator. A primary contact person in your organization for various Information Exchange support groups. See also Information Exchange Service Administrator.

Service Manager. A product offered by AT&T Global Network for customer administrators with responsibility to assist and manage users within customer accounts.

session. The period of time during which a user of a terminal can communicate with an interactive system, usually, elapsed time between logon and logoff.

session-level recovery. A method of restart and recovery within Expedite Base/AIX; no files are committed until the session ends normally.

SNA. Systems Network Architecture. string. A sequence of elements of the same nature, such as characters, considered as a whole.

SSL. See *Secure Socket Layer.*

syntax. The structure of expressions in a language.

system ID. The name that a manage network assigns to a system.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information, that is, the users, to be independent of and unaffected by the specific SNA network services and facilities used for information exchange.

T

TCP/IP. Transmission Control Protocol/Internet Protocol.

temporary distribution list. A distribution list that lasts only for the duration of your Information Exchange session.

trading partners. The business associates with whom users exchange information electronically.

Transmission Control Protocol/Internet Protocol. (TCP/IP). A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

U

UCS. Uniform Communication Standard.

Uniform Communication Standard (UCS). A standard EDI format used in the grocery industry. United Nations/Trade Data Interchange (UN/TDI). An EDI standard for administration, commerce, and transportation fields developed by the United Nations Economic Commission for Europe.

UN/TDI. United Nations/Trade Data Interchange.
user-initiated recovery. A method of restart and recovery within Expedite Base/AIX; checkpoints are taken after each COMMIT command, unless there is nothing to commit.

user class. A name that users can assign to their documents to identify them to trading partners.

user ID. A name that identifies a user to Information Exchange, within an account.

user message class. A category used to group mail. This category is agreed on among trading partners.

user profile. A description of a user that includes such information as password, user ID, and account. It also contains the characteristics of how a user works with Information Exchange.

W

wildcard character. A synonym for pattern-matching character.

X

X12. A specially formatted data stream.

Index

Q

R

S

T

U

V

W