DRAFT - Technical Report CSL/SNA
April 11, 1994

# Electronic Data Interchange Communications Platform Operations Manual

Paul Markovitz John Garguilo

U. S. Department of Commerce
National Institute of Standards and Technology
Computer Systems Laboratory
Systems and Network Architecture Division
Gaithersburg, MD 20899, USA

This work is a contribution
of the National Institute of Standards and Technology,
and it is not subject to copyright.

# Contents


**1 Introduction** **3**

**2 Overview** **5**
2.1 X.400 . . . 6
2.2 FTAM . . . 7
2.3 Dial-up . . . 7

**3 Installation** **8**

**4 Configuration** **11**
4.1 Local EDI User Configuration . . . 11
4.2 Trading Partner Configuration . . . 15
4.2.1 Pedi Example File . . . 16
4.2.2 P2 Example File . . . 18
4.2.3 FTAM Example File . . . 19
4.2.4 DIAL-UP Example File . . . 20
4.3 VAN Configuration . . . 20
4.3.1 XMODEM_STARTUP_DEFINITIONS . . . 20
4.3.2 VAN Script File . . . 26

**5 Operations** **28**
5.1 UA . . . 29
5.2 FTAM . . . 30
5.2.1 Sending FTAM Application . . . 30
5.2.2 Receiving FTAM Application . . . 30
5.3 VANs . . . 31
5.3.1 Sending Dial-up Application . . . 31
5.3.2 Receiving Dial-up Application . . . 31

**6 Conclusion** **32**

**Appendix A: Supported X.435 Services** **33**

**Appendix B: Definitions for Frequently Used Expressions** **35**

# List of Figures

2.1 NIST EDI Communications Platform . . . . . 5

3.1 Platform File Directory Structure . . . . . 9

# Chapter 1

# Introduction

The need to exchange information is critical within the business community. The information may be generic in nature, such as a purchase order or invoice, or specific to an organization, such as a customs declaration. Traditionally, paper copies of these documents have been mailed between companies. With many companies incorporating computers into their business practices, transferring the information electronically provides many advantages, such as reducing paperwork and transcription errors, minimizing cost, and improving response time. One of the major components of the electronic transmission of business data is Electronic Data Interchange (EDI).

EDI involves the exchange of data between two companies or *trading partners*. Information units, called EDI *interchanges*, are transmitted electronically between the trading partners. These interchanges are not structured to be human-readable, but rather to trigger responses from a computer application. For example, receipt of a shipping notice may result in the automatic updating of a company's inventory.

For meaningful communication to occur between computer applications, the format of the data being transmitted must be standardized. Although there are many standard formats for EDI interchanges, only two are widely recognized: the American National Standards Institute (ANSI) X12 and the Electronic Data Interchange For Administration, Commerce, and Transport (EDIFACT). These two standards are mandated for use within the Federal government (FIPS 161-1).

EDI standards specify how the data appears; they do not specify how the data is transmitted. The current trend in the EDI community is to replace asynchronous dial-up protocols (e.g., an XMODEM transfer) with more advanced technologies, such as electronic mail. Electronic mail provides a store-and-forward environment and a reliable audit trail.

Recognizing the need to promote functionally rich standards for the transmission of EDI data, the National Institute of Standards and Technology (NIST) initiated a project to design and implement an EDI communications package. The NIST EDI Communications Platform supports three types of applications for transmitting data: X.400 (i.e., electronic mail), File Transfer, Access and Management (FTAM), and dial-up protocols. The objective of the Platform [1] is to advance the use of Open Systems Interconnection (OSI) solutions for the communication of business data. The dial-up protocols are included to ease the transition for EDI applications using them to the more secure OSI protocols.

[1] The term *Platform* has several meanings. For purposes of this manual, the unqualified term *Platform* references the NIST EDI Communications Platform.

To present a comprehensive view of the NIST Communications Platform, this Operations Manual is organized as follows. Chapter 2 overviews the Communications Platform. Chapter 3 describes installation procedures. Chapter 4 details configuring the Platform. Chapter 5 specifies the operation of the Platform. Appendix A lists supported X.435 services, and appendix B provides a quick look-up for frequently used terms.

This paper makes many references to another NIST EDI project, called the Transaction Set Development System (TSDS). The TSDS addresses the need to design, test, and develop large sets of EDI transactions, especially by government agencies. For more information regarding the TSDS project, the following documents are available: **EDI Development Tools Design Document**, and **EDI Development Tools User's Guide**. Please note that these documents are unofficial draft internal reports.

# Chapter 2

# Overview

The NIST EDI Communications Platform is implemented on a Hewlett-Packard (HP) 9000 Series 827 running HP-UX version 8.02. It uses both the HP X.400 and FTAM products, version C.05.00. The Platform's source code is written in C++, compiled using the HP-UX C++ translator/compiler.

The Communications Platform comprises six major components: the EDI User Interface, the Local User Database, the Trading Partner Directory, and the X.400, FTAM, and Dial-up transmission media. These components are illustrated in figure 2.1.

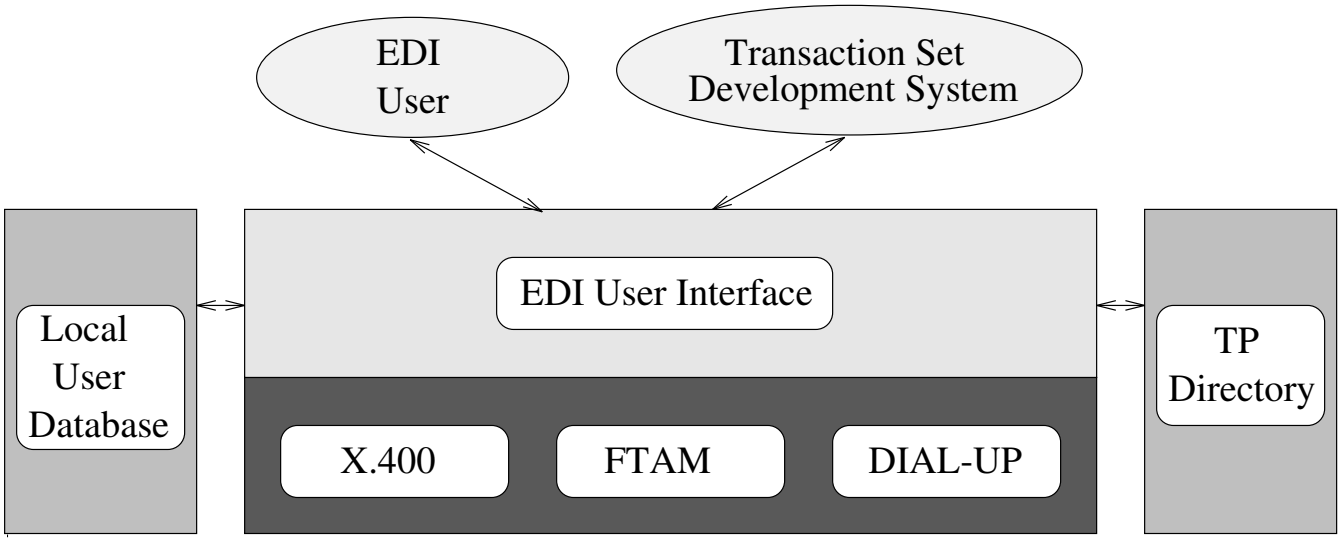


Figure 2.1: NIST EDI Communications Platform

The EDI User Interface transfers interchanges to and receives interchanges from EDI users. [1] Interchanges are conveyed between the EDI User Interface and local EDI users by one of two methods:

[1]The EDI User Interface will typically communicate with EDI translation software. For purposes of this manual, an EDI application and its translator are considered one entity, and referenced as an *EDI user*.

- as ASCII files that are copied to specific (i.e., preconfigured) file directories, or
- as data strings that are read and written to the TSDS.

In addition to local users, the Communications Platform interfaces to a Local User Database and a Trading Partner Directory. Both the Local User Database and the Trading Partner Directory are created during the Platform configuration process, and reside on the HP-UX file system. The Local User Database maintains information for each local EDI user, including: the name of the EDI user, the file directory into which inbound interchanges for the user are written, and other information. When an inbound interchange is received from a trading partner, the EDI User Interface uses the information stored in the Local User Database to convey the interchange to the appropriate local user.

The Trading Partner (TP) Directory is used to store trading partner information. When a local user submits an interchange for transmission, the EDI User Interface queries the TP Directory to obtain the information necessary to communicate the interchange to the trading partner. The TP Directory returns the transmission protocol to use for the communication, and all information germane to that protocol.

The remaining Platform components are the three transmission applications. These applications are discussed in the following sections.

## 2.1 X.400

The Platform's X.400 transmission application is a Message Handling System (MHS) User Agent (UA). The UA submits and receives X.400 messages conforming to the *Pedi* (i.e., X.435) and *P2* (i.e., X.420) protocol specifications. For outbound transmissions the EDI User Interface accepts an interchange from an EDI user, parses the name of the EDI trading partner from the interchange control header, and uses the name as a key to query the TP Directory. The TP Directory returns all information necessary to transmit the interchange, such as whether the message is to be formatted according to the *Pedi* or *P2* specification, the X.400 address of the message recipient, the message priority, and other message attributes. The UA then builds the message using the interchange as the message body, and submits the message for delivery to its recipient.

For inbound *Pedi* transmissions the UA parses the message to determine its *acceptability*. Acceptability means whether the UA makes the interchange available to a local EDI user. The criteria for acceptability are configured for each local EDI user using the *Pedi* service. For example, the Platform maintains knowledge of the EDI standards (i.e., X12 and/or EDIFACT) and versions of the standards supported by a local user. If an inbound interchange is determined to be unacceptable (e.g, the X12 version is not supported by the local user), the UA returns an EDI notification (if requested) to the message originator conveying the reason the interchange was rejected. Note that this capability is optional; the UA can be configured to accept all *Pedi* messages. Once the interchange is accepted, the EDI User Interface queries the Local User Database to determine how to transfer the interchange to the user. The interchange may be written to a specific file directory as an ASCII file, or transmitted to the TSDS.

To submit and receive X.400 messages the UA must communicate with a Message Transfer Agent (MTA). No MTA is supplied with the NIST Communications Platform. The Platform does, however, support the standard X.400 Application Program Interfaces (APIs) developed by the X/Open and API Association (APIA) for both the *Pedi* and *P2* protocols. Thus, the Platform can be used with any MTA supporting these standard interfaces.

## 2.2 FTAM

The NIST EDI Communications Platform can transmit EDI interchanges via its file transfer protocol. This protocol is specified by the International Standards Organization (ISO) FTAM standard. The FTAM standard provides a means to transfer, access and manipulate files in a uniform way.

For outbound transmissions the EDI User Interface accepts an interchange from an EDI user, parses the name of the EDI trading partner from the interchange control header, and uses the name as a key to query the TP Directory. The TP Directory returns all information necessary to transmit the interchange, such as a name and password used to login to the trading partner's system, and a file directory into which the EDI interchange is copied. The EDI User Interface writes this information to an ASCII file, called a *control file*. Every outbound interchange to be transferred via the FTAM protocol has a corresponding control file. The FTAM transmission application, which is a process that executes independently of the Platform process, uses the information in the control file to establish a connection with the trading partner's system, and transmits the interchange to that system.

For inbound FTAM transmissions, no action is required by the Platform. The receiving FTAM application is the FTAM responder provided by the system on which the Platform is installed. A trading partner's FTAM application establishes an association with the Platform's FTAM application, and transfers an EDI interchange. The Platform's EDI User Interface conveys the interchange to the appropriate local EDI user.

## 2.3 Dial-up

The NIST EDI Communications Platform can also transmit EDI interchanges via dial-up protocols. These protocols are used to communicate with Value Added Networks (VANs). Note that any of the transmission applications can communicate with VANs, provided that the VAN supports the transmission protocol.

For outbound transmissions the EDI User Interface accepts an interchange from an EDI user, parses the name of the EDI trading partner from the interchange control header, and uses the name as a key to query the TP Directory. The TP Directory returns the name of the VAN required to reach the trading partner. The EDI User Interface writes this name to a control file. Every outbound interchange to be transferred via a dial-up protocol has a corresponding control file. The dial-up transmission application, which is a process that executes independently of the Platform process, reads the VAN name from the control file, maps the name into the set of instructions required to communicate with the VAN (e.g., telephone number, modem speed, etc.), and calls the VAN to transmit the interchange.

To receive inbound transmissions from VANs, the dial-up transmission application periodically calls the VANs and reads any interchanges delivered to the Platform's mailbox. The times for which the VANs are called are configured into the system clock on which the Platform is installed. If any interchanges are waiting in the Platform's mailbox, they are received by the Platform, and conveyed to the appropriate local EDI users.

# Chapter 3

# Installation

The Communications Platform is installed using the UNIX tape file archiver utility *tar*. All Platform files are archived into one file named *platform.tar*. To install the Platform, the user copies the file *platform.tar* to the file directory where the Platform will reside, and types:

*tar -xvf platform.tar*

The *tar* command recreates the Platform's file directory structure on the new system, and restores each Platform file to its appropriate file directory. The name of each file is displayed to the screen as it is being restored. This completes the installation.

Once the installation is complete, the installing user is advised to confirm the existence of the Platform's file directories. Figure 3.1 illustrates the file directory hierarchy for the Platform. A brief description of each of the file directories is provided below. Note that file directories additional to those listed below may be present. These file directories are used for testing purposes by the NIST development team and are inconsequential to the normal operation of the Platform.

**platform/bin** This directory contains all the executable files, all the log files, and the Platform's configuration file.

**platform/error** If the Platform cannot transmit an outbound interchange, the interchange is relocated to this directory.

**platform/files** A file is written to this directory during FTAM and dial-up transfers, for purposes of access control.

**platform/ftam/control** For each interchange to be transmitted via the FTAM transmission application, a corresponding control file is created. The FTAM control files are written to this file directory.

**platform/ftam/interchange** Interchanges to be transmitted via the FTAM transmission application are written to this file directory.

**platform/ftam/lock** A file is written to this directory during FTAM transfers, to prevent the interchange and/or its corresponding control file from being modified or deleted.

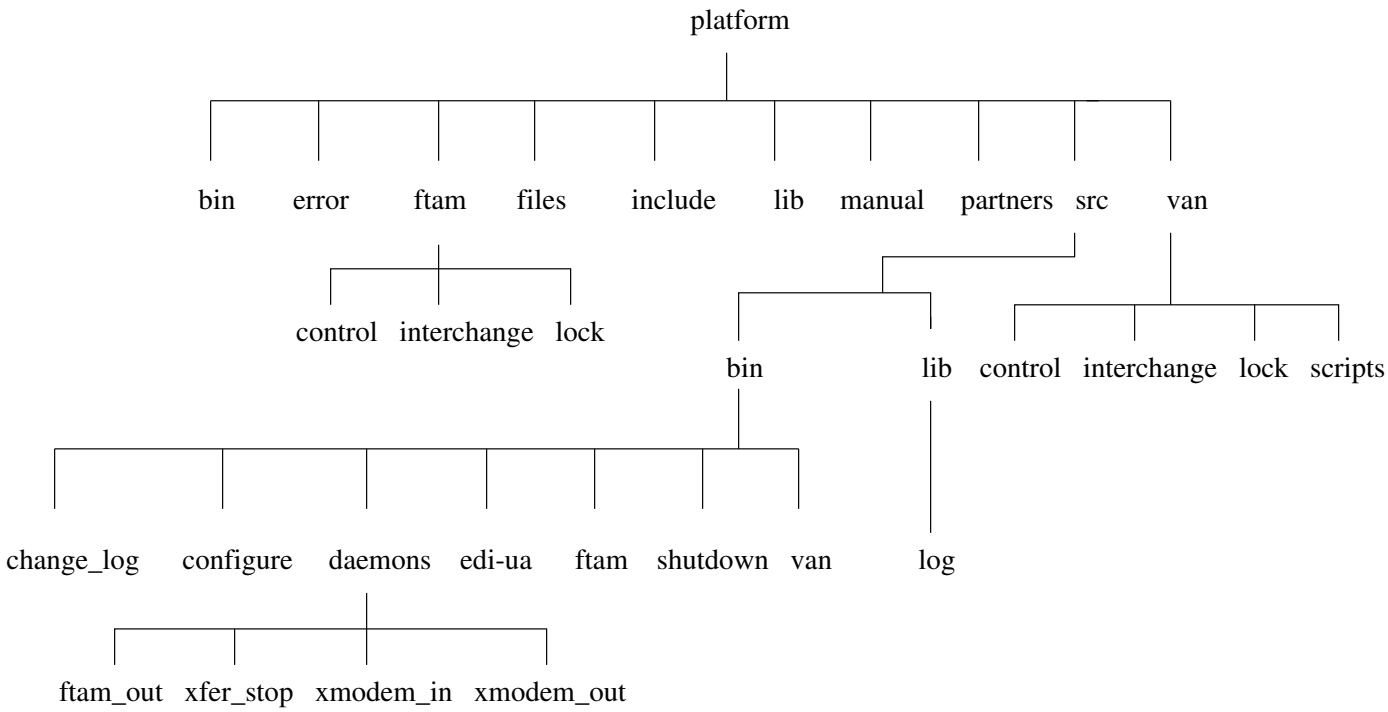


Figure 3.1: Platform File Directory Structure

**platform/include** This directory contains all the C++ header files.

**platform/lib** This directory contains all the Platform-defined libraries.

**platform/manual** This directory contains the Latex and PostScript representations of this manual.

**platform/partners** This directory contains the trading partners for all local EDI users (i.e., the TP Directory).

**platform/src/bin/change_log** This directory contains the C++ source code for modifying the Platform's logging options.

**platform/src/bin/configure** This directory contains the C++ source code for configuring the Platform.

**platform/src/bin/daemons/ftam_out** This directory contains the C++ source code for generating an outbound FTAM transfer.

**platform/src/bin/daemons/xfer_stop** This directory contains the C++ source code for gracefully halting execution of the processes responsible for inbound FTAM and dial-up transfers.

**platform/src/bin/daemons/xmodem_in** This directory contains the C++ source code for initiating a dial-up transfer from a VAN.

**platform/src/bin/daemons/xmodem_out** This directory contains the C++ source code for initiating a dial-up transfer to a VAN.

**platform/src/bin/edi-ua** This directory contains the C++ source code for the Platform's UA.

**platform/src/bin/ftam** This directory contains the low-level C code required for generating an outbound FTAM transfer.

**platform/src/bin/shutdown** This directory contains the C++ source code for gracefully halting executing of the Platform process.

**platform/src/bin/van** This directory contains the low-level C code required for generating a dial-up transfer.

**platform/src/lib/log** This directory contains the C++ source code for the logging functions performed by the Platform process.

**platform/van/control** For each interchange to be transmitted to a VAN via a dial-up protocol, a corresponding control file is created. The dial-up control files are written to this file directory.

**platform/van/interchange** Interchanges to be transmitted to a VAN via a dial-up protocol are written to this file directory.

**platform/van/lock** A file is written to this directory during dial-up transfers, to prevent the interchange and/or its corresponding control file from being modified or deleted.

**platform/van/scripts** The files in this directory contain all the information required to communicate with a VAN. A separate file is maintained for each VAN.

# Chapter 4

# Configuration

This chapter describes configuring the NIST EDI Communications Platform. Configuration is a three-staged procedure. The first stage configures EDI users local to the Platform. This process is described in section 4.1. The second stage configures the trading partners for the local EDI users. This process is described in section 4.2. The final stage configures information relevant to the VANs with which the Platform communicates, and is described in section 4.3.

In addition to configuring the Platform, the system administrator must configure the MTA and FTAM application on the system supporting the Platform. MTA and FTAM configurations are outside the scope of this manual. Please refer to the appropriate Systems Administration Guides for assistance.

## 4.1 Local EDI User Configuration

The Platform begins execution by reading its configuration file (i.e., *platform/bin/platform.cfg*). If the file does not exist, the Platform provides a sequence of prompts to assist with building the file. The prompts are terse, only containing relevant keywords. A more verbose description of the information to be entered is provided in this section. Each question prompted by the configuration program is detailed below.

Configuration file does not exist. Do you want to create it [y:n]?

This message appears if the configuration file does not exist or cannot be read. If the file exists, but cannot be read, the problem may be system-related, such as a file permission discrepancy.

Enter User Output Directory [.]

The Platform assumes that all outbound interchanges submitted by local EDI users will be written to one file directory. This file directory, referenced as the *Local-User-Outbound-File-Directory*, is entered here. As with all initial file directory prompts the default value of a period, which signifies the current working file directory, should not be used. If the Platform ever requires re-configuring, the value entered here previously will be provided as the default.

Enter Transmission Input Directory [.]

The Platform assumes that all interchanges received via inbound FTAM and dial-up transmissions will be written to one file directory. This file directory, referenced as the *Transmission-Inbound-File-Directory*, is entered here.

Enter FTAM Interchange Directory [.]

For an outbound FTAM transfer, the Platform relocates the interchange submitted by a local EDI user into a file directory which is polled by the FTAM transmission application. This file directory, referenced as the *FTAM-Interchange-File-Directory*, is entered here.

Enter FTAM Control Directory [.]

For outbound FTAM transfers, the Platform creates a control file for each interchange file. The control file contains all communications information required for the FTAM transfer. When an interchange is read by the FTAM transmission application, it searches the *FTAM-Control-File-Directory* for its control file. The name of the *FTAM-Control-File-Directory* is entered here.

Enter VAN Interchange Directory [.]

For an outbound dial-up transfer, the Platform relocates the interchange submitted by a local EDI user into a file directory which is polled by the dial-up transmission application. This file directory, referenced as the *VAN-Interchange-File-Directory*, is entered here.

Enter VAN Control Directory [.]

For outbound dial-up transfers, the Platform creates a control file for each interchange file. The control file contains all communications information required for the dial-up transfer. When an interchange is read by the dial-up transmission application, it searches the *VAN-Control-File-Directory* for its control file. The name of the *VAN-Control-File-Directory* is entered here.

Enter VAN script file [.]

The inbound dial-up transmission application accesses a file that contains data regarding VANs. This file, referenced as the *VAN-Script-File*, is described further in section 4.3. The name of the file, including its full file directory path specification, is entered here.

After the file directory information is obtained, the Platform requests information regarding local EDI users. The following prompts are displayed:

Enter the Id Code for the User:

The Id Code is the equivalent of the X12 Interchange Sender ID or the EDIFACT Sender Identification of the Interchange Sender segment.

Enter the Id Code Qualifier for the User:

The Id Code Qualifier is the equivalent of the X12 Interchange ID Qualifier or the EDIFACT Partner Identification Code Qualifier of the Interchange Sender segment.

Deliver Inbound Interchanges to the TSDS?

In the future, inbound interchanges may be delivered to local users via the TSDS. Since the TSDS is not currently operational, this answer must be *no*.

Enter the EDI User Input Directory:

If the TSDS is not used for the delivery of an inbound interchange, the Platform copies the interchange to a file directory to be processed by the local EDI user. This file directory name is entered here.

Does the local EDI user use the UA?

If the EDI user sends or receives interchanges via X.400 (i.e., *P2* or *Pedi*), answer *yes*.

If a *yes* response is given, the configuration program prompts for the X.400 address of the local EDI user. Each X.400 address attribute is displayed, allowing a value to be specified. If no value is to be entered for a specific address attribute, simply press *enter* at the prompt for that attribute.

Does the local EDI user use the X.435 service?

If the local EDI user sends or receives *Pedi* messages, answer *yes*. If a *yes* response is given, the configuration program prompts for the following information.

Enter the EDI User Additional Directory:

In a *Pedi* message, information additional to the interchange may be conveyed with the interchange (e.g., freeform text, diagrams). The Platform copies this additional information to a file directory to be processed by the user. This file directory name is entered here.

Reject an inbound X.435 message marked FOR COPY?

By default, a message is marked "For Action". The "For Copy" value is intended to convey that no action is required by the recipient EDI user; the interchange is copied to that user for information purposes only. If the local EDI user is not to receive interchanges marked "For Copy", answer *yes*.

Reject an inbound X.435 message marked as INCOMPLETE?

If the UA receives a message with either the interchange or an additional bodypart removed, the message is marked as "incomplete." If the UA is not to accept incomplete messages, answer *yes*.

Reject an inbound X.435 message if the sender is not a registered trading partner?

The Platform maintains a list of all trading partners for each local user. When a message is received, the UA can attempt to match the originator's Id Code, Id Code Qualifier, and X.400 address against those belonging to known trading partners for a local EDI user. If the UA is to reject the message if no match is found, answer *yes*.

Reject an inbound X.435 message if the standard is not supported?

The Platform maintains a list of EDI standards (as specified in X.435) for each local EDI user. When a message is received, the UA can test whether the EDI standard used to format the interchange is supported by the local user. If the UA is to reject a message if the EDI standard is not supported, answer *yes*. If a *yes* response is given, the configuration program prompts for the EDI standards supported by the local user. The list of possible EDI standards is shown below.

- EDIFACT ISO646
- EDIFACT T61
- EDIFACT Octet
- ANSI X12 ISO646
- ANSI X12 T61
- ANSI X12 Octet
- ANSI X12 EBCDIC
- UN/TDI ISO646
- UN/TDI T61
- UN/TDI Octet
- Private Octet
- Undefined Octet

The final information to be configured is the versions of the EDI standards supported by the local EDI user. The configuration program requests the data as follows:

ID-string >>

The ID string corresponds to the X12 standard identifier or the EDIFACT syntax identifier.

ID-version >>

The ID version corresponds to the X12 version identifier or the EDIFACT syntax version number.

Since the local EDI user can support multiple EDI standard versions, the prompts will display repeatedly until the configuring user enters a period as the only character in response to the "ID-string >> " prompt. The following prompt and response illustrate how to terminate input of the EDI standard versions.

ID-string >>  .

Note that if the UA is to accept **all** EDI standard versions, the configuring user enters a period in response to the first "ID-string >>" prompt. As a result, no EDI standard versions will be registered for the local EDI user, and all incoming values will be accepted.

This completes the configuration for one EDI user. The configuration program displays the following menu.

Menu Selections:
    [A]dd a user
    [D]el a user
    [E]xit configuration
Select A, D, or E:

The *Exit* option saves all information entered by the user, and exits the configuration program. The *Delete* option displays the ID Code and ID Code Qualifier for each local EDI User, and requests if the entry is to be deleted. The *Add* option returns the configuring user to the prompts for entering local EDI user information (i.e., ID Code and ID Code Qualifier). There is no limit to the number of local EDI users that may be registered with the Platform.

## 4.2 Trading Partner Configuration

The second step in configuring the Platform is to configure the trading partners for the local EDI users. Configuring trading partners is a manual process which comprises:

1. manually creating the appropriate file directories, and
2. manually populating these file directories with the appropriate files.

Under the file directory *platform/partners* the configuring user creates **one file directory per local EDI user**. The name of the file directory is the concatenation of the Id Code and Id Code Qualifier of the local user. For example, if the Id Code and Id Code Qualifier of a local user are *NIST_billing* and *01*, respectively, the configuring user creates the file directory *platform/partners/NIST_billing01*.

The configuring user then populates this file directory with trading partner files. There is **one trading partner file per trading partner**. As with the file directory name, the name of a trading partner file is the concatenation of the Id Code and Id Code Qualifier of the trading partner. For example, if the Id Code of a trading partner is *ABC_Company*, and no Id Code Qualifier is used, the configuring user creates the file *platform/partners/NIST_billing01/ABC_Company*.

Each trading partner file contains the information required to communicate interchanges to that trading partner. Interchanges may be conveyed by any one of the four transmission services:

- Pedi,
- P2,
- FTAM, or
- DIAL-UP.

Four example files are included in the file directory *platform/partners* to ensure that information is entered into the trading partner files correctly. The names of the example files are:

- pedi.example
- p2.example
- ftam.example
- dialup.example

To create a trading partner file, the configuring user is advised to copy the appropriate example file to the appropriate file directory, and to rename it according to the trading partner's Id Code and Id Code Qualifier. If in the above example, the local user *NIST_billing* communicates with the *ABC_Company* using the *P2* transmission service, the configuring user copies the file *p2.example* to the file directory *platform/partners/NIST_billing01*, and renames the file *ABC_Company*. The configuring user then edits the example file so that the file's contents reflect the appropriate information for the trading partner. Details for editing each of the example files is provided below.

### 4.2.1 Pedi Example File

The file *pedi.example* contains the following instructions.

# Enter the name of the Pedi service on this line
# Enter the X.400 address of the recipient on this line
# Enter the X.400 address of the EDIN Receiver on this line
# Enter the Positive Notification value on this line
# Enter the Negative Notification value on this line
# Enter the Forwarded Notification value on this line
# Enter the Responsibility Passing Allowed value on this line
# Enter the Message Priority value on this line

Each of the instructions is detailed below.

# Enter the name of the Pedi service on this line

Replace this line with the word *pedi*.

# Enter the X.400 address of the recipient on this line

Replace this line with the X.400 address of the trading partner. The address must have the syntax */tag=value/tag=value/.../tag=value/*. Note that the address must begin and end with a slash. The tags identify X.400 address attributes, and must match the following symbols:

| Tag | X.400 Address Attribute |
|---|---|
| ad | Administration Management Domain Name |
| cm | Common Name |
| co | Country Name |
| t1 | Domain Defined Attribute Type 1 |
| t2 | Domain Defined Attribute Type 2 |
| t3 | Domain Defined Attribute Type 3 |
| t4 | Domain Defined Attribute Type 4 |
| v1 | Domain Defined Attribute Value 1 |
| v2 | Domain Defined Attribute Value 2 |
| v3 | Domain Defined Attribute Value 3 |
| v4 | Domain Defined Attribute Value 4 |
| gn | Generation Qualifier |
| gv | Givenname |
| in | Initials |
| og | Organization Name |
| u1 | Organizational Unit Name 1 |
| u2 | Organizational Unit Name 2 |
| u3 | Organizational Unit Name 3 |
| u4 | Organizational Unit Name 4 |
| pd | Private Management Domain Name |
| su | Surname |

# Enter the X.400 address of the EDIN Receiver on this line

Using the *pedi* service, the message originator can request that EDI notifications be returned, not to the originator, but to a different user. If EDI notifications from the trading partner are to be returned to an address other than that of the originating local EDI user, replace this line with that X.400 address. Otherwise, delete the text on this line, but do not delete the line itself. If no EDIN Receiver is specified, there must be a blank line as a placeholder.

# Enter the Positive Notification value on this line

Replace this line with a *1* (i.e., one) if the UA is to request positive notification from the trading partner for X.435 messages, else a *0* (i.e., zero) if the UA is not to request positive notification.

# Enter the Negative Notification value on this line

Replace this line with a *1* if the UA is to request negative notification from the trading partner for X.435 messages, else a *0* if the UA is not to request negative notification.

# Enter the Forwarded Notification value on this line

Replace this line with a *1* if the UA is to request forwarded notification from the trading partner for X.435 messages, else a *0* if the UA is not to request forwarded notification.

# Enter the Responsibility Passing Allowed value on this line

Replace this line with a *1* if the trading partner's UA is allowed to forward responsibility for an X.435 message to another UA; replace with a *0* if responsibility forwarding is not permitted.

# Enter the Message Priority value on this line

Replace this line with a *0* to send messages to the trading partner as low priority, a *1* to send messages as normal priority, or a *2* to send messages as urgent priority.

This concludes the building of a *Pedi* trading partner file. Note that in the file *pedi.example*, a sample set of responses to the instructions described above is provided.

If the local EDI user accepts interchanges from registered trading partners only (i.e., during configuration of the local EDI user, a *yes* response is given to the question, "Reject an inbound X.435 message if the sender is not a registered trading partner?"), the configuring user creates a second file. This file:

- must reside in the file directory containing the trading partner files for the local EDI user, and
- must be named *addresses.tp*

The configuring user inserts the X.400 address of each trading partner into this file. These addresses are the same addresses entered in the *Pedi* trading partner file, in response to the instruction:

# Enter the X.400 address of the recipient on this line

Thus, if a local EDI user communicates with six trading partners, the file *addresses.tp* contains six lines, each line being the X.400 address of one of the trading partners. When an interchange for the local user is received, the Platform compares the originator's X.400 address against the addresses in the file *addresses.tp*. If no match is found, the Platform returns an EDI notification (if requested) to the message originator specifying that the interchange was not conveyed to the recipient EDI user because the message originator was not recognized.

### 4.2.2 P2 Example File

The file *p2.example* contains the following instructions.

# Enter the name of the P2 service on this line
# Enter the X.400 address of the recipient on this line
# Enter the Message Priority value on this line

Each of the instructions is detailed below.

# Enter the name of the Pedi service on this line

Replace this line with the word *p2*.

# Enter the X.400 address of the recipient on this line

Replace this line with the X.400 address of the trading partner. Instructions for formatting the address are detailed in section 4.2.1.

# Enter the Message Priority value on this line

Replace this line with a *0* to send messages to the trading partner as low priority, a *1* to send messages as normal priority, or a *2* to send messages as urgent priority.

This concludes the building of a *P2* trading partner file. Note that in the file *p2.example*, a sample set of responses to the instructions described above is provided.

### 4.2.3 FTAM Example File

The file *ftam.example* contains the following instructions.

# Enter the name of the FTAM service on this line
# Enter the FTAM user name on this line
# Enter the FTAM password that corresponds to the user name on this line
# Enter the file directory name in which the interchange is copied on this line
# Enter the X.500 Directory name on this line

Each of the instructions is detailed below.

# Enter the name of the FTAM service on this line

Replace this line with the word *ftam*.

# Enter the FTAM user name on this line

To establish an FTAM association, the FTAM transmission application specifies a valid login name for the trading partner's filestore. Replace this line with the valid login name.

# Enter the FTAM password that corresponds to the user name on this line

To establish an FTAM association, the FTAM transmission application specifies a valid password for the trading partner's filestore. Replace this line with the password that corresponds to the user name provided above.

# Enter the file directory name in which the interchange is copied on this line

Replace this line with a valid file directory specification on the trading partner's filestore. The FTAM transmission application will write the interchange into this file directory.

# Enter the X.500 Directory name on this line

The HP FTAM product requires an X.500 directory name , which is used to locate FTAM communications parameters, such as the network address of the trading partner's filestore. Replace this line with the X.500 directory name. The name must have the syntax */tag=value/tag=value/.../tag=value*. The tags identify locality and FTAM parameters, and must match the following symbols:

| **Tag** | **Paramter** |
|---|---|
| C | Country |
| L | Locality |
| O | Organization |
| OU | Organizational Unit |
| AP | Application Process |
| AE | Application Entity |

This concludes the building of an *FTAM* trading partner file. Note that in the file *ftam.example*, a sample set of responses to the instructions described above is provided.

### 4.2.4 DIAL-UP Example File

The file *dialup.example* contains the following instructions.

# Enter the name of the Dial-up service on this line
# Enter the name of the VAN on this line

Each of the instructions is detailed below.

# Enter the name of the Dial-up service on this line

Replace this line with the word *dialup*.

# Enter the name of the VAN on this line

Replace this line the name of the VAN to which the Platform connects to transmit the interchange. For more information regarding this name, see section 4.3.

This concludes the building of a *Dial-up* trading partner file. Note that in the file *dialup.example*, a sample set of responses to the instructions described above is provided.

## 4.3 VAN Configuration

The final step in configuring the Platform is to build the two files used by the dial-up transmission application to communicate with VANs. These files, *xmodem_startup_definitions* and the *VAN-Script-File*, are described in sections 4.3.1 and 4.3.2, respectively.

### 4.3.1 XMODEM_STARTUP_DEFINITIONS

The first file which needs to be configured is *xmodem_startup_definitions*. This file contains the communications information needed to send and receive interchanges between the Platform and the VAN(s). The file is configured by supplying specific information concerning *local hosts* and *destination hosts*, as well as the dialogue exchanged between them. Note that most of the communication parameter values associated with a specific VAN are provided by that VAN. An example *xmodem_startup_definitions* file is included with the Platform, located in the file directory *platform/van/control*.

The *xmodem_startup_definitions* file specifies three structured categories of information:

1. the *host* structure contains host (e.g., Platform) specific information,
2. the *tt_struct* contains communication specific information (e.g., telephone number, modem speed, and modem type), and
3. the *respond* structure contains specific dialogue information.

None of these structures needs to be changed to configure *xmodem_startup_definitions*. That is, no new parameters need to be created or deleted. Depending on the VAN, however, specific **parameter values** may need to be added, altered, and/or deleted.

Below is an example *xmodem_startup_definitions* file. This example file is included to demonstrate how parameter and corresponding parameter values are set. There is no automated script to build this file; it must be created manually (e.g., using a text editor).

The # (i.e., pound sign) at the beginning of a line denotes a comment and is used to explain subsequent lines or general information. Comments can also be added by putting information text between the lines **BEGIN_COMMENT_SECTION** and **END_COMMENT_SECTION**. Lines beginning with the words *HOST:*, *TT:*, and *RESPOND:* denote parameters values for the three categories of information. All lines will be prefixed with the comment character, comment section lines, one of the three words given above, or the * (i.e., asterisk) which denotes the beginning or ending of a category. Lines beginning with *HOST* give specific information (e.g., name, telephone number, default values) about the host (e.g., where the Platform resides). Lines beginning with *TT* give communication specific information (e.g., telephone number, modem speed, and modem type), and lines beginning with *RESPOND* give specific dialogue information needed to communicate with the desired VAN(s).

Communication definitions are delineated by the following lines: **BEGIN_TT_DEFINITION** and **END_TT_DEFINITION**. Within these lines are individual device definitions (e.g. modems and terminal), delineated by *TT:*BEGIN_DEFINITION=* and *TT:*END_DEFINITION* lines. Communication information is defined for each device.

Host definitions are delineated by **BEGIN_HOST_DEFINITION** and **END_HOST_DEFINITION** lines. Within these lines specific host definitions are delineated by *HOST:*BEGIN_DEFINITION=* and *HOST:*END_DEFINITION** lines. Transmission parameters for each host are defined here.

For each host a set of respond definitions is also specified (e.g., performing a login, starting a session, uploading a file, downloading a file, and disconnecting from the VAN). The required parameter values are derived from configuration manuals provided by VANs. As noted above, parameter value definitions may need to be added, deleted or altered.

Inline respond definitions (i.e., responses defined within host definitions) are delineated by the lines **BEGIN_RESPOND_INLINE** and **END_RESPOND_INLINE**. Non-inline respond definitions (i.e., responses defined outside a host definition) are delineated by the lines *RESPOND:*BEGIN_RESPOND_HOST=* and *RESPOND:*END_RESPOND_HOST**. Within these lines are specific respond definitions. These definitions are delineated by the lines *RESPOND:*BEGIN_DEFINITION=* and *RESPOND:*END_DEFINITION**.

Below is an example *xmodem_startup_definitions* file. Note that this example shows only a subset of the possible parameters values.

```
#
# Filename: xmodem_startup_definitions
#
***EDI_XMODEM_CONTROL_FILE***
#
# Define this host
#
THIS_HOST_DEFINITION=NIST
#
# Definition of 'tt_struct'
```

```
#
*BEGIN_TT_DEFINITION*
TT:*BEGIN_DEFINITION=/dev/cua0p7
*BEGIN_COMMENT_SECTION*
dial-out modem: on 869-1386
*END_COMMENT_SECTION*
TT:MDM_INDEX=HAYES
TT:DEFAULT_SPEED=B2400
TT:*END_DEFINITION*
#
TT:*BEGIN_DEFINITION=/dev/ttyd0p6
# dial-in modem: on 869-0096
TT:MDM_INDEX=HAYES
TT:DEFAULT_SPEED=B2400
TT:*END_DEFINITION*
#
TT:*BEGIN_DEFINITION=/dev/tty0p1
# modem, terminal - monitor line
TT:DEFAULT_SPEED=-1
TT:IN_USE=TRUE
TT:*END_DEFINITION*
#
TT:*BEGIN_DEFINITION=/dev/cua0p5
# modem, terminal - output loopback line
TT:HOST=NIST_LOCAL
TT:DEFAULT_SPEED=B9600
TT:MDM_INDEX=UNKNOWN
TT:*END_DEFINITION*
#
TT:*BEGIN_DEFINITION=/dev/ttyd0p4
# modem, terminal - input loopback line
TT:DEFAULT_SPEED=-1
TT:IN_USE=TRUE
TT:*END_DEFINITION*
*END_TT_DEFINITION*
#
# Next Host = NIST
#
HOST:*BEGIN_DEFINITION=NIST
HOST:TELNBR=8690096
HOST:DEFAULTS=SET_REMOTE_MODE|XMIT_BOTH|MOD_P_ST|MOD_G_ST|SPLIT_BOTH|P_XMODEM
HOST:TERMINATOR=\n
HOST:SEPARATOR='' ``
HOST:*END_DEFINITION*
#
# Next Host = Bell_Atlantic
#
HOST:*BEGIN_DEFINITION=Bell_Atlantic
```

```
HOST:TELNBR=12155743629
HOST:*END_DEFINITION*
#
# Next Host = GE
#
HOST:*BEGIN_DEFINITION=GE
# *** DEFAULTS MUST, repeat MUST, agree with RESPONSES ***
HOST:DEFAULTS=XMIT_BOTH|P_XMODEM|G_XMODEM
HOST:NULL_FILE=''NO DATA TO TRANSMIT''
HOST:TELNBR=3405565
#HOST:TELNBR=18002532209
HOST:*END_DEFINITION*
#
# Next Host = TRANS
#
HOST:*BEGIN_DEFINITION=TRANS
# *** DEFAULTS MUST, repeat MUST, agree with RESPONSES ***
HOST:DEFAULTS=XMIT_BOTH|P_XMODEM|G_XMODEM
HOST:NULL_FILE=''NO DATA TO TRANSMIT''
HOST:REC_LEN=80
HOST:TELNBR=14049947303
HOST:*END_DEFINITION*
*END_HOST_DEFINITION*
#
#
# Definition of 'respond' for each specified 'host'
#
*BEGIN_RESPOND_DEFINITION*
#
# Responses for NIST
#
RESPOND:*BEGIN_RESPOND_HOST=NIST
RESPOND:*BEGIN_DEFINITION=LOGIN
RESPOND:CMD_STRING=\n
RESPOND:TIME=15
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=''login: ``
RESPOND:ATTEMPTS=2
RESPOND:LOWER=TRUE
RESPOND:*NEXT_DEFINITION*
RESPOND:CMD_STRING=wallace\n
RESPOND:TIME=30
RESPOND:RESPONSE=password:
RESPOND:LOWER=TRUE
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:ATTEMPTS=1
RESPOND:*NEXT_DEFINITION*
RESPOND:CMD_STRING=tsunamis\n
```

```
RESPOND:TIME=30
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:ATTEMPTS=1
RESPOND:RESPONSE=''TERM = (hp) ``
RESPOND:ALT1=''\nlogin: ``
RESPOND:ALT1_ACTION=ACTION_RESTART
RESPOND:*NEXT_DEFINITION*
RESPOND:CMD_STRING=unknown\n
RESPOND:TIME=30
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:ATTEMPTS=1
RESPOND:RESPONSE=''<cs> ``
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=PUT_START
# this command puts the home directory in the log file
RESPOND:CMD_STRING=pwd\n
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=''<cs> ``
RESPOND:ATTEMPTS=1
RESPOND:*NEXT_DEFINITION*
#RESPOND:CMD_STRING=''cd ~/dirin\n''
RESPOND:CMD_STRING=''cd ~markovit/platform/bin\n''
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=''<cs> ``
RESPOND:ATTEMPTS=1
RESPOND:*NEXT_DEFINITION*
#RESPOND:CMD_STRING=~/bin/xtransfer\n
RESPOND:CMD_STRING=''xmodem_copy startup=/edi/xmodem_startup_definitions\n''
RESPOND:TIME=10
RESPOND:FLAGS=MOD_CMD
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=''xtransfer started\n''
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=EXIT
RESPOND:CMD_STRING=logout\n
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
RESPOND:*END_RESPOND_HOST*
#
# Responses for Bell_Atlantic
#
RESPOND:*BEGIN_RESPOND_HOST=Bell_Atlantic
```

```
RESPOND:*BEGIN_DEFINITION=LOGIN
RESPOND:CMD_STRING=\r
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=LOGON\r
RESPOND:ATTEMPTS=2
RESPOND:*NEXT_DEFINITION*
RESPOND:CMD_STRING=FAKEID,FAKE\r
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:ATTEMPTS=2
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=PUT_START
RESPOND:CMD_STRING=TRANSFER,X,EDFROM,4096\r
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=CONTINUE\r
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=PUT_END
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=SAVED\r
RESPOND:ATTEMPTS=1
RESPOND:*NEXT_DEFINITION*
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=EDIREADY\r
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=GET_START
RESPOND:CMD_STRING=RECEIVE\r
RESPOND:TIME=45
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=TRANSFER,X,EDTO,*\r
RESPOND:ALT1=COLLECTING\r
RESPOND:ALT1_ACTION=ACTION_RESET_TIME
RESPOND:ALT2=EMPTY\r
RESPOND:ALT2_ACTION=ACTION_EXIT
RESPOND:ATTEMPTS=1
RESPOND:*NEXT_DEFINITION*
RESPOND:CMD_STRING=CONTINUE\r
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
#
#
```

```
RESPOND:*BEGIN_DEFINITION=GET_END
RESPOND:CMD_STRING=SAVED\r
RESPOND:TIME=10
RESPOND:NO_RESPONSE=ACTION_EXIT
RESPOND:RESPONSE=EDIREADY\r
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
#
RESPOND:*BEGIN_DEFINITION=EXIT
RESPOND:CMD_STRING=LOGOFF\r
RESPOND:ATTEMPTS=1
RESPOND:*END_DEFINITION*
RESPOND:*END_RESPOND_HOST*
#
RESPOND:*END_RESPOND_HOST*
#
*END_RESPOND_DEFINITION*
```

### 4.3.2 VAN Script File

The second file requiring configuration is the *VAN-Script-File*. The name of this file is specified by the configuring user (see section 4.1) in response to the prompt:

Enter VAN script file

This file contains two categories of information. The first is the full file directory and file name specification of the *xmodem_startup_definitions* file. The entry must use the syntax:

ctrl=file directory specification/xmodem_startup_definitions

The second category provides instructions to the processes responsible for transmitting interchanges to and from the VANs. Each entry must use the syntax:

HostName ControlFile Time Protocol Parameters

where,

**HostName:** This value identifies the VAN. The name matches the VAN name in the *xmodem_startup_definitions* file specified by the instruction, **HOST:*BEGIN_DEFINITION=**.

**ControlFile:** For this value, enter an asterisk. The asterisk means that the VAN communications information is contained in the *xmodem_startup_definitions* file.

**Time:** For this value, enter an asterisk. The asterisk means that the VAN is called every time the system clock signals the Platform to query for inbound interchanges from the VANs.

**Protocol:** For this value, enter **XMODEM**.

**Parameters:** These flags are used to specify transmission options, specifically:

| Flag | Transmission Option |
|---|---|
| b | Binary Mode |
| t | Text Mode |
| u | UNIX Text Mode [1] |
| y | YMODEM Batch Mode |
| B | MODEM7 Batch Mode |
| K | Use 1K Packets |
| c | Select CRC Mode on Reception |
| l | Log XMODEM Header and Trailer Information |
| x | Log XMODEM Debugging Information |

Flags are preceded by a + (i.e., plus) or a - (i.e., minus) to either activate or deactivate the flag. The default flags are +*ctl-x*. An example *VAN-Script-File* is provided below.

ctrl=/users/edi/vans/xmodem_startup_definitions
TRANS * * XMODEM +cx
GE * * XMODEM +c-x
Bell_Atlantic * * XMODEM

Note that the parameters described above are used for normal (i.e., non-testing) Platform operations. For a complete list of options and their purpose, enter the command:

*platform/bin/xmodem_copy help*

[1] UNIX text mode differs from text mode in that the line termination character is a newline (i.e., 0x0A) as opposed to a carriage return, line feed (i.e., 0x0D 0x0B).

# Chapter 5

# Operations

This chapter describes the operation of the Platform. The Platform comprises three transmission applications. The first application interfaces with the MTA. This application is described in section 5.1. The second application performs outbound FTAM transfers. This application is described in section 5.2. The third application performs asynchronous modem transfers with VANs, and is described in section 5.3.

Before describing the specific applications, some general information, such as starting and stopping the Platform, must be presented. The Platform is started by the *root* user from the file directory *platform/bin*. The following command starts the Platform as a background process:

*edicomp <log-level> <log-file>*

The *log-level* and *log-file* parameters are optional. The *log-level* parameter assumes one of the following values. Note that the *log-level* parameter is used primarily by the UA.

**0** No logging; only errors are written to the log file.

**1** Basic logging; the originator, recipient(s), and identifiers of X.400 messages are logged.

**2** Interchange logging; basic logging plus information relating to the interchange that is contained in the X.400 message header is logged.

**3** Trace logging; interchange logging plus trace information, such as distribution list expansion, external trace data, and others, is logged.

**4** Verbose logging; all X.400 message attributes are logged.

The default value is *basic logging*.

If (and only if) the *log-level* parameter is present, a *log-file* parameter may be specified on the command line to start the Platform. The *log-file* parameter is used to provide the file directory path and the name of the file in which logging information is written. If no *log-file* parameter is specified, the Platform logs information to the file *platform/bin/platform.log*. The log file is never overwritten by the Platform. With each successive running of the Platform, logging information is appended to the existing log file.

While the Platform is executing, a user has the option of modifying the *log-level* and *log-file* parameters. By entering the command:

*logcomp <log-level> <log-file>*

a user can set a new *log-level* or *log-file*, without having to stop and restart the Platform. The *log-level* and *log-file* parameters are used identically as when starting the Platform.

To stop the Platform, a user enters the command:

*stopcomp*

This command gracefully shuts down the Platform, after it has finished any current EDI transmissions.

## 5.1 UA

The UA application also controls the general processing of the Platform. After the Platform is started, it reads the configuration file (see chapter 4), then repeatedly performs the following functions:

1. Reads the *Local-User-Outbound-File-Directory*. If any interchanges are present, it parses the originator and recipient EDI names from the interchange, and uses these names to query the TP Directory. The query returns all relevant communications information. The Platform uses this information to transmit the interchange via the X.400, FTAM, or dial-up applications.
2. Reads the *Transmission-Inbound-File-Directory*. If any interchanges are present, it parses the recipient EDI name from the interchange, and matches the name against its list of local users. It relocates the interchange into the file directory specified for inbound interchanges for that user.
3. Queries the inbound MTA queue. If any EDI messages are present, it parses the message, extracts the interchange from the message, and relocates the interchange into the file directory specified for the message recipient (i.e., a local EDI user). If an EDI notification or Message Transfer Service (MTS) report is read from the MTA queue, the notification or report is logged according to the requested *log-level*.
4. Tests whether the user has requested a change in the *log-level* and *log-file*. If so, it performs the requested modifications.
5. Tests whether the user has requested to stop the Platform. If so, it shuts down the Platform gracefully.

If, at any time during the processing of an interchange, the Platform determines that an interchange cannot be delivered to either a local EDI user or a trading partner, the Platform relocates the interchange to a special error file directory. The name of this file directory is *platform/error*.

If no inbound or outbound interchanges are to be transmitted, the Platform suspends execution for a period of time (initially set to 5 seconds), then resumes execution.

## 5.2 FTAM

The FTAM transmission medium comprises two distinct applications: a sending application and a receiving application. Both applications execute independently of the Platform process. Section 5.2.1 describes the sending FTAM application, and section 5.2.2 describes the receiving FTAM application.

### 5.2.1 Sending FTAM Application

The sending FTAM application consists of a server process (*platform/bin/ftam_out*), whose function is to recognize the interchanges and FTAM control files created by the Platform, and a transfer agent process (*platform/bin/ftam_copy*), whose function is to transfer a single EDI interchange. There is always one server and zero or more transfer agent processes executing. This configuration increases Platform performance by allowing multiple simultaneous FTAM transfers.

The FTAM server is a daemon process (i.e, always executing) which periodically polls the *FTAM-Interchange-File-Directory* and the *FTAM-Control-File-Directory*. If an interchange file is present, the server creates a transfer agent process to transfer the interchange.

The FTAM transfer agent is an instantiation of an FTAM initiator-sender. The transfer agent initiates an FTAM association based on information contained in the FTAM control file. This information includes:

**a user name** which identifies a valid login name on remote FTAM filestore,

**a user password** which identifies the password corresponding to the "user name",

**a file name** which identifies the file directory name into which the interchange being transmitted is written, and

**an X.500 directory name** which is required by the HP FTAM product, and is used to locate FTAM communication parameters (e.g., the network address of the remote filestore).

Once the FTAM association is established, the transfer agent sends the contents of the interchange file to the remote filestore. The transfer agent exits after the transfer completes.

In the file directory *platform/bin* the transfer agent creates a log file for the outbound FTAM transfer. The name of the file has the format xlog_XXXXXX, where XXXXXX is a sequence of six alphanumeric characters. The name of the log file is guaranteed to be unique. If the transfer of the interchange is successful, the transfer agent deletes the log file. Thus, only log files containing errors will remain in the *platform/bin* file directory.

### 5.2.2 Receiving FTAM Application

The receiving FTAM application is the local FTAM responder provided by the host system. A remote FTAM application initiates an association with the local FTAM responder and transfers a file containing an EDI interchange to the *Transmission-Input-File-Directory*. The Platform's User Interface makes the interchange available to the appropriate local EDI user. Depending the on the *log-level* specified during the configuration of the Platform, either by a brief information message or the entire inbound interchange will be written to the Platform's log file (i.e., *platform.log*).

## 5.3 VANs

The dial-up transmission medium comprises two distinct applications: a sending application and a receiving application. Both applications execute independently of the Platform process. Section 5.3.1 describes the sending dial-up application, and section 5.3.2 describes the receiving dial-up application.

### 5.3.1 Sending Dial-up Application

The sending dial-up application consists of a server process (*platform/bin/xmodem_out*), whose function is to recognize the interchanges and dial-up control files created by the Platform, and a transfer agent process (*platform/bin/xmodem_copy*), whose function is to transfer a single EDI interchange. There is always one server and zero or more transfer agent processes executing. This configuration increases Platform performance by allowing multiple simultaneous dial-up transfers.

The dial-up server is a daemon process (i.e, always executing) which periodically polls the *VAN-Interchange-File-Directory* and the *VAN-Control-File-Directory*. If an interchange file is present, the server creates a transfer agent process to transfer the interchange.

The dial-up transfer agent uses the VAN name parameter from the control file to access the information required to communicate with the VAN (e.g., telephone number, modem speed, etc.). All communications information for all VANs resides in the file *xmodem_startup_definitions*. The transfer agent reads this file, then locates the information relevant to the VAN identified in the control file. This information is used to establish a connection with the VAN, and to upload the interchange. The transfer agent exits after the transfer completes.

In the file directory *platform/bin* the transfer agent creates a log file for the outbound dial-up transfer. Logging is the same as described in section 5.2.1.

### 5.3.2 Receiving Dial-up Application

As with the sending dial-up application, the receiving application consists of a server process (*platform/bin/xmodem_in*) and a transfer agent process (*platform/bin/xmodem_copy*). The server process is executed by the system clock (e.g., *cron*) at regularly scheduled intervals. The intervals are configured by the system administrator.

The server process reads the *VAN-Script-File* which identifies the communications information file (i.e., *xmodem_startup_definitions*) and various parameters for creating the transfer agent processes. One transfer agent process is created for each VAN.

The transfer agent processes use the information in the file *xmodem_startup_definitions* to establish a connection with a VAN, and download any interchanges delivered to the Platform's mailbox. Inbound interchanges are written to the *Transmission-Inbound-File-Directory*, then conveyed by the Platform to the appropriate local EDI users.

In the file directory *platform/bin* the transfer agent creates a log file for the inbound dial-up transfer. Logging is the same as described in section 5.2.1.

# Chapter 6

# Conclusion

Traditionally, business information has been communicated using paper and the postal system. With the advent of EDI, the mode of transmission has become electronic. Early EDI transmissions used direct connections and dial-up protocols that provided no means of tracking data or guaranteeing that the data reached its destination. In response to the need for more reliable transmissions, the use of electronic mail to communicate EDI data is growing in popularity.

Perceiving the need to promote this functionally rich transmission medium, the NIST has designed and implemented an EDI Communications Platform. This Platform allows the communication of EDI data via the X.400 electronic mail protocol. It also provides direct connections via the FTAM and dial-up protocols. This prototype will become public domain software to promote OSI technology within the EDI community.

## Appendix A

# Supported X.435 Services

This appendix explains the provision of some of the X.435 services. The purpose of this appendix is to provide the reader with a general understanding of how the NIST EDI Communications Platform generates outbound X.435 messages. To accomplish this objective, the setting of specific X.435 message content attributes is outlined below.

Before listing specific X.435 attributes, some general notes must be presented.

- The Platform does not support EDI forwarding. Forwarded EDI messages are received and parsed correctly, but cannot be generated.
- The Platform does not support the generation of EDI additional bodyparts. If additional bodyparts are received, the contents are conveyed to the appropriate local EDI users via files.
- The Platform supports the generation and receipt of EDI notifications. Received EDI notifications are logged according to the *log-level* specified during the configuration of the Platform.

When generating X.435 messages, the data used to set the message content attributes is obtained primarily from three sources: the interchange to be transmitted, the entry for the local EDI user in the Local User Database, and the entry for the trading partner in the Trading Partner Directory. Specifically, X.435 content attributes are set as follows:

| | |
|---|---|
| EDI Message: | |
| Heading: | |
| This EDIM: | |
| User | set to X.400 address from Local User Database |
| User Relative Id | set to the current time (i.e., a timestamp) |
| Originator | set to X.400 address from Local User Database |
| Recipients: | |
| Recipient | set to X.400 address from Trading Partner Entry |
| Action Request | set to "for-action" |
| EDI Notif Requests | set from Trading Partner Entry |
| Resp Passing Allowed | set from Trading Partner Entry |
| Interchange Recip | set from interchange |
| Recipient Ref | set from interchange |
| Interchange Control Ref | set from interchange |
| Processing Priority | set from interchange |
| Ack Request | set from interchange |
| Comm Agreement Id | set from interchange |
| Test Indicator | set from interchange |
| Authorization Info | set from interchange |
| Recip Extensions | do not set |
| EDIN Receiver | set from Trading Partner Entry |
| Responsibility Forwarded | set to FALSE |
| EDI Bodypart Type | set from interchange |
| Incomplete Copy | set to FALSE |
| Expiry Time | do not set |
| Related EDI Messages | do not set |
| Related Non-EDI Messages | do not set |
| Obsoleted EDIMs | do not set |
| EDI App Security Elements | do not set |
| Cross Ref Info | do not set |
| EDI Message Type | do not set |
| Service String Advice | set from interchange |
| Syntax Identifier | set from interchange |
| Interchange Sender | set from interchange |
| Date and Time of Prep | set from interchange |
| Application Ref | set from interchange |
| Heading Extensions | do not set |
| Primary Bodypart | set from interchange |
| Additional Bodyparts | do not set |

## Appendix B

# Definitions for Frequently Used Expressions

This appendix provides a quick reference for some terms used frequently throughout this manual. The terms are presented in alphabetical order.

**FTAM-Control-File-Directory** Every outbound interchange to be transmitted via the FTAM transmission application has an associated control file. These control files contain the communications information required for the transmission. The Platform's User Interface writes the FTAM control files to this file directory. The name of this file directory is user-configurable.

**FTAM-Interchange-File-Directory** The FTAM transmission application polls a file directory for the presence of interchanges ready to be transmitted. The Platform's User Interface copies interchanges submitted by local EDI users for FTAM transmission to this file directory. The name of the file directory is user-configurable.

**Local-User-Outbound-File-Directory** The Platform assumes that all local EDI users will write all outbound interchanges to one file directory. The Platform's User Interface polls this file directory, and appropriately transmits any interchanges present. The name of this file directory is user-configurable.

**Transmission-Inbound-File-Directory** The Platform assumes that all inbound FTAM and dial-up transfers will be written to one file directory. The Platform's User Interface polls this file directory, and conveys any interchanges present to the appropriate local EDI users. The name of this file directory is user-configurable.

**TSDS** The Transaction Set Development System. This system provides an application program interface by which the Platform can communicate with local EDI users. Currently, the TSDS is not complete. References to the TSDS are placed in the Platform's configuration program for future use only.

**VAN-Control-File-Directory** Every outbound interchange to be transmitted to a VAN via a dial-up protocol has an associated control file. These control files contain the communications information required for the transmission. The Platform's User Interface writes the VAN control files to this file directory. The name of this file directory is user-configurable.

**VAN-Interchange-File-Directory** The dial-up transmission application polls a file directory for the presence of interchanges ready to be transmitted. The Platform's User Interface copies interchanges submitted by local EDI users for dial-up transmission to this file directory. The name of the file directory is user-configurable.

**VAN-Script-File** The inbound dial-up transmission application reads a file which identifies the file containing communications information for all VANs, and parameters for creating processes to establish connections with the VANs. The name of the file is user-configurable.