Digital DEC/EDI

Tru64 Unix - User Support Manual

Revised for Software Version: V4.0

Compaq Computer Corporation
Houston, Texas

November 2001

©Compaq Computer Corporation 1990,2001

Compaq, the Compaq logo, and VMS Registered in U.S. Patent and trademark Office.

OpenVMS and Tru64 are trademarks of Compaq Information Technologies Group, L.P. in the United States and other countries.
UNIX is a trademark of The Open Group in the United States and other countries.

All other product names mentioned herein may be trademarks of their respective companies.

Confidential computer software. Valid license from Compaq required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. government under vendor's standard commercial license.

Compaq shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is provided "as is" without warranty of any kind and is subject to change without notice. The warranties for Compaq products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

This document is the property of, and is proprietary to Compaq Computer Corporation. It is not to be disclosed in whole or in part without the express written authorization of Compaq Computer Corporation. No portion of this [enter document] shall be duplicated in any manner for any purpose other than as specifically permitted herein.

Compaq service tool software, including associated documentation, is the property of and contains confidential technology of Compaq Computer Corporation. Service customer is hereby licensed to use the software only for activities directly relating to the delivery of, and only during the term of, the applicable services delivered by Compaq or its authorized service provider. Customer may not modify or reverse engineer, remove, or transfer the software or make the software or any resultant diagnosis or system management data available to other parties without Compaq's or its authorized service provider's consent. Upon termination of the services, customer will, at Compaq's or its service provider's option, destroy or return the software and associated documentation in its possession.
Printed in the U.S.A.

The following are trademarks of Compaq Computer Corporation:
DEC, DEC/EDI, DIGITAL, OpenVMS, and the Compaq logo.
Adobe is a registered trademark of Adobe Systems Incorporated.
BT is a registered trademark of British Telecommunications plc.
InstallShield is a registered trademark of InstallShield Corporation.
MS and Windows are registered trademarks, and Windows 95, Windows 98, Windows NT and Windows 2000 are trademarks, of Microsoft Corporation.
Oracle is a registered trademark of Oracle Corporation.
SAP is a registered trademark of SAP AG.
UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Ltd.
All other trademarks not listed above are acknowledged as the the property of their respective holders.

Contents

Preface

Part I Configuration

Chapter 1 Creating a Normal Configuration

Chapter 2 Configuring the Pedi Gateway

Chapter 3 Configuring the OFTP Gateway

Chapter 4 Configuring the Import/Export Gateway

Chapter 5 Configuring the SMTP/ MIME Gateway

Chapter 6 Configuring the 3780 Gateway

Chapter 7 Scheduling Jobs

Chapter 8 Setting Up Other Routing Options

Chapter 9 Testing the Configuration

Part II Maintaining

Chapter 10 Monitoring

Chapter 11 Maintaining the Server

Chapter 12 System Performance

Chapter 13 Secondary Archive and Retrieve

Part III Problem Solving

Chapter 14 Finding and Fixing Problems

Chapter 15 Problem Solving – Pedi Gateway

Chapter 16 Problem Solving – OFTP Gateway

Chapter 17 Digital Problem Solving – SMTP /MIME Gateway

Chapter 18 Problem Solving – 3780 Gateway

Part IV Appendices

Appendix A Pedi Gateway: Supported Elements of Service

Appendix B Document and Transmission File Status Flows

Appendix C Environment Variables

Appendix D Files and Processes Used by Digital DEC/EDI

Appendix E Sample OFTP Logs and Traces

Appendix F Sample SMTP/MIME Logs and Traces

Appendix G Sample 3780 Logs and Traces

Appendix H SAP Status Generator

Appendix I Monitoring DEC/EDI from Tru64

Preface

Purpose of this Guide

This guide is provided to help you set up and get a Digital DEC/EDI system working, and then to keep it working.

It does this by describing the following:

- Configuring — how to create a working Digital DEC/EDI system.
- Maintaining — how to keep the system working.
- Problem Solving — what to do when something goes wrong.

Readership

This book is intended for anyone responsible for the installation, configuration and maintenance of a Digital DEC/EDI system.

It is recommended that you have read the Introduction to EDI book (referred to overleaf). This will provide you with an appreciation of general EDI and Digital DEC/EDI terminology.

Digital DEC/EDI Documentation

This is one of a set of Digital DEC/EDI books. The complete list is as follows:

- *Digital DEC/EDI: Introduction*

  This book introduces general EDI concepts, and Digital's EDI system, Digital DEC/EDI. It describes the main components of the Digital DEC/EDI system, and how business documents are processed and communicated to trading partners. The book seeks to establish the concepts and terms used by Digital DEC/EDI. These are also summarized in a glossary.

  You are strongly recommended to become familiar with the material in this book before proceeding to install or use Digital DEC/EDI.

- *Digital DEC/EDI: Installation*

  This book describes how to install the Digital DEC/EDI software, how to perform basic system configuration and how to verify such an installation. It describes how to install the Application Client, Server, Cockpit and CommandCenter components.

- *Digital DEC/EDI: Application Development*

  This book describes the Application Client interfaces and the means of connecting business applications to the Application Client. It also details the creation and deployment of mapping tables as part of the process of integrating applications with Digital DEC/EDI.

- *Digital DEC/EDI: User's Guides (Tru64 UNIX and Open VMS)*

  These guides contain information on setting up and operating Digital DEC/EDI systems. They also contain information covering configuration, maintenance and problem solving.

  The term *User's Guides* is used throughout this book to refer to the following books which are provided along with the Digital DEC/EDI Server they describe.

Tru64 UNIX	*Digital DEC/EDI: UNIX User Support Manual*
OpenVMS	*Digital DEC/EDI: OpenVMS User Support Manual - Volume 1*
	Digital DEC/EDI: OpenVMS User Support Manual - Volume 2

- *Release Notes*

  Further to the above, each software kit contains a set of release notes applicable to that software. These release notes contain information about known product problems (with workarounds where appropriate) and any operational tips or hints not provided as part of the above documentation set. You are strongly recommended to review these release notes before installing the software. Refer to the appropriate installation guide for information on how to locate the release notes.

- On-line Documentation

  Comprehensive on-line documentation is supplied with the Digital DEC/EDI software: for example, on-line help libraries and UNIX man page help information. In addition the Digital DEC/EDI Cockpit and CommandCenter kits contains the *Digital DEC/EDI: Error Messages Help Library.* This contains all error messages the product may log along with a description of why the message occurred and what to do about it.

It is provided in MS-Windows help library format. In addition, the CommandCenter CD-ROM provides on-line versions of all Digital DEC/EDI books in Adobe Acrobat format.

Digital DEC/EDI InfoCenter

For further information on Digital's EDI and Electronic Commerce Solutions and Services, please visit the EDI InfoCenter on the World Wide Web. The location is:

`http://www.decedi.com`

Related Third Party Documentation

Refer to the documentation provided with third-party products for installation and configuration details.

Typographical Conventions

Some information within this guide is specific to the database product you are using in conjunction with Digital DEC/EDI. The following conventions are used to indicate such text:

Oracle 8i — This indicates the adjoining paragraph contains information specific to running Digital DEC/EDI with the Oracle 8i database.

Server — The term Server is used in this document to reference a DEC/EDI environment where the environment may be running on either one physical AlphaServer system or on multiple AlphaServer systems connected in a TruCluster Server.

Digital DEC/EDI / Compaq DEC/EDI — The ownership of DEC/EDI was transferred to Digital GlobalSoft Ltd, a subsidiary of Compaq Computer Corporation based in India with effect from May 1, 2001. Consequent to this transfer, the name of the product was changed to Digital DEC/EDI. There may be references made to the existing name of the product, Compaq DEC/EDI in various sections of the documentation and screen display. We are in the process of implementing the name change across the product code and documentation. This is expected to be completed within the next couple of months. Pending the completion of this, all references to Compaq DEC/EDI in the documentation

pertain to the Digital DEC/EDI product. Please refer to the product website at `www.decedi.com` for further information on the transfer of ownership.

Prerequisites

Skills

A small number of UNIX® system commands are required to set up some elements of the Digital DEC/EDI system. These are explained where necessary. A sound knowledge of issuing simple UNIX commands (such as commands for file management, creating simple script files, accessing man page help) is recommended.

The main tool used in the detailed configuring the Digital DEC/EDI Server is the CommandCenter. This is a PC-based application that runs under Microsoft®-Windows™. Its style, appearance, and the controls it employs are intended to be typical of other professional Windows applications. If you are not familiar with such applications, you are recommended to review the MS-Windows tutorial supplied with MS-Windows (accessible from Program Manager/Help). The descriptions and explanations provided in this guide do not attempt to duplicate this knowledge.

System

Before performing any of the tasks described in this book, you should ensure that:

- You know what configuration of Digital DEC/EDI components you are trying to achieve. That is, you have selected the nodes on which you wish to run the Digital DEC/EDI Server and Application Clients, and you have decided which services you wish to run on the Server.
- You have decided which PCs are to run the CommandCenter and Cockpit.
- You have successfully completed and verified the physical installation of those components on their respective nodes in accordance with the instructions provided in the installation guide:
- The underlying network layer exists and the Server knows about each of the other nodes in the Digital DEC/EDI network. Conversely each of the other nodes in the network knows about the Server node. You should be able to test this by using `ping` or a similar networking utility, for

example, to check the network link to a node called `edisrv` from another UNIX machine:

```
# /sbin/ping edisrv.edi.dec.com
PING edisrv.edi.dec.com (16.36.60.234): 56 data bytes
64 bytes from 16.36.60.234: icmp_seq=0 ttl=254 time=4 ms
64 bytes from 16.36.60.234: icmp_seq=1 ttl=254 time=3 ms
...
```

If the output from the ping utility is not similar to the above, you have a network problem that you need to fix before proceeding.

If TCP/IP is Installed

- You can successfully run the Network Tester provided with the CommandCenter

Part I Configuration

This part of the Digital DEC/EDI User's Guide describes the configuration activities needed to create a working Digital DEC/EDI system. It also describes how you should set about testing the configuration.

Chapter 1 Creating a Normal Configuration

This chapter presents the tasks you need to complete to configure a running Digital DEC/EDI system assuming the normal routing of data through the Server. The configuration for other options for routing data through the Server is described in Chapter 8 *Setting Up Other Routing Options*. The style of description assumes you are doing this for the first time.

Introduction

Before you begin this configuration, you must have completed the initial configuration activities described in the Installation manual.

The steps involved in configuring a Digital DEC/EDI system are presented in the order that you are recommended to do them. You do not have to stick to this order, but if you do not, you may end up typing more, for example, if you register your applications before you create trading partner agreements, then the application name is available from a pick list when you create the agreement. If you perform these tasks in the other order, then you have to enter the application name twice.

An Overview of Configuration

Configuration involves the following stages:

1. Defining which Digital DEC/EDI Translation and Communications Services are to run on the Server being configured. See *Defining Which Services to Run* on page 1-2.
2. Creating the definitions of which EDI documents you wish to exchange with your trading partners. See *Creating EDI Document Definitions* on page 1-4.
3. Developing and registering your business applications, the nodes on which they run, and any Mapping Tables they require. See *Developing and Registering Applications* on page 1-10.
4. Defining which communications gateways you wish to run, and which connections you want to establish to your Trading Partners. See *Defining Communications Links* on page 1-11.
5. Defining your Trading Partners and the agreements you have with them for the exchange of EDI data. See *Defining Trading Partner Agreements* on page 1-16.

Defining Which Services to Run

You need to define which of the various Translation and Communications Services you intend to run on the Server.

The main purpose of this is to tell Digital DEC/EDI, which components to start when you next issue the Digital DEC/EDI system start-up command. You could elect to start all services, but you are recommended to define only those that you are licensed to use. The one you intend to use as unlicensed services do not start, and unused services consume resources.

If you have not installed sufficient product licenses to enable all the components you have requested to be started, those that do not have licenses will not start, and corresponding errors will appear in the Error Log. In addition, starting services that you do not intend to use is wasteful of system resources.

When running the DEC/EDI Server on multiple systems in a TruCluster Server, each system on which you plan to start DEC/EDI must have its own unique set of licenses.

You define the services you want to run by using the CommandCenter Management Services Editor "Configure Services" option. In this release of Digital DEC/EDI, you can select to run:

- EDIFACT/ODETTE Translation Services.
- X12/TDCC Translation Services
- TRADACOMS Translation Services.
- Import/Export Communications Services.
- OFTP Communications Services.
- 3780 Communications Services
- SMTP/MIME Communications Services
- Pedi (X.400/X.435) Communications Services.

You should select to run at least one Translation Service and at least one of the Communications Services options.

The EDIFACT and X12 Transmission File Builders (TFBs) provide a default build interval of 5 minutes, which you can change if you wish. Any subsequent change you make to this information will take effect when you next start the Digital DEC/EDI Server.

Changing the Build Interval

Use the Configure Build Intervals option to change the build intervals.

Creating EDI Document Definitions

Several stages are involved in the creation of EDI Document definitions for the data you wish to exchange with your Trading Partners. The amount of work involved depends on whether you want to exchange standard documents, whose formats are exactly as defined by the EDI standards bodies, or whether you want to exchange modified or entirely new definitions. The following stages are involved:

- Loading the definitions for the version of EDI standard on to the Server by using the latest Message Update Service kit. See *Loading EDI Document Definitions* on page 1-4.
- Using the EDI Table Editor to make any changes to the document definitions you want to use. See *Customising Existing EDI Document Definitions* on page 1-7.
- Defining data labels for the document. See *Creating Data Labels* on page 1-8.

Loading EDI Document Definitions

Digital ships EDI Document definitions for the different versions of the EDI standards that Digital DEC/EDI supports. This facility is referred to as the Message Update Services (MUS). With the Digital DEC/EDI Server, you have a version of the MUS kit that was current when the Server was produced. Later MUS kits and versions are available on `http://www.decedi.com`. If your support contract entitles you to receive updates, contact your Digital DEC/EDI Service provider for details on the availability of the latest MUS kits for use with your version of Digital DEC/EDI.

These MUS kits contain both EDI Document definitions and the accompanying dictionaries for the corresponding versions of EDI standards. Even if you want to use an EDI document entirely of your own design, you can save yourself significant effort if you base it on a version of an EDI standard for which MUS tables are supplied. You should expect to install at least one version of an EDI Standard on your Server. In practice you may choose to install several different versions.

The Message Updates Services allows you to perform the following actions:

- Installing one or more versions of an EDI Standard.
- Copying one installed version of an EDI Standard to another version.
- Copying data labels from an installed version of an EDI Standard to another.
- Deleting an installed version of an EDI Standard.

The first of these is important in the initial setup of a Digital DEC/EDI Server and is described in a following section.

Checking if Message Updates Services are Installed

You should have installed the subsets that contain the Message Updates Service when you installed Digital DEC/EDI Server. If you are unsure whether you have installed the Message Update Service subsets, the following command shows you which Digital DEC/EDI Message Update Service subsets are installed:

```
# /usr/sbin/setld -i | grep DEDIAMSG
```

This displays a list of the Digital DEC/EDI Message Update Service subsets, indicating those that are installed and those that are not. Check the list of subsets installed includes the Message Update Services.

To load EDI standards you need to ensure you have both the Message Updates Service Base subset installed and at least one Message Update Service standard (for example, EDIFACT) subset.

Installing a New Version of an EDI Standard

You invoke the Message Update Services with the following command:

```
# decedi_must
Digital DEC/EDI Message Updates Service Tool V2.1 for Digital
DEC/EDI V4.0
Copyright (C) 1990-2001 Digital Computer Corporation. All
rights reserved.
1)  Install new versions of EDI standards from the Digital
DEC/EDI  IMPDEF directory.
2)  Copy an existing version of a standard to a new version of
the same standard.
3)  Copy the data labels associated with one version of a
standard to a different version of the same standard.
4)  Delete a version of a standard from Digital DEC/EDI.
5)  Exit.
Enter an option (1-5): 1
```

Selecting option 1 causes a list of the available EDI Standards to be displayed, similar to the following:

```
List of standards and versions to install:
1)  Edifact   1        19) TDCC      002007    37) X12  003010
2)  Edifact   901      20) TDCC      002008    38) X12  003020
3)  Edifact   902      21) TDCC      002009    39) X12  003021
4)  Edifact   911      22) TDCC      003000    40) X12  003022
5)  Edifact   912      23) TDCC      003001    41) X12  003030
6)  Edifact   921      24) TDCC      003020    42) X12  003031
7)  Edifact   D93A     25) TDCC      003030    43) X12  003032
8)  Edifact   D94B     26) TDCC      003040    44) X12  003040
9)  Edifact   D95A     27) TDCC      003050    45) X12  003041
10) Edifact   D95B     28) TDCC      003060    46) X12  003050
11) Edifact   D96A     29) All       TDCC      47) X12  003051
12) Edifact   D96B     30) Tradacoms 89        48) X12  003052
13) Edifact   S93A     31) Tradacoms 93        49) X12  003060
14) All       Edifact  32) All       Tradacoms 50) X12  003061
15) Odette    1        33) X12       002001    51) X12  003062
16) Odette    3        34) X12       002002    52) All  X12
17) All       Odette   35) X12       002003
18) TDCC      002006   36) X12       002040
E) Exit
Enter the menu number of the standard and version to install:
```

Enter the menu number that corresponds to the version of the EDI Standard you wish to install. Displayed messages indicate the progress of loading the selected version into the Server database.

Once the process is complete, select the `Exit` option.

Customising Existing EDI Document Definitions

The Messages Update Services allow you to load EDI document and dictionary definitions into your system as defined by the corresponding EDI Standards bodies. If you are expecting to use entirely standard EDI documents, you need not perform any of the activities outlined in this section. However, in practice, you may need to make at least small changes to EDI documents as supplied to suit the particular documents you have decided to exchange with your trading partners; for example, to make the presence of a certain piece of data *optional* where the EDI Standard defines it to be *mandatory*. Equally, you may wish to create entirely new document definitions, where the EDI Standards bodies have not already defined one that corresponds to your business requirements. Or, you may simply wish to look at the individual details of the EDI document definitions.

The EDI Table editor within the CommandCenter allows you to:

- View the contents of existing document definitions as loaded on a specific Digital DEC/EDI Server.
- Make modifications to existing document definitions for use by all trading partners or a single named trading partner, or a group of Trading Partners.
- Create entirely new document definitions for use by all Trading Partners, or a single named Trading Partner, or a group of Trading Partners.
- View the dictionary components of an EDI Standard (segments and elements).
- Make changes to dictionary components for use by all Trading Partners, or a single Trading Partner, or a group of Trading Partners.
- Create entirely new dictionary components for use by all Trading Partners, or a single named Trading Partner, or a group of Trading Partners.
- Modify enveloping and syntax defects.

The EDI Table editor allows you to load definitions from a Server to the PC. You can then review and make changes on the PC, and save those changes back to the Server. You need access to the dictionary components on a Server when making changes.

The Server uses any changes you have stored in the following circumstances:

- When it is next started.
- If you explicitly request the Translation Services components to reload their EDI Tables information (reload the Tables cache).
- When the Translation Services components require a copy of the relevant definitions and do not already have a copy loaded into the in-memory Tables cache.

You may also need to rebuild the Profile Cache and (or) regenerate the Data Labels, depending on the changes you have made.

See *Modifying Configuration Information* on page 1-21 for more information on how and when the different Server components take account of changes in configuration information, including changes to EDI Tables.

Creating Data Labels

Once you have installed a version of an EDI Standard, you need to create data labels for each element of data defined within that standard. These labels are used during the process of creating Mapping Tables. You create data labels for a version of a standard by using the Data Label Generator, `decedi_dlg`.

This tool gives you a choice over the manner in which the data label names are constructed. Two main options are presented for the naming of data labels:

```
<Segment ID> + <Element position> [+ <Sub-element position>]

<Segment ID> + <Element ID> [+ <Sub-element ID>]
                                              [+ <Repeat Count>]
```

The first of these is the default. The second is referred to as the ID format.

You can also elect to separate each of the constituent fields of the data label name with either a hyphen, or (by default) an underscore character, to make it easier to read.

On-Line Help

For a more complete description of all the parameters and options provided by the Data Label Generator, refer to its on-line help:

```
# man decedi_dlg
```

Using the Data Label Generator

To use the Data Label Generator, or to access its man page information, you must be logged in to the `root` account.

The following is an example of using the Data Label Generator to create data labels for EDIFACT, Version 921. It uses the second of the above two naming schemes, and specifies (by default) underscores as the separator character:

```
# decedi_dlg -s=EDIFACT -v=921 -f=ID
Digital DEC/EDI Data Label Generator V4.0
© Digital Computer Corporation 1990-2001. All Rights Reserved.
Now generating the data labels... please wait
processing segment AGR
processing segment AJT
processing segment ALC
processing segment ALI
processing segment API
...
...
processing segment TSR
The data labels were successfully generated
NOTE: Data labels are cached, so replace the relevant tables
caches for these data labels to take effect.
Now exiting the Data Label Generator...
```

Normally, the tool generates data labels for a complete version of an EDI Standard. However additional command line switches are available to generate the necessary data labels for Trading Partner and Trading Group specific documents within an EDI Standard.

You must generate data labels for a version of an EDI Standard before you can reference the corresponding fields of data in a Mapping Table.

If you make any changes to the dictionary for an EDI Standard by using the Table Editor, you must re-run the Data Label Generator before the changes are fully effective.

Developing and Registering Applications

This activity comprises a number of separate parts, most of which are covered in *Digital DEC/EDI: Application Development* and are not detailed here. Specifically, that book covers the activities associated with interfacing or creating business applications with the Digital DEC/EDI Application Client and in producing the necessary Mapping Tables.

Before these business applications can be run, they need to be registered with the Server. This is done by using the Management Services Editor. Each business application that uses a Server must be registered, and the node or nodes on which it is to run must be declared. Specifically the name, or *application_ID* of each business application must be registered. Business applications that are not registered, or nodes from which an application has not been declared to exist, have their requests rejected by the Digital DEC/EDI Server when they attempt to post, fetch or track objects.

Information on registered business applications is held in memory within the Digital DEC/EDI Server in the Profile Cache. When you make additions or changes to the list of registered applications, or the nodes on which they run, you are prompted to decide whether you wish the Profile Cache to be rebuilt and reloaded. If you decline to do this (perhaps because you want to make a series of changes before committing them all), the changes you make do not take effect until you rebuild, and either:

- Explicitly reload the cache.
- Restart the Server, whereupon the Profile Cache is automatically loaded.

Note that while the Profile Cache is automatically loaded when you start the Server, it is only rebuilt when you explicitly request it. You can do this by using any of the CommandCenter editors.

Once you have registered your business applications, you need to copy your compiled Mapping Tables to the Mapping Table Repository on the Server. The Mapper accesses this repository for all Mapping Tables your business applications instruct it to use. You copy compiled Mapping Tables to the repository by using the Mapping Table editor.

The Mapper caches Mapping Table information in memory. However, when you copy a new version of a Mapping Table into the repository, the Mapper replaces the copy it has in memory with the new copy the next time it needs to use that table.

Defining Communications Links

The communication services the server provides are defined in Chapter-15. Each of the different possible services is provided by a corresponding communications gateway. For each gateway, you also define individual connections. The information you define for each connection specifies the communications link between your system and your Trading Partners or VANs. Similarly, for each connection, you also define node information. This specifies on which node(s) the connection can be registered.

Note: *If you are adding a new connection to a running server, you must disable and then re-enable the Gateway (not the connection) to make new connection recognized by the relevant DEC/EDI Gateway.*

The task of defining the communications links for your system has three main stages:

- Defining operational parameters for each gateway.
- Defining operational parameters for each connection.
- Defining operational parameters for each node.

These tasks are performed by using the CommandCenter Communications Editor.

The following sections provide more detail on what you need to do. More specific details are provided in later chapters and include any additional setup of related products that you need to do.

Defining Communications Gateway Parameters

For each gateway you intend to use, you have to define its gateway parameters. While these parameters vary depending on the type of gateway, they define operational characteristics of the gateway as a whole, and affect all connections defined for that gateway.

The help libraries provided within the Communications Editor provide more specific information on what each of the parameters is, and what values you should enter for them. Any subsequent changes you make to these gateway parameters are effective when you next start the appropriate gateway, explicitly or when you next start Digital DEC/EDI.

Defining Communications Connection Details

You need to define the operational details of each separate communications link you expect to use to send and receive data. You use the Communications Editor to do this. Each connection you define is identified with a name that you choose that needs to be unique within the Server. This name is the *Connection_ID*.

While the details for Import/Export connections are fairly simple, the necessary details for some of the other communications gateways, such as OFTP and Pedi connections, are more complex. Many parameters for such connections need to match corresponding details already configured in other underlying products, for example, X.25 in the case of OFTP, and MAILbus 400 in the case of Pedi connections.

See the help library within the Communications Editor for more specific information of the details of defining the parameters for individual connections and how they need to match corresponding elements of configuration in related products.

Any new connections you define or changes you make to existing connection details are effective when the next request is made to the corresponding gateway to use that connection. They are not effective for transfers already in progress.

Enabling and Disabling Gateways and Connections

As you create new definitions for individual gateways and connections, they are all enabled by default. A gateway or connection can be used to send or receive EDI data only if it is enabled.

You can explicitly enable and disable gateways and connections by using the Communications Editor. You might choose to temporarily disable an operational gateway or connection to prevent it being used, perhaps while you are changing one or more of its parameters, or because you know that attempts to use a communications link will fail, and you want to prevent unnecessary logging of errors where the system will otherwise try to use it.

Requests from the Communications Editor to enable a gateway or connection are effected immediately. Requests to disable a gateway or connection are effective the next time a connection is used.

Defining Jobs

Some communications gateways are 'job based'. This means that they can run a number of different types of jobs. The Import/Export gateway is an example of this type of gateway where it supports three built in jobs, those of IMPORT, EXPORT and IMPEXP (a combination of Import and Export).

Other communications gateways, such as the 3780 Gateway, also support user defined jobs. These jobs can do such things as send files to a VAN, receive files from a VAN, or query the VAN. You use the Communications Editor to create or amend jobs. Each job you define will have its own unique job id, and be associated with a connection id. Each 3780 job must have a unique Job Identifier, be associated with a script file based on the TCL programming language and be associated with a specific connection identifier. You can use the same Job Identifier for different connections but not the same connection.

Defining Communications Node Details

In cluster environment, the few Connection details can be node specific information. You need to define the operational details of each separate communications link you expect to use to send and receive data, which are node specific details. Each node you define is identified with a name that you choose that needs to be unique within the Connection. This name is the Node name. This node name with Connection ID can be used to identify the node within the server.

While the details for Import/Export connections, OFTP, PEDI and SMTP are fairly simple, the necessary details for some of the other communications gateways, such as 3780 are more complex detail. Many parameters for such connections (3780) need to have more details. If no node is specified under 3780 connections, then it will take the default parameter values specified in Connection.

See the help library within the Communications Editor for more specific information of the details of defining the parameters for individual connections and how they need to match corresponding elements of configuration in related products.

Any new nodes you define or changes you make to the existing node details are effective when the next request is made to the corresponding gateway to use that connection. They are not effective for transfers already in progress.

Coping with Errors in Transmission

As data is sent and received by each of the communications gateways, various errors may occur. These may range from configuration problems generic to the whole gateway or connection, or specific to an individual file being sent or received or specific to an individual transmission. Digital DEC/EDI is designed to accommodate this, and keeps track of any errors it encounters.

Import/Export, Pedi and 3780 Gateways

Each gateway has a *connection error limit* defined. This is a number you can define to control how many consecutive errors are to be tolerated for each connection link before it becomes disabled automatically. When the number of consecutive errors on a particular connection reaches the defined connection error limit, that connection is disabled automatically. Manual intervention is required to enable the connection before it may be used again.

Note that this process is designed to count consecutive errors and not simply count all errors. This is designed to catch solid faults that affect all transmissions on a connection, and to provide a means of tolerating intermittent faults. While the detection of any fault causes an error to be recorded in the Error Log, the error count for a connection is reset to zero after each successful transmission.

OFTP and SMTP/MIME Gateways

The OFTP and SMTP/MIME gateways have similar facilities, but the real-time nature of the connection implies slightly different handling. The connection error limit value is set on a per-connection basis. Otherwise the connection error mechanism is the same - as the number of consecutive errors for a particular connection is reached that connection becomes disabled.

OFTP, SMTP/MIME and 3780 Gateways

In addition, some gateways implements a file retry mechanism. As part of defining the parameters for a connection, you define a *file retry limit* value. This is a limit on the number of unsuccessful attempts to send a particular transmission file before the gateway marks the transmission file as **`FAILED`**. The audit trail for each transmission file contains a *retry count* field. These gateways increment this field each time it fails to send that transmission file. If the retry count reaches or exceeds the file retry limit defined for that connection, that transmission file is set to **`FAILED`**. You use the Cockpit to view the retry count for a transmission file. You also use the Cockpit to reset the transmission to cause it to be reprocessed (in which case its retry count is also reset to zero) or to cancel it from further processing.

Note that the file retry checking mechanism applies only to outbound data. The gateway does increment the retry count value for each failure, but does not take any action when it reaches the limit defined for that connection.

Defining Trading Partner Agreements

The definition of trading partner agreements is the central piece of the configuration. It is the point in the configuration where you bring together the other pieces of configuration that you have already defined: the applications you are running, the EDI documents you exchange and the communications connections you use to send data to, or receive data from, your trading partners.

You use the Trading Partner editor to define your trading partner information. This information is defined at three levels:

- The trading partner level.
- The application level.
- The document level.

The trading partner level forms the top of the hierarchy. You need to define each of your trading partners at this level. Each is separately identified with a name you choose for that trading partner, referred to throughout the system as the *Partner_ID*. Each name that you choose needs to be unique within the Server.

Also at the trading partner level you specify the names of one or more communications connections that you expect to use for this trading partner. These connections are defined using their *Connection_ID*. Each *Connection_ID* you refer to must already have been created by using the Communications editor. (See *Defining Communications Connection Details* on page 1-12). For each of the connections that you specify, you can define additional parameters that are used by the TFB as it builds transmission files.

For each connection, these control:

- The maximum size of an outbound transmission file.
- The maximum number of documents per group.
- The maximum number of groups per interchange.
- The maximum number of interchanges per transmission file.

An important additional connection parameter is the *Connection Specific Data* field. This can be used to provide trading partner specific information to be used by the selected communications gateway, when sending transmission files. See *Overriding Connection Details By Using Connection Specific Data* on page 3-8 for more details of using *Connection Specific Data*.

For the connections you specified at the trading partner level, you specify one to be used as the default for the trading partner. The others that you specify can be used in place of the default, for the exchange of individual types of document. This is explained later in this section.

For each trading partner that you define, you must define one or more agreements at the application level. Each agreement covers the exchange of EDI documents between the trading partner and a named Digital DEC/EDI application, specified by using its *application_ID*. For each agreement, you specify the parameters that control the enveloping and formatting of the EDI data you exchange. This information includes:

- The EDI Standard and numbering scheme to be used.
- Delimiters (for example, segment terminator, element separator, release character).
- Interchange enveloping (UNB, ISA, BG, STX) parameters.
- Group enveloping (UNG, GS) parameters, if you are using Functional Groups.

For each agreement that you define at the application level, you must define one or more documents. Information at the document level defines, for the relevant agreement, which type of EDI document you intend to exchange. For example, an EDIFACT 90.1 INVOIC. At this level you also define a number of indicators including the *direction_indicator* and the *test_indicator*. The *direction_indicator* specifies whether the definitions are to be used for outbound only, inbound only or bidirectional (both outbound and inbound) exchanges of data. The *test_indicator* is used to specify whether the data being exchanged is live or test data.

Also at the document level, you may specify a different *connection_ID* to be used that overrides the default previously configured at the trading partner level. You can only choose to use *connection_IDs* from the list that you defined at the trading partner level. You may find this useful to specify a different communications link to use for one particular transaction, for example, because it is exchanging test data, or because the data needs some special processing by the trading partner.

In a typical Digital DEC/EDI Server configuration, you may define many trading partners, and for each that you exchange EDI data with, you will have at least one agreement and document defined. For some trading partners you may have multiple agreements, and for some agreements you may have specified that you will exchange more than one type of document.

The Digital DEC/EDI Server uses an in-memory copy of this trading partner data. Once you have used the Trading Partner editor to define the data, you must build a copy of the data, referred to as the Profile Cache. The Profile Cache contains this trading partner data, and other configuration data, in a form that is optimised for access by the different components of the Digital DEC/EDI Server. This Profile Cache is automatically loaded into memory when the Server is started. If you make any changes to your trading partner configuration, and you wish the Server to start using the changed data, you must rebuild the Profile Cache, and, if the Server is already running, request the new cache to be loaded into memory. You do this by using any of the CommandCenter editors. See *Modifying Configuration Information* on page 1-21.

To aid new users, a Trading Partner Wizard is available within the Trading Partner Editor. This leads you through setting up a simple trading partner agreement.

Starting and Stopping the Server

Starting

The Digital DEC/EDI Server needs to be started before it can be used to exchange any EDI data. The Server is started by issuing the following command on the Server from the root account:

```
# decedi_start
```

You can either do this interactively or you can include a link to it in the start-up procedure for your UNIX system. If your Server is configured to run in a TruCluster Server then you should do this on each member of the TruCluster that will participate in your DEC/EDI Server.

The start-up procedure checks to see if DEC/EDI has already been started on this system, runs any Server and/or system specific setup tasks that you have defined, verifies that it can access the required DEC/EDI resources (the database, Memory Queues, caches, etc.) and sees if it should start as a member of a TruCluster Server enabled environment. If this is the first or only member of the DEC/EDI environment then it reloads the work queues (Memory Queues) either from the file saved during shutdown or directly from the DEC/EDI Database if the save file is missing. If there are already members of the DEC/EDI environment running in the TruCluster Server then the startup procedure simply joins that environment and uses the existing work queues. Finally, the startup procedure starts the services that you have requested (see *Defining Which Services to Run* on page 1-2).

As each service starts, it checks the existence of valid license PAKs before searching for work to do in the Memory Queues.

Stopping

To stop the Server, issue the following command on the Server from the root account:

```
# decedi_stop
```

If your Server is configured to run in a TruCluster Server then you should do this on each member of the TruCluster that is participating in your DEC/EDI Server instance.

The shutdown procedure tells all of the background processes (daemons) associated with the services running on the DEC/EDI Server member to finish what they are doing and to shutdown cleanly. The shutdown procedure will monitor each DEC/EDI process to ensure that it is has not stalled. If a process is detected as stalled (that is, it gathers no CPU time) for a period of 30 seconds then DEC/EDI will send it another shutdown request. It sends it 3 requests until it finally terminates the process as being unresponsive.

Once all processes have shutdown, the shutdown procedure takes a copy of DEC/EDI's work queues (the Memory Queues) and saves the contents to a file so that the startup procedure can reload the contents during startup.

Note that if you are shutting down an instance of DEC/EDI that is running on multiple members of a TruCluster Server then only the last member to run decedi_stop will save the contents of the work queues; all other members will simply say that they are leaving the Server environment.

Modifying Configuration Information

This section describes how each of the main pieces of configuration data are accessed by a running Digital DEC/EDI Server.

Once a Digital DEC/EDI Server is running, any changes you make to configuration data do not necessarily take effect immediately.

Modifying Mapping Tables

When the Mapper is first used after a Digital DEC/EDI Server is started, it reads the specified Mapping Table from the Mapping Table Repository into an in-memory cache. It adds further Mapping Tables to the cache as they are needed. Before the Mapper re-uses a Mapping Table in its cache, it checks to see if there is a more recent copy of the table in the repository. If there is, it replaces the table in the cache with the newer one from the repository.

This means any changes you make to Mapping Tables in the repository are effective when the table is next used.

When the cache becomes full, that is, you are trying to use more different Mapping Tables than the cache can hold, the Mapper replaces the least recently used Mapping Table in the cache with any new tables it needs to use.

The number of Mapping Tables that can be held in the in-memory cache is governed by the environment variable `DECEDI_MAX_MAPPING_TABLES`. See Appendix C *Environment Variables* for more information on this and other environment variables.

Modifying EDI Document Definitions

Once a Digital DEC/EDI Server has been started, the Translation Services components read EDI document definitions into an in-memory *tables cache* when they are needed. This means the performance in handling the first document of a particular version of an EDI standard may be relatively slow, but subsequent documents are handled much more quickly.

If the cache becomes full, the Translation Services components replace the least recently used version of a standard with additional versions, as required. Otherwise, provided the cache does not become full, any changes or additions you make to EDI Tables data become effective only when you issue an explicit replace cache command from the EDI Table editor, or when you restart the Digital DEC/EDI Server.

Two environment variables define the maximum size of the in-memory cache. They are `DECEDI_MAX_DOCUMENTS` and `DECEDI_MAX_SEGMENT_TABLES`. See Appendix C *Environment Variables* for more information on these and other environment variables.

Modifying Trading Partner Profile Data

The profile cache contains information from the Trading Partner Tables (created by using the Trading Partner Editor), information describing registered business applications and the nodes on which they run (created by using the Management Services Editor), and the header information of the EDI documents (created by using the EDI Tables Editor). The profile cache needs to be built explicitly before it can be loaded into memory (or reloaded) and used by the Digital DEC/EDI Server. Any changes you make to information contained in the cache are not built into a new version of the cache unless you request it explicitly. The relevant CommandCenter editors remind you of this and ask you whether you want to rebuild the cache after you make changes.

When you start a Digital DEC/EDI Server, it loads the latest built copy of the profile cache into memory. If you rebuild the profile cache to include changes, you also need to reload the cache to cause a running Digital DEC/EDI Server to use the newly built cache.

Modifying Communications Data

The information that defines the configuration of the Digital DEC/EDI Communications Services splits into two parts:

- Gateway data.

  This configuration data refers to an entire communications gateway. The gateway reads this data each time it is enabled. Any changes you make to this data are effective only if disable and then re-enable the gateway or if you restart the whole Digital DEC/EDI Server.

- Connection data.

  This configuration data refers to individual connections. Each connection has data defined for it. The gateway reads this data each time a connection is used. So, any changes you make to this data are effective the next time that connection is used.

If you add a new connection and/or delete an existing connection to a running DEC/EDI Server then you must disable and re-enable the gateway to get it to recognize the new connection. If you delete a connection from one gateway and add the connection to a different gateway using the same connection id then you must disable and re-enable both gateways.

Chapter 2 Configuring the Pedi Gateway

The Digital DEC/EDI Pedi gateway can send and receive the following X.400 message types:

- 1984 and 1988 P0.
- 1984 and 1988 P2 (IA5TEXT Bodypart).
- 1984 and 1988 P2 (Bilaterally defined Bodypart).
- Pedi (X.435). Appendix A details the individual X.435 elements of service that the Pedi gateway supports.

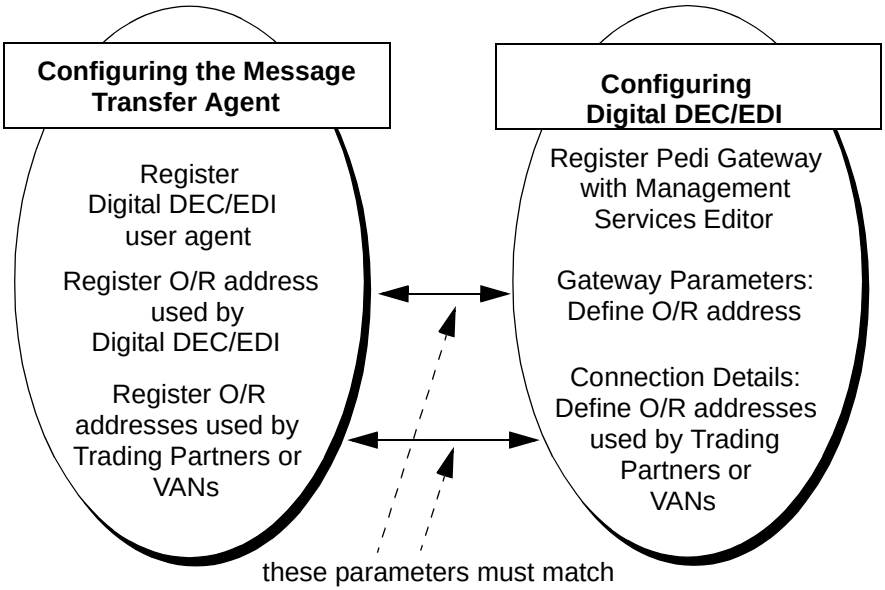
It does this by using X.400 services provided by the MAILbus 400 Message Transfer Agent (MTA). This, in turn, requires services provided by other products, in particular by the X.500 Directory Services product.

Setting Up

To set up a working Digital DEC/EDI Pedi gateway, you also need to configure the MAILbus 400 and X.500 products. It is important that configuration parameters you enter for these products match corresponding gateway and connection parameters that are entered by using the Communications Editor, and that form part of the configuration of Digital DEC/EDI. This chapter describes how you do this.

The following figure shows a summary of what you need to do to set up the Pedi gateway.

Figure 2-1 Configuring Digital DEC/EDI and the MAILbus 400 MTA



Terminology: Gateway and User Agent

Digital DEC/EDI uses the term *gateway* to refer to a component that, as far as Digital DEC/EDI is concerned, provides communications services for a given communications protocol. Digital DEC/EDI has several different communications gateways. The one relevant to this chapter is the Pedi gateway.

The MAILbus MTA uses the term *user agent* to refer to a piece of software that connects to the MTA to send or receive messages.

The Pedi gateway that is provided by Digital DEC/EDI, acts as a user agent of MAILbus 400.

This chapter uses the terms *user agent* and *gateway* interchangeably, depending on the context.

Before You Start

The procedures described in this chapter assume the following:

- You have set up a *Routing Domain* within which your MTA and the Digital DEC/EDI Pedi gateway operate.

  A Routing Domain is a MAILbus 400 term that describes a group of MTAs that access the same routing information. This routing information is held in an X.500 directory.
- You have set up an MTA.
- The MTA verification procedure (IVP) runs successfully.

Refer to the *MAILbus 400 MTA Planning and Setup Guide* for more information, if you have not already completed the above activities.

Configuring the MAILbus MTA

This section describes the following:

- How to register the Digital DEC/EDI Pedi gateway as a user agent of the MAILbus 400 Message Transfer Agent (MTA).
- How to set up the O/R address that the user agent serves, that is, the user agent O/R address.
- How to define O/R addresses for each Trading Partner or VAN to whom the MTA sends messages. In Digital DEC/EDI, the parameters that comprise the O/R address need to be defined for each separate connection-id.

  These addresses can be complete or partial O/R addresses. They include a *routing instruction* to enable the MTA to determine how to deliver messages to this O/R address. See the *MAILbus 400 MTA Planning and Setup Guide* for details of defining O/R addresses for individuals in other routing domains.

Registering the Pedi Gateway as a MAILbus User Agent

To register the Digital DEC/EDI Pedi gateway as a user of the MTA, take the following steps:

Step 1: Use a text editor of your choice to open your MTA start-up script, `/var/mta/scripts/start_mta.ncl`.

Step 2: In the section that defines user agents, add the following lines:

```
create mta agent "DECEDI" type xapi

enable mta agent "DECEDI"
```

Step 3: Save the file.

Step 4: Run your MTA start-up script to make the changes take effect:

```
# ncl
ncl> do /var/mta/scripts/start_mta.ncl
ncl> exit
```

Defining the O/R Address for the Pedi Gateway

The Digital DEC/EDI Pedi gateway acts as a single O/R address. This O/R address is the Originator address for any outgoing EDI message. The values you need to register by using the X.500 Directory Services must match exactly with the corresponding values entered as part of the Pedi gateway configuration within Digital DEC/EDI.

For example, suppose that the user agent formed by the Digital DEC/EDI Pedi gateway is in the Routing Domain `myco-rd`, and needs to have the following characteristics:

Country Code	gb
Administration Domain	ptt-admd
Private Domain	myco-prdm
Organization Domain	myco-purchase
MTA Server Name	myco-mta

Defining the Address

To define this address, and show that it is served by the Digital DEC/EDI user agent, you would need to add the following entry to the X.500 directory:

```
ncl>  set mts "/mts=myco-rd" oraddress -
_ncl>  "c=gb; a=ptt-admd; p=myco-prmd; o=myco-purchase" -
_ncl>  routing instruction [action=deliver, -
_ncl>  server mta="myco-mta", agent="decedi", -
_ncl>  definitive oraddress="c=gb; a=ptt-admd; -
_ncl>  p=myco-prmd; o=myco-purchase;"]
```

The following table shows you the fields which make up the O/R address, and indicates the names used to refer to each separate piece both as ncl O/R address qualifiers and as seen on the Pedi gateway details screens within the CommandCenter Communications Editor.

The values you enter into Digital DEC/EDI and the values you enter into X.500 must match exactly, including the case of the letters

.

Table 2-1 Pedi Communications: Gateway Parameters

Ncl O/R Address Qualifier	Digital DEC/EDI Gateway Parameter
C	Local Country Code
A	Local Administration Management Domain
P	Local Private Management Domain
O	Local Organization
OU1	Local Unit Name 1
OU2	Local Unit Name 2
OU3	Local Unit Name 3
OU4	Local Unit Name 4
CN	Local Common Name
G	Local Given Name
I	Local Initials
S	Local Surname
Q	Local Generation
X.121	Local X.121 Address
T-ID	Local Terminal ID
N-ID	Local Unique UAID

Defining O/R Addresses for Trading Partners and VANs

This section describes how to define your Trading Partner's or VAN's O/R address to the MTA.

Before you can define your Trading Partner's or VAN's address, you need to do the following:

1. Create a domain entry for the remote domain, if the Trading Partner or VAN is in a remote routing domain.

   For example, if the remote domain is `acme-rd`:

   ```
   ncl> create mts "/mts=myco-rd" domain "acme-rd" -
   _ncl> different ccitt domain true
   ncl> set mts "/mts=myco-rd" domain "acme-rd" description -
   _ncl> "the acme-rd routing domain; direct connection"
   ```

2. Specify a routing instruction that defines how the remote domain is served. For example:

   ```
   ncl> set mts "/mts=myco-rd" domain "acme-rd" -
   _ncl> routing instruction -
   _ncl> [action=transfer to domain, boundary mta="myco-mta"]
   ```

The ncl command that you use to describe an O/R address in another routing domain is very similar to the one you used in the previous section to define the O/R address that Digital DEC/EDI serves. For example, suppose that your Trading Partner is in the United States, with the following address details:

Country Code	us
Administration Domain	usadmd
Private Domain	acme
Organization Domain	sales

In this case, you would need to add the following entry to the X.500 directory:

```
ncl> set mts "/mts=myco-rd" oraddress -
_ncl> "c=us; a=usadmd; p=acme; o=sales" -
_ncl> routing instruction [action=transfer to domain, -
_ncl> server domain="acme-rd"]
```

The following table shows you the fields which make up the O/R address, and indicates the names used to refer to each piece both as ncl O/R address qualifiers and as seen on the Pedi connection details screens within the CommandCenter Communications Editor. The values you enter into Digital DEC/EDI and the values you enter into X.500 must match exactly, including the case of the letters.

Table 2-2 Pedi Communications: Connection Parameters

Ncl O/R Address Qualifier	Digital DEC/EDI Connection Parameter
C	Country Code
A	Administration Management Domain
P	Private Management Domain
O	Organization
OU1	Unit Name 1
OU2	Unit Name 2
OU3	Unit Name 3
OU4	Unit Name 4
CN	Common Name
G	Given Name
I	Initials
S	Surname
Q	Generation
X.121	X.121 Address
T-ID	Terminal ID
N-ID	Unique UAID
DDA	Domain Defined Attribute

Using a Remote MAILbus MTA

The Digital DEC/EDI Pedi gateway uses the XAPI interface provided by MAILbus 400. This means it is possible to run the MTA on a different node to that running the Digital DEC/EDI Pedi gateway. All of the configuration activities described in the preceding sections of this chapter still apply, although they need to be performed on the node where the MTA is installed.

If you have elected to run the MTA on a different node to the Digital DEC/EDI Pedi gateway, you must still install the MAILbus 400 MTA Base subset on the Digital DEC/EDI node and perform some limited configuration of it.

Specifically, you need to configure the components of the MTA Base subset to specify the location of the MTA, and the transport to use to reach the MTA.

The following is an extract from the MAILbus 400 user documentation:

> In the file `/var/mta/mta_api_server_address`, the string `LOCAL` should be changed to either of the following, depending on the transport type on the system where the Agent [Digital DEC/EDI] is installed:
>
> - For connections that use OSI transport (TP4), change the first line of `mta_api_server_address` to:
>
>  ```
>  OSITP mta_nsap_address
>  ```
>
> where `mta_nsap_address` is the NSAP address of the node where the MTA that serves the Agent is running. For example:
>
>  ```
>  OSITP 49002aaa00040066aa21
>  ```
>
> - For connections that use TCP/IP, change the first line of `mta_api_server_address` to:
>
>  ```
>  TCP mta_node_name
>  ```
>
> where `mta_node_name` is the name of the node where the MTA that serves the Agent is running. For example:
>
>  ```
>  TCP mta-node7
>  ```

For more details on performing this configuration, refer to the *MAILbus 400 MTA Planning and Setup Guide*.

Dealing With Pedi EDI Notifications

You can request an EDI Notification (EDIN) from a remote user agent for any outgoing Pedi message. These only apply to X.435 messages.

To request an EDIN from a remote user agent, you set the EDI Notification Request field on the appropriate connection details screen to Y.

Where you have requested receipt of an EDIN, an outbound transmission file remains at a state of **DELIVERED** until the EDIN is received. When the EDIN has been received, and assuming the EDIN is a positive notification, the state is updated to **PURGEABLE**. If the EDIN is a negative notification, the state is updated to **FAILED**.

If you have not requested receipt on an EDIN, the state is updated from **DELIVERED** to **PURGEABLE** automatically.

The User Agent creates and sends out an EDIN, if requested in the EDI message header of any messages that are received.

Setting Record Attributes

On the Details 2 tab of a Pedi connection, you will see a box containing the Record Attributes. This information controls how your EDI document (or non-EDI data if you used the Bypass interface) will look when placed in a P0 or P2 type bodypart. It has no effect on PEDI bodypart types.

The first option allows you to place the entire transmission file in the bodypart as a single line (also knows as stream format). To achieve this, set the ‘Record Length’ to zero (0).

The second option allows you to send fixed length lines in the P0 or P2 bodypart and to control which characters are used as the ‘end of line’. For example, if your trading partner wants you to send 80 character lines with a line terminator of CRLF (carriage return/line feed) then you would enter 80 as the record length, 13 (the ASCII code for a carriage return) as the first terminator and 11 (the ASCII code for a line feed) as the second terminator. If you looked at the resulting bodypart then you would see your transmission file listed in 80 character fixed length lines except for the last line that would be a maximum of 80 characters.

Note that if your trading partner only wants you to send one terminator then leave the second terminator as zero. If you leave both terminators as zero then DEC/EDI will send in stream format.

Also note that when your trading partner sends you a document using a P0 or P2 bodypart, DEC/EDI will use the values you enter here to determine what to strip out of the incoming message. Your trading partner must send to you in the same format that you send to them.

The final option allows you to send variable length lines with each line containing a separate EDI segment. If you set the Record Length to 32767 then DEC/EDI will look at the EDI files being sent and will add any terminators that you have defined to the end the segment. So, for example, if you are using the X12 standard and your trading partner wants you to send CRLF as the segment terminator so that they see each segment on its own line, set your segment terminator in the Trading Partner Agreement to CR, set the Record Length in your Pedi connection to 32767 and set the first terminator as 11 (the ASCII code for a line feed). Leave the second terminator as zero.

Configuring PEDI in a TruCluster Server

The PEDI Gateway will operate in parallel in a TruCluster Server. Each member of the cluster will send to the MTA and receive from the MTA simultaneously.

However, if you wish to split your workload so that only certain systems send files for a given connection then highlight the connection in the main window and choose Edit ->Add->node and enter the short node name (as shown by the ‘hostname –s’ command from the UNIX prompt) for the system. Only that system will then be allowed to send files using that connection. All nodes however will receive messages for all connections.

The default is to allow all systems to send and receive files for all connections. This is typically the most efficient setting.

Additional Traces and Logs

If you find problems while testing the Pedi gateway after you have configured it, you may find it useful to turn on one or more of the various logs and traces that can provide additional information to assist problem solving. Chapter 15 *Problem Solving – Pedi Gateway* provides information on how to do this.

Chapter 3 Configuring the OFTP Gateway

The Digital DEC/EDI OFTP gateway sends and receives messages by using the Odette File Transfer Protocol (OFTP). It does this by using the networking services provided by the X.25 software product.

This chapter describes how to set up Digital DEC/EDI, and the X.25 software, so that you can exchange data with your Trading Partners by using the Digital DEC/EDI OFTP gateway.

Before You Start

The procedures described in this chapter assume the following:

- You have configured the X.25 software by using the `wansetup` utility.
- The X.25 Configuration Test Procedure (CTP) runs successfully.

If you have not already completed the above activities, see the *X.25 Configuration Guide* for details on how to do so.

Setting up X.25 Filters

X.25 *filters* need to be created so that the X.25 software can match inbound X.25 calls to the Digital DEC/EDI OFTP gateway. These filters are created within the X.25 product.

Note: You need to read this section only if your system needs to handle incoming X.25 calls from your Trading Partners. If you decide not to set up any filters, it does NOT mean you can't receive any inbound data. However it does mean that to receive any inbound data, your gateway needs to initiate the call.

The X.25 software tries to match an inbound X.25 call to a filter. Once a call has been matched to a filter, the application associated with that filter is started. If the application is already running, the X.25 call is routed to it. The Digital DEC/EDI OFTP gateway is such an application.

As a configuration activity you need to use the CommandCenter Communications editor to enter the names of these filters into the OFTP gateway parameter screens.

When the OFTP gateway starts, it associates itself with the filter names specified in the OFTP parameter details. This enables the X.25 software to route to the OFTP gateway, all inbound X.25 calls that match the specified filters.

If no filters are specified, the OFTP gateway is unable to handle any inbound X.25 calls.

Call Parameters Used

X.25 filters can use any of the following X.25 call parameters when attempting to match to a filter :

- Incoming DTE Address
- Call Data Mask
- Call Data Value
- DTE Class
- Sending DTE Address
- Receiving DTE Address
- Group Name
- Originally Called Address
- Redirect Reason
- Called Address Extension Mask
- Called Address Extension Value
- Called NSAP

Creating Filters

Creating an X.25 filter can be done in one of two ways :

- By using the `wansetup` procedure.
- By creating the ncl commands manually.

These are described on the following pages.

Setting Up Filters By Using the “wansetup” Procedure

This section describes how to create an X.25 filter by using the `wansetup` procedure.

Step 1: Invoke the `wansetup` procedure by using the following command:

```
# /usr/sbin/wansetup advanced
```

Step 2: Select “Modify an existing configuration script” from the main menu.

Step 3: Move down the Sections menu to the Filters option.

Step 4: Select “Add a Filter” from the Filter Options menu.

Step 5: Fill in the required values into the input fields. Note down the Filter Name you have chosen. You need to enter this Filter Name into the OFTP gateway parameter screens by using CommandCenter Communications Editor.

Step 6: Repeat above process to create as many filters as required.

Step 7: Exit the Filter Options menu by selecting “Go to Sections Menu”.

Step 8: Select the “Create the ncl Script” option.

Step 9: Exit `wansetup`.

The next time you start X.25, the required filters are created.

Setting Up Filters By Creating the “ncl” Commands Manually

You can manually edit the ncl commands to create the required filters. You need to ensure that you save the ncl commands in a file that gets executed each time X.25 is started up. A natural choice is the file `/var/dna/scripts/x25_extra_create.ncl`. The commands in this file are executed each time X.25 is started.

The following is a sample of some ncl commands that create filters:

```
create node 0 x25 access filter "EDI Sending DTE Filter"
set node 0 x25 access filter "EDI Sending DTE Filter" -
  sending dte address 12345678
set node 0 x25 access filter "EDI Sending DTE Filter" -
  security filter Default

create node 0 x25 access filter "EDI Incoming DTE Filter"
set node 0 x25 access filter "EDI Incoming DTE Filter" -
  incoming dte address 87654321
set node 0 x25 access filter "EDI Incoming DTE Filter" -
  security filter Default

create node 0 x25 access filter "EDI User Data Filter"
set node 0 x25 access filter "EDI User Data Filter" -
  call data value '72000000'H, call data mask 'FFFFFFFF'H
set node 0 x25 access filter "EDI User Data Filter" -
  security filter Default
```

If the above three filters were created on a system to route inbound X.25 calls to the OFTP gateway, then you would need to define the following filter names for the OFTP gateway, by using the Communications editor:

- “EDI Sending DTE Filter”
- “EDI Incoming DTE Filter”
- “EDI User Data Filter”

For more information on creating X.25 filters, refer to the *X.25 Configuration Guide*.

Setting up X.25 Templates

When the OFTP gateway makes an outbound X.25 call, you can instruct it to use a predefined X.25 template. You need to enter the name of this template in the OFTP connection details.

To make a call, a number of call parameter values, such as *DTE class* and *DTE address* must be specified. To alleviate the need to specify common or frequently used call parameters each time a call is made, those parameters can be defined in a template. A template is therefore used to specify the facilities required when making an X.25 call.

Templates containing details associated with an outgoing call are created by the network manager and given template names.

The “Default” template is created during X.25 system configuration and is used if no template name is specified.

The values for *DTE class*, *DTE User Data*, *DTE Address* and *DTE Sub-Address* can be specified directly in the OFTP connection details. If these fields are sufficient for your needs, you do not need to use a template. However, if you need to include other call parameters, for example, *packet size*, when making an outbound X.25 call on this connection, you need to use a template.

Creating a template can be done in one of two ways :

- By using the `wansetup` procedure.
- By creating the ncl commands manually.

These are described in the following sections.

Setting Up Templates By Using the “wansetup” Procedure

This section describes how to create a template by using the `wansetup` procedure.

Step 1: Invoke the `wansetup` procedure by using the following command:

```
# /usr/sbin/wansetup advanced
```

Step 2: Select “Modify an existing configuration script” from the main menu.

Step 3: Select “Add a Template” from the Template Options menu.

Step 4: Fill in the required values into the input fields, such as ‘packet size’. Note down the Template Name you have chosen. You need to enter this name into the OFTP connection details for each OFTP connection that needs to use this template.

Step 5: Repeat above process to create as many templates as required.

Step 6: Exit the Template Options menu by selecting “Go to Sections menu”.

Step 7: Select the “Create the ncl Script” option.

Step 8: Exit `wansetup`.

The next time you start X.25, the required templates are created.

Setting Up Templates By Creating the “ncl” Commands Manually

You can edit the ncl commands manually to create the required templates. You need to ensure that you save the ncl commands in a file that gets executed each time X.25 is started up. A good file to choose for this would be the file: `/var/dna/scripts/x25_extra_create.ncl`. The commands in this file are executed each time X.25 is started.

The following are samples of some ncl commands that create a template:

```
create node 0 x25 access template EDI_PS_256
set node 0 x25 access template EDI_PS_256 -
                                packet size 256
set node 0 x25 access template EDI_PS_256 -
                               throughput class request [0..0]
set node 0 x25 access template EDI_PS_256 -
                                reverse charging false
set node 0 x25 access template EDI_PS_256 -
                                fast select not specified
set node 0 x25 access template EDI_PS_256 -
                                charging information false
set node 0 x25 access template EDI_PS_256 -
                                transit delay selection 0
set node 0 x25 access template EDI_PS_256 -
                                end-to-end delay [0..0]
set node 0 x25 access template EDI_PS_256 -
                                expedited data not specified
set node 0 x25 access template EDI_PS_256 -
                                nsap mapping false
```

The template `EDI_PS_256` can now be specified as the template name on all connections that require a packet size of 256.

For more information on creating templates, refer to the *X.25 Configuration Guide*.

Overriding Connection Details By Using Connection Specific Data

It is possible to over-ride certain connection parameters that you have created for OFTP connections. This is achieved by using *Connection Specific Data*. This data field comes either from the Trading Partner tables (as created by the CommandCenter Trading Partner Editor), or from the `trade` command used when you post data into the Digital DEC/EDI

system. When specified in the Trading Partner tables, the value can be entered either at the Trading Partner level, or at the Trading Partner document level. Whether you provide any value for Connection Specific Data from the Trading Partner tables, or from the `trade` command, varies depending on whether the values you provide are a fixed part of the configuration, or whether they apply to individual documents.

The OFTP gateway expects the string of data that you specify in Connection Specific Data to be in the following form:

```
oftp_id\vfn\sfid_user\orig_oftp_id\file_format\record_length
                                                    \sfid_resvThese
fields are described in the following table.
```

Table 3-1 OFTP Connection Specific Data Fields

Field Name	Field Description
`oftp_id`	Relates to the final OFTP destination ID for transmission files. By default, this is obtained from the Remote OFTP ID specified in the OFTP connection details.
`vfn`	Specifies the virtual file name assigned to the transmission files. By default, this is time-stamped in the format: ABBBBBBBCCCC.
`sfid_user`	Relates to the value of the SFID user field sent to the remote OFTP partner. By default, this is left blank.
`orig_oftp_id`	Can be used to overide the originator OFTP ID field in the SFID sent to the remote OFTP partner. By default, this is obtained from the Local OFTP ID field specified in the OFTP connection details.

Table 3-1 OFTP Connection Specific Data Fields

Field Name	Field Description
`file_format`	Relates to the value of the SFID file format field sent to the remote OFTP partner. Obtained from the File Format field by default, specified in the OFTP connection details. `F` - for Fixed record files `V` - for Variable record files `U` - for Unstructured record files `T` - for Text record files
`record_length`	Controls the maximum length of records transmitted to the OFTP partner. By default, this is obtained from the Record Length field specified in the OFTP connection details.
`sfid_resv`	Relates to the value of the SFID reserved field sent to the remote OFTP partner. By default, this is left blank.

If any field is not required, then it can be omitted. For example, if only the `record_length` needs to be specified, the field should be entered as:

```
\\\\\80
```

Configuring OFTP in a TruCluster Server

If your X.25 environment has been configured so that each system in the TruCluster Server has the same dte name referencing the path to your trading partner and either your trading partner can receive from multiple X.25 sub-addresses or the X.25 server masks your X.25 sub-address then is then sending to your trading partner using multiple systems in the TruCluster Server is possible as long as your trading partner allows multiple simultaneous connections from you.

Similarly, if your network configuration allows it, DEC/EDI will accept multiple inbound communications sessions from the same connection.

However, both of these cases take very careful planning by your network administrator.

You can use the multiple OFTP services available in a TruCluster Server by assigning specific connections to individual systems. Highlight the connection in the main window and choose Edit ->Add->node and enter the short node name (as shown by the ‘hostname –s’ command from the UNIX prompt) for the system. Only that system will then be allowed to send and receive files using that connection.

For details on how to configure your TruCluster Server to use X.25 services please refer to your Compaq representative or the Compaq support desk.

Additional Traces and Logs

If you find problems while testing the OFTP gateway after you have configured it, you may find it useful to turn on one or more of the various logs and traces that can provide additional information to assist problem solving. Chapter 16 *Problem Solving – OFTP Gateway* provides information on how to do this.

Chapter 4 Configuring the Import/Export Gateway

The Digital DEC/EDI Import/Export gateway sends and receives messages by using files located in specific directories.

This chapter describes how to set up Digital DEC/EDI to exchange data with your trading partners using the Digital DEC/EDI Import/Export gateway.

Choosing Directories

When Digital DEC/EDI is installed a default directory is created for importing or exporting files. This directory is: `/var/adm/decedi/impexp`.

If you do not wish to use this directory or want to create multiple directories so that specific security restrictions can be applied to them, then you must create these directories using the `mkdir` command. For example if you want Applications A and B to both be able to read exported files, but not to have either of them be able to see the others then you would create two export directories and associate these with two different Import/Export connections.

You can specify default import and export directories at the gateway level or override them on a per-connection basis.

Using Pre-Defined Jobs

The Import/Export gateway is a job-based gateway. The jobs being:

- IMPEXP - Imports and exports files from and to a directory
- IMPORT - Imports files from a directory.
- EXPORT - Exports files to a directory.

These jobs can either be started interactively via the Communications Editor, or via the `decedi_manage` command. By default, high priority outbound transmission files automatically run the EXPORT job.

Job scheduling for the Import/Export gateway is achieved by using the `decedi_manage` command in scripts that are run by the UNIX scheduler. For more details see Chapter 7 *Scheduling Jobs*.

Using Post-Export and Pre-Import Commands

You can optionally specify UNIX commands to be run after a file is exported or prior to importing a file. An example of this might be to ensure the file ownership is sufficient for the recipient to read it. So, a post-export script file might contain:

```
chown appa /usr/users/appa/export/*
```

It is also possible to pass certain parameters to the pre-import or post-export commands. These parameters are:

- #CONN — the connection_id.
- #IMPLOC — the import directory location.
- #EXPLOC — the export directory location.
- #EXPFIL — the name of the transmission file.

The #EXPFIL is only valid and useful for the post-export field. When a file is exported and the #EXPFIL flag is specified, then the command specified is run for each file exported. Otherwise the command is simply run when all files have been exported. So, the previous example could have been written as:

```
chown appa #EXPLOC/#EXPFIL
```

Finally, to show how useful this can be when first testing the Digital DEC/EDI Server where files are commonly exported and then looped back via the import command you could define an executable script file /usr/examples/decedi/server directory that contains:

```
# decedi_loopback
#
#    Script to loopback exported files to the import directory
#    and reimport them under a different name.
#
#    To use this script, On the Import/Export connection
     screen,
#    set the Post Export job to:
#
#    /usr/examples/decedi/server/decedi_loopback.sh #EXPFIL
     #EXPLOC #IMPLOC
#
#    Parameters:
#          $1 : Exported file name (excludes directory)
#          $2 : Export directory
#          $3 : Import directory

sourcefile=$2/$1
sourcefilename=`echo $1 | awk -v FS=. '{print $1}'`
targetfile=$3/I_${sourcefilename}.IMPORT
mv $sourcefile $targetfile
```

The post-export job for the connection could then specify:

```
/usr/examples/decedi/server/decedi_loopback #EXPFIL #EXPLOC
#IMPLOC #CONN
```

On export, this script would automatically move the exported file to the import directory for the connection, prefixing it with 'I_' to avoid duplicate names, and issue an IMPORT job for the connection.

Configuring Import/Export in a TruCluster Server

DEC/EDI will allow a connection to run on any system in a TruCluster Server configuration. However, only one system at a time may process a connection and DEC/EDI will prevent more than one system at a time trying to process a specific connection.

If you wish to direct the Import/Export Gateway to run a connection on a specific node then you can highlight the connection in the main window and choose Edit ->Add->node and enter the short node name (as shown by the ‘hostname –s’ command from the UNIX prompt) for the system. Only that system will then be allowed to send and receive files using that connection.

Chapter 5 Configuring the SMTP/ MIME Gateway

The Digital DEC/EDI Internet SMTP/MIME gateway sends and receives messages using Internet electronic mail (E-mail) as defined in RFC 1767.

Simple Mail Transfer Protocol (SMTP) is the basic mechanism for exchanging E-mail within the Internet community. SMTP is defined in RFC 821 and 822. It was initially designed for simple text messages. Only 7-bit ASCII characters were supported and text lines could not be longer than 1000 characters

Since its introduction, SMTP has been extended with a new specification called Multipurpose Internet Mail Extensions (MIME). MIME is specified in RFC 1521 and 1522.

About MIME

This new specification removes the restrictions on the original concepts by adding support for the following:

- Multi-part messages.
- Unlimited text line lengths.
- Non-ASCII character sets.
- Binary or application-specific files.
- Image, audio, video and multi-media files.

MIME Header Fields

The MIME specification currently defines five new header fields. They are:

- MIME-Version.
- Content-Type (+ sub-type and parameters).
- Content-Transfer-Encoding.
- Content-ID.
- Content-Description.

RFC 1767 describes how MIME is used to send and receive EDI data. It describes how the new header fields, defined by MIME, are used for EDI transmissions.

The following illustrates MIME and EDI with an example:

```
To: edi_user@widgets.co.uk
Subject: <<Subject Here>>
From: edigate@makers.co.uk
Date: <<date here>>
Mime-Version: 1.0
Content-Type: Application/EDIFACT
Content-Transfer-Encoding: QUOTED-PRINTABLE

<<standard EDIFACT Interchange here>>
```

The basic fields used for EDI are:

- Mime-Version.
- Content-Type.
- Content-Transfer-Encoding.

Version

Currently there is only one version of MIME so the *Mime-Version* field should be set to `1.0`.

Content Type

The *Content-Type* field should be set to one of the following:

- `Application/EDIFACT` — for an EDIFACT Interchange.
- `Application/EDI-X12` — for an X12 Interchange.
- `Application/EDI-consent` — for any other format, such as TRADACOMS.
- Multipart/Mixed supported for Inbound direction.

Content-Transfer-Encoding

The *Content-Transfer-Encoding* field describes the encoding used to place the EDI data into the SMTP message. Its possible values are:

- `BASE64` — used for binary files, lines no longer than 76 characters, human unreadable.
- `QUOTED-PRINTABLE` — used mostly for ASCII text, lines no longer than 76 characters, still human readable.
- `7-BIT` — no explicit encoding.
- `8-BIT` — no explicit encoding.
- `BINARY` — no explicit encoding.
- `X-<type>` — unregistered types; for example, X-UUE for UUencoded.

Before you Start

This chapter assumes that you have a running *sendmail* system. Digital UNIX is supplied with *sendmail*, a SMTP server/router. A mail server/router is a program that takes a mail message and decides where it should go and how to get it to its recipients.

It is assumed that you have configured your system to allow SMTP access from your system/domain to all your trading partner systems/domains. For more information on sendmail, refer to its on-line manual pages, that is, issue the following command:

```
# man sendmail
```

Creating the Mail Account

The gateway configuration requires you to specify an account (login) name to use for sending and receiving e-mail.

Choosing `decedi` as the account is allowable, but it does mean that the gateway mail commands have full read/write access to anything owned by `decedi`. A better way would be to use another independent account, say `edimail`.

When you are creating this new account, you should ensure that it belongs to the same group as the existing `decedi` account. For example, if the `decedi` account belongs to the primary group `edi`, then ensure that the account `edimail` also has `edi` as its primary group.

Securing Work Directories

The SMTP gateway uses two specific work directories.

The directory `/var/adm/decedi/smtp/` is a general purpose work directory and is used to hold temporary working files.

The directory `/var/adm/decedi/smtp_store/` is a temporary store for new inbound mail messages. When the SMTP gateway wakes up on its regular inbound poll timer, it fetches (using `binmail`) all new inbound mail messages and place them in this directory. It then processes each message file in the directory. Files in this directory have the format:

```
ABBBBBBBCCCC_CONNID.MESSAGE
```

Later, the `CONNID` part, is replaced with the 'real' connection id when it is known, but up until this point it is hard coded as `CONNID`.

Security Setting

The security setting on both of these directories is very important. In this example, with the account being `edimail` and the group being `edi`, the directories should be secured as:

```
# chown edimail /var/adm/decedi/smtp/ \
                            /var/adm/decedi/smtp_store/
# chgrp edi /var/adm/decedi/smtp/ /var/adm/decedi/smtp_store/
# chmod g+rwx /var/adm/decedi/smtp/ \
                            /var/adm/decedi/smtp_store/
#
# ls -ld /var/adm/decedi/smtp/ /var/adm/decedi/smtp_store/

drwxrwxr-x 2 edimail edi 512 ... /var/adm/decedi/smtp/
drwxrwxr-x 2 edimail edi 512 ... /var/adm/decedi/smtp_store/
```

You should amend the above commands to reflect the account and group names you are using.

Substitution Tags

Substitution tags are reserved upper case words prefixed with a hash (#) character. They can be entered as text into most connection configuration fields. They are not supported gateway parameters configuration fields. The use of substitution tags is optional.

Substitution tags allow you to specify into a field, a non-static value. Typically, configurations would contain hard coded text values. Using substitution tags, makes it possible to alter the values of configuration fields at run time.

For example, if you specified in the `Subject:` field the value `EDI Interchange: #TRF/#ICN`, the computed run-time value used by the gateway would be something like:

```
EDI Interchange:A34BC78912A4_CONN1/15397526
```

The `#TRF` tag would be replaced with the transmission file name and the `#ICN` would be replaced with the Interchange Control Number.

Supported Tags

The following substitution tags are supported:

- #TRF — The transmission filename, for example ABBBBBBBCCCC_CONN1.
- #CONN — The connection id, e.g. CONN1.
- #ICN — Interchange Control Number.
- #AREF — Application reference.
- #ISI — Interchange Senders Id.
- #ISQ — Interchange Senders Qualifier.
- #IRI — Interchange Recipient Id.
- #IRQ — Interchange Recipient Qualifier.
- #CSD1 — 1st parameter of the Connection Specific Data.
- #CSD2 — 2nd parameter of the Connection Specific Data.
- #CSD3 — 3rd parameter of the Connection Specific Data.
- #CSD4 — 4th parameter of the Connection Specific Data.
- #CSD5 — 5th parameter of the Connection Specific Data.
- #CSD6 — 6th parameter of the Connection Specific Data.
- #CSD7 — 7th parameter of the Connection Specific Data.
- #CSD8 — 8th parameter of the Connection Specific Data.
- #CSD9 — 9th parameter of the Connection Specific Data.
- #IF — the input file specification for external processing.
- #OF — the output file specification for external processing.

To find out which tags are supported for which fields, select HELP on the appropriate field, when editing connection details within the Communications Editor.

Restrictions

The following restrictions apply:

- The #IF and #OF are only valid in the external processing command fields (inbound and outbound commands).
- The #CSDn tags refer to values from the Connection Specific Data (CSD) field. The CSD value comes either from the Trading Partner tables (as created by the CommandCenter Trading Partner Editor), or from the `trade` command used when you post data into the Digital DEC/EDI system.

  When specified in the Trading Partner tables, the value can be entered either at the Trading Partner level, or at the Trading Partner document level.

- The CSD field contains a set of values separated by a \. For example if the Connection Specific Data value was `first\second\\fourth` then the CSD tags would have the following values:
  - `#CSD1 = "first"`
  - `#CSD2 = "second"`
  - `#CSD3 = ""`
  - `#CSD4 = "fourth"`
  - `#CSD5 = ""`
  - `#CSD6 = ""`
  - `#CSD7 = ""`
  - `#CSD8 = ""`
  - `#CSD9 = ""`
- The tags #ICN, #AREF, #ISI, #ISQ, #IRI and #IRQ are only valid for outbound non-BYPASS transmissions, that is, transmissions that have been through the Translation Services.

For any transmissions that do not meet these criteria, these tags will be equated to a null string.

Matching Inbound Message to a Connection ID

To be successfully processed, the gateway must be able to match an inbound message to a defined connection id. If there is no match the inbound message is failed.

The matching process is as follows:

- The `From:` address in the inbound message is examined, its value is used as a key into the connections table for a match on any record with a matching value in the `To:` field.
- If a match is found then the connection id has been identified.
- If no match is found, the `From:` value is then used as a key again into the connections table, but this time for a match on any record with a matching value in the `Alias` field.
- If a match is found then the connection id has been identified.
- If none of the above match then DEC/EDI will look for a connection id with a value of ALL in the Alias field. If one is found then this connection id will be used.

The `Alias` field in the connection details is used only for this purpose of matching inbound messages to connection id's. It is not used during construction or posting of outbound messages.

As a default, the `Alias` field can be given the same value as the `To:` field. Otherwise, it is simply an alias address for the remote trading partner.

To illustrate the matching process with an example, assume that the connection id being used is TEST and is the only one configured. It is configured with the following values:

```
From  : edimail@some.org
To    : Fred Bloggs <edi@widgets.org>
Alias : edi@widgets.org
```

Outbound

On outbound the `Alias` field is not used, but the `From:` and `To:` fields will be used as specified in the connection details. So, all outbound messages will have the following headers:

```
From  : edimail@some.org
To    : Fred Bloggs <edi@widgets.org>
```

Inbound

On inbound, suppose that a message has the following headers in it:

```
From  :  Fred Bloggs <edi@widgets.org>
To    : edimail@some.org
```

In this case the `From:` address in the inbound message is used to match to any connection record with a matching value in the `To:` field. In this case, it would successfully match to connection id TEST.

Another inbound message might have the following headers in it:

```
From  :  edi@widgets.org
To    : edimail@some.org
```

Notice that the message may have originated from the same place as the first (`edi@widgets.org`) but it simply does not contain the initial comment (`Fred Bloggs`). The first match to any connection record with a matching value in the `To:` field will fail. The second attempt is to match to any connection record with a matching value in the `Alias` field. In this case it would successfully match to connection id TEST.

Inbound messages that do not match to any connection id, are forwarded to the administrators e-mail address specified in the SMTP gateway parameters.

Security Processing

Security processing is activated on a per connection basis, using the Communications Editor. If security processing is set to None, the Inbound and Outbound command fields can be left empty.

When security processing is set to type External, you must specify operating system commands in both the Inbound and Outbound command fields.

Security processing of type External allows you to modify the complete contents of inbound and outbound messages. This feature gives you the ability to secure or modify transmissions in any way you choose.

The minimum processing required by the commands specified, is to create the output data file from the input data file.

The substitution tags #IF (Input File) and #OF (Output File) can be specified in the command fields. These tags refer to files that contain only the data portion of the message. Other files are created to hold the message and content headers, these are described later.

Specifying security processing of type External and using the following commands in the command fields would be equivalent to setting security processing to None.

```
Security Type: External
Outbound Command: cp #IF #OF
Inbound Command: cp #IF #OF
```

The external processing commands can do more than simply create the output data file. For example, they could also modify the contents of the message and content header files as well.

Sending Duplicate Messages

A word of caution if you intend to specify e-mail addresses in the `CC:` and `BCC:` fields in the connection details.

When using the `/usr/sbin/sendmail` executable to post messages, it is impossible for sendmail to notify the SMTP gateway which e-mail deliveries failed and which worked.

The only status returned from `/usr/sbin/sendmail` is whether the mail was successfully delivered to all recipients (`To:`, `Cc:` and `Bcc:`), or if the mail failed to be delivered to some or all of the recipients.

In the case where delivery to the `To:` address is successful but the delivery to the `Cc:` address failed, the SMTP gateway would assume the worst case scenario, that the message delivery failed to the primary recipient, in this case the transmission is assumed to have failed.

In general terms, each failure to send the message to a `To:`, `Cc:`, `Bcc:` recipient could actually be sending the message successfully to the other recipients.

To avoid this problem, do not specify e-mail addresses in the `Cc:` or `Bcc:` fields. Specify just a single address in the `To:` field.

If e-mail addresses do need to be specified in the `Cc:` and `Bcc:` fields, ensure that they are robust/high-availability addresses.

Using Background/Queued Delivery Modes

It is recommended that you avoid using the Background or Queued delivery modes in sendmail. The reason for this is that the SMTP gateway under these modes of delivery, cannot determine if the messages were successfully delivered or not.

Once a message has been successfully accepted by sendmail for Background or Queued delivery the SMTP gateway assumes that the delivery is successful and marks the transmission as **SENT**.

To avoid this problem, use only the Interactive delivery mode for sendmail.

If Background or Queued delivery modes must be used (e.g. to offload message delivery overhead from the SMTP gateway), ensure that the addresses used are robust/high-availability addresses.

Configuring SMTP/MIME in a TruCluster Server

DEC/EDI will allow a connection to run on any system in a TruCluster Server configuration and, for outbound files, files may be sent on the same connection in parallel. However, for inbound files, only one system at a time may process incoming mail. DEC/EDI will automatically prevent more than one system at a time trying to process incoming mail without any action on your part.

If you wish to direct the SMTP Gateway to run a connection on a specific node then you can highlight the connection in the main window and choose Edit ->Add->node and enter the short node name (as shown by the ‘hostname –s’ command from the UNIX prompt) for the system. Only that system will then be allowed to send files using that connection. However, the first free gateway in the cluster will attempt to read incoming mail.

Chapter 6 Configuring the 3780 Gateway

The 3780 gateway uses the CLEO 3780Plus Protocol Emulator to exchange files with a VAN using IBM's Binary Synchronous Communications Protocol (Bisync). This is a character-oriented protocol that uses special characters to delineate the various fields of a packet and to control the necessary protocol functions.

Before You Start

The procedures described in this chapter assume the following:

- Your account on the VAN has been set up. You should receive full instructions about how to set up your account when you register to use the VAN.
- The modem for communicating with the VAN has been set up. See your modem documentation for details of how to do this.
- The 3780Plus SYNCcable+ hardware has been installed.
- The 3780Plus software has been installed.

  Follow the installation procedures in the CLEO 3780Plus User Manual for OSF/1 (Digital UNIX).

  Note: if you have installed the X.25 WAN Device Drivers kit, it may replace the scc driver with its own version. This can change the characteristics of the ttynn ports to be synchronous. The ports need to be asynchronous.

  If you have installed V3.0 or earlier version of the WAN Device Drivers kit, then this may cause a problem.

Checking Port Characteristics

– To see the port's characteristics, execute the Unix event report command, 'uerf' and look at the last startup message:
uerf -R -r 300.

– See if `sscc0:` is assigned to the WAN Device Driver Interface.

– If it is, re-run `wansetup kernel` and de-configure the scc by not selecting scc.

Defining Jobs

When using the 3780Plus gateway, you exchange transmission files by running a pre-defined job. To allow you to take advantage of cheaper rates for connecting to VANs, the 3780Plus gateway sends transmission files based on a job schedule. You must define both the job and the schedule used for sending and receiving transmission files.

A 3780 gateway job specifies how to set up a connection with a VAN, including:

- Sending transmission files.
- Receiving transmission files.
- Requesting VAN reports (optional).
- Processing VAN reports (optional).

To each VAN, the 3780Plus gateway looks like a 2780 or 3780 terminal device. Each VAN has their own set of commands to exchange data between itself and the 3780 terminal session. Digital DEC/EDI uses the Tool Control Language (Tcl, pronounced *tickle)* to control the job session and act as a *glue* between Digital DEC/EDI, CLEO 3780Plus, and the VAN.

Tcl Scripts

The Tcl commands are supplied as script files; one or several can make up a job. A concatenated series of Tcl scripts is generally called a 'Tcl stick'(pronounced Tickling Stick).

Tcl Scripts

In general, scripts perform the following:

- Pre-processing commands. These are Tcl commands or UNIX shell script files that are executed before connecting to the VAN.
- Login commands for a specific connection to a specific VAN.
- VAN-specific commands that are executed during the connection to the VAN.
- Logout commands.
- Post-processing commands. These are Tcl commands or UNIX shell script files that are executed after closing the connection to the VAN.
- Error commands. These are Tcl commands or UNIX shell script files that are executed if an error occurs during the connection to the VAN.

When you set up the 3780Plus gateway you define jobs in terms of these scripts. You define a job for a specific task.

Any variables used by the scripts must have their values defined in the 3780 gateway parameter or connection details.

Since the gateway can have more than one connection sharing the same physical line to the modem, it is important *not* to define job schedules that overlap each other. Otherwise a job may not be executed.

Supplied TCL Scripts

Digital supplies two Tcl scripts for users to use as they are or as a template (copied into a new file name) when defining their own unique jobs. One is an example of a communication session with the GEIS VAN and the other with the TYMENET VAN. Each script file contains commands that set up a sequence of Login commands to log into the VAN, send any data that is "Awaiting Transmission" for that VAN connection, receive any data from the VAN mailbox, and log off. There are also options to request certain reports from the VAN.

The two supplied Tcl scripts are:

/var/adm/decedi/data/decedi_geis_sndrcv.tcl - for the GEIS VAN

/var/adm/decedi/data/decedi_mcdd_sndrcv.tcl - for the TYMENET VAN

Certain variables MUST be defined in the 3780 gateway parameter details (if applied to all 3780 connections) or in the specific 3780 connection details (if applied to an individual connection).

Two values are actually mandatory and must be defined in the gateway parameter details;

- CLEODIR var/adm/decedi/3780Plus

  - Working directory used for 3780Plus connections.

- CLEOLOG 3780.LOG

  - Log file into which connection info is written. This is then copied into a timestamped log file and placed in the CLEODIR directory when the job is finished. The original 3780.LOG is always written to the CLEOCODE directory.

The other variables are mandatory, but the values can be user-specified:

- CLEOCODE user-specified

  - The name of the directory where the 3780 emulator has been installed (or copied to).

- CLEO_VIDEO user-specified

  - The name of the CLEO supplied file that is used for obtaining all of the connection setup and status messages. Use the full path name for this variable.

There are some variables that must be specified in the connection details for each VAN type. For the GEIS VAN:

- TRANSPARENCY Y or N

  - Turns Transparency on or off.

- TRANSLIT_TABLE NONE or filename

  Uses a separate Translation table.

- PRINTFILENAME

  filename such as PRINT File into which received reports go.

For the TYMENET VAN:

- TRANSPARENCY Y or N

  Turns Transparency on or off.

- PRINTFILENAME filename such as PRINT

  - File into which received reports go.

Other Connection Parameters:

The Connection Details screen help contains information about the other 3780 emulator parameters required to correctly configure a connection.

Configuring 3780 in a TruCluster Server

DEC/EDI will allow a connection to run on any system in a TruCluster Server configuration. However, because each system may have its modems attached to different ports, you can highlight the connection in the main window and choose Edit ->Add->node and enter the short node name (as shown by the ‘hostname –s’ command from the UNIX prompt) for the system. You can then override the default device specification, modem type, and line type.

Note that only one system at a time can process a specific connection id.

Chapter 7 **Scheduling Jobs**

Once the communications job has been defined, it can be started in three different ways:

- by using the Communications Editor.
- by using `decedi_manage`.
- by using a schedule (`decedi_manage` with the UNIX `cron` utility).

This chapter describes the last of these options.

Scheduling a Job

To create a schedule for running a job, the UNIX utility `cron` and `decedi_manage` must be used. The `decedi_manage` parameter to be used is the `-sc` (start connection) parameter as in the following example:

```
# decedi_manage -sc GEIS TEST
```

The `-sc` parameter takes two values; the first is the Connection Id and the second is the Job Id. Please see the `decedi_manage` man page for further information about `decedi_manage`.

To schedule the execution of the `decedi_manage` command, the `cron` utility's `crontab` file must be set up. Digital DEC/EDI provides a template file called `/var/adm/decedi/crontab.template` to assist in setting up a job schedule. Information on how to use this file is contained in the template file itself.

Crontab

A special file (crontab) contains various commands which are automatically executed at specific times by the UNIX system. The date and time at which the specified commands are executed is user defined. The job schedules will need to be installed within the crontab file.

The scheduled commands within the crontab file are entered using the following format:

```
<minutes><hours><day-of-month><month><weekday><command-line>
```

Table 7-1 Crontab File Entry Format

Field	Description	Range
minutes	Minutes after the hour	0-59
hours	Hour of the day	0-23 (0 = Midnight)
day-of-month	Numeric day within the month	1-31
month	The month of the year	1-12
weekday	The day of the week	0-6 (0 = Sunday)
command-line	The UNIX command that is to be executed	Any valid UNIX command.

So, to have connection GEIS run job TEST every 15 minutes the cron entry would be:

```
0,15,30,45 * * * * decedi_manage -sc GEIS TEST
```

To run the same job but every half hour, between the hours of 8.00 am to 8.00 pm during Mondays-Fridays only would be:

```
0,30 8-20 * * 1-5 decedi_manage -sc GEIS TEST
```

For more information on how to use crontab, refer to the UNIX man command *# man crontab*

Your UNIX system may not have a crontab file created. In this situation you will be required to create one. To check to see whether your UNIX system has a crontab file installed enter the following command at the UNIX prompt:

```
# crontab -l
```

If there is no output then there is no existing crontab file. This means you must create a crontab file. If there is an existing crontab file then you will be required to append your entry to it.

Creating a New Crontab Schedule

Ensure that you do not currently have a crontab file installed. The previous section describes how to check this. If your system is currently using a crontab file and you continue with this section, then you will overwrite it. This can corrupt your UNIX system.

Digital DEC/EDI has supplied you with an empty crontab file. This file is located at `/var/adm/decedi/crontab.template`

The file contains information on setting up and using crontab. To create the new crontab file, copy the template file to your site specific one, for example:

```
# cp /var/adm/decedi/crontab.template /var/adm/decedi/crontab
```

Edit your site specific crontab file to add your schedules. Once schedules have been added for the first time, make the file available to the cron utility by issuing the following command:

```
# crontab /var/adm/decedi/crontab
```

The new crontab file (`/var/adm/decedi/crontab`) will now be used.

Modifying an Existing Crontab Schedule

First you need to make a local copy of the existing crontab schedule file. Enter the following UNIX command:

```
# crontab -l > /var/adm/decedi/crontab
```

This file will contain the existing scheduled commands. To schedule new commands, edit the file and add them using an available editor, for example:

```
# vi /var/adm/decedi/crontab
```

Edit your site specific crontab file to add your schedules. Once schedules have been added here for the first time, make this file available to the cron utility by issuing the following command:

```
# crontab /var/adm/decedi/crontab
```

The new scheduled commands will now be installed within the crontab.

Chapter 8 Setting Up Other Routing Options

Previous chapters of this guide have concentrated on the configuration of normally routed EDI data. That is, data that is routed outbound or inbound, via each of the three main Digital DEC/EDI Services in turn: Mapping Services, Translation Services and Communications Services.

This chapter describes the configuration necessary to support other forms of routing data through the system:

- Bypass routing.
- Application-to-Application routing.

You can use these routing options in conjunction with normal routing of other data. Individual documents and transmission files are routed according to a combination of parameters you specify to the Application Client `post` and `fetch` commands, and values you specify in the Server configuration.

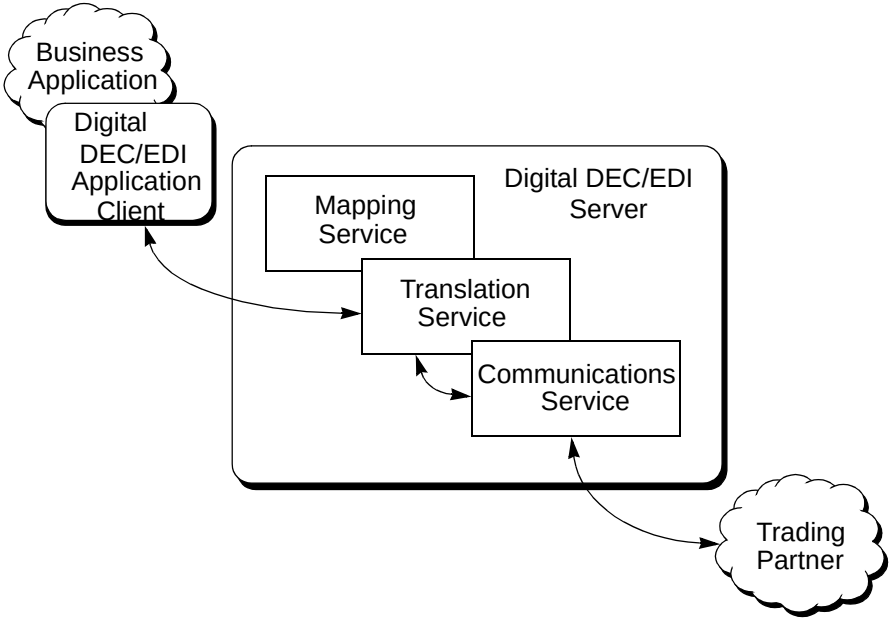
Setting Up Bypass Routing

Bypass routing is the general term used to refer to the routing of EDI data through the Server to bypass processing by one or more of the three services. Bypass routing can be applied to outbound and inbound data, and is achieved through a combination of configuration parameters and values you specify to the Application Client `post` or `fetch` commands. The two types of bypass routing are:

- Mapper Bypass Routing

  Data bypasses the Mapping Services, and uses the Translation and Communications Services. You would use this if the files sent and received by the application are already in Internal file format.

Figure 8-1 Mapper Bypass Routing



The main reason for doing this is to assist the migration of applications from earlier versions of Digital DEC/EDI, where applications often built Internal Format files themselves.

Note that with this form of bypass routing, the business application is not protected from any future changes to the EDI standards.

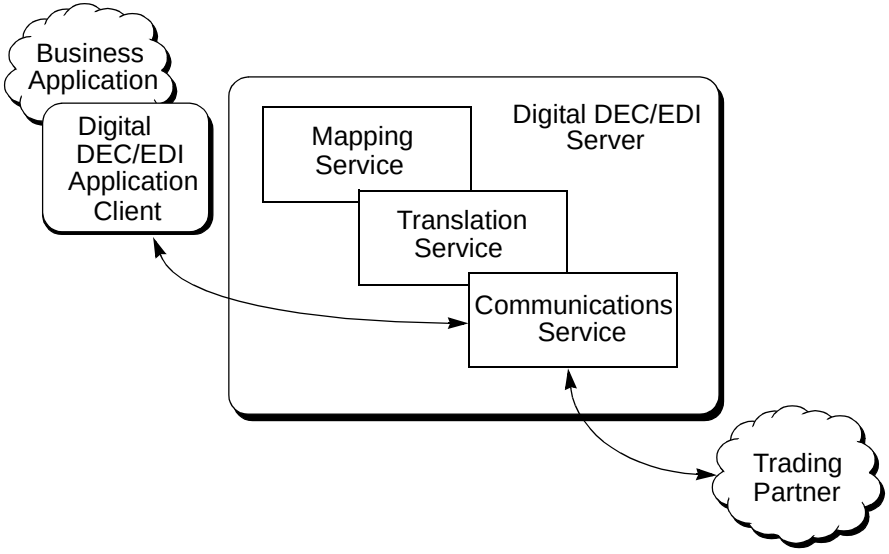
For example, if in future, different items of data are required by a particular version of an EDI document definition, the application will need to be modified to take account. With the normal routing of data through Digital DEC/EDI, the Mapping Services can often handle this, but with Mapper Bypass Routing, this is not possible.

- Translator Bypass Routing

  Data bypasses the Mapping and Translation Services, and uses the Communications Services. Here the Server is used simply as a gateway, so that applications can exchange files with trading partners.

  One reason for doing this is to send files of binary data that cannot be processed into EDI format. You would have to agree such transactions with your trading partner beforehand. Alternatively you might want to send files that are already in EDI format, perhaps because they have been imported from another EDI system.

Figure 8-2 Translator Bypass Routing



Specifying Outbound Bypass Routing

The routing of outbound data is determined from parameters you specify to the `post` command of the Application Client. As you provide the data you specify the type of data you wish to post and this implies the type of routing to be used. This is done by using the `-type` option as in the following table.

Table 8-1 Specifying Outbound Routing Options with the "-type" Option

-type=...	Routing Used
`application_file`	The file to be posted or fetched is an application file, to be normally routed via the Mapper.
`document`	The file to be posted or fetched is a single document in Internal File format. It is posted or fetched bypassing the Mapper.
`transmission_file`	The file to be posted or fetched is a transmission file. It is posted or fetched bypassing both the Mapper and Translation Services.

See the *Digital DEC/EDI: Application Development* book for more information on specifying parameters to the Application Client `post` command.

Specifying Inbound Bypass Routing

Inbound bypass routing is achieved by a mixture of the parameters you specify to the Application Client during the `fetch` operation and the parameters you configure for a particular *connection id*.

While the parameters you specify to the fetch command instruct the Server to provide you with information of the implied type (see the table in the previous section), the configuration parameters you need to specify ensure the Server routes the data correctly, so it is available to be fetched.

The remainder of this section describes the configuration parameters you need to specify. See the *Digital DEC/EDI: Application Development* book for more information on specifying parameters to the Application Client `fetch` command.

A transmission file received inbound may contain one or more of the following:

- A single EDI interchange.
- More than one EDI interchange.
- Non-EDI data (strictly the term “non-EDI” here is used to refer to the fact that the format of the data does not conform to any of the EDI syntax definitions that are supported by Digital DEC/EDI.

The processing required by each of these differs. Accordingly, Digital DEC/EDI provides several different options that may be configured. Two different configuration fields are used:

- The *Syntax* field specified for each connection, by using the Communications editor.
- The *Use Translator On Inbound* field specified for each agreement, by using the Trading Partner editor.

The following table describes how each different value you can select in the *Syntax* field dictates the routing applied to an inbound transmission file, and, where appropriate, how that routing is also affected by the *Use Translators On Inbound* flag.

`EDIFACT, X12, TRADACOMS`	These are appropriate to the normal routing of EDI data and assumes that the received transmission file contains only EDI interchanges conformant with a single EDI syntax. The Communications Services route the entire transmission file intact to the Translation Services.
`MULTIPLE`	Like the previous option, this option is used when the received data is expected to contain only EDI interchanges, conformant with one or more EDI syntaxes. However, where the received transmission file contains more than one interchange, each interchange is split into a separate new transmission file. Each new transmission file contains a single interchange, and each is individually routed to the Translation Services. This is the default option. Any parts of the original transmission file that contain non-EDI data are placed in one or more new transmission files and marked as **`FAILED`**. While the extra processing imposed by this option may in some cases affect performance, the advantage of this option over the previous option is that it allows the processing of the separate elements of the original transmission to be more closely monitored within the Translation Services, and that multiple EDI syntaxes, for example, both EDIFACT and X12, can be received using the same connection.

BYPASS — This option instructs the Communications Services to route the received transmission file intact, bypassing the Translation Services. This option would be used for receiving files of non-EDI data, or where the data being received was EDI data to be processed by using a third-party EDI translator package.

SPLIT — Like the *BYPASS* option, the *SPLIT* option assumes that all data is to be routed bypassing the Translation Services. However if this option is selected, the Communications Services separate the received transmission file into its separate constituent parts. As with the *MULTIPLE* option, each EDI interchange, and each piece of non-EDI data is placed in a separate new transmission file. These new transmission files are then individually made available to be fetched.

`MIXED` The *MIXED* option makes use of an additional element of configuration data. Each agreement defined within the Trading Partner tables, by using the Trading Partner editor, contains a flag to denote whether to *Use Translators On Inbound*.

As with the *SPLIT* option, the received transmission file is split into separate transmission files, each containing either a single EDI interchange or non-EDI data. The transmission files containing non-EDI data are routed bypassing the Translation Services.

For each transmission file containing a valid interchange of EDI data, the Trading Partner tables are searched to find an agreement whose details match the data in the transmission file. If there is a matching agreement, and if the corresponding *Use Translators On Inbound* flag is set, then that transmission file is routed to the Translation Services. Otherwise, if the flag is not set, or if there is no matching agreement, the transmission file is routed bypassing the Translation Services.

This routing option is useful in separating inbound data, some of which is to be processed by Digital DEC/EDI and some of which is to be processed by a second EDI system. Both pieces of data can be received in the same transmission file, and Digital DEC/EDI can separate the two pieces and route them accordingly.

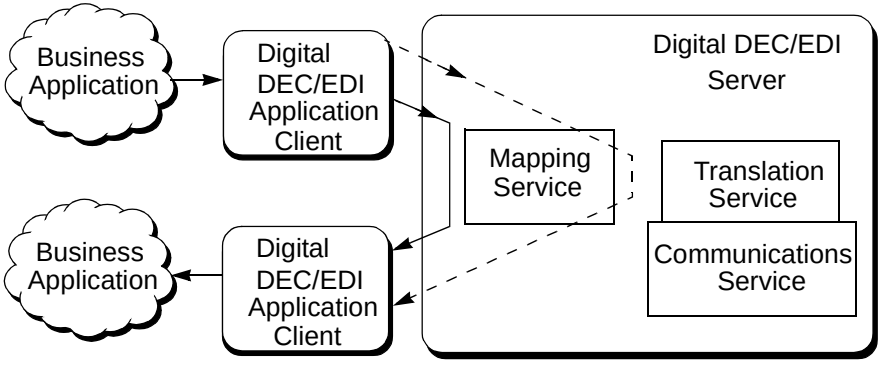
Setting Up Application-to-Application Routing

Application-to-Application routing is the term that applies to data routed from one Application Client to another, via the Server. This might be a useful option for routing EDI data between departments or organizations within the same company, where an intra-company network exists, and the need to use external communications services is avoided.

With Application-to-Application routing, the Server is involved in the transaction in three ways:

- It allows the Mapper to be used to process outbound and inbound data.
- It keeps an audit trail of each transaction, in a similar manner to that maintained for normally routed data.
- It maintains the necessary routing and registration information.

Figure 8-3 Application-to-Application Routing



As the preceding figure shows, the Translation and Communications Services are not used for data routed by using Application-to-Application routing.

Application-to-Application routing is configured by using the Management Services Editor.

When you use the Management Services editor to register a business application, you register the name of the application, and the node or nodes on which that application is to run. All applications need to be registered with the Server, regardless of the style of routing they use. Some additional configuration is necessary for applications that post data to other applications.

As an example, if you want an application app1 on node1 to route data to a second application app2 on node2, you need to configure the following, by using the editor:

Step 1: Register application details for app1 (by using the configure application: details tab).

Step 2: Register app1 as existing on node1 (by using the configure application: node tab).

Step 3: Register application details for app2 (by using the configure application: detail tab).

Step 4: Register app2 as existing on node2 (by using the configure application: node tab).

Step 5: For application app1, register app2 as a destination application (by using the configure application: application->application tab). At the same time register each type of document to be exchanged.

Note that you do not need to register anything for app2 in respect of app1.

When posting normally routed data through the Application Client, you can specify a destination trading partner. For data routed by using Application-to-Application routing, that same parameter is interpreted as the name of the destination application (app2 in this example).

Whether the Mapper is used to process data sent or fetched by using Application-to-Application routing depends on what values you specify to the Application Client, (specifically the `-type` option) when posting and fetching the data, and not on anything you need to configure. For instance, if you post by using `-type=application_file`, then the Mapper is involved. If you post by using `-type=document`, then the Mapper is not involved. In both cases, documents that match the application-to-application routing details are generated.

See the *Digital DEC/EDI: Application Development* book for more details on how to use the Application Client, `post` and `fetch` commands.

Chapter 9 Testing the Configuration

This chapter contains general guidance on testing a Digital DEC/EDI system. Every system is different and the specifics of testing each system differ. However most of the advice presented in this chapter applies to all systems.

Overview

This chapter attempts to break down the task of testing your system into individual steps, that can be tested and debugged in isolation. You do not have to perform any or all of the activities described in this chapter. You could decide to run the whole system and see what happens. Whether or not you do this depends on your familiarity with setting up Digital DEC/EDI systems and how confident you are of detecting and fixing any problems that may arise.

You should plan to perform your testing and development activities on a machine other than the one you intend to use for production EDI data. This helps to protect the processing of you production EDI data from any problems that may arise while testing a new configuration or using different data.

Using Logs and Traces

As part of testing a Digital DEC/EDI system, it is often helpful to enable one or more of the different logs or traces the system can provide. These often provide very useful information for use in a test scenario. However, the system performance can often be degraded if you make significant use of these features.

While this is less of a problem when you are commissioning a system, it may prove intolerable in a live production operation. For this reason, you are recommended to keep a note of the different logs and traces you enable, so that you can remember to disable them when they are no longer needed.

Testing Outbound Data

This section describes an approach to testing an outbound configuration that breaks the task into separate stages. You are recommended to work through these stages in the order specified as each stage builds on the previous one. As a more experienced user of Digital DEC/EDI, you may feel confident to try the complete configuration and, where problems arise, to fix them as appropriate.

As a new user, the step by step approach to testing advocated here allows you to check out small parts of the configuration individually.

Using Test Indicators to Test an Outbound Configuration

The `test_indicator` is a flag that can be used to control the processing of EDI data by the Digital DEC/EDI system. The test_indicator flag, as applied

to outbound documents, can take one of four different values, as shown in the following table:

Table 9-1 Test Indicator Values for Outbound Data

Test Indicator Value	Meaning
`mapper_test`	The mapper produces inhouse files, but these files are not passed to the Converter. This is intended to be used to test the function of Mapping Tables. Information about any generated document is not entered into the Digital DEC/EDI document audit trail, but the mapper audit trail can contain evidence of the generation of each of these documents.
`translation_test`	As with `mapper_test`, the mapper produces inhouse files, but this time the documents are entered into the Digital DEC/EDI document audit trail and the documents passed to the Converter for conversion into External Format files. This is intended to be used to test the EDI tables and trading partner tables. The Converter produces a **CONVERTED** document that is immediately set to **PURGEABLE**. The TFB does not attempt to build this document into a transmission file.
`partner_test`	As with the `translation_test`, the Converter converts the documents it receives, but this time the documetns are passed to the TFB, which build them into a transmission file. The communications gateway selected for the documents processes the result as an otherwise valid transmission. The documents are marked as having a TEST status.
`live`	A normal production status document.

The test_indicator applied to a particular document is defined or defaulted as a parameter to the post command, or overidden during the mapping phase.

The test_indicator applied to an outbound document is determined in the following precedence:

- The default is `live`.
- Any value specified as a Mapping Table Attribute overrides the default.
- Any value specified to the `post` command overrides any default value applied.
- Any explicit value applied during the mapping process overrides all other values.

Setting About Testing Outbound Data

The following steps form a logical progression in testing the correct processing of normally routed outbound data. This sequence assumes you have an example of an application file that you wish to send and that you have configured your Digital DEC/EDI system in accordance with the information contained in previous chapters, and the Application Development book:

Step 1: Verify the interface to your application and the Mapping Table. See *Verifying the Application Interface and Mapping Table* on page 9-5.

Step 2: Verify the configuration of the Translation Services. See *Verifying the Configuration of the Translation Services* on page 9-7.

Step 3: Export the transmission file using the Import/Export gateway. See *Exporting a Transmission File* on page 9-7.

Step 4: Send a test transmission file to your trading parter. See *Sending Test Data to Your Trading Partner* on page 9-8.

Step 5: Exchange production data with your trading partner. See *Exchanging Live Data with Your Trading Partner* on page 9-8.

Each of these steps relies on the successful completion of the previous step. The following descriptions reflect this. If you need to change any part of the Digital DEC/EDI Server configuration as a result of these tests, refer to *Modifying Configuration Information* on page 1-21 of Chapter 1, for information on how to make any changes effective.

Verifying the Application Interface and Mapping Table

This stage of testing assumes you have one or more example application files you can use to test with and that you have completed the development of the Mapping Table or Tables you intend to use. You should have successfully compiled the Mapping Table and copied it to the Mapping Table Repository on the Server. See *Developing and Registering Applications* on page 1-10 of Chapter 1.

Use the Application Client `post` function, with a `test_indicator` of `mapper_test`, to post an example application file into the system. You should use a command of the following form:

```
# trade post application_name \
  -test_indicator=mapper_test application_file_name \
  -type=application_file -table_name=table_name \
  -local_test=output_file_name
```

If successful, this command produces a file on the Server in the location specified by `output_file_name`. This file contains the data, in internal file format, for each document that would be produced by mapping the supplied application file.

No documents are entered into the Digital DEC/EDI audit trail, but you can use Cockpit to review the Mapper audit trail. If the mapping was successful, the auditing information may have been transferred to the archive database rather than the current database, before you have had a chance to view it.

If the mapping is not successful, you may see messages similar to the following displayed following the `post` command

```
# trade post .....
Failed to post file 1
Map Failed
```

Mapping Failed In this case, perform the following checks:

1. Check the parameters and options you specified for the `post` command.
2. Ensure that the specified application has been registered by using the Management Services editor. See *Developing and Registering Applications* on page 1-10.
3. Ensure that the specified Mapping Table has been compiled without errors, and that the compiled table has been copied to the Mapping Table Repository on the Server. See *Developing and Registering Applications* on page 1-10.
4. Use Cockpit to view the Mapper current audit trail to see if any information has been generated by the Mapper relating to the failures. If one or more HARD ERROR or SOFT ERROR records have been generated for the specified Mapper run, then examine the record details to view the error message test. If an error message has been generated, refer to the *Digital DEC/EDI Application Development* book for information on the error and what action to take.
5. If no errors have been logged in the Mapper Audit Trail, it may be necessary to obtain further debugging information by running the post command again using either or both of the `debug` or `error_log` options. For more information on using these options, refer to the *Digital DEC/EDI: Application Development* book.

Verifying the Configuration of the Translation Services

Using the same data as in the previous step, submit the application file, but use a `test_indicator` value of `translation_test`. This extends the previous test, to include outbound conversion to external file format. This tests the EDI Tables and Trading Partner configuration information. You should use a command similar to the following:

```
# trade post application_name \
  -test_indicator=translation_test application_file_name \
  -type=application_file -table_name=table_name
```

If this test is successful, the Mapper creates one or more documents from the application file. These are entered into the Digital DEC/EDI audit trail and set to a status of **QUEUED** for the Converter. The Converter attempts to convert each of these files. If successful they are set to **PURGEABLE** and then removed from the current audit trail to the archive audit trail by the Archive Server. If the test is not successful, the documents may be set to **FAILED**.

You should use the Cockpit to view the audit trail produced for each of the documents that are produced. See *Tracking Document and Transmission File Problems* on page 14-1 of Chapter 1, for more information on tracking and fixing these problems.

Exporting a Transmission File

Create a connection for the Import/Export gateway that allows you to export transmission files. Resubmit the application files from the previous tests, this time using a `test_indicator` of `partner_test`. Select the connection by ensuring it is the one configured for use by the trading partner configuration.

If this test is successful, one or more transmission files are produced by the TFB, and exported to the specified directory by the Import/Export gateway. See *Tracking Document and Transmission File Problems* on page 14-1 for information on tracking and fixing any problems that arise.

This test verifies that you can produce outbound transmission files marked as test data. You are now ready to send this test data to your trading partner.

Sending Test Data to Your Trading Partner

This test uses the same data as the previous test, but uses the communications gateway to send data to an external trading partner. Assuming the gateway and connection are defined and ready to be tested, you need to ensure the trading partner configuration data refers to this connection, rather than the export connection used in the previous test.

The successful outcome of this test is that transmission files are produced and sent to your trading partner. The transmission files are set to **PURGEABLE** and removed to the archive audit trail by the Archive Server.

You need to contact your trading partner directly to confirm that the data was correctly processed on the trading partner's EDI system.

Specific advice on problem solving with the different communications gateways is contained in Part III of this book.

Exchanging Live Data with Your Trading Partner

When you and your trading partner are happy that you have completed sufficient testing of your EDI systems by the exchange of test transmissions, you can proceed to exchange live EDI data.

The default test indicator for the `post` command is `live`. You can either specify this value explicitly, or remove the specification of the `test_indicator` option completely.

Testing Inbound Data

The `test_indicator` values that can be applied to outbound data do not apply to inbound data in the same way. The following order of testing is recommended:

Step 1: Import an example transmission file into the system, and use translator bypass to fetch the transmission file. This will verify the basic operation of the Server and the Application Client connection.

Step 2: Route the transmission file through the Translation Services and use Mapper Bypass routing to fetch the resulting documents in internal file format. This will verify the configuration of the Translation Services components: EDI tables, data labels, trading partner profile information.

Step 3: Repeat the previous stage, but use a Mapping Table to fetch the inbound documents via the Mapper. This will verify the Mapping Table.

Step 4: Repeat the previous stage, but use the real communications gateway and ask you trading partner to send you representative test data. This will verify that the Digital DEC/EDI system configuration you verified in stage 3 can handle actual EDI data supplied by your trading partner.

Step 5: Once you and your trading partner are happy with the exchange of test EDI data, you can exchange production EDI data.

The first stages in this sequence imply that you have a representative transmission file. You can either create one by using the outbound Digital DEC/EDI system, or you can request one from your trading partner.

Part II Maintaining

This part of the Digital DEC/EDI User’s Guide describes the activities you need to perform on a running Digital DEC/EDI Server.

Chapter 10 **Monitoring**

What is Monitoring ?

This chapter describes activities you should expect to perform regularly to monitor the flow of EDI data through the Server, and to check for any errors that occur. You should expect to check for the following:

- The general flow of EDI data through the Server. (See *Using Cockpit to Monitor Data Flow* on page 10-3).
- Documents or transmission files that fail within the Server. (See *Checking for Failed Data* on page 10-6).
- Documents or transmission files that have become ‘stuck’ within the Server. (See *Checking for Stuck Data* on page 10-7).
- Errors that arise within the Communications Services. (See *Checking for Communications Services Errors* on page 10-10)
- Digital DEC/EDI system level errors that arise. (See *Checking for Digital DEC/EDI System Errors* on page 10-11)
- Available disk space. (See *Checking for Available Disk Space* on page 10-15).

Most of these activities are performed by using Cockpit, although some are performed directly on the Server. The descriptions assume you have access to a working Cockpit installation and that you can use that product while you are reading this book.

Deciding How Much To Do

The frequency with which you need to perform the different monitoring activities depend on:

- The amount of data being processed by the system.

  The more data flowing through the system, the more quickly simple problems can escalate to become major ones. For example, a problem that derives from a lack of disk space, can be relatively easily handled where transactions rates are low, but with very high transaction rates, such a problem can quickly become difficult to manage.

- The importance of ensuring certain data is processed within time limits.

  It may be important to you to ensure that particular data is processed within a certain time. For example, the batch of data you received during the afternoon must be completed before 6pm that evening.

- The maturity and stability of your implementation.

  A stable system is more likely to remain operating correctly than one on which changes have been made. For example, if you have recently configured the Server to support several new trading partners, it is possible you may have inadvertently modified some of the existing configuration.

- Your experience in operating the system.

  If you know that, for example, batches of documents arrive first thing in the morning and first thing in the afternoon, then these may be good times to monitor the system more closely, rather than when the system is less heavily used.

You may discover that you need to monitor the performance of your system, hourly or more or less frequently depending on the above factors. Your experience of the system and the frequency with which errors occur is the best guide.

Using Cockpit to Monitor Data Flow

This section describes how you use Cockpit to obtain a summary of the state of EDI data flowing through the Server, and then how you obtain more detailed information about individual items of data.

Viewing Data By Using the Summary Screen

The Cockpit *summary* screen allows you to see, at a glance, the overall status of data flowing through a Server. You can use the summary screen to view the contents of either the Current Audit Trail, or the Archive Audit Trail. The Current Audit Trail contains information about documents and transmission files that still have work to be done on them, for example, they are waiting to be fetched. Once all work has been completed on them, they are moved to the Archive Audit Trail. Normally you will be more concerned with the work outstanding in your system, and therefore monitoring the Current Audit Trail.

For the Current Audit Trail, the summary screen provides a graphical display of the total count of documents and transmission files that are either:

- **FAILED**

  Documents or transmission files that have failed a stage of processing within the Server and need manual intervention before any further processing can take place.

- **ACTIVE/OTHER**

  Documents or transmission files currently being processed, or queued within the Server for processing.

- **PURGEABLE/CANCELLED**

  Documents or transmission files that have completed their processing within the Server, and are awaiting archiving.

This display can provide a single snapshot of data within a Server, or you can configure it to provide automatic updates of the data, for example, every 5 minutes.

You can configure the frequency with which the summary screen is updated with new information from the Server. However, the more frequently you ask for these updates to be made, the more the Server is interrupted to provide this information. If you, and other users of Cockpit on different PCs, are requesting very frequent updates, you can start to reduce the performance of the Server.

By using the summary screen to monitor data you should expect to find the following. On a lightly used Server you should expect to find:

- No **FAILED** documents or transmission files.
- A small (compared with the total daily work load) number of **ACTIVE** documents and transmission files.
- No **PURGEABLE** documents or transmission files, unless you are using X12/TDCC Functional Acknowledgements or EDIFACT CONTRL messages where some such documents may still be waiting to receive acknowledgements.

On a more heavily used Server, you should expect to find:

- No **FAILED** documents or transmission files.
- A correspondingly larger number of **ACTIVE/OTHER** documents and transmission files (than on a lightly used system).
- A small number of documents or transmission files, marked as **PURGEABLE** or **CANCELLED**, that are awaiting processing by the archiver. If you are using X12/TDCC or EDIFACT/ODETTE, this number may be higher where such documents may still be waiting for corresponding functional acknowledgements to arrive.

Provided the summary screens shows numbers of documents and transmission files that correspond to the above guidelines, and these match with your expectations and experience of the work being processed by the Server, the above describes the extent of monitoring data flowing through a Server.

Accessing More Detailed Information

The categorisation of data in the Server into one of three main states: **FAILED**, **ACTIVE/OTHER**, **PURGEABLE/CANCELLED** provides a high level picture of the overall state of a Server. However, you may wish to get more detailed information about the individual processing states, in particular those that are summarised as **FAILED**.

The options for the summary screen allow you to enable more detailed displays of documents and transmission files at each of the different outbound and inbound processing stages. A further option allows information on functional acknowledgements to be displayed.

Selecting either, or both, of these directions, adds further fields to the summary screen display. These extra fields show all the individual states that together are summarised as Active and the number of documents and transmission files at each of those states. This extra display allows you to see if, for example, the work being processed by the Server is evenly spread, or whether it is bottlenecked at particular states.

Appendix B *Document and Transmission File Status Flows* contains a full description of the states through which documents and transmission files flow, and which components of the Digital DEC/EDI Server effect the transitions between the different states.

Checking for Failed Data

The total counts of documents and transmission files are displayed on the Cockpit summary screen. From this you can quickly check if there are any failures.

The display also shows two coloured indicators, one for failed documents, and the other for failed transmission files. These indicators change colour to further emphasise the presence of **FAILED** data in the system. The colour coding of these indicators is described in the following table.

Table 10-1 Cockpit Summary Screen - Coloured Indicators

Colour	Meaning
Green	There are no **FAILED** documents or transmission files in the Server.
Yellow	There are some **FAILED** documents or transmission files in the Server, but the number has not increased since the last update to the summary screen.
Red	There are more **FAILED** documents or transmission files in the Server since the last update to the summary screen.

See *Determining the Point of Failure* on page 14-2 for information on how to further diagnose and fix any **FAILED** documents or transmission files that are detected.

Checking for Stuck Data

Getting a Summary of Stuck Data

Stuck data is defined as any document or transmission file that has been within the Server for longer than you would expect. In practice you have to exercise an element of judgement to determine how this applies to your system. Data may become stuck for a number of reasons. The most common of which are:

- Inbound documents are at a state of **AVAILABLE**, but the application that should fetch them has not been run.
- Outbound documents are at a state of **QUEUED**, waiting for the Transmission File Builder to build them into a transmission file.
- Outbound transmission files are stuck at a state of **AWAIT TRANSMISSION** within the Communications Services until a gateway or connection becomes enabled.

How long data can expect to spend in the system depends on the way in which you have chosen to operate your Digital DEC/EDI system, and the loading you put on the system (that is, the degree to which you run the system close to its maximum throughput capability). You might, for example, regard data that has been in the Server for longer than 24 hours to be “stuck”. Also the type of state the data is in may affect what decisions you make. For example, a **PURGEABLE** document stuck in the Current Audit Trail waiting for a returned functional acknowledgement for a few days, may not need investigating, whereas a document stuck at **CONVERTING** should be investigated much sooner.

You can use the Cockpit summary screen to find the total of those documents or transmission files that have been in the Server for longer than expected. When you first request the summary screen to be displayed, you are given the opportunity to enter parameters that are used to filter the data before it is displayed. These parameters allow you to display data that entered the system before or after dates and times you specify.

You can check for stuck data, by requesting to have displayed only data that entered the system before a specified date and time, for example, by specifying yesterday's date.

The information you then see consists of only those documents and transmission files that entered the system before that date and time.

If there are any documents or transmission files listed as being in the Server for longer than you expect, you should investigate the cause and fix as appropriate.

Getting More Detailed Information About Stuck Data

If the summary screen shows there is stuck data in the system, you may need to access more detailed information for individual documents and transmission files to see why data is stuck, and therefore what further action to take.

The summary screen contains a count of the individual totals of documents and transmission files. The counts are displayed, both in figures, and graphically, in bar chart form.

To get further information about data in the Server, you can double click on the bar that corresponds to the data you want to see. For example, if the summary screen shows you have 10 documents that have been at a state of **AVAILABLE** for over 2 days, you double click on the bar that shows this.

A dialogue box appears that invites you to define how you want the detailed information categorised. For documents, as in this example, you can choose to categorise the information by parameters such as *application ID*, *partner ID* and *document type*. For transmission files you have similar options.

For our example of 10 documents stuck at a state of **AVAILABLE**, you might want to see which applications they were waiting for. You would do this by selecting to categorise the information by application ID, that is, by selecting that option from those offered in the dialogue.

The window that appears is a breakdown of those original 10 documents categorised by application ID. That is, you can now see which applications have documents that are available to be fetched, and that are, in this case, more than 2 days old. Double clicking one of the selected applications produces a window, listing each of the documents waiting to be fetched.

You can continue to get further information on these documents by using the other features of Cockpit. The different reasons why individual documents and transmission files get stuck varies and the amount of investigation may vary correspondingly. In the above example, you may have determined that all the unfetched documents that were older than 2 days were destined for the same application and the reason they were stuck was because that application had not been run recently.

What To Do Next

If you have data that is stuck in the Server, you can either wait to see if the data becomes unstuck, or you can investigate further. For example, if you can see documents at a state of **AVAILABLE**, and you know the corresponding application has not run for several days, but will run shortly, then there is nothing further you can do.

To investigate further, refer to *Stuck Documents and Transmission Files* on page 14-12 of Chapter 14.

Checking for Communications Services Errors

When the number of consecutive errors for a particular connection exceeds a specified limit, a connection becomes disabled and cannot be used for sending or receiving any further data until you have re-enabled it.

By specifying a value for any of the `DECEDI_MAIL_<gateway>` environment variables, you receive mail whenever a connection is disabled. Appendix C *Environment Variables* provides more details on the use of this environment variable.

If you have not set these environment variables, you need to check periodically to see whether any connections have become disabled.

The easiest way to do this is to use the Communication Editor to display a list of all connections for your Server. This display shows the state and current error count for each connection. The state is displayed as either enabled or disabled.

For each connection that has become disabled, you need to reset the corresponding error count to zero and re-enable it. If you do not do this, transmission files that expect to use a disabled connection become stuck at **`AWAIT TRANSMISSION`**.

The use of error limits and retry counters as they appear in each of the different gateways within the Communications Services is more fully described in *Coping with Errors in Transmission* on page 1-14 of Chapter 1.

Checking for Digital DEC/EDI System Errors

Each of the different components of the Digital DEC/EDI Server is capable of logging information about events or errors that occur during operation. This information is recorded in the Digital DEC/EDI Error Log file. The information is recorded in a binary format. This means that you can't view it directly. The recommended interface to the Error Log is provided by Cockpit. In simple cases it may be easier to use the `decedi_look` function on the Server.

The Error Log file is `/var/adm/decedi/logs/decedi_errors.log`.

You should check the Error Log file regularly for any errors that may be logged. The following sections describe the format and content of information in the error log and how you use either Cockpit or the `decedi_look` tool to review its content.

You should review the error log at least daily.

Formatting of Data in the Error Log

Information describing events or states is logged into the Error Log file. The following information is logged in each case:

- The date and time the information was logged.
- The name of the process that logged the information.
- The process id (PID) of the process that logged the information.
- One or more messages that describe the event or condition being recorded.

Each of the individual message are recorded with a *severity* level. Four different severity levels are possible, as shown in the following table.

Table 10-2 Digital DEC/EDI Error Log - Message Severities

Severity Level	Meaning
Informational	The message contains information about an event, either in support of other more serious messages (for example, the name of a file that cannot be found), or as a matter of record (for example, a process that has started up normally).
Warning	The message contains information about an activity that has completed, but not necessarily with the expected outcome. You may need to review such messages, and in some cases take action.
Error	The message describes an event that did not complete successfully, and that has caused a particular task to be stopped. Subsequent tasks can proceed unaffected. You should expect to investigate all error messages, and take action to either recover the original problem, and perhaps further action to prevent such a problem recurring.
Fatal	Such a message describes a severe error that prevents a process from continuing its normal processing of data. After a fatal message has been logged, the overall operation of the system may be compromised, and a full shut down and restart may be needed.

Viewing the Error Log by Using Cockpit

When you use Cockpit to show you the Error Log on a Server, you are initially positioned at the end of the file. That is, the display shows those messages that have been logged most recently.

Cockpit allows you to jump to the top or bottom of the file and scroll through line by line or screen by screen. Also, you can search for particular messages, messages logged by particular components, and messages logged at particular times.

CommandCenter/Cockpit User Access Controls

User Access Controls allow a suitably privileged user to restrict individual user's access to the CommandCenter and Cockpit functions on a per Server basis. These restrictions are set up using the User Access Control Editor.

You can use these controls to restrict Cockpit users to only seeing data for 'their' applications. Certain users can be limited to only viewing documents and transmission files, while others may perform operations like resetting failed documents. You may also prevent users from using the CommandCenter editors.

For full details of the controls that can be applied see the on-line help for the User Access Control Editor.

Viewing the Error Log by Using “decedi_look”

You need to be logged in to the `root` account to use the `decedi_look` tool. Invoke the tool by using the following command:

```
# decedi_look
```

This prints the current contents of the Error Log file. Unless the file is short, you may need to redirect the output from `decedi_look` to a separate file and use a text editor of your choice to review its content:

```
# decedi_look > errors.log
```

Groups of messages output by `decedi_look` appear similar to the following example:

```
Mon May 22 13:11:00 1995 PID =  3736 NAME = Track Server
(CORBA)
DECEDI__PCCAPPL (i), application DOC
DECEDI__PCCREPCAF (i), client application Node
ediclt.edi.dec.com
DECEDI__NOTAUTHORIZED (e), user session not authorized
```

In this example, an unauthorized application called `DOC` on node `ediclt.edi.dec.com` attempted to issue a `track` command at the time and date indicated. The request was rejected.

Viewing Other Log Files

In the same directory as the Error Log file is another Operator Log file called `decedi_opcom.log`. This contains a subset of the more serious messages that are also logged in the Error Log. If the Error Log file is cluttered with many informational messages, it can be difficult to look for the more serious messages. Looking at the Operator Log may be easier in these circumstances.

You can avoid logging messages of lower severity by using the environment variable DECEDI_LOG_SEVERITY detailed in Appendix D.

You can use both Cockpit and `decedi_look` to review the Operator Log file. When you use Cockpit, you should specify the name of the Operator Log file to view. Similarly, while `decedi_look` reviews the content of the Error Log, by specifying the name of the Operator Log as a parameter you can review the contents of that file:

```
# decedi_look -f /var/adm/decedi/logs/decedi_opcom.log
```

When manipulating the database using the various database vendor-specific commands, decedi_config captures all input and output. In the event of a failure in the command, the user is told the particular database operation failed and the contents of the input and output are appended to the text file:

```
/var/adm/decedi/logs/db_errors.log
```

This file can be viewed using any editor, for example ‘vi’.

Checking for Available Disk Space

The Digital DEC/EDI Server uses and consumes disk space as it is running. The information in this section describes what you need to check to ensure the Server has enough disk space, and how you should perform the checks.

Why Digital DEC/EDI Requires Disk Space

It is very important you do not allow your Digital DEC/EDI Server to run out of disk space. When there is no remaining disk space the system can no longer operate.

In general, disk space is consumed in three ways:

- Due to the exchange of EDI documents. These are held on disk until the Secondary Archive takes them away. Until they are removed, this build up produces:
  - More files in the store directories. See *Checking for Store Directory Disk Space* on page 10-16.
  - More records in the Archive Audit Trail (in the database). See *Checking the Database* on page 10-19.
- You add new versions of EDI standards that cause the database to fill up. See See *Checking the Database* on page 10-19.
- The build up of information in log files. See *Checking Other Areas* on page 10-20.

The following sections describe, in detail, the checks you need to make.

Checking for Store Directory Disk Space

What is a Store Directory ?

Store directories are locations where Digital DEC/EDI saves copies of the files that contain the EDI data you send through the system. The store directories contain the document files (internal format files, external format files etc.) and transmission files. By default, when you install and configure the Server, you have a single store directory: `/var/adm/decedi/store_1`. You can have more store directories if you need to. Additional store directories must appear alongside `store_1` and must be numbered consecutively from 1 onwards. Thus the first additional store directory must be called `store_2`. Likewise for further store directories. When the Server is started it determines the number of store directories by searching for `store_1`, then `store_2` until it fails to find a directory. If it finds, for example, only `store_1`, `store_2` and `store_3` but no `store_4` then it concludes there are three store directories (regardless of whether or not other `store_*` directories exist).

When creating new files for documents and transmission files, the Server processes access the store directories randomly. This is done to balance the I/O load between store directories. However all files for a document are in the same directory. All files for a transmission file are also in the same directory.

While the store directories must all appear in `/var/adm/decedi/store_*`, the individual directories may be linked to directories that are on other physical disks. Thus, although the directories appear to Digital DEC/EDI as existing in one place, you can actually spread them over different physical disks. This is important is helping to spread the I/O load over more than one disk and thus in helping to optimise the performance of the Server.

Checking Disk Space

The store directory space can be checked by using the UNIX `df` command, as follows:

```
# df /var/adm/decedi/store_*

Filesystem    512-blocks  Used   Avail Capacity Mnted on
/dev/re0g       3160756 1575154 1269526  55%    /usr
/usr/users/edi1 2912542 2104208  517078  80%    /usr/users/edi
```

Since the store directory used for a new document or transmission file is selected on a random basis, the disk on which least space is available for store directories is the most critical. In the above example, this is the disk `/usr/users/edi`.

The amount of space needed by the directory depends on the number of current and archive documents. Use the following formula to estimate the amount of disk space (in bytes) required for each store directory:

$$space = \frac{6 \times docsize \times (maxcurrentdocs + maxarchivedocs)}{numstoredirs}$$

where:

- *docsize* is the average size of a document in bytes
- *maxcurrentdocs* is the maximum number of documents in the current system
- *maxarchivedocs* is the maximum number of documents in the archive system
- *numstoredirs* is the number of store directories

This formula makes a number of assumptions. In particular it assumes that the envelope overhead is small, compared with data content, and that the data in the internal format file is on average a third of the size of the corresponding data label.

The above formula should only be used as a guide to disk space requirements. Switching on ‘detailed listings’, and having lots of documents that contain errors may cause more disk space to be used.

Increasing the Amount of Available Disk Space

The two easiest ways to increase the amount of disk space available to the Server are:

- To perform a Secondary Archive to remove completed transactions that you no longer need to keep on the system. See *Removing Old EDI Data By Using Secondary Archive* on page 11-2 of Chapter 11.
- Creating additional store directories (or relocating existing ones) on additional disks. See *Creating and Moving Store Directories* on page 11-8 of Chapter 11.

Checking the Database

Checking Database Free Space

Refer to Oracle Documentation on how to determine the free space.

Increasing Space for an Oracle Database

The two ways to increase the amount of free space available to the database are:

- To perform a Secondary Archive to remove completed transactions that you no longer need to keep on the system. See *Removing old EDI Data by Using Secondary Archive* in Chapter 16.
- To increase the size of the datafiles that comprise the database. See *Changing the Size of Your Database* in Chapter 16 for instructions on how to do this.

For more information on calculating how big your database needs to be for a specified workload, refer to *Deciding Where to Locate the Database* in Chapter 3.

Checking Other Areas

Other areas that need periodic monitoring with respect to disk space are:

- `/var/adm/decedi/arch_reports` - this contains Secondary Archive reports.
- `/var/adm/decedi/logs` - this contains the Error Log file and other system log files.

If the disk that contain these become full then either ‘link’ them to a different file system or remove some of the older logs to another place outside of the scope of the Digital DEC/EDI system.

The following directories should also be checked. These contain only temporary files and should not normally consume much disk space. While the Digital DEC/EDI Server is not running, they may be cleaned out:

- `/var/adm/decedi/backup` - this is used as a working directory by Secondary Archive and Retrieve.
- `/var/adm/decedi/temp` - this is used by the Server as a general temporary working directory.
- `/tmp` - this is used by the system as a temporary working directory.

Chapter 11 **Maintaining the Server**

This chapter describes activities you should expect to perform periodically or occasionally, to ensure your Server remains working effectively.

Overview

Periodic activities are those that you should plan to perform according to a regular schedule that you need to determine. *Occasional* activities are those you may need to perform in response to external events.

In addition to the monitoring activities described in the previous chapter, the activities you should perform periodically are:

- Removing old EDI data by using Secondary Archiving. (See *Removing Old EDI Data By Using Secondary Archive* on page 11-2).
- Starting a new Error Log file. (See *Starting a New Error Log File* on page 11-6).
- Stopping and starting the Server. (See *Stopping and Starting the Server* on page 11-7).

Activities you may need to perform occasionally are:

- Changing the configuration, for example, adding new trading partners. (See *Changing the Configuration* on page 11-7).
- Creating additional store directories or moving existing ones. (See *Creating and Moving Store Directories* on page 11-8).
- Changing the size of the database. (See *Changing the Size of Your Database* on page 11-10).
- Retrieving EDI data from a Secondary Archive. (See *Retrieving EDI Data From a Secondary Archive* on page 11-11).
- Resending EDI data. (See *Resending EDI Data* on page 11-11).
- Repairing stuck EDI data. (See *Repairing EDI Data* on page 11-13).

Performing Periodic Maintenance Activities

Periodic maintenance activities are those for which you should establish a regular timetable. Each of the following sections contain suggestions for how often you should carry out the activities described.

Removing Old EDI Data By Using Secondary Archive

A more complete reference description of the facilities available for Secondary Archiving is described in Chapter 13 *Secondary Archive and Retrieve*. Information is also available on line in the man pages entry for *decedi_arch(8)*.

Deciding How Often to Run Secondary Archiving

Primary Archiving is the automatic process by which finished documents and transmission files are moved from the Current Audit Trail to the Archive Audit Trail.

Secondary Archiving is the process of removing EDI data and audit trail information from the Archive Audit Trail to an off-line area for longer term storage. You need to consider the following when deciding how often to perform Secondary Archiving. Factors in favour of performing more frequent Secondary Archiving are:

- You minimise the amount of disk space you need.
- You keep the archive audit trail as small as possible, and thus help preserve server performance (although trying to run secondary archive constantly would be counter productive, by causing too many conflicts with the archiver).

Factors in favour of less frequent Secondary Archiving are:

- You can have more immediate access to the data, should you need to examine it, or resend it.
- You avoid generating many different archives that need storing and cataloguing.

Your experience running the Server with your particular pattern of work and your general business requirements should be a guide in how you approach this. As general recommendations:

- Aim to perform more frequent Secondary Archiving rather than less.
- Perform Secondary Archiving at least weekly.
- Schedule Secondary Archiving to run at a time when your Server is least busy, for example, this might be at night or over a weekend.

Performing a Secondary Archive

To perform a Secondary Archive you must be logged into the `root` account of your Server. The following is an example of performing a Secondary Archive:

```
# decedi_arch -v -b 01031995 -o ./decedi_archive.tar -c it \
 -f "-l -i"

initializing environment at Fri May  5 06:22:13 1995
selecting documents to archive at Fri May  5 06:22:16 1995
total of 335 documents selected by Fri May  5 06:22:18 1995
selecting transmissions to archive at Fri May  5 06:22:18 1995
total of 67 transmissions selected by Fri May  5 06:22:19 1995
selecting maps to archive at Fri May  5 06:22:19 1995
total of 67 maps selected by Fri May  5 06:22:23 1995
calling Operating System archive shell script
/usr/sbin/decedi_arch_script at Fri May  5 06:22:23 1995
Operating System archive completed by Fri May  5 06:22:27 1995
deleting archived objects at Fri May  5 06:22:27 1995
deleted 469 database objects and 31 files so far
deleted 469 database objects and 531 files so far
deleted 469 database objects and 1031 files so far
deletion completed by Fri May  5 06:27:44 1995
```

In this example all objects created before the 1st March 1995 (as specified by `-b 01031995`) are archived. The archive steps are output in verbose mode (`-v`). As well as the default of archiving the audit trail contents, the in-house and transmission files associated with the audit entries are also placed in the archive (`-c it`).

The archive is placed in `./decedi_archive.tar` (as specified by `-o ./decedi_archive.tar`), and the archive command is passed flags (as specified by `-f "-l -i"`), to tell the tar command to ignore checksum errors, and warn if all links to files are not resolved.

The script used to perform the actual archive is either the default one supplied with Digital DEC/EDI, `/usr/sbin/decedi_arch_script`, or one that you have supplied as `/usr/sbin/decedi_arch_syscript`.

Deciding What Information to Archive

The `decedi_arch` command has several options that allow you to control what data is processed. Of particular interest are the flags: `-a`, `-b` and `-c`. The `-a` option allows you to specify a date for the Secondary Archive to process only data created after that date. The `-b` option allows you to specify a date for the Secondary Archive to process only data created before that date. You can use `-a` and `-b` together or individually. By using these two options together, you can pick which data is processed, on the basis of their creation dates. Conversely, you might use just the `-b` option to specify that Secondary Archive process all data older than 1 week (assuming today's date is 27th May 1995):

```
-b 20051995
```

The `-c` option allows you to specify which pieces of information are included in the archive. It is important to think carefully about any options you choose here. After a successful Secondary Archive run, all data selected for the archive is deleted from the system. If you have not requested particular information to be placed into the archive, you will have lost it. By default, all audit trail information is archived, with a copy of every transmission file. If you want to save more than this, for example, copies of external format files, you must specify this.

If you want to be able to retrieve and then resend individual outbound documents, you must save external format files. This is not the default option. If you want to be able to resend individual outbound transmission files, you must save transmission files.

If you plan to include more items that the default in the archive:

- The less chance there is of losing data you may need to refer to later.
- The resulting archive is larger.
- The Secondary Archive takes longer to complete.

Starting a New Error Log File

New information is appended to the end of an existing Error Log file. Thus the most recently logged information is located at the end of the file. The oldest information is at the start of the file. As more messages are logged, one or more of the following may become true:

- The file becomes too cumbersome and slow to view when you try to look for particular messages.
- The file consumes too much disk space.
- The reduced performance of writing information to the file may become significant to the overall system.

As a minimum you should start a new Error Log file each week. You may wish to save old Error Log files in case you need to review them at a later stage. You are recommended to keep at least one previous Error Log file.

Create a new Error Log by renaming the existing file, for example:

```
# mv /var/adm/decedi/logs/decedi_errors.log
decedi_errors.old_log
```

You should perform this operation from the `root` account, with the Server shut down. A new file is created automatically when the next message is written to it.

The environment variable `DECEDI_LOG_SEVERITY` can be used to restrict the logging of unwanted messages of informational and warning severity. Appendix C *Environment Variables* provides more information on this and other environment variables used by Digital DEC/EDI.

Stopping and Starting the Server

To perform several of the maintenance activities described in this chapter, it is necessary to stop and restart the Server. In addition, it is good practice to plan to stop the Server periodically, for example, once every week or two. This allows:

- The startup procedure to recover any documents or transmission files that have become stuck in the Server.
- Any changes to environment variables (See Appendix C *Environment Variables*) to become effective.
- Any resources used by the Server, and its dependent products, to be released.
- If you believe that the DEC/EDI Memory Queues are out of sequence with the documents listed in the database then you should shut down DEC/EDI on all systems in the TruCluster and delete the file /var/adm/decedi/memoryQueuesSaveFile.do_not_delete. When you run the startup procedure, DEC/EDI will automatically recover from the DEC/EDI database.

From the `root` account on your Server, use `decedi_stop` to stop the Server and `decedi_start` to start it.

Performing Occasional Maintenance Activities

Changing the Configuration

Once a Digital DEC/EDI Server is running any changes you make to its configuration data do not necessarily take effect immediately. Some changes are effective the next time the data is required, others require the Server to be restarted and others require further explicit action before new configuration information is used.

Not all changes are effective when you restart the Server. See Modifying Configuration Information on page 1-21*, for information on what changes are effective when.*

Creating and Moving Store Directories

The creation of additional store directories can be a very important means of increasing or maintaining the performance of a Digital DEC/EDI Server. For more information about store directories, see *What is a Store Directory ?* on page 10-16.

Creating Additional Store Directories

The following is an example of creating an additional store directory, called `store_2`, on another disk, and linking it to make it appear as `/var/adm/decedi/store_2`:

Step 1: Create the directory that is to be used.

```
# mkdir /usr/users/edi/store_2
```

Step 2: Make sure that the owner of the directory is Digital DEC/EDI.

```
# chown decedi /usr/users/edi/store_2
```

Step 3: Create a link to the directory from `/var/adm/decedi/store_N`.

```
# ln -fs /usr/users/edi/store_2 /var/adm/decedi/store_2
```

Step 4: Make sure that the owner of link is also Digital DEC/EDI.

```
# chown decedi /var/adm/decedi/store_2
```

Step 5: Check that the linked directory can be seen.

```
# ls -al /var/adm/decedi/store_2
```

When the Server is next started it sees `/var/adm/decedi/store_2` and determines that there is now an extra store directory to use.

When deciding where to place additional store directories, Digital DEC/EDI benefits from using fast disks, connected locally to the CPU. You are recommended not to use network mounted disks.

Moving a Store Directory

You might elect to move a store directory, perhaps to a faster disk or a disk that has more disk space to the one on which the directory currently exists.

You must only perform this operation with the Server stopped. If you try with the Server still running, you are very likely to lose EDI data.

The steps involved are illustrated in the following example. A store directory in the physical location `/usr/old/edi/store_2` is to be moved to a new location of `/usr/new/edi/store_2`.

Step 1: Create the new directory.

```
# mkdir /usr/new/edi/store_2
```

Step 2: Make sure that the owner of the directory is Digital DEC/EDI.

```
# chown decedi /usr/new/edi/store_2
```

Step 3: Copy the contents of the old store directory to the new one.

```
# cp /usr/old/edi/store_2/*.* /usr/new/edi/store_2/
```

Step 4: Create a link to `/var/adm/decedi/store_N`.

```
# ln -fs /usr/users/new/store_2 /var/adm/decedi/store_2
```

Step 5: Make sure that the owner of link is also Digital DEC/EDI.

```
# chown decedi /var/adm/decedi/store_2
```

Step 6: Check that the linked directory can be seen.

```
# ls -al /var/adm/decedi/store_2
```

After this, when you start the Server it should automatically start using the new store directory. Remember to delete the old directory when you are satisfied that the new directory is working correctly.

Changing the Size of Your Database

Changing the Size of Your Oracle Database

Refer to Oracle Documentation on how to change the size of your oracle database.

Sizes for Audit Trail Tables in the Database

The following are some of the more important tables within the database in terms of their potential effect on performance. They have the following space allocated to them with the default database as shipped with the kit:

Table 11-1 Sizes for Audit Trail Tables in the Database

Description	Table	Allocation (Records)
Current Document Audit Trail	TLF	1000
Current Document History Trail	THF	6000
Current Transmission Audit Trail	CLF	500
Current Transmission History Trail	CHF	3000
Archive Document Audit Trail	TLA	10000
Archive Document History Trail	THA	60000
Archive Transmission Audit Trail	CLA	5000
Archive Transmission History Trail	CHA	18000

If the number of records in each of the tables increases beyond the space allocated, the database storage area that is used to hold that table is extended.

Retrieving EDI Data From a Secondary Archive

As described previously in this chapter, an important periodic maintenance activity is using Secondary Archive to remove old EDI data. This process removes information that is no longer needed online, from the Archive Audit Trail to an offline archive.

You may wish to retrieve some or all of this information at a future date, perhaps simply to view information, or perhaps to resend a document or transmission file.

The Retrieve utility allows you to process the contents of an offline archive, and do one or more of the following:

- Retrieve individual named documents and transmission files from the archive.
- Retrieve all documents and transmission files from the archive.

The following shows an example of retrieving a single named document from an archive to the Archive Audit Trail.

```
# decedi_retr -d "APPL1_O_0000028345" -t "" -m "" -i /dev/rz6h
```

See *The Retrieve Utility* on page 13-16 for a more complete description of the facilities available for retrieving data. Some of this information is also on line in the man page entry for *decedi_retr(8)*.

Resending EDI Data

There may be occasions when you need to be able to resend outbound documents or transmission files. For example, because your trading partners system has failed, a document that you sent correctly was lost within the trading partner system.

With Digital DEC/EDI, you can resend copies of documents or transmission files from statuses of either **PURGEABLE** or **CANCELLED**. Such documents or transmission files would normally be in the Archive Audit Trail.

Note that if you have retrieved the document or transmission file from an offline archive, by using the retrieve function, you are only able to resend if you have a copy of the external format file or transmission file. This depends on what options you selected when you performed the original Secondary Archive.

Use Cockpit to perform the resend function. Resent documents are created with a status of **CONVERTED** and are queued for processing by the TFB. Resent transmission files are created with a status of **AWAIT TRANSMISSION** and are queued for processing by the Communications Services.

Digital DEC/EDI gives the resent data new document ID and transmission file names as appropriate, the data content is otherwise exactly as was sent before (although the TFB provides a resent document with new control numbers as it is built into a new transmission file).

The document name after reprocessing would be same. So, for example 20011122171120ABQDCP_ACME1 after reprocessing would be 20011122171120ABQDCP_ACME1. For resent transmission files, the name of the resent transmission file is prefixed by Rnnn, where nnn is the resend count. So, for example, a transmission file named 20011211204036ACL0B4_CVP_E would be resent the first time as R001_20011211204036ACL0B4_CVP_E.

Repairing EDI Data

In abnormal cases, documents and transmission files can become stuck in the Current Audit Trail. The `decedi_repair` tool gives you the ability to repair the audit trail in such circumstances.

The *decedi_repair* tool also gives you the ability to cancel documents and transmission files which the Cockpit would not allow you to cancel due to some processing still being outstanding on these. For instance, a completely corrupt transmission file which causes the EDIFACT Translator to crash, will be recovered next time the system is started, and cause the Translator to crash again. In this case the *decedi_repair* tool could be used to cancel the document whilst it was a **`SEPARATING`** state so that it would not be reprocessed again, and the translator could get on with processing the other documents.

Please refer to the man page entry for *decedi_repair (8)* for more details.

It should be noted that the decedi_repair tool is there to keep your system going: it corrects the symptoms, but does not solve the problems which caused the symptoms in the first place. When using it, please note down anything it does repair, and investigate what caused the problem in the first place.

Chapter 12 **System Performance**

The Digital DEC/EDI system is formed from many different components that need to act together to successfully process your EDI data. To get the best performance from the system, you need take care in configuring certain parts of the system.

This chapter describes the most important factors that can effect performance. While most of these relate to configuration settings, it is important to remember that many of the maintaining activities described in previous chapters are vital to ensuring you keep a well running system. With the Digital DEC/EDI system, you must actively maintain it, if peak performance is to be sustained over long periods.

The information provided within this chapter is aimed at operation of production systems where throughput is important and error rates are likely to be very low, rather than at test systems where throughput is probably less important and error rates are likely to be higher.

What is Performance

The periodic maintenance activities described in the preceding chapter should be sufficient to preserve reasonable performance of a stable well configured Digital DEC/EDI Server. However over time, you may change the characteristics of the work you are asking the Server to handle and you may place other loads on the system (running applications other than Digital DEC/EDI on the same CPU).

Best performance derives from Digital DEC/EDI, when the software configuration is best matched to the available hardware resources for the current task. The hardware provides CPU, memory and I/O resources (and in some cases network resources) to the software. So, performance is being limited when the hardware is not able to deliver enough of one or more of these types of resource to the software that is trying to run. Increasing the availability of one or more of these resources improves performance. However that is not always possible, and often changing the software configuration may provide equally significant improvements in performance.

This section provides some guidelines on checking and improving Digital DEC/EDI system performance. It provides some simple examples of UNIX system commands you can use to check on resource usage, but it does not attempt to provide an in depth guide to UNIX system tuning.

Improving Performance

To try to improve or optimise the performance of Digital DEC/EDI, you can take one or more of the following approaches:

- Reduce the loading on the hardware of other non-Digital DEC/EDI software making use of the same hardware resources. For example, are there other applications running on the same machine that you could move to another machine ? Could you tune other applications to make better use of the system ?
- Increase the available hardware resources. For example, use more disks to spread the I/O load or buy extra memory to increase the amount available for caches
- Change the Digital DEC/EDI configuration to make more effective use of available resources.

The above takes a simple view of the Digital DEC/EDI system, making the assumption that Digital DEC/EDI presents a reasonable constant loading on the available resource. In practice this is very rarely the situation. The Digital DEC/EDI system is comprised of many separate elements, some of which are running continuously, or at least for as long as there is work to do (for example, the converter), other components of which run to a defined schedule (for example, the TFB) or in response to an external event (for example, the Mapper). The work that you request the system to perform is likely to fluctuate significantly over a 24 hour period.

This means that at many times during the day, the hardware is more than capable of dealing with the resource demands placed on it, but at peak times it may not be.

Before trying to improve the system performance, you need to decide what it is about the way in which the current system is performing that you are trying to improve. Only then can you make sensible decisions about what changes, if any, you need to make.

Normally, you are looking to improve the performance of the system at periods of peak loading. You should approach this by looking to see what the CPU, I/O and memory usage characteristics of your system are, during periods of peak loading. A well tuned system should be showing no obvious hardware limitations. A badly tuned system shows, for example, that one or more disks in the system are receiving I/O requests in excess of the rate they can service.

What follows is not a guide to tuning UNIX systems, but an indication of some of the commands you can use to see how the UNIX system is performing.

Checking System Performance

The following sections provide you with a general idea of what areas of the system, if any, are limiting performance. Following sections in this chapter provide information on configuration and usage options for improving Digital DEC/EDI performance. These represent good general practice, but they may not all always deliver significant performance improvements to your system. You should look to see how the performance limitations identified above may be affected by the suggestions offered in that chapter. Remember that improvements made in one area, may deliver benefits in other areas.

Checking System CPU Usage

The following example command shows the CPU usage of the main processes in the system:

This example is taken from a system running Oracle Rdb

```
# ps auwx | head

USER   PID   CPU   MEM ... ...TIME    COMMAND
dbsmgr 12447 38.0  3.4 ... ...0:07.44 v61/rdbserver
root   12414 21.0  2.0 ... ...0:04.39 /usr/sbin/decedi_etrnd
dbsmgr 12428 10.0  3.8 ... ...0:04.64 v61/rdbserver
root       0  8.0 14.4 ... ...19:28.16 [kernel idle]
dbsmgr 12467  6.0  3.8 ... ...0:04.54 v61/rdbserver
dbsmgr 12419  3.0  1.6 ... ...0:02.12 /usr/sbin/decedi_etfsd
root   12238  1.0  1.7 ... ...0:01.76 /usr/sbin/decedi_impexpd
```

In this example, almost 60% of the CPU time is being taken by the EDIFACT translator process (`/usr/sbin/decedi_etrnd`) and its corresponding `rdbserver` process. See Appendix D for more information on the names of the separate Digital DEC/EDI processes.

Checking System I/O Rates

The following command shows the I/O rates for the system. The format of the command is:

```
iostat <interval> <number_of_reports>
```

For example:

```
# iostat 1 5

  tty        rz6       fd0       re0       dk3           cpu
tin tout  bps tps  bps tps  bps tps  bps tps   us  ni  sy  id
 0    9    0   0    0   0   12   1    0   0    2   0   1  98
 0    0    0   0    0   0  462  60    0   0   31   0  16  53
 0    0    0   0    0   0  219  29    0   0   12   0   7  81
 0    0    0   0    0   0    0   0    0   0    0   0   1  99
 0    0    0   0    0   0  472  94    0   0    3   0   5  92
```

In this example, all the I/O is being done to a single device. This may indicate a performance bottleneck around that one device. Although you need to check whether, for the disk in question, it is able to respond to this rate of requests as defined by the manufacturers specification for the disk. If it is not, you may need to reconfigure the system.

Checking Virtual Memory Usage

The following command shows virtual memory usage for the system. The format of the command is:

```
vmstat <interval> <number_of_reports>
```

For example:

```
#vmstat 1 5

Virtual Memory Statistics: (pagesize = 8192)
  procs    memory     pages           intr        cpu
  r  w  u  act  free wire fault ...  in  sy  cs  us  sy  id
  2149 18   11K  506 2312   3M  ...  21 432 348   2   1  98
  2147 20   11K  485 2312  122  ...  83  2K  1K  32  25  43
  2149 18   11K  483 2312   77  ...  85  2K  1K  33  22  45
  5148 16   11K  480 2312   77  ...  89  2K  1K  35  22  43
  5148 16   11K  477 2312   77  ... 120  2K  1K  26  20  54
```

Of particular importance is the free memory pages. If this continues to decline or is at a very low number then the system will not perform as efficiently as it could.

General Guidelines for Improving Performance

The following general configuration details can affect Digital DEC/EDI system performance:

- How well tuned the database is. Database tuning is covered extensively in the documentation for the database product.
- Correct placement of store directories. Much I/O takes place to the store directories. For best performance, they should be located on fast disks (this should exclude network mounted disks), and any store directories spread over the disks you have to ensure the best balance of I/O between disks. It is functionally possible to have all store directories on the same disk. However this may result in the I/O to that one disk being a significant bottleneck on system performance. Splitting the store directories over more disks can improve performance dramatically. See *Creating and Moving Store Directories* on page 11-8 for information on creating and moving store directories.
- Correct placement of the database. Similar reasoning to the placement of the store directories applies to the placement of the database. The system performs many I/O operations to the database. The better access you can provide to the database, the more effectively the system performs.
- Amount of data maintained in the archive database. The more data you maintain in the archive database the slower the performance is for the components that are accessing it. As part of your regular maintenance you should perform Secondary Archive operations to clean the database of data you no longer need to keep on-line. It is possible that more frequent Secondary Archives may improve system performance by helping to maintain a smaller database.
- By using high priority versus normal priority for outbound data. When outbound documents enter the Server they are set to be processed at one of two priorities. Unnecessary use of the high priority setting may result in a lower overall performance. High priority is intended to be applied to a minority of documents. It is not indended to be used to deliver faster performance for every document. By setting a document to high priority, the TFB builds that document into a transmission file with no other documents. This is efficient for that one document, but not for the remainder of the system.
- Writing unnecessary messages to the Error Log file. You can reduce the amount of lower severity, informational messages that are written to the

Error Log file by setting the DECEDI_LOG_SEVERITY environment variable See Appendix C for more information on this environment variable. By writing less information to the Error Log you can help improve performance.

- Requesting logging of archive server actions. By the setting of an environment variable, DECEDI_AS_VERBOSE, you can instruct the archive server to log information about the data is it processing. Any use of this facility can cause performance to be reduced.
- Electing to turn on the logging of history information. By the setting of an environment variable, DECEDI_MAINTAIN_HISTORY, you can instruct the Server to maintain history records about documents and transmission files flowing through the Server. The logging of this extra history information can produce a significant reduction in performance. Only turn this logging on if you need this extra information.
- Many concurrent Cockpit users. The Cockpit allows you and other users to query the state of data in the Server, and in effect to query the database. Excessive use of this causes conflicts with the Server's acccess to the same database. If you, and several other users, configure Cockpit to provide regular and frequent updates to displays, you can cause the Server to become bottlenecked in accessing the database. Multiple concurrent applications using the `decedi_track` API or `trade track` CLI call can also cause this.
- Copying between Application Client and Server by using shared disks. As you post data to or fetch data from the Server via the Application Client, the creation and copying of data files is implied. When you register an Application with the Server by using the Management Services editor, you can define whether or not data files need to be explicitly copied between client and server (which is the slower option), or whether the two components share one or more disks (such as would be the case for network mouted disks). In this latter case the Server can read and write files directly and the performance of posting and fetching data can be significantly improved.

Improving Performance of the Mapping Services

The following can affect the performance of the Mapper at run time:

- The amount of auditing you request the Mapper to perform. By default, Mapper auditing is set to a level that provides a minimum of information, sufficient to permit data tracking. It is possible to increase or decrease the level of auditing information provided. See *Specifying Mapper Audit Levels* on page C-13 for guidance on how to change auditing levels. You should only increase the level of auditing above the default if you are sure you need the extra information and can accept the probability of reduced performance.
- The use of significant numbers of hook routines when running Mapping Tables. Each call to a hook routine reduces the effective performance of the Mapper. For best performance, try to reduce your use of hook routines.
- Mapping Tables that request significant amounts of information to be saved in history files. The more information you request the Mapper save in history files, the slower the mapping process is, and, the slower other Server components, such as Secondary Archive perform.
- Incorrect sizing of the Mapper Tables cache. By default, the Mapper can hold up to 10 Mapping Tables cached in memory. Performance is optimised if you make repeated use of a small number of Mapping Tables and the cache can hold all of these tables in memory. The default size of the Mapper cache can be overridden by using the environment variable `DECEDI_MAX_MAPPING_TABLES`. See Appendix C for more information on this environment variable. Increasing the cache size to hold more Mapping Tables in memory may improve performance. However you need to ensure you don't increase it to a level where the operating system has insufficient memory to allow the Mapper to run in memory permanently.

Improving Performance of the Translation Services

The following can affect the performance of the Translation Service:

- Enabling the following features: detailed listings, code validation in conversion/translation. The extra processing and I/O required to sustain each of these functions can significantly reduce the performance of the converter and translator components. The flags that control these settings are individually settable within each trading partner. Only turn these flags on where you need the features they provide and can tolerate the probable reduction in performance.
- Incorrect settings for the size of the EDI tables cache. Each of the Translation Services components have a copy of the EDI tables cache in memory. This cache contains EDI document definitions and versions of EDI standards dictionaries. By using environment variables, you can change the default size of this cache. See Appendix C for information on the environment variables: `DECEDI_MAX_DOCUMENTS` and `DECEDI_MAX_SEGMENT_TABLES`. Up to the point where your system runs out of physical memory, increasing the size of this cache can increase performance where you need more EDI tables than can otherwise be maintained in the default cache.
- Incorrect setting of the transmission file build interval. See the following section.

Setting the Transmission File Build Interval

The Digital DEC/EDI Transmission File Builder (TFB) build intervals define the frequency with which the TFB's build transmission files. By default the Digital DEC/EDI TFB build intervals are set to 5 minutes.

You can override this default value by using the Management Services Editor “Configure Build Interval” option. You should normally set this to a value in the range of 1 minute to 1 hour.

Experience of the throughput and dynamics of your Digital DEC/EDI system are the best guide as to what to set this value to. If you set the value too low:

- You produce smaller transmission files than may be necessary because the TFB may not be able to take advantage of documents that would otherwise have been available to go into the same transmission.
- On a busy system, overall performance may be reduced because the TFB is not performing efficiently.

If you set the value too high:

- On a busy system, the TFB can become the bottleneck for the processing of outbound data with a large queue of documents waiting to be built.
- On a quiet system, you may be waiting unnecessarily for the TFB to build simple transmission files.

Any change you make to the setting take effect immediately if the TFB is running, or when you next start Digital DEC/EDI.

Improving Performance of the Communications Services

The following sections describe for each gateway, the features which can affect their performance.

Improving Performance of the Import/Export Gateway

Much of the work of the Import/Export gateway is associated with reading and writing files to disk directories you specify. Performance can be affected by slow disks.

The Import/Export Gateway is a job based gateway, with the jobs being scheduled by the user using the `cron` utility described in Chapter 7 *Scheduling Jobs*. There is a fine balance to be made between scheduling jobs too often, and too little. An EXPORT job causes the database to be searched for transmissions waiting to be exported for that connection. Excessive export jobs will cause an extra load on the database, whilst if the jobs are not scheduled frequently enough then there will be long delays between when files are ready to be exported and when they actually are. Similarly on inbound, an IMPORT job, causes the import directly to be scanned for any files waiting to be imported. Excessive import jobs cause an extra load on the disk device of the import directory, whilst if the jobs are not scheduled frequently enough then there will be long delays between when files are ready to be imported and when they actually are.

Improving Performance of the Pedi Gateway

Chapter 15 *Problem Solving – Pedi Gateway* describes the different options you can use to save copies of different data files to assist problem solving. If you have these enabled, you should expect a significant reduction in performance of the Pedi gateway. Only these options when you are fault finding.

One of the configuration parameters for the Pedi gateway is the *Delivery Queue Poll Interval*. This is the time period between successful polls by the Pedi gateway of the MTA to check for inbound transmission files. If you set this value too high, you can cause the gateway to perform many unnecessary checks. If you set this value too low, you can cause data to be held within the MTA for unnecessarily long periods. The default for this value is 5 minutes.

Improving Performance of the OFTP Gateway

Chapter 16 *Problem Solving – OFTP Gateway* describes the different trace options you can use to obtain diagnostic information to assist problem solving. If you have these enabled, you should expect a significant reduction in performance of the OFTP gateway. Use only these trace options when you are fault finding.

Improving Performance of the SMTP/MIME Gateway

Chapter 17 *Digital Problem Solving – SMTP /MIME Gateway* describes the different trace options you can use to obtain diagnostic information to assist problem solving. If you have these enabled, you should expect a significant reduction in performance of the SMTP/MIME gateway. Use only these trace options when you are fault finding.

Improving Performance of the 3780 Gateway

Chapter 18 *Problem Solving – 3780 Gateway* describes the different trace options you can use to obtain diagnostic information to assist problem solving. If you have these enabled, you should expect a significant reduction in performance of the 3780 gateway. Use only these trace options when you are fault finding.

Chapter 13 Secondary Archive and Retrieve

Earlier chapters within this guide described Secondary Archiving and Retrieve as operations to be performed in the context of other maintenance activities. This chapter presents these two functions in more detail, describing the different options available, the reports that are produced, and the extent to which you can customise their behaviour.

The Secondary Archive utility allows you to remove obsolete data from the Digital DEC/EDI Archive area to off-line media, such as a tape or a WORM (Write Once Read Many) device, or to a specified location on a disk. The Retrieve utility allows you to restore some or all of this information back into the Digital DEC/EDI Archive should you require the information for Audit purposes, or even to resend some data.

In the explanations that follow, the term ‘objects’ is used to refer equally to documents, transmissions files or mapper runs.

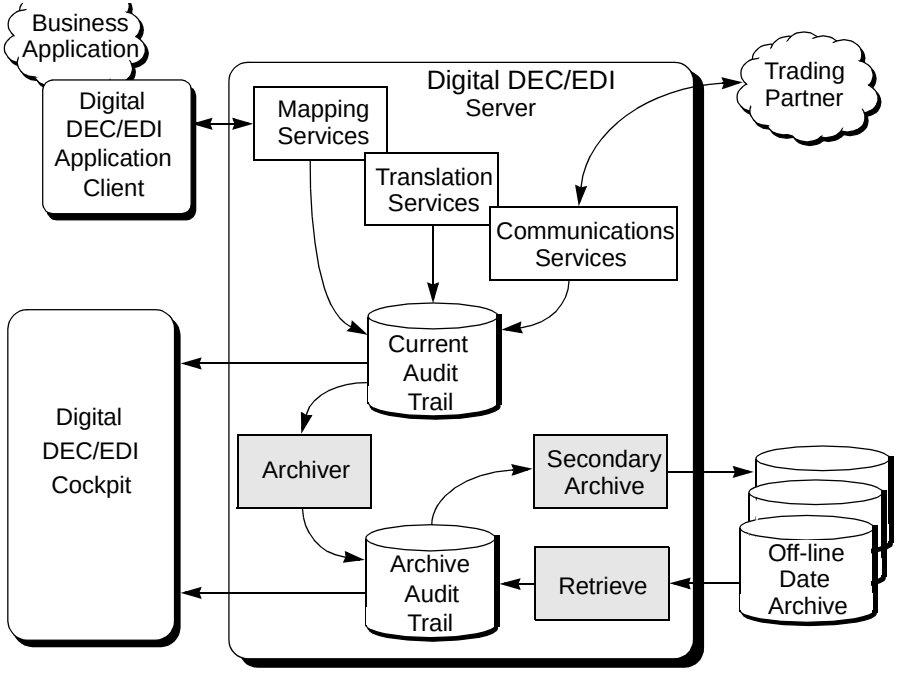
A subset of this information is also contained in the man page help for these two commands: *decedi_arch(8)* and *decedi_retr(8)*.

Secondary Archive

Digital DEC/EDI is capable of generating and recording large amounts of information about the EDI data it is processing. This information is stored on disk, both in store directories and in the Current and Archive Audit Trails. Unless action is taken, the amount of disk space used increases to the point where no more is available and the Server halts.

The Secondary Archive function allows you to free up database and disk space by moving any obsolete objects from the Archive database and storage areas into a more flexible environment, that is, onto a magnetic tape or writeable Compact Disk. The information can then be retained indefinitely, or as fits your business practices. The data can be restored at any time by using the Retrieve utility.

Figure 13-1 Audit Trails and Archiving



The Secondary Archive function produces reports that detail the information included in each archive. These reports are produced in the directory `/var/adm/decedi/arch_reports`. These reports allow you to search for specific objects by using the unique identifers associated with the objects. The format of these reports is described in *Secondary Archive Report Format* on page 13-11.

What Secondary Archiving Does

The Secondary Archive utility takes the database entries for obsolete objects, and, optionally, the files associated with these objects and places in a location you specify. This can be either an off-line media such as a tape or WORM device or into a specific file. By default, Secondary Archive uses the UNIX `tar` utility to produce archives. However, you can specify a different utility be used.

You can choose which information is included in a Secondary Archive run by specifying:

- A date, such that only objects created before this date are included.
- A date, such that only objects created after this date are included.
- Whether or not to include all information associated with an object. By default only audit trail information and a copy of each transmission file are included. You can specify to include different amounts of data.

In operation, Secondary Archive begins by verifying that the last archive performed on this system did not fail. If it detects the previous Secondary Archive did fail, then it attempts to recover from that previous failure by using the new arguments that you have specified. The details of where recovery begins depend on where the previous failure occurred.

The utility then ensures that it has the appropriate access rights to the directories that it requires. This includes write access to the following directories:

- `/var/adm/decedi/backup`
- `/var/adm/decedi/data`

The utility checks that no one else is already running a Secondary Archive or Retrieve and checks to see if you have provided a customised script that describes the archive, or whether the Digital supplied script is to be used.

The utility then starts selecting documents to archive. Each audit entry in the database for the selected documents is copied to the Document Audit file and, if requested and available, all of the historical entries associated with the documents are copied to the Document History file. Also, if requested, an entry is written to the Archive Input file for the in-house, external, error listing and detailed listing files associated with the document. Finally, a line is added to the Archive Report detailing the document selected.

Once the selection of documents has completed, the utility then begins selecting transmission files to archive. Each audit entry in the database for the selected transmission files is copied to the Transmission Audit file and, if requested and available, all of the historical entries associated with the transmission files are copied to the Transmission History file. Also, if requested, an entry is written to the Archive Input file for the transmission file any error listing file associated with the transmission. Finally, a line is added to the Archive Report detailing the Transmission File selected.

Once the selection of transmission files has completed, the utility then begins selecting mapper run audit information to archive. Each audit entry in the database for the mapper run selected is copied to the Mapper Audit file and, if requested, pointers to all of the historical files associated with the mapper run are written to the Archive Input file.

The utility then writes the names of the Document, Transmission and Mapper Audit and History files to the Archive Input file and calls the archive script that creates the requested archive from the list of files specified in the Archive Input file.

If the script returns a success (that is zero) status, then the utility begins removing all objects and associated files from the Digital DEC/EDI database and storage directories.

Finally, it updates the date and time stamps at the beginning of the Archive Report and exits.

Secondary Archive Command Syntax

The Secondary Archive utility is invoked by using the `decedi_arch` command. You must either have `/usr/sbin` as part of your path list or prefix the command with the `/usr/sbin/` directory name. The format of the command is:

```
decedi_arch [-v] [-c [n][h][i][e][l][d][t][m]] [-a DDNNYYYY]
 [-b DDNNYYYY] [-o outnam] [-f "outflg"]
```

where the square brackets ([]) delimit optional arguments. The flags listed above perform the following modifications to the command:

-v Causes the utility to write informational messages to the screen to indicate each stage in its operation. This includes the date and time that various phases begin and end, and, for every 500 objects selected, a message indicating the number of objects selected so far. The default is not to print any informational messages.

-c Causes the utility to modify the amount of information saved in the archive. Note that the audit information associated with objects must always be archived and cannot be deselected. The argument following this flag can contain any of the following subflags:

- n: archive nothing but the audit information from the database.
- h: if available, archive the historical information maintained in the database.
- i: if available, archive the in-house file associated with each selected document.
- e: if available, archive the external file associated with each selected document.
- l: if available, archive the error listings associated with each selected document, and the error listings associated with each selected transmission.
- d: if available, archive the detailed listings associated with each selected document.
- t: if available, archive the transmission files associated with each selected transmission.
- m: if available, archive the Mapper History files associated with each selected mapper run.

By default, the utility saves audit (mandatory) and transmission files only. This is equivalent to specifying `-c t`. To deselect the archival of transmission files, use `-c n`. If you want to archive just in-house files, then use `-c ni` as `n` clears the information selected and `i` selects the in-house files. The order in which you specify these arguments can be important. For example, specifying `-c in` causes the utility to archive only the audit information as the `n` clears the flag set by the `i`. That is, specifying `-c ni` generates a different action to specifying `-c in`.

-a Causes the utility to select objects created in Digital DEC/EDI on or after midnight of the date specified by the next argument. The next argument must be a date in the format `DDNNYYYY` where `DD` is the day of the month, `NN` is the numeric representation of the month between 01 (for January) and 12 (for December) and `YYYY` is the full year specification (that is 1995, rather than 95).

For example, specifying `-a 02031995`, would result in the selection of objects that were created on or after the 2nd of March 1995.

-b Causes the utility to select objects created in Digital DEC/EDI before midnight of the date specified by the next argument. The next argument must be a date in the format `DDNNYYYY` where `DD` is the day of the month, `NN` is the numeric representation of the month between 01 (for January) and 12 (for December) and `YYYY` is the full year specification (that is 1995, rather than 95).

For example, specifying `-b 01041995`, would result in the selection of objects that were created before the 1st of April 1995. An object created at midnight on the 1st of April 1995 would not be selected.

-o Indicates that the next argument is the name of the output device. This argument is passed directly to the archive script for processing. The default is to pass a file name of `/var/adm/decedi/backup/DECEDI_YYYYNNDD` where `YYYYNNDD` is the current date. If you want to archive to a tape or WORM device then pass the name of the device. See the documentation on the Digital UNIX tar command for further information. The archive script also uses the `-f` tar flag to control where the information is placed.

If your system contains a customized archive script, please contact your system manager to ask what output device specification you can use.

-f Indicates that the next argument is a quoted (that is, uses double quotes "..") string containing additional flags that should be passed to the archive script. These are intended to be used to control whichever Digital UNIX utility is used to create the actual archive. These flags are useful if you wish to pass values that indicate actions like “do not rewind the tape” or “append to the existing archive”.

For more information on the flags available, refer to the documentation on the tar utility. If you are using a customized script then please ask your system manager.

Modifying Secondary Archive With Customized Scripts

If you wish to incorporate your organization's own backup strategy into the Digital DEC/EDI Archive Utility you may do so by writing your own archive script.

The Secondary Archive utility searches for a customized script called `/usr/sbin/decedi_arch_syscript`. If you create a script of your own, you should give it this name instead. If this script exists, it is called in preference to the Digital supplied script in `decedi_arch_script`.

Note: Digital strongly recommends that you examine and understand the workings of the Digital supplied archive script before attempting to write your own. Do not change `/usr/sbin/decedi_arch_script`*, the existing archive script.*

The script must accept 3 arguments that are passed to it. These are:

$1 The name of the archive to create. This is a copy of the argument passed to the archive utility associated with the `-o` flag.

$2 The name of the file containing a list of files to archive. This file contains each filename, including its full directory specification. Each filename is terminated with a new-line character. Note that these filenames may contain symbolic links for directories, so you should ensure that you instruct whatever utility you use to create the archive, to follow all symbolic links and to copy the physical file associated with the link.

$3 Any specialized arguments that were passed to the archive utility by using the -f flag.

The script is only required to create the archive requested by using the files listed in the file specified by the second argument. If you believe it necessary, then you may also verify the arguments to check existence, the ability to write to (first argument) and the ability to read from (second argument).

If it executes successfully, the script should return a value of zero. Any other value causes the Secondary Archive utility to assume that the script has failed,and it terminates indicating a script failure. The output and error streams from the script are written to the file `decedi_arch_script.outerr` in `/var/adm/decedi/backup`. Digital

strongly recommends setting any and all flags that enable diagnostic analysis to help track down problems within the script.

Files Involved in Secondary Archive

The following files are invoked or used by the archive utility. Unless otherwise specified, all files reside in the `/var/adm/decedi/backup` directory:

- `decedi_arch` - The Digital DEC/EDI Secondary Archive Utility. Resides in the `/usr/sbin` directory.
- `decedi_arch_script` - The Digital supplied Digital DEC/EDI Secondary Archive Script. Resides in the `/usr/sbin` directory.
- `decedi_arch_syscript` - The customized Digital DEC/EDI Secondary Archive Script. If present, resides in the `/usr/sbin` directory.
- `decedi_arch_script.outerr` - Contains the output and error streams from the last failed Digital DEC/EDI Secondary Archive Script run. This file is automatically deleted after a successful run.
- `decedi_arch_CF.tmp` - Control file used by the Secondary Archive Utility to remember what objects and associated files it has archived. This file is deleted after a successful run.
- `decedi_arch_IF.tmp` - Archive Input file containing a list of files to place in the archive. The name of this file is passed to the archive script as its second argument. This file is deleted after a successful run.
- `decedi_arch_CPF.dat` - The archive utility's CheckPoint file that tells the utility if the last archive was successful, and, if not, at which point it failed. This file is deleted after a successful run. If present, resides in the `/var/adm/decedi/data` directory.
- `decedi_arch_DA.v`*nn* - The Document Audit file used to hold Audit information from the Digital DEC/EDI database for all documents within the archive. This file is saved to the archive, and is deleted after a successful run. *nn* is the version of Digital DEC/EDI, for example 30 (for Digital DEC/EDI V3.0) or 31 (for Digital DEC/EDI V3.1).
- `decedi_arch_DH.v`*nn* - The Document History file used to hold Historical information from the Digital DEC/EDI database for all documents within the archive. If your Digital DEC/EDI system does not gather historical information or you elected not to save the historical

information then this file is. The file is saved to the archive, and is deleted after a successful run. *nn* is the version of Digital DEC/EDI, for example 30 (for Digital DEC/EDI V3.0) or 31 (for Digital DEC/EDI V3.1).

- `decedi_arch_TA.v`*nn* - The Transmission Audit file used to hold Audit information from the Digital DEC/EDI database for all transmissions within the archive. This file is saved to the archive, and is deleted after a successful run. *nn* is the version of Digital DEC/EDI, for example 30 (for Digital DEC/EDI V3.0) or 31 (for Digital DEC/EDI V3.1).
- `decedi_arch_TH.v`*nn* - The Transmission History file used to hold Historical information from the Digital DEC/EDI database for all transmission files within the archive. If your Digital DEC/EDI system does not gather historical information or you elected not to save the historical information then this file is empty. The file is saved to the archive, and is deleted after a successful run. *nn* is the version of Digital DEC/EDI, for example 30 (for Digital DEC/EDI V3.0) or 31 (for Digital DEC/EDI V3.1).
- `decedi_arch_MA.v`*nn* - The Mapper Audit file used to hold Audit information from the Digital DEC/EDI database for all mapper runs within the archive. This file is saved to the archive, and is deleted after a successful run. *nn* is the version of Digital DEC/EDI, for example 30 (for Digital DEC/EDI V3.0) or 31 (for Digital DEC/EDI V3.1).
- `decedi_arch_report_D, decedi_arch_report_T, decedi_arch_report_M` - All three of these files are temporary files used to hold the information that goes into producing the final Archive Report. After a successful run, these files are deleted.
- `decedi_arch_YYYYNNDD` - The Secondary Archive Report that resides in the `/var/adm/decedi/arch_reports` directory. The `YYYYNNDD` section of the file name is replaced by the date that the archive was written. For example, an archive run on the 24th of June 1995 would be called `decedi_arch_19950624`.

Secondary Archive Report Format

The Secondary Archive report is formatted to be both human readable and computer processable. If you wish to write your own utility to process the reports then please note the following format rules that have been used:

- Each line of the report is terminated by a new-line character.
- Each line of the report is a maximum of 132 characters long, including the new-line character.
- Any lines beginning with an asterisk "*" are comment lines that may be ignored.
- Any lines beginning with the formfeed character (ASCII character 12) or that have a zero length can be ignored.
- Any lines beginning with the characters "DOC" begin a two line section detailing a document that has been included in the archive. The first line of the section contains the following information. Each piece of information is separated by one or more space characters.

Table 13-1 Secondary Archive Report, Document Data, Line 1

Columns	Description
1-3	"DOC"
5-24	The Application Id associated with the document.
26	The Direction ('I' for Inbound, 'O' for Outbound) associated with the document.
28-37	The Document Count associated with the document.
39-58	The Partner Id associated with the document.
60-83	The date and time that the document was created.
85-119	The User Reference associated with the document.

The second line of the section contains the following information.

Table 13-2 Secondary Archive Report, Document Data, Line 2

Columns	Description
5-8	The EDI standard associated with the document. This is set to one of the following values: • `EDIF` for EDIFACT documents • `ODET` for Odette documents. • `TRAD` for TRADACOMS documents. • `X12 ` for X12 documents (note the trailing space). • `TDCC` for TDCC documents.
10-19	The version of the standard associated with the document. This is the version used to locate Digital DEC/EDI tables rather than the information from the interchange, group or document envelopes.
25-30	The document type associated with the document.
33-46	The Interchange Control Number associated with the document.
49-62	The Group Control Number associated with the document.
65-78	The Document Control Number associated with the document.
80-129	The name of the transmission file associated with the document.

- Any lines beginning with the characters “TRF” describe a transmission file that has been included in the archive. The line contains the following

additional information. Each piece of information is separated by one or more space characters.

Table 13-3 Secondary Archive Report, Transmission File Data

Columns	Description
1-3	"TRF"
5-54	The name of the transmission file.
56-79	The date and time that the transmission file was created.
81-90	The size of the transmission file, in bytes (right justified, space padded).
97	The Direction ('I' for Inbound, 'O' for Outbound) associated with the transmission file.
105-108	The EDI standard associated with the transmission file. This is set to one of the following values: • `EDIF` for EDIFACT transmission files • `ODET` for Odette transmission files. • `TRAD` for TRADACOMS transmission files. • `X12 ` for X12 transmission files (note the trailing space). • `TDCC` for TDCC transmission files. This field may also contain 'PASS' to indicate a transmission file that has been routed by using one of the bypass routing options, 'MULT' to indicate that several EDI standards may be contained within the transmission file, 'MIXD' to indicate that the transmission contains both information aimed at the BYPASS interface and information to be processed by one of the Digital DEC/EDI translation components or 'SPLT' that indicates that the transmission file has been split out into multiple transmission files each of which has been made available to the BYPASS interface.

- Any lines beginning with the characters "MAP" begin a two line section detailing a mapper run that has been included in the archive. Note that the amount of information saved and the amount presented in the report depends on the level of auditing selected for the mapper. The first line of

the section contains the following information. Each piece of information is separated by one or more space characters.

Table 13-4 Secondary Archive Report, Mapper Data, Line 1

Columns	Description
1-3	"MAP"
6-11	The Mapper run id number.
14-37	The date and time that the mapper run occurred.
39-68	The User Reference associated with the mapper run.
70-124	The comment associated with the mapper run.

The second line of the section contains the name of the application file associated with the mapper run. This field spans column 5 to column 130.

Secondary Archive Examples

1. The example below archives all objects created on or after midnight on the 1st of April 1995 but before midnight on the 1st of May 1995 (that is all of April's documents, transmission files and mapper runs) and writes then to the tape drive pointed to by the block special file `/dev/rz6h`. The archive includes the default audit and transmission files, and also the in-house files and any mapper history files. A special flag is passed to the tar utility by using the `-f` flag. The flag indicates that the tape is 40960 blocks long, that is, it can hold 20 Mb.

```
# decedi_arch -a 01041995 -b 01051995 -o /dev/rz6h -c im \
    -f "-S 40960"
```

2. The following example archives all objects from the very first in the database up to, but not including, midnight today, and writes them to the tape device `/dev/rz2h`, that points to a WORM device. The special flag `-r` is passed to the tar utility to tell it to append to the existing archive, rather than rewind the device and start at the beginning again. Only audit information and transmission files are saved.

```
# /usr/sbin/decedi_arch -o /dev/rz2h -f "-r"
```

3. The following command archives all objects, from the very first in the database, up to, but not including, midnight today, and writes them to the

file DECEDI_9505Archive.tar. The command also requests that the utility provide informational messages as it works.

```
# decedi_arch -o DECEDI_9505Archive.tar -v

initializing environment at Tue May 30 11:37:50 1995
selecting documents to archive at Tue May 30 11:37:54
1995....
total of 300 documents selected by Tue May 30 11:37:56
1995.
selecting transmissions to archive at Tue May 30 11:37:56
1995....
total of 100 transmissions selected by Tue May 30 11:37:57
1995.
selecting maps to archive at Tue May 30 11:37:57 1995....
total of 100 maps selected by Tue May 30 11:38:01 1995.
calling Operating System archive shell script
/usr/sbin/decedi_arch_script at Tue May 30 11:38:01 1995...
Operating System archive completed by Tue May 30 11:38:05
1995
deleting archived objects at Tue May 30 11:38:05 1995...
deleted 500 database objects and 0 files so far...
deleted 500 database objects and 500 files so far...
deleted 500 database objects and 1000 files so far...
deletion completed by Tue May 30 11:39:57 1995
```

The Retrieve Utility

The Retrieve utility takes an archive produced with the Secondary Archive utility and restores all or a selected subset of the objects contained in the archive back into the Digital DEC/EDI Archive database and storage areas. By default, Retrieve uses the UNIX tar utility to process the archives. However if you have created an archive by using a different utility, by providing a customised archive script, you must provide a similar script to allow Retrieve to access the data in the archive.

The utility allows you to specify which objects to include in its retrieval by selection arguments through which you can indicate which documents, transmission files and/or mapper runs to retrieve.

The utility begins by verifying that the last Secondary Archive performed on this system did not fail. If it detects that it did fail, then it issues a warning message indicating that you should rerun the Secondary Archive utility so that it may leave a clean archive environment.

The utility checks that it has the appropriate access rights to the directories that it requires. In particular, this includes write access to the `/var/adm/decedi/backup` directory.

The utility then ensures that someone else is not already running a Secondary Archive or Retrieve. It also checks that either a customized script or Digital supplied Retrieve script is available.

The utility restores the content of the archive to a temporary directory created under the `/var/adm/decedi/backup` directory by using the Digital supplied Retrieve script `/usr/sbin/decedi_retr_script` or, where you have provided a customised script in `/usr/sbin/decedi_retr_syscript`, selecting the latter.

The utility then begins selecting documents, transmission files and mapper runs to retrieve depending on the selection criteria given. For each entry that it finds that matches the criteria, it adds the entry back into the Digital DEC/EDI Archive database and copies any of the associated files that were archived back into the storage directories. Different store directories are selected by the Retrieve utility to those that objects may have used when they were originally in the system,

Once the requested objects have been restored, the utility removes the temporary directory and files and then exits.

Retrieve Command Syntax

The Retrieve utility is invoked by using the `decedi_retr` command. You must either have `/usr/sbin` as part of your path list, or prefix the command with the `/usr/sbin/` directory name. The format of the command is

```
decedi_retr -i inam [-v] [-d "docsel"] [-t "trfsel"]
 [-m "mapsel"] [-f "inflg"]
```

where the square brackets ([]) delimit optional arguments. The flags listed above perform the following modifications to the command:

-i Indicates that the next argument, `inam`, is the name of the input device. This argument is passed directly to the Retrieve script for processing. There is no default: a value must be supplied. If the archive is on a tape or WORM device then pass the name of the device. See the documentation on the Digital UNIX tar command for further information. Note that the Retrieve script uses the `-f` tar flag to control where the information is to be read from.

If your system contains a customized Retrieve script then please contact your system manager to ask what input device specification to use.

-v Causes the utility to write informational messages to the screen.

-d Causes the utility to retrieve only documents that match the quoted document id that follows. The document id must be within double quotes and may contain the wildcards “%” (match any single character) or “*” (match 1 or more characters). For example, specifying `-d "*_I_*"` would select all incoming documents, while specifying `-d "APPL1_*"` would select all documents sent or received by the application APPL1. The document id is not case sensitive. Any values you enter are converted to uppercase before processing. The default is to select all documents. To select none, specify an empty quoted string, that is, `-d ""`.

-t Causes the utility to retrieve only transmission files that match the quoted transmission file name that follows. The transmission file name must be within double quotes and may contain the wildcards "%" (match any single character) or "*" (match 1 or more characters). For example, specifying `-t "*_exmple"` would select all transmission files associated with the `EXMPLE` connection id. The transmission file name is not case sensitive. Any values you enter are converted to uppercase before processing. The default is to select all transmission files. To select none, specify an empty quoted string, that is, `-t ""`.

-m Causes the utility to retrieve only mapper runs that match the quoted mapper run id that follows. The mapper run id must be within double quotes and may contain the wildcards "%" (match any single character) or "*" (match 1 or more characters). For example, specifying `-m "0005*"` would select all mapper runs in the range 000500 to 000599 inclusive. The default is to select all mapper runs. To select none, specify an empty quoted string, that is, `-m ""`.

-f Indicates that the next argument is a quoted (that is, by using double quotes "...") string containing additional flags that should be passed to the Retrieve script to control whichever Digital UNIX utility is used to read the actual archive. These flags are useful if you wish to pass to the utility, characteristics like the size of the tape.

For more information on the flags available, refer to the documentation on the tar utility. If you are using a customized script then please ask your system manager.

Modifying Retrieve With Customized Scripts

If you wish to incorporate your organization's own backup strategy into the Digital DEC/EDI Retrieve Utility you may do so by writing your own retrieve script.

The Retrieve utility searches for a customized script called `/usr/sbin/decedi_retr_syscript`. If this script exists, it is called in preference to the Digital supplied script in `decedi_retr_script`.

Note: Digital strongly recommends that you examine and understand the workings of the Digital supplied retrieve script before attempting to write

your own. Do not change `/usr/sbin/decedi_retr_script`*, the existing retrieve script.*

The script must take 3 arguments. These are:

- $1 The name of the archive to read. This is a copy of the argument passed to the Retrieve utility associated with the `-i` flag. The only difference is that the Retrieve command expands the file name so that it contains a full directory path.
- $2 The name of the directory into which all files within the archive must be placed. If the Digital UNIX utility does not allow restoration of all files into the same directory then the script must ensure that all files are moved into this directory after they have been restored.
- $3 Any specialized arguments that were passed to the retrieve utility by using the -f flag.

The script is only required to restore the archive requested to the directory requested in the second argument. If you believe it necessary, then you may also verify the arguments to check existence, the ability to read from (first argument) and the ability to write to (second argument).

After a successful execution, the script should return a value of zero. Any other value causes the Retrieve utility to assume that the script has failed and it terminates indicating a script failure. The output and error streams from the script are written to the file `decedi_retr_script.outerr` in the directory `/var/adm/decedi/backup`. Digital strongly recommends setting any and all flags that enable diagnostic analysis to help track down problems within the script.

Files Involved in Retrieve

The following files are invoked or used by the Retrieve utility. Unless otherwise specified, all files reside in the `/var/adm/decedi/backup` directory.

- `decedi_retr` - The Digital DEC/EDI Retrieve utility. Resides in the `/usr/sbin` directory.
- `decedi_retr_script` - The Digital supplied Digital DEC/EDI Retrieve script. Resides in the `/usr/sbin` directory.
- `decedi_retr_syscript` - The customized Digital DEC/EDI Retrieve script. If present, resides in the `/usr/sbin` directory.
- `decedi_retr_script.outerr` - Contains the output and error streams from the last failed Digital DEC/EDI Retrieve script run. This file is automatically deleted after a successful run.

  `decedi_arch_CPF.dat` - The Secondary Archive utility's CheckPoint file that tells the Retrieve utility if the last Secondary Archive was successful. . This file is automatically deleted after a successful run. If present, resides in the `/var/adm/decedi/data` directory.

Retrieve Examples

1. The example below retrieves all outbound documents from application `APPL1`, all transmission files from connection id `EXMPLE` and no mapper runs, from the tape drive pointed to by the block special file `/dev/rz6h`. Note that a special flag is passed to the tar utility by using the `-f` flag. The flag indicates that the tape is 40960 blocks long, that is, it can hold 20 Mb:

```
# decedi_retr -d "APPL1_O_*" -t "*EXMPLE" -m "" \
 -i /dev/rz6h -f "-S 40960"
```

2. The following command retrieved all inbound documents from applications whose name begins with `APPL`, all transmission files and no mapper runs, from the file `DECEDI_9505Archive.tar`, and requests that the utility provide informational messages as it works:

```
# decedi_retr -d "TEST*_I_*" -t "*" -m "" \
 -i DECEDI_9505Archive.tar -v

initializing retrieve environment...
calling Operating System retrieve shell script
/usr/sbin/decedi_retr_script...
document APPL-1_I_0000000289 retrieved
document APPL-1_I_0000000290 retrieved
document APPL-1_I_0000000291 retrieved
document APPL-4_I_0000000295 retrieved
document APPL-4_I_0000000296 retrieved
document APPL-4_I_0000000297 retrieved
6 documents retrieved
```

```
retrieving all transmissions...
100 transmissions retrieved
no maps requested for retrieval
```

Part III Problem Solving

This part of the Digital DEC/EDI User’s Guide describes what you need to do when things go wrong.

Chapter 14 Finding and Fixing Problems

This chapter provides information on “Problem Solving” for a running Digital DEC/EDI Server. This includes the following classes of problems:

- Documents and transmission files that fail or are stuck. See *Tracking Document and Transmission File Problems* on page 14-1.
- Errors appearing in the Error Log file. *Diagnosing Errors in the Error Log File* on page 14-15.
- Problems starting or stopping the server. See *Problems Starting or Stopping the Server* on page 14-17.

Tracking Document and Transmission File Problems

The descriptions in this section assumes you have used Cockpit as described in Chapter 10 *Monitoring*, and have identified one or more documents or transmission files that are marked as failed, or that are stuck within the Server.

For a document or transmission file that is marked as **FAILED**, you need to determine where it has failed, and then refer to the relevant section that follows, to determine the likely reason for the failure, and how to proceed.

For a document or transmission file that has become stuck, see *Stuck Documents and Transmission Files* on page 14-12.

You may find it helpful to refer to Appendix B *Document and Transmission File Status Flows* for information on the different statuses, including failure states, through which documents and transmission files flow within the Server.

Determining the Point of Failure

The Cockpit summary screen, and the document and transmission file screens list failed data as **`FAILED`**. To find where the document or transmission file has failed, you need to access the corresponding *Document Details* or *Transmission File Details* screens. These provide the information on where the failure occurred.

Read the notes provided in the following section that provide general notes on dealing with failures, then refer to the following table for the failure you have, and go to the indicated section for further details.

Table 14-1 Determining the Point of Failure

Direction	Displayed Failure Text	Go to Section
Outbound	Document failed during conversion	*Document Failed During Conversion* on page 14-4.
Outbound	Transmission file failed while building	*Transmission File Failed While Building* on page 14-5.
Outbound	Failed to send a transmission file	*Failed to Send a Transmission File* on page 14-5.
Outbound	Transmission file not delivered to trading partner	*Transmission File Not Delivered to Trading Partner* on page 14-8.
Outbound	Transmission file failed after delivery to trading partner	*Transmission File Failed After Delivery to Trading Partner* on page 14-8.
Inbound	Attempt to receive transmission file aborted	*Attempt to Receive Transmission File Aborted* on page 14-8.
Inbound	Unable to identify EDI syntax	*Transmission File Failed During Separation* on page 14-9
Inbound	Transmission file failed during separation	*Transmission File Failed During Separation* on page 14-9.

Table 14-1 (continued) Determining the Point of Failure

Direction	Displayed Failure Text	Go to Section
Inbound	Document failed during separation	*Document Failed During Separation* on page 14-10
Inbound	Document failed during translation	*Document Failed During Translation* on page 14-11
Inbound	Document fetch aborted	*Document Fetch Aborted* on page 14-12.

General Notes on Dealing With Failures

This section contains general advice on coping with failed documents and transmission files. It introduces the activities you normally have to perform in dealing with failures. More specific additional advice on each of the separate failure conditions is contained in following sections.

The general approach should be:

- Review the sources of information detailing the failure. This includes the Digital DEC/EDI Error Log File, described in *Diagnosing Errors in the Error Log File* on page 19-14.
- Determine if the problem is fixable.
- If the problem is fixable, then fix it and reset the data to be reprocessed.
- If the problem is not fixable, then cancel the data.

Reviewing Data

Where failures occur within the Digital DEC/EDI Server there are often separate error listing files produced specific to a document or transmission file that fails. Where these are produced, they contain important information relating to the failure that has occurred. Failures relating to a document may produce a *Document Error Listing* file. Similarly transmission file failures can produce a *Transmission File Error Listing*.

In addition to using Cockpit to look at audit trail information, you can also use Cockpit to review the contents of files within the Server. For documents, these files include the *Internal Format* file, the *External Format* file, the *Document Error Listing* file and the *Detailed Listing* file.

For transmission files these include the transmission file itself and the *Transmission File Error Listing* file.

While investigating document or transmission file failures on your system, you should normally expect to have to review the contents of one or more of these files. For example, if an outbound document fails during conversion, the Document Error Listing file might show that the document was missing a specified piece of mandatory data.

Resetting the Data

Documents and transmission files normally fail within the Server for one of two reasons:

- The data they contain is not correct or not complete (that is, does not match that expected, given the system configuration).
- The server configuration is not correct.

In many cases it is possible to fix the cause of the failure. In other cases it may not. In either case, the data remains in its failed state until you take explicit action to reset it. By resetting a document or transmission file, you can elect to cancel it, or to queue it for reprocessing. Cancelling data implies it is marked as **CANCELLED**, and removed from the Current Audit Trail by the Archiver. Electing to reprocess the data implies resetting the data back to a previous processing state and queuing it for reprocessing. (Appendix B *Document and Transmission File Status Flows* contains full information on which states reprocessed data is reset to). Unless you have fixed the reason for the original failure, the failure is likely to be repeated.

You use Cockpit to reset documents and transmission files.

Document Failed During Conversion

Outbound document failures during conversion are typically the result of a mismatch between the document data supplied within the Internal Format file and the definition of the EDI document to be produced (as created and viewed by using the EDI Tables Editor).

An Error Listing file is produced detailing the errors detected. An External Format file is also produced to the extent possible given the detected errors. A Detailed Listing file may exist if your configuration requested one. The Detailed Listing may provide more information about the context in which any errors have been detected.

You should expect to review the Error Listing file and probably the Internal Format and External Format files. You need to determine whether the data supplied in the document matches what you have configured. You need to establish whether the document data is at fault or whether the configuration is at fault. It is also possible that you may need to amend the relevant Mapping Table used to produce this document.

Where multiple failures are indicated in the Error Listing, the first error reported should be considered the most important. Errors reported after the first error may, in some cases, be consequent on the first error, and thus fixed automatically when the first error is fixed.

Transmission File Failed While Building

This failure status can be given to both documents and transmission files. If the problem is with a single document, the TFB fails just that document. However if the problem is with the transmission file the TFB fails the transmission file and all the documents within it.

Under normal conditions, this failure state is unlikely to occur. Most of the reasons why the TFB may fail documents or transmission files relate to the absence of one or more pieces of configuration data, or some problem with the EDI data itself. Since these have all been checked by earlier Server components, they are far less likely to cause a problem in the TFB, unless the configuration has been changed since the document entered the Server.

Note that if you are using TRADACOMS, there is no separate TFB. The role of TFB is carried out by the corresponding converter.

If a document has failed, use Cockpit to check whether there is an associated transmission file that is also failed. If there is one, first investigate the reasons why the transmission file failed.

Failed to Send a Transmission File

Failures of this sort are due to incorrect configuration of communications connection details (for an individual connection) or gateway parameters (for the gateway as a whole). This configuration is maintained by using the Communications Editor.

Where the gateway relies on other products to provide lower level communications services, such as is the case with the Pedi and OFTP

gateways, it is also possible that failures can be due to these other products being wrongly configured.

If a connection or gateway is disabled, the transmission file is not failed. It remains at **AWAIT TRANSMISSION**.

Failed to Send — Import/Export Gateway

The following are typical errors that can occur for when sending a transmission file by using the Import/Export gateway:

- The destination directory does not exist.
- The destination directory is inaccessible, or is write protected.
- The disk containing the destination directory has insufficient disk space to allow the transmission file to be copied to it.

Failed to Send — Pedi Gateway

For connections that use the Pedi gateway, typical problems are that:

- The connection configuration data is incorrect.
- The MTA configuration is incorrect and/or is incorrectly installed.
- The O/R name addressing information you have configured in both Digital DEC/EDI and X.500 are not exactly the same.

Chapter 15 *Problem Solving – Pedi Gateway* contains details, specific to the Pedi gateway, on additional logs and traces you can turn on to provide more information to assist problem solving.

Failed to Send — OFTP Gateway

For OFTP connections, if the gateways fails to send a transmission file, it tries again. Each time it retries, it increments a file retry counter for the transmission file. This means that any transient problems that can occur when using OFTP are automatically recovered, and no user intervention is required.

However, when the file retry counter reaches the file retry limit defined for the connection, the transmission file is marked as **FAILED**. Failures to send transmission files when using OFTP are normally due to one or more of the following:

- Configuration data for the connection and gateway being incorrect.
- Configuration of the X.25 software and related products being incorrect.
- The trading partners system being inactive — wait until you know the trading partner's system is active and reset the transmission file.

Chapter 16 *Problem Solving – OFTP Gateway* contains details, specific to the OFTP gateway, on additional logs and traces you can turn on to provide more information to assist problem solving.

Failed to Send — SMTP/MIME Gateway

For connections that use the SMTP/MIME gateway, typical problems that cause this state are:

- The file retry count has exceeded its maximum, due to:
  - sendmail cannot transfer data to the next hop, for instance because the receiving machine is down.
  - sendmail can't transfer data to the next hop, because it is the last hop (i.e. this is a direct point-to-point link) and it does not recognize the `To:` address.
  - The external processing hook failed.

Chapter 17 *Digital Problem Solving – SMTP /MIME Gateway* contains details, specific to the SMTP/MIME gateway, on additional logs and traces you can turn on to provide more information to assist problem solving.

Failed to Send — 3780 Gateway

For connections that use the 3780 gateway, typical problems that cause this state are:

- Reached maximum file retry count when attempting to send the transmission file. This can be caused by:
  - The TCL script has a syntax error after the transmission files were set to **`SENDING`**.
  - 3780Plus emulator had problems sending or receiving data.
  - Telephone number being dialled is busy.

Chapter 18 *Problem Solving – 3780 Gateway* contains details, specific to the 3780 Gateway, on additional logs and traces you can turn on to provide more information to assist problem solving.

Transmission File Not Delivered to Trading Partner

This failure state only applies to transmission files sent by using the Pedi gateway. It implies the gateway received a non-delivery report for the transmission file. That is, the MTA was unable to deliver the transmission file to the trading partner.

In the event of a non-delivery, an error is logged in the Error Log file indicating the transmission file name and the non-delivery report is saved in:

```
/var/adm/decedi/logs/transmission-file-name.REPORT
```

The non-delivery report explains why the delivery failed. This is almost always due to a misconfiguration of the Digital DEC/EDI connection data, or the corresponding information in the MTA or X.500. After you have corrected the configuration error, you can use Cockpit to reset the transmission file to be resent.

Transmission File Failed After Delivery to Trading Partner

This failure state applies only to transmission files sent by using the Pedi gateway, and for which Pedi enveloping (as opposed to P0 or P2 enveloping) was used. It implies the gateway has received a negative EDI Notification message for the transmission.

This means your trading partners EDI system has declined to accept responsibility for the data in the transmission file you sent. An error is reported in the Error Log file and the audit trail for the transmission file contains the reported reason for the transmission being rejected. You should discuss with your trading partner how you want to handle this failure.

Attempt to Receive Transmission File Aborted

This failure indicates a serious problem with the gateway unable to create the transmission file, possibly due to lack of disk space, or due to an inability to add new records to the audit trail. The errors are unlikely to be due to the particular transmission file.

The Error Log contains appropriate errors explaining the cause. You need to address these errors, and ask your trading partner to resend the transmission.

Transmission File Failed During Separation

The Transmission File Splitter is unable to split the entire contents of a transmission file into documents. If the Splitter cannot process the entire transmission file correctly, it marks as failed, both the transmission file, and any documents it has produced from that transmission file. Examine the Transmission File Error Listing to see the reasons for the failure.

Note that if you are using TRADACOMS or X12, there is no separate Transmission File Splitter. The splitting functions are carried out by the corresponding Translators.

No Matching Trading Partner Agreement

A very common reason for failures during separation is that the trading partner agreement information, that is contained within the transmission file, does not match the configuration contained in the profile cache. The error reported is *failed to find TP agreement*.

You have to change the trading partner agreement information contained in the profile cache or cancel this transmission file.

Note that the profile cache has to be built explicitly from the data you have configured. Do this by using the Trading Partner editor. If you have elected not to rebuild AND reload the cache after you have made changes, or additions to the configuration, the Server components may be using data that is different from that which you can see by using the CommandCenter editors. If you suspect this may be the case, you can use the Profile Cache Verification tool. This tool provides a listing of the contents of the in-memory copy of the profile cache being used by a Server once it has been started. See man page help for *decedi_pcv(8)* for more details on this tool.

Incorrect or Missing EDI Standard/Version

The trading partner information may reference a version of an EDI Standard that is not installed on your Server. The Splitter then reports problems in trying to interpret the data in the transmission file.

If you want to process the data, you need to either install the correct version of the EDI Standard, or change the profile information to reference the correct version (and rebuild and reload the profile cache).

Bad Transmission File Data

Transmission files that contain EDI data that do not conform to the syntax rules implied by the selected EDI standard, cause the Splitter to fail the transmission file. It is possible for the data to be either incorrectly formatted (perhaps a minor error) or otherwise corrupted (perhaps due to a communications error).

If the data is bad, you have to cancel the transmission file and request the trading partner to resend the data.

Constituent Document Error

For X12/TDCC if any document within the document fails to be translated then the entire transmission is marked as failed. This is so the transmission file can be reprocessed once the problem has been corrected, and the appropriate functional acknowledgement generated, if applicable. Any documents within the transmission which were successfully translated will still proceed to be made available to the fetching application.

Document Failed During Separation

For X12, the process of separation also includes that of translation, where the process of translation is similar to the outbound conversion, in terms of the errors that can cause a document to be failed.

Inbound document failures during separation are typically a result of a mismatch between the data contained within the transmission file and the definition of the EDI document (as created and viewed by using the EDI Tables Editor).

An Error Listing File is produced detailing the errors detected. An Internal format file is produced to the extent possible, given the detected errors. A detailed listing file may exist if your configuration requests one. The detailed listing may provided more information about the context in which any errors have been detected.

You should expect to review the Error Listing and probably the Internal Format and Transmission file. You need to determine whether the data supplied in the document matches what you have configured. You need to establish whether the document data is at fault or whether the configuration is at fault.

Where multiple failures are indicated in the Error Listing, the first error reported should be considered the most important. Errors reported after the first error may, in some cases, be consequent on the first error, and thus fixed automatically when the first error is fixed.

EDIFACT CONTRL Message

The CONTRL message returned after a transmission can report success, failure or partial failure of the transmission as required. It is set up with the CommandCenter Trading Partner Editor (see its On-line Help for further details).

Document Failed During Translation

The process of translation is similar to outbound conversion, in terms of the likely errors that may cause a document to be failed.

Inbound document failures during translation are typically the result of a mismatch between the data contained within the External Format file and the definition of the EDI document (as created and viewed by using the EDI Tables Editor).

An Error Listing file is produced detailing the errors detected. An Internal Format file is produced to the extent possible, given the detected errors. A Detailed Listing file may exist if your configuration requests one. The Detailed Listing may provide more information about the context in which any errors have been detected.

You should expect to review the Error Listing and probably the Internal Format and External Format files. You need to determine whether the data supplied in the document matches what you have configured. You need to establish whether the document data is at fault or whether the configuration is at fault.

Where multiple failures are indicated in the Error Listing, the first error reported should be considered the most important. Errors reported after the first error may, in some cases, be consequent on the first error, and thus fixed automatically when the first error is fixed.

Document Fetch Aborted

It is possible for inbound documents to be failed by the Mapper. Such documents had a status of **`AVAILABLE`**, before the Application Client `fetch` command caused the Mapper to fetch the document or documents. During the mapping process, the Mapper could not correctly process the document and, in consequence, sets the offending document to **`FAILED`**.

It is possible that the Mapper audit trail may contain sufficient information about the failure to allow you to locate and fix the problem. Where this is not the case, you need to refer to the Mapper debug log file. The debug log file contains detailed information about the mapping run, along with any errors that occurred. Such a log file is not produced by default, but can be requested by using the `-debug` option on the `fetch` command line:

```
# trade fetch ..... -debug=filename.log ...
```

If a debug log was not requested at the time the document was failed, reset the document and reissue the `fetch` command, adding the debug option.

Stuck Documents and Transmission Files

The term *stuck* is used to refer to those documents and transmission files that have been within the Server for longer than a specified time; longer than can be explained by the Server trying to process a backlog of data. Reasons

why documents and transmission files may be stuck at particular states are described in the following table.

Table 14-2 Stuck Documents and Transmission Files

Status	Explanation
FAILED	The document or transmission file is waiting for you to decide whether to cancel or reprocess it. See the earlier sections in this chapter for more information on dealing with individual failures.
QUEUED	If the document(s) is part of a batch, the TFB is waiting until all documents that constitute the batch are available to it. Check for any documents from the batch that may have failed during conversion. Use the Management Services Editor to check when the TFB is next scheduled to run. It is possible the build interval may not yet have completed. If the document(s) are not part of a batch and the build interval should have completed by now then it may be possible that the Memory Queues have become unsynchronized. In that event, follow the steps listed in the next section.
AWAIT TRANSMIS-SION	The gateway or connection may be disabled, or does not exist. Use the Communications Editor to check. Enable the gateway or connection and use the editor to issue a Start Connection command.
SENT	An OFTP transmission file for which no EERP has been received. Check with the trading partner, to see if the transmission file has been received. Either wait until the EERP has been received, or cancel the transmission file by using Cockpit. A Pedi/X.400 transmission file for which no delivery notification has been received. Check with the trading partner, to see if the transmission file has been received. Check with the MTA administrator to determine the state of the message. Either wait for the notification to arrive, or cancel the transmission file by using Cockpit.

Table 14-2 (continued) Stuck Documents and Transmission Files

Status	Explanation
DELIVERED	A Pedi transmission file, that is using Pedi enveloping, is waiting for receipt of an EDI notification message. Check to see if your trading partner has generated an EDI notification. Either wait for the notification to arrive or cancel the transmission file by using Cockpit.
PURGEABLE	Outbound X12/TDCC documents may remain in a **PURGEABLE** state in the current audit trail until such time as the functional acknowledgement they are expecting arrives. Check how long ago the document got to **PURGEABLE** by reviewing its details using the Cockpit. If the functional acknowledgement is overdue then contact your Trading Partner to find out why he has not sent it. If the functional acknowledgement will not be received then cancel the expectation of a functional acknowledgement using the Cockpit to reset the document.
AVAILABLE	An inbound document is waiting to be fetched by an application.
TRANS AVAILABLE	An inbound transmission file, that has been routed by using Translator Bypass Routing, is waiting to be fetched by an application.

For documents or transmission files in other states, you may need to restart the Server. This allows the recovery processing on startup to recover the data back to a state where it can be reprocessed.

Regenerating the Memory Queues”

DEC/EDI uses a series of memory resident queues to determine what work is required of it. You can see these queues by looking at the Memory Queue Monitor in the PC based Cockpit.

Typically, the number of entries you see in the Memory Queue Monitor should equal the number of documents and transmission files that you see in

the summary screen of the Cockpit although there may be more. If there are less then this is an indication that you need to rebuild the Memory Queues.

To rebuild the Memory Queues, shutdown DEC/EDI on all systems of the current environment. The last system to shutdown will save the Memory Queue contents to a file called /var/adm/decedi/memoryQueuesSaveFile.do_not_delete. Despite its name, delete this file and restart DEC/EDI. When the DEC/EDI startup procedure sees that this file is missing, it will go back to the database and begin rebuilding the Memory Queues from the contents of the database thus effectively synchronizing the database and the Memory Queues.

Diagnosing Errors in the Error Log File

If any errors are logged in the Error Log file, you should investigate why they were logged and take any necessary corrective action and any preventative measures to ensure the errors will not occur again.

If no errors or messages are output to the Error Log, check that the Error Log exists and that the setting, if any, of the environment variable DECEDI_LOG_SEVERITY does not prevent messages being logged. See *Appendix D: Environment Variables* for more details of this environment variable. Check also that there is some free disk space on the disk that contains the Error Log File.

Errors that can occur are described in the Error Messages Help Library. This is a Windows help file, that is installed with Cockpit and CommandCenter.

You can access the file from Cockpit when you have the error displayed in the Error Log view. Double clicking the error displays information about that error. If you do not have the error displayed within the Error Log view, you have to access the help library directly. Do this by running the help library, for example, from File Manager.

Oracle 8i only

Some errors logged by Digital DEC/EDI are in fact errors returned by the database product, as in the following example of an Oracle 8i error message:

```
Wed Nov 20 13:59:55 1996 PID = 11958 NAME = DECEDI_MUS
DECEDI__TRACEMOD (i), calling module =
DECEDI__DB_CONNECT_DATABASE
DECEDI__SQLERROR (e), SQL error : SQLCODE = -1034 : ORA-01034:
ORACLE not availa
```

```
ble
ORA-07429: smsgsg: shmget() failed to get segment.
DEC OSF/1 (AXP) Error: 2: No such file or directory
```

To find out more details of what the error means (module ORA, error 7429, in this case) use the Oracle 8i `oerr` utility as in the following example:

```
# $ORACLE_HOME/bin/oerr ORA 7429
07429, 00000, "smsgsg: shmget() failed to get segment."

// *Cause:  A shared memory segment used for all part of //
the SGA could not be retrieved.                    // *Action:
Use the system error number in the error    //         message
to determine why the segment could not be //
retrieved. If it does not exist, shutdown the //
database using the "abort" option, and then restart //
it.                                                  //
If the get failed because the permissions are //
incorrect, make sure that the ownership of the

   //          oracle executable is the same as that on the
shared //           memory segment.
```

Database Full Errors

See *If you are using Oracle8, see Changing the Size of Your Oracle8 Database.* on page 11-10 for information on increasing the size of the database.

Problems Starting or Stopping the Server

This section describes problems that can arise when starting or stopping the Server.

Diagnosing Problems With Starting the Server

This section describes how the Server is started and the various problems that can arise when starting the Server.

How the Server is Started

The startup procedure, initiated by using decedi_start, takes place in three main stages. They are:

1. Verification Stage

   The startup procedure first verifies that the environment can support DEC/EDI. The tasks involved in this are as follow…

   Ensure that DEC/EDI is not already running on this system.

   Execution of the decedi_sysetup, decedi_systart and decedi_systart_<system-name> where <system-name> is the name of the current system as shown by the 'hostname –s' UNIX command.

   Verifies that the Oracle 8i (or later) database is available for DEC/EDI's use and that the database is at the correct revision level to support DEC/EDI.

   Verifies that the Memory Queues can be created or accessed either through Shared Memory sections or through the Memory Channel facility in a cluster configuration.

2. Environment Setup Stage

   The startup procedure then waits for the 'startup lock'. In a clustered environment this lock prevent multiple concurrent startups or shutdowns of DEC/EDI.

   Connects to the DEC/EDI database.

   Evaluates whether DEC/EDI is being started as a member of a clustered environment.

   Attaches to the Memory Queues.

   If this is the first instance of a server in the DEC/EDI environment then the startup procedure looks for the Memory Queues save file

/var/adm/decedi/memoryQueuesSaveFile.do_not_delete. If this file is found then the procedure loads the Memory Queues with its contents. If the file is not found then the procedure will interrogate the DEC/EDI database to rebuild the Memory Queues.

If this is not the first instance of a server in a TruCluster configured DEC/EDI environment then the startup procedure simply joins the existing environment and begins using the Memory Queues that have been maintained by Server instances running on other members of the cluster.

3. Process Startup Stage

   Finally, the procedure starts the relevant background processes that you have requested through the CommandCenter's Management Services Editor and through the '_START' environment variables in the /usr/sbin/decedi_sysetup file.

FINDING AND FIXING PROBLEMS

Example of Starting the Server

The following is an example of the output that the startup procedure might produce, in this case with a TruCluster configured system with DEC/EDI already running on other systems in the cluster that starts 3 EDIFACT Translation Services components and the Import/Export, OFTP and Pedi Communications Services:

```
# decedi_start
    Checking if DEC/EDI is already running...
    Running decedi_setup...
    Running decedi_systart...
    Verifying Oracle database access...
    Initiating startup sequence...
    Waiting for startup lock...
    Connecting to the database...
    Looking for active server instances...
        Server instances found; joining existing environment.
    Attaching to memory section...
    Scanning for services to start...
    Starting server processes...
        Starting 1 instance of the Archive Server
        Starting 1 instance of the Port Server
        Starting 1 instance of the Communications Controller
        Starting 3 instances of the EDIFACT TFB
        Starting 3 instances of the EDIFACT Converter
```

```
        Starting 3 instances of the EDIFACT Translator
        Starting 3 instances of the EDIFACT TFS
        Starting 1 instance of the Import/Export Gateway
        Starting 1 instance of the PEDI/X.400 Gateway
        Starting 1 instance of the OFTP Gateway
        Total of 18 processes started.
    Waiting 5 seconds for components to start.....
    Finished.
```

The following is an example of the output that the startup procedure might produce, in this case with a single system with DEC/EDI that starts 2 X12 Translation Services components and the Import/Export, SMTP and 3780 Communications Services:

```
# decedi_start
    Checking if DEC/EDI is already running...
    Running decedi_setup...
    Running decedi_systart...
    Verifying Oracle database access...
    Initiating startup sequence...
    Waiting for startup lock...
    Connecting to the database...
    Looking for active server instances...
        No server instances found.
    Attaching to memory section...
    Initializing memory section...
        Save file found. Loading...
        8 entries loaded. Renaming save file...
        Done.
    Scanning for services to start...
    Starting server processes...
        Starting 1 instance of the Archive Server
        Starting 1 instance of the Port Server
        Starting 1 instance of the Communications Controller
        Starting 2 instance of the X12 TFB
        Starting 2 instance of the X12 Converter
        Starting 2 instance of the X12 Translator
        Starting 1 instance of the Import/Export Gateway
        Starting 1 instance of the 3780 Gateway
        Starting 1 instance of the SMTP Gateway
        Total of 12 processes started.
    Waiting 5 seconds for components to start.....
    Finished.
```

In addition, and provided that the setting of the DECEDI_LOG_SEVERITY environment variable does not prevent it, the Error Log contains

informational messages logged by each component as it first starts, for example:.

```
Tue May 30 15:07:06 2001 PID = 12240 NAME = EDIFACT Translator
DECEDI__STARTING (i), startup under way
```

An additional message is logged when a process has completed its own startup processing:

```
Tue May 30 15:07:25 2001 PID = 12240 NAME = EDIFACT Translator
DECEDI__STARTED (i), startup completed
```

Problems During Stages 1 and 2

It is during this phase that most problems arise. Typically these will involve database connection issues or memory section connection issues.

If you see database connection issues then contact your DBA and read the error message displayed by the startup procedure. These are usually related to the fact that the database has not been started or that the Oracle TNS service is either incorrectly configured or has not been started. If you do not have an Oracle DBA then please refer to the Oracle documentation for further details on the error message displayed.

If you see Memory Section connection errors then DEC/EDI will display the name of the system procedure that generated the error along with the appropriate error message. The help (man) pages on UNIX for the system procedure shown (usually shm_open on single systems and imc_api_init on a TruCluster Server configured system) will list the possible error messages returned and what they mean.

Problems During Stage 3

As each process is created it logs an informational (`DECEDI__STARTING`) message in the Error Log. The process then performs a series of checks on its environment and complete its initialization before logging the `DECEDI__STARTED` message.

You can check to see whether the Server processes have been created as in the following example. This lists the Digital DEC/EDI Server processes running on the system. (See Appendix E Files and Processes Used by Compaq DEC/EDI for more information on the names of the separate Digital DEC/EDI processes). You should see something similar to the following:

```
# ps -e | grep 'decedi' | grep –v grep
1283521 pts/3    S        0:00.11 decedi_cstd 0
1283522 pts/3    S        0:00.28 /usr/sbin/decedi_smtpd 1
1284251 pts/3    S        0:00.38 /usr/sbin/decedi_3780d 1
1285097 pts/3    S        0:00.28 /usr/sbin/decedi_xtrnd 1
1285644 pts/3    S        0:00.23 /usr/sbin/decedi_ccid 1
1285660 pts/3    S        0:00.48 decedi_csfd 0
1285674 pts/3    S        0:00.28 /usr/sbin/decedi_xcnvd 1
1285715 pts/3    S        0:00.30 /usr/sbin/decedi_impexpd 1
1285832 pts/3    S        0:00.15 /usr/sbin/decedi_psd 1
1285846 pts/3    S        0:00.17 /usr/sbin/decedi_asd 1
1285860 pts/3    S        0:00.14 decedi_csgd 0
1285906 pts/3    S        0:00.24 /usr/sbin/decedi_xtfbd 1
1285912 pts/3    S        0:00.24 /usr/sbin/decedi_tcnvd 1
```

Processes may log errors in the Digital DEC/EDI Error Log indicating one or more of the problems listed below.

- When checking for LMF licenses, no valid license was found. You have to obtain and register a valid license before you can continue.
- No store directories can be found. You must create at least one store directory. See *What is a Store Directory ?* on page 10-16 and *Creating and Moving Store Directories* on page 11-8 for more information.
- One of more of the pre-requisite products for that process is missing, not configured or not started. Refer back to earlier sections of this guide for instructions on what other products you need to have installed, and how they need to be configured.
- The process cannot connect to the database. This is unlikely, since stage 1 accessed the database but if your system has run out of resources then the process may be unable to connect. Also check the access protection on the database to make sure it is owned and accessible to your Oracle installation account.

If no errors or messages are output to the Error Log, check that the Error Log exists and that the setting, if any, of the environment variable DECEDI_LOG_SEVERITY does not prevent messages being logged. See Appendix D Environment Variables for more details of this environment variable.

Check also that there is some free disk space on the disk that contains the Error Log File.

The final thing to check is the process output file in /var/adm/decedi/logs/*.out. These files contain informational messages regarding cache update detection but may also contain information on why a processes failed to start.

If startup didn't create all the processes you thought should have been created, for example, no OFTP gateway was started, check that you have requested it to be started. You use the Management Services Editor to specify which Translation and Communications Services components are to be started. If you have not asked for, in this case, the OFTP gateway to be started, decedi_start does not start it, even if you have defined connections for it by using the Communications Editor.

If you see messages in the Error Log similar to the following, this means that while you have used the Management Services Editor to request a particular gateway to be started, you have not also used the Communications Editor to define gateway parameters for the gateway.

```
Thu Jul 13 10:28:09 1995 PID = 10544 NAME = Import/Export
Gateway
DECEDI__STARTFAILED (e), startup failed
DECEDI__ERRRPAR (e), error reading communications parameter
record IMPEXP
DECEDI__USEEDITPAR (e), please use EDIT PARAMETER first
DECEDI__RNF (e), record not found
```

The gateway fails to start. You need to use the Communications Editor to define gateway parameters for the gateway and restart the Server.

Diagnosing Problems With Stopping the Server

This section describes how the Server is stopped and the various problems that can arise when stopping the Server.

How the Server is Stopped

Shutdown is divided into 3 stages also.

1. Stage 1 allows the shutdown procedure to join the DEC/EDI environment as a functional member. This prevents the Memory Queues memory section from being disbanded when the other processes on the system shuts down so that stage 3 can be performed.
2. Stage 2 sends a shutdown request to all the processes on the system. Most processes will respond immediately and begin their shutdown

sequence. Some processes may be busy doing work (for example, sending or receiving transmissions or translating incoming files). After the shutdown procedure sends shutdown requests to the processes, it keeps a careful eye on who responded and who did not.

Processes that are accumulating CPU time are assumed to be 'busy' and should be left alone to complete their work before the shutdown procedure continues. The shutdown procedure will wait for them to finish before continuing.

Processes that are NOT accumulating CPU time are sent reminders after 1 and 2 minutes in case they missed the first request. If after 3 minutes they still have not responded or accumulated any CPU time and DEC/EDI cannot find a valid reason for them to be 'stalled' then DEC/EDI will terminate the process under the assumption that some resource problems are causing it to become unresponsive.

Note, however, that none of the Communications Gateways are terminated in this manner as they may be waiting for child processes to complete their operations before exiting. If you believe that your Communications Gateway has become unresponsive due to other circumstances, you should take action to clear the process from the system and allow the shutdown procedure to continue.

3. The final stage of the shutdown procedure looks to see if this is the last instance of the server running in the current environment. On single system configurations, this will always be true. In a TruCluster Server configured environment, it will be true only if no other members of the cluster are running DEC/EDI.

   If this is the last instance of the server running then the shutdown procedure will save a copy of the Memory Queues into the save file /var/adm/decedi/memoryQueuesSaveFile.do_not_delete.

Example of Stopping the Server

The following is an example of the output that the shut down procedure might produce on a single system environment or the last instance in a cluster.

```
#decedi_stop
    Initiating shutdown sequence...
    Waiting for shutdown lock...
    Connecting to the database...
    Attaching to memory section...
```

```
    Shutting down DEC/EDI Server processes...
        20 processes to shutdown; sending requests...
    Looking for active server instances...
        No other server instances found; saving Memory
        Queues...
        Saved as
        /var/adm/decedi/memoryQueuesSaveFile.do_not_delete
Removing shared memory IDs
```

FINDING AND FIXING PROBLEMS

The most likely problem that you will come across is either an unresponsive or ‘stuck’ process. These fall into 3 categories; Communications Gateways, Client Servers and general processes.

For Communications Gateways, the shutdown procedure will just list for you the processes that it is waiting on. For processes that use system commands to do their work like the Import/Export gateway with pre-import or post-export options defined or the SMTP gateway or the 3780 gateway, the process may be waiting on a script to complete.

For all of these Gateway processes, you can find out if they have child processes active by using the PS command. For example, if the shutdown procedure reports the following message…

```
    Import/Export Gateway process (1289306) has been stalled
for 5 minutes
```

then you can take the process identifier listed in the parentheses and search the system to see if it has any child processes as follows…

```
# ps -ef | grep 1289306 | grep -v grep
root 1289306 1048577  0.0 16:17:16 pts/3 0:00.32
/usr/sbin/decedi_impexpd 1
root 1289773 1289306  0.0 16:17:16 pts/3 0:07.12
/usr/local/bin/post_export
```

The first line shows the DEC/EDI process that the shutdown procedure is waiting for (the 1289306 process identifier is in the second column) and the second line shows that the it has a child process (the 1289306 process identifier is in the third column) that it is waiting for. This process may be in the middle of copying files around so you can just leave it until it finishes. However, if you look at what the post_export job is doing and you believe it to be stuck (for example, in a never-ending loop or waiting for a disk to become available) then you should kill the CHILD process (process id 1289773 in the above example) which will free the Import/Export Gateway to shutdown cleanly.

The Client Servers (the Post/Fetch Server, the Track Server and the CommandCenter Server) may become stalled after network problems with a remote client. If this happens, Server may actually gain small amounts of CPU time while it attempts ‘keep alive’ messages to the client so the shutdown procedure will see it sometimes as busy and sometimes as stalled. If you can confirm that nobody is trying to use the CommandCenter or that there are no remote trade post, fetch or track commands running then you may need to kill this process.

General processes typically will either be busy or will respond immediately to the shutdown request. However, there may be circumstances where a process will hang. For example, if the Oracle database encounters problems this can cause the DEC/EDI processes to wait until the Oracle problems have been resolved. Typically, DEC/EDI will detect these processes and shut them down itself but, as with the Client Servers, the process may ‘creep’ CPU time in which case you will need to kill them manually.

Chapter 15 Problem Solving – Pedi Gateway

This chapter provides additional problem solving information and testing tips specific to the Pedi gateway.

Enabling MTA Message History Logging

If, during testing, you have problems with message delivery, you may find it useful to turn on the message history logging within your MAILbus MTA. To do this, use the following ncl command:

```
ncl> set mta message history state on
```

You can then use ncl commands to see details of what is logged. To see all logged history, for example, you use the command:

```
ncl> show mta all
```

There is an MTA performance degradation when you turn on logging, and the logging also uses up disk space. Remember to turn off the logging when you have finished your testing.

Using Pedi Gateway Debug Environment Variables

By using the Pedi gateway debug environment variable `DECEDI_PEDI_KEEP_MSGFILE`, you can make Digital DEC/EDI keep all the following:

- Outbound message content, before it is encoded into ASN.1. The file that is written is:

  `/var/adm/decedi/logs/transmission-file-name.MESSAGE`

- Inbound message content, after decoding from ASN.1. The file that is written is:

  `/var/adm/decedi/logs/transmission-file-name.MESSAGE`

- All MTA delivery reports. (Delivery reports that indicate non-delivery are saved by default regardless of the setting of the above environment variable). Delivery reports are saved as:

  `/var/adm/decedi/logs/transmission-file-name.REPORT`

Add the following line to `/usr/sbin/decedi_sysetup` to turn on this logging:

```
# DECEDI_PEDI_KEEP_MSGFILE=1
```

You can also save copies of all outbound and inbound EDI notifications. To do this, you put the following line in `/usr/sbin/decedi_sysetup`:

```
# DECEDI_PEDI_KEEP_EDIN=1
```

The notifications are each saved as:

```
/var/adm/decedi/logs/transmission-file-name.NOTIFICATION
```

Remember to delete old files from your `/var/adm/decedi/logs` directory if you use these environment variables.

Decoding the ASN.1 Message

It may be useful to see the format of the ASN.1 encoded message (envelope and/or content). To decode the ASN.1 message and make it readable, use the MTA decoder tool provided with the MAILbus 400 kit,as in the following example:

```
#/usr/sbin/mta/mta_decoder -f \
     /var/mta/bad_msgs/C6AD7BDF17C1CE11801908002BBBC9EF -c
```

For more information about this tool see *MAILbus 400 Tuning and Problem Solving Guide*.

Chapter 16 Problem Solving – OFTP Gateway

This chapter describes the general problem solving techniques that you should use to detect and solve problems with the OFTP gateway.

Using the Cockpit

The Digital DEC/EDI Cockpit can be used to display information about all OFTP transmission files. By using the Cockpit, it is possible to determine the following:

- The current status of the transmission file.
- The number of attempts to send or receive the file.

A transmission file being processed by the OFTP gateway can be in any one of the states described in the following table.

Table 16-1 Statuses for OFTP Transmission Files

Status	Meaning
AWAIT TRANSMISSION	The file is ready to be sent to the trading partner. It is sent the next time a connection is made to, or a connection is received from the trading partner. You can manually force a connection to be made by using the Start Connection function of the Communications Editor. An attempt may have already been made to send the file. The retry count tells you how many attempts there have been to send this file. If the retry count is non-zero then you need to look in the error log file for determine why the previous attempt failed.
SENDING	The file is currently being sent to the trading partner.
SENT	The file has been sent to the trading partner, but an End-to-End Response (EERP) from the trading partner is still outstanding.
DELIVERED	An End-to-End Response (EERP) has been received from the trading partner.
PURGEABLE	The transmission file has been processed and will be archived.
RECEIVING	The transmission file is currently being received from the trading partner.

Table 16-1 (continued) Statuses for OFTP Transmission Files

Status	Meaning
RECEIVED	The transmission file has been received from the trading partner.
PARTRCV	An attempt has been made to receive the file, but due to an error (see the error log file) the attempt was aborted. The file is retried the next time a connection is made to or a connection is received from the trading partner. You can manually force a connection to be made by using the Start Connection function of the Communications Editor. The retry count tells you how many attempts there have been to receive this file.
FAILED (Failed to send)	The number of attempts to send the file have exceeded the file retry limit specified in the connection details.

Using the Error Log File

All OFTP errors are reported to the Digital DEC/EDI Error Log file. Examining the Error Log file is the first step in diagnosing OFTP problems. Descriptions for the OFTP error messages can be obtained by using the online Error Messages Help Library supplied with Cockpit. In most cases the descriptions for these error messages should be sufficient to understand the error and to determine the actions required to fix the problem.

Enabling OFTP Trace for More Information

If the error message descriptions do not provide you with sufficient information to solve the problem, then activating the OFTP trace for the required connections might provide useful additional data.

The OFTP gateway provides a trace facility. This trace facility can be switched on or off on a per connection basis by using the `Trace Protocol`

flag. Use the Communications editor of Digital DEC/EDI CommandCenter to enable trace for all required connections, by setting the flag appropriately.

When submitting problem reports to Digital support centers, providing a trace file with the problem report would assist Digital to solve the problem.

Reading an OFTP Trace File

An OFTP trace file contains various types of records. These are described in the following table.

Table 16-2 OFTP Trace File Record Types

Record Type	Description
X.25 Level Trace	Trace records starting with `X25:` contain the X.25 level trace. These lines trace the events occuring at the X.25 level.
Message Level Trace	Trace records starting with `MSG:` contain the trace of all OFTP messages sent and received. Messages Sent are denoted by `MSG: <--- IN` and messages sent are denoted by `MSG: ---> OUT`.
Data Messages	OFTP Data messages have a format of their own. An OFTP data exchange buffer (message) is split into sub-records. The OFTP trace displays the contents of each sub-record. A Sub-record line starts with the text `SR (nnn)` where `nnn` denotes the number of bytes in that sub-record. There can be a maximum of 63 data bytes in the sub-record. Following the `SR (nnn)` text there may optionally be an `EOR` string. This indicates that this sub-record is the last piece of an OFTP logical record (eg. a transmission file record). Following this is the actual data displayed in ASCII where possible. Data bytes that cannot be displayed in ASCII are prefixed by a ~ character and its value is then displayed as a pair of hex digits. For example, `~0d` denotes a Carriage Return character.
Status Change Trace	Trace records starting with `STC:` trace the state changes of the OFTP protocol engine.

Table 16-2 (continued) OFTP Trace File Record Types

Record Type	Description
File Status Changes	Trace records starting with `FSU:` trace the status changes to transmission files on the communications audit trail
Error Log Trace	Trace records starting with `LOG:` trace all the messages that the OFTP gateway writes to the Digital DEC/EDI error log file. Having them displayed in here is useful, because it not only displays the error messages but also indicates the exact point at which the messages were logged. Note that the OFTP trace file does not contain all messages that appear in the Digital DEC/EDI Error Log file. When using OFTP trace, it is still wise to inspect the Digital DEC/EDI Error Log file for additional error messages. Error messages that do not appear in the OFTP trace file include messages generated by underlying software such as the database access routines.
General Trace	All other records, not falling into the above categories are used to trace general OFTP gateway processing, such as the event that reads a connection record, or an event that reads/writes data from/to a transmission file.

Appendix E lists some sample OFTP trace files.

OFTP Trace Example 1

The following is an example of how OFTP trace can be used to solve a problem. Suppose the following error message was logged in the Digital DEC/EDI error log file (as viewed by using `decedi_look`):

```
Wed May 31 12:31:16 1995 PID = 30794 NAME = OFTP Server 30794
DECEDI__ESIDTRANS (w), end session request transmitted
DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for SSID
DECEDI__CONNID (i), connection id = OUT
DECEDI__ESIDR04 (i), invalid password
DECEDI__SSIDNEGFAIL (w), SSID start session negotiation failed
```

You could ascertain from the error message that the problem is with the received remote OFTP password. If OFTP trace was enabled you would see the following in the OFTP trace file in the directory `/var/adm/decedi/logs`:

```
MSG: <--- IN
SSID - Start Session ID
SSIDCMD  (Command)                 = X
SSIDLEV  (Protocol Version Level)  = 1
SSIDCODE (Initiators ID code)      = IN-ID
SSIDPSWD (Initiators Password)     = INBOUNDD
SSIDSDEB (Exchange Buffer Size)    = 02048
SSIDSR   ((S)end (R)eceive (B)oth) = R
SSIDCMPR (Compression)             = Y
SSIDREST (Restarts)                = Y
SSIDSPEC (Special Logic)           = N
SSIDCRED (Exchange Buffer Credits) = 003
SSIDRSV1 (Reserved)                = RESV
SSIDUSER (User Field)              = USER
SSIDCR   (Carriage Return)

Received SSID with incorrect Password...
- received INBOUNDD
- expected INBOUND
LOG: DECEDI__SSIDNEGFAIL (w), SSID start session negotiation
failed
LOG: DECEDI__ESIDR04 (i), invalid password
LOG: DECEDI__CONNID (i), connection id = OUT
LOG: DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for
SSID
LOG: DECEDI__ESIDTRANS (w), end session request transmitted

MSG: ---> OUT
ESID - End Session ID
ESIDCMD (Command)      = F
ESIDREAS (Reason Code) = 04 (Invalid Password)
```

The trace file shows that an SSID message has just been received from the remote OFTP system and that the remote password is incorrect. It also displays that the password received, `INBOUNDD` does not match what was expected, `INBOUND`.

Immediately after this the gateway transmits an ESID (End Session) request with a reason code of 04 (Invalid Password).

At this stage you would either notify the remote OFTP partner that they are sending the wrong password (and what it is) or modify the password at your local end to match what is received.

OFTP Trace Example 2

Suppose the following OFTP error messages were logged in the Digital DEC/EDI Error Log file:

```
Wed May 31 13:13:57 1995 PID = 30026 NAME = OFTP Server 30026
DECEDI__UNEXPDATA (e), unexpected data received
DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for SSRM
DECEDI__CONNID (i), connection id = OUT2

Wed May 31 13:13:57 1995 PID = 30026 NAME = OFTP Server 30026
DECEDI__ESIDTRANS (w), end session request transmitted
DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for SSRM
DECEDI__CONNID (i), connection id = OUT2
DECEDI__ESIDR01 (i), command not recognized

Wed May 31 13:13:57 1995 PID = 30026 NAME = OFTP Server 30026
DECEDI__OFTPERROR (e), an OFTP error has occured, aborting
session
DECEDI__OFTPSTATE (i), OFTP state = Waiting for X.25 call to be
cleared
DECEDI__CONNID (i), connection id = OUT2
```

You could ascertain from the error message that the problem is with the message received from the remote OFTP partner. The message we were expecting was a SSRM (Start Session Ready Message) message, but what we received was not recognized, hence the reason for sending an ESID (End Session Request) with a reason code of 01.

If OFTP trace was enabled, you would see the following in the OFTP trace file in the directory `/var/adm/decedi/logs`:

```
Successfully read connection record by connection id OUT2
Successfully locked connection record
X25: opened X.25 port
X25: encoded parameters...
DTE Class       = X25-OUT
DTE Address     = 99999999
DTE Sub-Address = 99
User Data (hex) =  1 0 0 0
X25: encoded parameter list for making call
X25: X.25 call established using template
X25: got X.25 port status
```

```
port status = X25S_RUNNING
STC: State Changed to I_WF_RM (Initiator waiting for SSRM)
X25: waited for an X.25 event...
- X25EV_DATA event triggered
X25: read 47 byte(s) of data
LOG: DECEDI__CONNID (i), connection id = OUT2
LOG: DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for
SSRM
LOG: DECEDI__UNEXPDATA (e), unexpected data received
Unexpected Data Received, Data Follows...

MSG: <--- IN
*** message not recognized ***
~0d~0d~0a~0d~0d~0a (nodeab.xyz.abc.com)
ttyp5~0d~0d~0a~0d~0d~0a~0d
login:
LOG: DECEDI__ESIDR01 (i), command not recognized
LOG: DECEDI__CONNID (i), connection id = OUT2
LOG: DECEDI__OFTPSTATE (i), OFTP state = Initiator waiting for
SSRM
LOG: DECEDI__ESIDTRANS (w), end session request transmitted

MSG: ---> OUT
ESID - End Session ID
ESIDCMD (Command)      = F
ESIDREAS (Reason Code) = 01 (Command not Recognized)
```

From this trace, you can see the following:

- An X.25 call request was issued with the specified call parameters.
- The X.25 call was established successfully (port status = X25S_RUNNING).
- 47 bytes of data were received.
- The message received was not recognized.
- The data received was:

  ```
  "~0d~0d~0a~0d~0d~0a (nodeab.xyz.abc.com)
  ttyp5~0d~0d~0a~0d~0d~0a~0d login: "
  ```

  The ~ indicates that the next two characters are to be intepreted as a hexadecimal value.
- An ESID (End Session Request) was transmitted with a reason code of 01 (Command not Recognized).

Looking at the data message actually received it is clear that the X.25 call has not been accepted by an OFTP system, but by an X.29 login server. The

X.29 login server has sent back a welcome message to login into the host. The OFTP gateway was expecting an SSRM message, not a login screen.

Using X.25 Trace

It is very unlikely that you should need to resort to X.25 trace to track OFTP problems. Enabling OFTP trace should provide you with sufficient information to track the problem.

Where you do need to use X.25 trace, refer to the *DEC CTF (Common Trace Facility) User's Guide*.

Chapter 17 Digital Problem Solving – SMTP /MIME Gateway

This chapter describes the general problem solving techniques that you should use to detect and solve problems with the SMTP/MIME gateway.

Outbound Processing

All files referred to in this section are located in the SMTP gateway work directory: `/var/adm/decedi/smtp/`. The files all have a common name of `ABBBBBBBCCCC_<connid>` but differing file extensions.

When an outbound transmission file is processed, the following files are created in the SMTP work directory:

```
ABBBBBBBCCCC_<connid>.ENVELOPE
ABBBBBBBCCCC_<connid>.HEADER
ABBBBBBBCCCC_<connid>.RAW
```

The `.ENVELOPE` file contains all the message headers (RFC 822). This includes the user defined additional headers from the connection details. An example of this file is:

```
From: edimail@widgets.org
Subject: 16APR199612513845_SMTP
Sender: Digital DEC/EDI <edimail@widgets.org>
To: An EDI System <edimail@widgets.org>
Mime-Version: 1.0
Message-Id: <16APR199612513845_SMTP@widgets.org>
Additional-Header-1: some value
```

The `.HEADER` file contains the content header lines. An example of this file is:

```
Content-Type: Application/EDIFACT
Content-Transfer-Encoding: Base64
```

```
<blank line>
```

Note the blank line at the end of the file.

The .RAW file is an exact copy of the Digital DEC/EDI transmission file from the Digital DEC/EDI store directories.

Once the above three files have been created (.ENVELOPE, .HEADER and .RAW) the Security Type setting on the connection details is examined.

If Security Type is specified as None the .RAW file is simply renamed to create the .DATA file. Otherwise it is the responsibility of the Outbound command to create the .DATA file. If after executing the outbound command, a .DATA file has not been produced, an error is generated.

The .DATA file is then encoded to the required format (Base64 or Quoted-Printable) to create a .CONTENT file.

Once the final three files have been created (.ENVELOPE, .HEADER and .CONTENT) the final .MESSSAGE file is constructed by appending the final files (.ENVELOPE + .HEADER + .CONTENT).

The .MESSAGE file is then posted using /usr/sbin/sendmail.

Once a .MESSAGE file has been posted successfully using sendmail a copy of it is placed in the Digital DEC/EDI store directories. This .MESSAGE file is archived along with the .TRANSMISSION file during secondary archive.

Archiving the .MESSAGE file ensures the auditability of the data before and after the external processing commands.

For a more detailed tracing of the processing of an outbound transmission, enable the trace flag on the connection, using the CommandCenter Communications Editor, and examine the trace file produced. Example trace files are given in Appendix F *Sample SMTP/MIME Logs and Traces*.

Inbound Processing

A new inbound message is first located in the directory: /var/adm/decedi/smtp_store/. This .MESSAGE file is then broken down into the following three files in the /var/adm/decedi/smtp/ directory:

```
ABBBBBBBCCCC_CONNID.ENVELOPE
ABBBBBBBCCCC_CONNID.HEADER
ABBBBBBBCCCC_CONNID.CONTENT
```

The .ENVELOPE file contains all the message headers (RFC 822) found in the inbound message. An example of this file is:

```
Date: Tue, 16 Apr 1996 12:51:39 +0000
From: edimail@widgets.org
Subject: 16APR199612513845_SMTP
To: An EDI System <edimail@widgets.org>
Mime-Version: 1.0
Message-Id: <16APR199612513845_SMTP@widgets.org>
```

The .HEADER file contains the content header lines. An example of this file is:

```
Content-Type: Application/EDIFACT
Content-Transfer-Encoding: Base64
<blank line>
```

Note the blank line at the end of the file.

The .CONTENT file contains the remaining data from the .MESSAGE file (i.e. all data after the first blank line).

At this stage the From: address in the inbound message is examined and matched to a connection id as described earlier.

The .CONTENT file is then decoded from the required format (Base64 or Quoted-Printable) to create a .DATA file.

At this stage the Security Type setting on the connection details is examined.

If Security Type is specified as None the .DATA file is simply renamed to create the .RAW file. Otherwise it is the responsibility of the Inbound command to create the .RAW file. If after executing the inbound command, a .RAW file has not been produced, an error is generated.

The .RAW file is used to create the final transmission file in the Digital DEC/EDI store directories. At this stage a copy of the .MESSAGE file is placed in the Digital DEC/EDI store directories. This .MESSAGE file is archived along with the .TRANSMISSION file during secondary archive.

Archiving the `.MESSAGE` file ensures the auditability of the data before and after the external processing commands.

For more detailed tracing of the processing of an inbound transmission, enable the trace flag on the connection, using the CommandCenter Communications Editor, and examine the trace file produced. Example trace files are given in Appendix F *Sample SMTP/MIME Logs and Traces*.

Reprocessing Failed Inbound Transmissions

The SMTP Gateway can fail to receive transmission files for a number of reasons, for example:

- Cannot match the sender or receiver to addresses in the SMTP connection details.
- External processing is selected and the inbound processing call fails.

In the event of the transmission file being received hitting these or other errors, the transmission file is failed. If an audit trail entry has be created for the transmission file then its status is set to FAILED_TO_RECEIVE, otherwise no audit trail is created for it as the the intended receiving connection is not known.

In either case an error is reported in the main Digital DEC/EDI error log, the problem transmission file is saved and its location also put in the main Digital DEC/EDI error log. For example:

```
Tue Jun 4 17:07:26 1997 PID =  2920 NAME = Internet SMTP/MIME
Gateway
DECEDI__SMTPINERR (e), error processing inbound message
DECEDI__SAVMSGFILE (i), message file saved as
/var/adm/decedi/smtp_store/04JUN199717072412_CONNID.MESSAGE_FA
ILED
DECEDI__EMAPFROM (e), error mapping from: address to
connection id
DECEDI__FROMADDR (i), from = <edimail@edisrv.reo.dec.com>
```

To reprocess a failed inbound transmission file, first correct the problem that caused the initial failure. In the above example, this means adding <edimail@edisrv.reo.dec.com> as a valid address defined in an SMTP connection. To reprocess the file, simply mail the failed file to your server's EDI mail account. For example, using the failed transmission above, and assuming our servers EDI mail account is 'edimail' then the following would

reprocess it:

```
mail edimail < \
/var/adm/decedi/smtp_store/04JUN199717072412_CONNID.MESSAGE_FA
ILED
```

Note: *The SMTP gateway uses the SMTP message id as a unique key in the Communications audit trail. When reprocessing a failed transmission file, if a communications audit trail with that message id already exists and providing the original failed during receipt or was cancelled, its audit trail entry with be reused for the reprocessed transmission. If the audit trail entry with the matching message id was not failed during receipt or cancelled, then the resent transmission will fail due to the non-uniqueness of its SMTP message id.*

Chapter 18 Problem Solving – 3780 Gateway

This chapter describes the general problem solving techniques that you should use to detect and solve problems with the 3780 gateway.

Log, Error and Data Files

The 3780 gateway generates various log and data files when you run a job. Some of these files are specific to 3780Plus and some are specific to the job's script.

- `/var/adm/decedi/3780Plus/connid_3780_date_time.LOG`

  This file is created by the 3780Plus emulator. It contains a listing of 3780Plus activity. This file is generated only if the Logging Option field is set to CREATE when you define the connection; this specifies that a new log file (with a time-stamp in the name) is to be created on each occasion that the connection is started.

  If the Logging Option is set to APPEND then this log file is created only on the first occasion that the connection is started, and re-used on subsequent occasions by appending the logging data to the same file.

- `/var/adm/decedi/3780Plus/connid_MON_date_time.LOG`

  This is an optional log file generated by the 3780Plus emulator. It contains a complete character-by-character trace of the Bisync protocol used for communicating with the VAN.

  A log file is generated only if the Line Monitoring field is set to YES when you define the connection; this specifies that a new monitor file (with a time-stamp in the name) is to be created on each occasion that the connection is started.

- `/var/adm/decedi/3780Plus/conn_id_data_date.dat`

  This file is created by the Tcl script.

  If the connection is lost or interrupted when the VAN is downloading data, then the file receiving data from the VAN, most likely incomplete, is copied to the above filename.

- `/var/adm/decedi/3780Plus/conn_id_mailbox.dat`

  This file is created by Cleo's 3780Plus as specified in the Tcl script.

  This contains the received data/transmission files from the VAN.

- `/var/adm/decedi/vanreports/connid_date.DAT`

  This file is created by the 3780Plus emulator and the Tcl script.

  When a connection has been established with the VAN, various files are downloaded. The files contain reports which supply information about the account you logged into. Multiple files may be downloaded by the VAN. The 3780Plus emulator saves these files as `PRINT.000`, `PRINT.001` etc. The Tcl script appends all the PRINT files into a single file and saves it into the *vanreports* directory.

- `/var/adm/decedi/vanreports/`
  `connid_inphist.saved_version`

  This file is created by the Tcl script.

  If there are PRINT files remaining from a previous connection when a new job is submitted, then these files are appended into one file and then saved in the *vanreports* directory. This situation would occur if the Tcl script crashed half way through a communication session.

- `/var/adm/decedi/logs/connid_job_id_date_time.TRACE`

  The trace file is a file containing log information about a job session. It is used for debugging purposes and is not meant to be used during normal operation.

  The trace file is located in the `/var/adm/decedi/log` directory. A new trace file is created each time a new job is run. It is created by the 3780 gateway.

  Example:

  ```
  /var/adm/decedi/logs/
                GEIS_TEST_30JAN199611014696_3780.TRACE
  ```

The use of the trace file can be turned on by switching on the Trace Session field in the Connection Details screen in the Communications Editor.

The file is created and starts logging messages after the job's process has been started and the Connection Details record has been read. The messages will describe the sequence of events required to process a job. Information unique to the job will be logged and this will help diagnose problems when errors occur.

Appendix H, *Sample 3780 Logs and Traces* provides examples of some of these tracing options.

Part IV Appendices

The appendices provide reference information about Digital DEC/EDI that you may need to refer to periodically.

Appendix A Pedi Gateway: Supported Elements of Service

The EDI Message (EDIMS) EDI Messaging (EDIMG) System Kernel Functional Profile contains 36 mandatory X.435 Service Elements. At this release, Digital DEC/EDI supports 30 out of the 36. The following Service Elements are not supported:

- Cross Reference Information
- Deferred Delivery
- Deferred Delivery Cancellation
- Disclosure of Other Recipients
- Distribution List Expansion History Indication
- Multi-part Body

The full list of 36 mandatory Service Elements is shown in the following table.

Table A-1 X.435: Mandatory Service Elements

Service Element	Function
Access Management	Enables a User Agent (UA) and Message Transfer Agent (MTA) to establish access to one another, and to manage information associated with access establishment.
Alternate Recipient Allowed	Enables an originating user to request the Physical Delivery Access Unit (PDAU) to provide additional rendition facilities.
Character Set	Allows the originator to indicate in the heading of the EDI message, the character set used in the EDI body of the message.
Content Type Indication	Enables the originating User Agent (UA) to indicate the content type of each submitted message. A recipient User Agent (UA) can have one or more content types delivered to it. An example of a content type is the contents generated by the Inter Personal Mail (IPM) class of cooperating UAs.
Conversion Prohibition	Enables an originating User Agent (UA) to instruct the Message Transfer System (MTS) that the implicit encoded information type conversion(s) should not be performed for a particular message.
Converted Indication	Enables the Message Transfer System (MTS) to indicate to a recipient User Agent (UA) that the Message Transfer System (MTS) performed encoded information type conversion on a delivered message. The recipient User Agent (UA) is informed of the resulting types.

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Cross Reference Information (on reception only)	Allows the originator to indicate in the heading of an EDI message, information that can be used for cross referencing between application specified reference IDs within an EDI interchange, and body parts of this, or other EDI messages.
Deferred Delivery	Enables an originating User Agent (UA) to instruct the Message Transfer System (MTS) that a message being submitted is delivered no sooner than a specified date and time. Delivery takes place as close to the date and time specified as possible, but not before. The date and time specified for the deferred delivery is subject to a limit that is defined by the originator's management domain.
Deferred Delivery Cancellation	Enables an originating User Agent (UA) to instruct the Message Transfer System (MTS) to cancel a previously submitted deferred delivery message. The cancellation attempt may or may not always succeed. Possible reasons for failure are: deferred delivery time has passed, or the message has already been forwarded within the Message Transfer System (MTS).
Delivery Notification	Enables an originating User Agent (UA) to request that the originating User Agent (UA) be explicitly notified when a submitted message has been delivered successfully to a recipient User Agent (UA) or access unit. The notification is related to the submitted message by means of the message identifier, and includes the date and time of delivery.

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Delivery Time Stamp Indication	Enables the Message Transfer System (MTS) to indicate to a recipient User Agent (UA) the date and time at which the Message Transfer System (MTS) delivered a message. In the case of physical delivery, it indicates the date and time at which the Physical Delivery Access Unit (PDAU) has taken responsibility for printing and further delivery of the physical message.
Disclosure of Other Recipients	This enables the originating User Agent (UA) to instruct the Message Transfer System (MTS) when submitting a multi-recipient message, to disclose the O/R (Originator/Recipient) names of all other recipients to each recipient User Agent (UA), upon delivery of the message. The O/R (Originator/Recipient) names disclosed are as supplied by the originating User Agent (UA). If the distribution list expansion has been performed, then only the originator Distribution List (DL) name is disclosed, and not the names of its members.
Distribution List (DL) Expansion History Indication	Provides to a recipient, at delivery, information about the distribution list(s) through which the message has arrived. It is a local matter as to how much of this information is presented to the recipient.
EDI Message Type(s)	Allows the originator to indicate in the heading of an EDI message the type(s) of EDI messages contained in the EDI interchange (for example, invoices, purchase orders, and so on).

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
EDI Notification Request	Allows the originating EDI User Agent (UA) to request that it be notified of a recipient's acceptance, refusal or forwarding of EDI Message (EDIM) responsibility, in any combination, for the message carrying this request. The originating EDI User Agent (UA) conveys this request to the recipient EDI User Agent (UA)/MS or AU.
EDI Standard Indication	Enables the originating EDI User Agent (UA) to indicate, in the heading of an EDI message, the type of EDI standard that is contained in this EDI message (for example, EDIFACT).
EDI Message Identification	Enables co-operating EDI UAs to convey a globally unique identifier for each EDI message sent or received. The EDI message identifier is composed of an O/R (Originator/Recipient) name of the originator and an identifier that is unique with respect to that name. EDI UAs and EDI Message (EDIM)G users use this identifier to refer to a previously sent or received EDI message (for example, in EDI notifications).
EDI Message (EDIM) Responsibility Forwarding Allowed Indication	Allows an originating EDI User Agent (UA) to indicate that the EDI Message (EDIM) responsibility for this EDI message may be forwarded on by the recipient EDI User Agent (UA).
EDIN Receiver (on reception only)	Allows the originator, or a forwarding EDI User Agent (UA)/MS (Message Store), to indicate to a recipient, that O/R (Originator/Recipient) address that requested notifications should be returned to.

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Expiry Date/Time Indication (on reception only)	Allows the originator to indicate to the recipient the date and time after which the originator considers the EDI message to be invalid. The intent is to state the originator's assessment of the current applicability of an EDI message. The particular action on behalf of a recipient by the recipient, or by the recipient's EDI User Agent (UA), is unspecified. Possible actions might be to file or delete the EDI message after the expiry date has passed.
Grade of Delivery Selection	Enables an originating User Agent (UA) to request that transfer through the Message Transfer System (MTS) is urgent or non-urgent, rather than normal. The time periods defined for non-urgent and urgent transfer are longer and shorter, respectively, than that defined for normal transfer. This indication is also sent to the recipient with the message.
Incomplete Copy Indication	Allows a forwarding EDI User Agent (UA) to indicate that the forwarded EDI message is an incomplete copy of an EDI message with the same EDI message identification, in that one or many body parts of the original EDI message are absent.
Interchange Header	Allows the originating EDI User Agent (UA) to place data elements of the EDI interchange headers in corresponding fields in the EDI Message (EDIM).

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Message Identification	Enables the Message Transfer System (MTS) to provide the User Agent (UA) with an unique identifier for each message or probe submitted or delivered by the Message Transfer System (MTS). UAs and the Message Transfer System (MTS) use this identifier to refer to a previously submitted message in connection with elements of service,such as delivery and non-delivery notification.
Multi-destination Delivery	Enables an originating User Agent (UA) to specify that a message being submitted is to be delivered to more than one recipient User Agent (UA). Simultaneous delivery to all specified UAs is not implied.
Multi-part Body (on reception only)	Allows an originator to send to a recipient an EDI message with a body that is comprised of several parts. The nature and attributes, or type, of each body part are conveyed along with the body part.
Non-delivery Notification	Enables an Message Transfer System (MTS) to notify an originating User Agent (UA) if a submitted message was not delivered to the specified recipient User Agents (UAs). The reason the message was not delivered is included as part of the notification.
Obsoleting Indication (on reception only)	Allows the originator to indicate to the recipient that one or more EDI messages previously sent by the originator are obsolete. The EDI message that carries this indication supersedes the obsolete EDI message(s).

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Original Encoded Information Types Indication (EIT)	Enables an originating User Agent (UA) to specify to the Message Transfer System (MTS) the encoded information types of a message being submitted. When the message is delivered, it also indicates to the recipient User Agent (UA) the encoded information types of the message specified by the originating User Agent (UA).
Originator Indication	Allows the identity of the originator to be conveyed to the recipient.
Recipient Indication	Allows the originator to provide the names of one or more EDI Message (EDIMG) users, or Distribution List (DL)s, who are intended recipients of the EDI message. In addition, it is possible to specify an action request qualifier for each recipient, such as, For Action; Copy; Other (as defined bilaterally).
Related Message(s) (on reception only)	Allows the originator to associate with the EDI message being sent, the globally unique identifiers of one or more other messages that share the same identification space (for example, IP messages). This enables the recipient's EDI User Agent (UA), for example, to retrieve a copy of the referenced messages from storage.
Requested Delivery Method	Allows a user to request, on a per-recipient basis, the preference of method or methods of message delivery (such as, through an access unit). Non-delivery results if preference(s) cannot be satisfied.

Table A-1 (continued) X.435: Mandatory Service Elements

Service Element	Function
Submission Time-stamp Indication	Enables the Message Transfer System (MTS) to indicate to the originating User Agent (UA) and each recipient User Agent (UA) the date and time at which a message was submitted to the Message Transfer System (MTS). In the case of physical delivery, this also enables the PADU to indicate the date and time of submission on the physical message.
Typed Body	Permits the nature and characteristics of the body of an EDI message to be conveyed along with the body. Permissible body part types are EDI body, forwarded EDI message (EDIM) body, and externally defined body parts.
User/User Agent (UA) Capabilities Registration	Enables a User Agent (UA) to indicate to its Message Transfer Agent (MTA), through registration, the unrestricted use of any or all of the following capabilities with respect to received messages. The Message Transfer Agent (MTA) will not deliver to a User Agent (UA) a message that does not match, or exceeds, the registered capabilities.

Appendix B Document and Transmission File Status Flows

This appendix details the individual states that documents and transmission files go through as they are processed within the Digital DEC/EDI Server and the Memory Queues that they transition through. Different sections describe the status flow for:

- Outbound documents.
- Outbound transmission files.
- Inbound transmission files.
- Inbound documents.

Documents processed by using Application-to-Application routing are also described where relevant within the sections for outbound and inbound documents.

The final state of all documents and transmission files in the system is **PURGEABLE** or **CANCELLED**. Documents and transmission files that have been processed successfully (despite any possible recovered failures en-route) are given the final status of **PURGEABLE**. Those that fail and are not recoverable (for example because their content is invalid) are given a final status of **CANCELLED**.

The Archive Server takes documents and transmission files in either of these two final states and removes their audit trail information to the Archive Audit Trail.

Document flow can be monitored using the DEC/EDI Cockpit.

The Memory Queue button in the Cockpit will show you how many documents or transmission files are in each queue and you can double-click the queues to find out which documents are in them.

The Summary screen in the Cockpit will show you what the last saved status of the document of transmission file was by allowing you to examine the contents of the database. Remember that while some statuses have a direct relationship to which queue the document or transmission file is in, many do not.

Note that the term EDIFACT is used in the following sections to reference both EDIFACT and ODDETTE standards as is X12 to reference both X12 and TDCC (UCS) standards.

Outbound Document Flow

Outbound documents begin their life cycle in the Port/Fetch Server. The Server adds the document to the database with a status of **QUEUED** and then examines the EDI syntax of the document. EDIFACT documents are placed in the *EDIFACT Converter Queue*, X12 documents in the *X12 Converter Queue* and TRADACOMS in the *TRADACOMS Converter Queue*.

For EDIFACT and X12 documents, once the relevant Converter has finished creating the External Format File, the document is moved to one of the *EDIFACT TFB Queue*, the *EDIFACT –High TFB Queue*, the *X12 TFB Queue* or the *X12 –High TFB Queue* depending on the EDI Syntax of the document and its priority setting. If the document fails conversion however, the status in the database is updated to **FAILED_TO_CONVERT** and the document is removed from the Memory Queue. If you fix the problem and use the Cockpit to reset the document then the status is reset back to **QUEUED** and the CommandCenter Server puts the document back into the relevant Converter's Memory Queue.

For high priority EDIFACT and X12 documents, the TFB (Transmission File Builder) immediately takes the document and wraps it in the appropriate EDI envelope (UNB for EDIFACT and either an ISA or BG for X12/TDCC). The now enveloped document is placed into its own Transmission File, the document's status is updated to **AWAITING_TRANSMISSION** in the database and the document moved to the *Built Documents Queue (unsent)*.

For normal priority EDIFACT and X12 documents, the TFB will wake up after the interval you have specified and will 'batch' together documents for the same trading partner into one Transmission File. All documents contained in the Transmission File are then updated to

AWAITING_TRANSMISSION in the database and moved to the *Built Documents Queue (unsent)*.

For TRADACOMS documents, the conversion and building procedures are all done in one step. The document is taken from the *TRADACOMS Converter Queue*, converted to an external format, enveloped and placed in a new Transmission File. The document's status is then updated to **AWAIT_TRANSMISSION** in the database and the document moved to the *Built Documents Queue (unsent)*.

Note that if a document of any standard fails in the building phase of the procedure because of, for example, lack of disk space then the document is updated with a status of **FAILED_TO_BUILD** and it is removed from the Memory Queues. A transmission file is NOT created. If you then reset the document, its status is reset to **CONVERTED** and it is placed in the queue from which it originally came (the relevant TFB queue or the *TRADACOMS Converter Queue*).

From here on, documents mirror the status of the transmission files to which they belong. Once the transmission file (and thus the documents) is set to a **PURGEABLE** status, the documents are moved to the *Archive Server Queue* unless the documents are waiting for an EDIFACT CONTRL message, an X12 997 transaction or a TDCC 999 transaction in which case they are moved to the *Ack. Exp.Documents Queue*. The documents wait in this queue until the expected document arrives (or you tell them not to wait for the expected document) at which point they also are moved to the *Archive Server Queue*.

The Archive Server moves the documents from the Current area to the Archive area and removes the entry from the *Archive Server Queue*.

Application to Application Document Flow

Application to application documents enter the system in the same manner as normal documents but because they have been defined as Application to Application using the CommandCenter's Management Services Editor, no actual outbound document is created. Instead, an INBOUND document is created with a status of **AVAILABLE** and the document placed in the *Available Documents Queue* Memory Queue. From there on, it follows the same path as any other AVAILABLE document (see Inbound Document Flow later in this appendix).

Outbound Transmission File Flow

Outbound transmission files are created either by the EDIFACT TFB, X12 TFB, TRADACOMS Converter or by utilizing the –type=transmission_file qualifier to the trade post command.

Transmission files are created with a status of **AWAIT_TRANSMISSION** and they are placed into the *Await Trans. File Queue* waiting for the relevant Communications Gateway to wake up and send them.

If the gateway fails to complete transmission of the file, then the transmission file is set to **FAILED_TO_SEND** and it is removed from the Memory Queues. If you then reset/reprocess the transmission file, it is set back to **AWAIT_TRANSMISSION** and placed back in the *Await Trans. File Queue*.

Once successfully sent to the trading partner, some gateways will change the status of the transmission file to **SENT** and move it to the *Sent Trans. File Queue*. Usually, this status means that the transmission file is waiting for some communications level acknowledgement from your trading partner or Value Added Network. For example, transmission files sent over using the Pedi Gateway would be waiting for an X.400 delivery receipt and those sent over OFTP would be waiting for an End-to-End Response Packet (EERP). If a negative acknowledgement is received then the status of the transmission file is changes to **FAILED_TO_DELIVER** and the transmission file is removed from the Memory Queues. If you reset the transmission file for reprocessing then its status is changed back to the original **AWAIT_TRANSMISSION** and it is placed back in the *Await Trans. File Queue*.

Some gateways (notably the Pedi Gateway where X.435 EDI Notifications have been enabled in the Connection details) then change the status of the transmission file to **DELIVERED**. If a negative acknowledgement is received then the status of the transmission file is changes to **FAILED_AFTER_DELIVERY** and the transmission file is removed from the Memory Queues. You cannot reset a transmission file from this status for reprocessing; you must cancel the transmission file and then either resubmit the documents or resend the transmission file.

If no communications level acknowledgements are used or if all positive communications level acknowledgements are successfully received then the transmission file eventually ends up at the **PURGEABLE** status and is placed

into the *Archive Server Queue* where the Archive Server moves it to the Archive area and removes it from the Memory Queues.

Inbound Transmission File Flow

Inbound files arriving from your trading partners or Value Added Network are initially created with a status of **RECEIVED** in the DEC/EDI database and placed in the *Comms. Controller (Inbound) Queue* Memory Queue with the exception of the OFTP Gateway which initially places it in the *OFTP's Partially Received Queue*. Once the file has been fully received by the OFTP Gateway and the gateway has sent the End-to-End Response Packet (EERP) to acknowledge receipt, the transmission file is moved to the *Comms. Controller (Inbound) Queue*.

The Communications Controller's job is to look at the file received and determine what it is and where you want it to go based on your settings of the Syntax field in the associated connection.

- If the connection is set to X12, EDIFACT, TRADACOMS or MULTIPLE, the Communications Controller separates each EDI Interchange that it finds in the file into separate transmission files and appends a unique identifier (an underscore followed by a base 36 3 character counter) to the original transmission file name to create new unique transmission file names. The original transmission file is set to the **PURGEABLE** status and added to the *Archive Server Queue* to be moved to the Archive area.

  EDIFACT/ODETTE transmission files (interchanges) are set to a status of **TRANSLATABLE** and places on the *EDIFACT TFS Queue* Memory Queue.

  X12/TDCC transmission files (interchanges) are set to a status of **TRANSLATABLE** and places on the *X12 Translator Queue* Memory Queue.

  TRADACOMS transmission files (interchanges) are set to a status of **TRANSLATABLE** and places on the *TRADACOMS Translator Queue Memory Queue*.

  If the Communications Controller encounters anything in the transmission file other than formatting characters such as end-of-line or null characters, these characters are placed in a separate transmission file and that transmission file is given a status of **FAILED_TO_IDENTIFY**.

You cannot reset a transmission file from this status for reprocessing; you must review the transmission file and then cancel it. If the file contains corrupted information from your trading partner then you must ask your trading partner to resend the file or ask your Value Added Network to re-queue it for retrieval.

- If the connection is set to MIXED then the Communications Controller performs the same processing as listed for the X12, EDIFACT, TRADACOMS or MULTIPLE settings except that for each interchange discovered, it examines your Trading Partner Profile and attempts to match the incoming interchange envelope with a defined trading partner, agreement and document.

  If it finds a match then the 'Use Translator On Inbound' flag in the agreement is reviewed. If the flag is checked then the interchange is forwarded to the relevant EDIFACT, X12 or TRADACOMS Memory Queue with the **`TRANSLATABLE`** status.

  If a trading partner profile cannot be matched , if 'Use Translator On Inbound' flag is clear or if the information in the transmission file is not recognizable as an EDI Interchange then the transmission file is set to a status of **`TRANS_AVAILABLE`** and added to the *Available Documents Queue* Memory Queue where it can be retrieved using the '-type=transmission_file' flag on the trade fetch command.

- If the connection is set to SPLIT then the Communications Controller attempts to identify each individual interchange in the file and create separate entries for them but will set the status of all files to **`TRANS_AVAILABLE`** and added to the Available *Documents Queue* Memory Queue where it can be retrieved using the '-type=transmission_file' flag on the trade fetch command. The Trading Partner Profiles are not checked.
- If the connection is set to BYPASS then the Communications Controller simply updates the status of the incoming transmission file to **`TRANS_AVAILABLE`** without even looking at the contents of the file. The transmission file is then added to the *Available Documents Queue* Memory Queue where it can be retrieved using the '-type=transmission_file' flag on the trade fetch command.

Transmission files added to the *EDIFACT TFS Queue*, *X12 Translator Queue* or *TRADACOMS Translator Queue* Memory Queues are parsed by the relevant component and the documents contained within each

interchange are extracted. Once successfully processed, the transmission file is updated with a status of **PURGEABLE** and moved to the *Archive Server Queue* Memory Queue to be moved to the Archive area. If the interchange cannot successfully be processed then its status is updated to **FAILED_TO_SEPARATE** and removed from the Memory Queues.

For EDIFACT/ODETTE and TRADACOMS Interchanges, no documents will have been created if the Transmission File has a **FAILED_TO_SEPARATE** status. This typically happens when you have a mismatch between information in the envelope. For example, the envelope may say there are 20 invoices in the interchange but DEC/EDI can only find 19. You must examine the Error Listing associated with the transmission file and, if you can correct the problem, you may reset the transmission file to be reprocessed at which point its status is reset to **TRANSLATABLE** and it is placed back into the *EDIFACT TFS Queue* or the *TRADACOMS Translator Queue* Memory Queues. If you cannot correct the problem then you must cancel it and ask your trading partner to correct and resend the file.

For X12/TDCC Interchanges, some documents may have been successfully processed and some may not. If a transmission file has a **FAILED_TO_SEPARATE** status then the problem may be with either the envelope or the document contents. Always start by looking at the transmission file's error listing for the problem report and then look at the document's error listing to determine the problem. If you can correct the problem with either the envelope or document then reset the transmission file to reprocess and its status is reset to **TRANSLATABLE** and it is placed back into the *X12 Translator Queue*. Note that when you reprocess an X12 Transmission File, it will look for any successfully processed documents and will not replicate them. If you cannot correct the problem then you must cancel it and ask your trading partner to correct and resend the interchange if no documents were successfully processed or only the documents that failed if some documents failed.

Transmission files that were placed in the *Available Documents Queue* and then retrieved using the 'trade fetch' command with the '-type=transmssion_file' qualifier are updated to a status of **PURGEABLE** and added to the *Archive Server Queue*. If for some reason the trade fetch command fails (for example, lack of disk space) then the status of the transmission file is changed to **FAILED_TO_TRANSFETCH** and is removed from the Memory Queues. Resetting the transmission file to

reprocess changes the status back to **AVAILABLE** and adds the transmission file back into the *Available Documents Queue*.

Inbound Documents Flow

Inbound documents are created by one of the following processes.

- The EDIFACT Transmission File Splitter (TFS) simply separates the documents from their transmission files and sets the status of the documents to **SEPARATED**. The documents are placed on the *EDIFACT Translator Queue* where each document is processed by an EDIFACT Translator, set to a status of **AVAILABLE** and moved to the *Available Documents Queue* Memory Queue.

  If the document should fail translation for some reason (for example, an unrecognized weight qualifier) then the document will be set to the **FAILED_TO_TRANSLATE** status and removed from the Memory Queues. Review the error listing associated with the document and if you can correct the problem, do so and reset the document to reprocess. This will change the document's status back to **SEPARATED** and add it back into the *EDIFACT Translator Queue*.
- The X12 Translator performs the both split and translate jobs in one pass and creates the documents with a status of either **AVAILABLE** or **FAILED_TO_SEPARATE**. Available documents are added to the *Available Documents Queue* while failed documents are removed from the Memory Queues.

  For failed documents, after reviewing the error listings associated with the original transmission file and determining if you can correct the problem, you can reset the transmission file (not the document) which will also have a **FAILED_TO_SEPARATE** status to reprocess. This will cause the currently failed documents to all be cancelled and then new documents created to replace them.
- The TRADACOMS Translator also performs the both split and translate jobs but in 2 passes in a similar fashion to EDIFACT. Please see the description of the EDIFACT process for further details.

Once the document is **AVAILABLE** and on the *Available Documents Queue* Memory Queue, it can be retrieved using the 'trade fetch' command. If the document is successfully fetched then the status of the document is updated to PURGEABLE and it is moved to the *Archive Server Queue* to be

transferred to the Archive area. If the trade fetch fails (for example, due to mapping problems) then the document is updated to the **FAILED_TO_FETCH** status and removed from the Memory Queues. Resetting the document to reprocess will change its status back to **AVAILABLE** and add it back to the *Available Documents Queue*.

Appendix C **Environment Variables**

Various functions of the Digital DEC/EDI Server can be modified by setting one or more of the environment variables described in this Appendix.

These variables are defined in the file: `/usr/sbin/decedi_sysetup`. This is a text file, that you should edit by using a text editor of your choice. Note that changes made to the settings of these variables are not effective until the Server is restarted. This file is not replaced on installation, so any changes you have made are preserved.

Do not make any changes to the file `decedi_setup` *in the same directory. This file contains important data required by Digital DEC/EDI, that you should not alter.*

To set an environment variable add one or more lines to the bottom of the `decedi_sysetup` file, for example:

```
#! Switch on archiver logging
DECEDI_AS_VERBOSE=ALL
```

Note: Any lines added should be after the `set -a` *line so that the variables are automatically exported for the rest of Digital DEC/EDI to see.*

The following table describes each of the environment variables used by the Digital DEC/EDI Server.

Table C-1 Environment Variables

Environment Variable Name	Effect
DECEDI_LOG_SEVERITY	This variable provides a means of avoiding the logging of groups containing only lower priority messages from the Error Log. Where not specified, all messages are logged. The following values may be specified: - WARNING : suppresses all messages of informational severity. - ERROR : suppresses all messages of informational or warning severity.
DECEDI_LOG_TRANS_ SEVERITY	Instructs at what severity the X12 Translation service should start logging messages. It can be set to one of: • ALL • WARNING suppresses all messages of informational severity. • ERROR suppresses all messages of informational or warning severity. If not defined, or define to some other value it will default to ALL.
DECEDI_MAIL_IMPEXP DECEDI_MAIL_PEDI DECEDI_MAIL_OFTP DECEDI_MAIL_SMTP DECEDI_MAIL_3780	These variables specify which mail users should receive mail notification when a connection is disabled for the specified gateway. The value should be specified as a comma separated list of recipients. A member of the list may also reference a mailing list if the first character is “@”. When not specified, no mail is sent.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_ MAINTAIN_HISTORY	Setting this environment variable to any value other than zero causes the Server to record history records in the audit trail. A record is added to the database each time a document or transmission file changes from one state to another, listing the time at which the transition occurred. This history information is viewable by using Cockpit. The logging of history information results in many additional database transactions and can significantly slow down a Server. Only use the option to log history records, if performance is not a problem, and you have a need to view this extra information.
DECEDI_MAX_DOCUMENTS	This variable defines the maximum number of EDI document definitions that can be held in memory by the Translation Services components. Where not specified, a value of 10 is assumed. Setting a value in excess of the default, may increase system performance, but consumes more memory for each extra document definition.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_ MAX_MAPPING_TABLES	The mapper caches Mapping Tables as they are used. This can significantly improve mapping performance where the same mapping tables are used repeatedly. This variable defines the maximum number of Mapping Tables that can be cached at one time Where not specified, a value of 10 is assumed. The maximum number of cache entries supported is 100. Once the number of tables cached reaches the specified maximum value, the least recently used table is eliminated from the cache, to make space for a new table. If a newer copy of a cached mapping table is placed in the Mapping Table Repository, the newer copy replaces the existing one in the cache when the Mapping Table is next used.
DECEDI_ MAX_SEGMENT_TABLES	This variable defines the maximum number of EDI standards dictionaries that can be held in memory by the Translation Services components.Where not specified, a value of 2 is assumed. Setting a value in excess of the default, may increase system performance, but consumes more memory for each extra dictionary.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_NOTIFY_MAIL	This variable specifies which mail users should receive mail notification of important system messages. The value should be specified as a comma separated list of recipients. A member of the list may also reference a mailing list if the first character is "@". Where not specified, no mail is sent.
DECEDI_ NOTIFY_OPERATOR	This variable specifies the device to which any important system messages are to be sent. Where not specified, no messages are sent.
DECEDI_PEDI_KEEP_EDIN	If this variable is defined, with any value, it specifies that copies of any Pedi Notification messages, sent or received, are kept in /var/adm/decedi/logs. By default, these messages are not kept.
DECEDI_ PEDI_KEEP_MSGFILE	If this variable is defined, with any value, it specifies that copies of any Pedi messages sent or received are kept in /var/adm/decedi/logs. MTA delivery reports are also saved.. By default, these messages and reports are not kept.
DECEDI_ USE_TEST_INDICATOR	This variable defines whether or not test_indicator is used to select whether a particular document is to be fetched. If this variable has any value, and where the fetch request specifies a test indicator, only documents of that type are fetched. Where this variable is not defined, test indicator is not taken into account when searching for documents to fetch.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_X12_ DISABLE_APPFIL	By default the X12 Translator places multiple in-house files destined for the same application into a single physical file for performance reasons. if this flag is set then each in-house file is placed in a separate file.
DECEDI_X12_ DISABLE_B501	The B501 element in the TDCC 999 document is no longer used but Digital DEC/EDI sets it to to 999. If this variable is defined this field is supplied empty.
DECEDI_X12_ ENABLE_ACK_REPORT	By default Digital DEC/EDI only mails received functional acknowledgements which indicate errors. If this variable is set then all incoming functional acknowledgements are mailed to the nominated mail account(s).
FBR_ RUNTIME_AUDIT_LEVEL	See *Specifying Mapper Audit Levels* on page C-13.
DECEDI_AS_START_COUNT	The number of Archive Server processes to start on each system.
DECEDI_CCI_START_ COUNT	The number of Communications Controller processes to start on each system.
DECEDI_ETFB_START_ COUNT	The number of EDIFACT TFB processes to start on each system.
DECEDI_ECNV_START_ COUNT	The number of EDIFACT Converter processes to start on each system.
DECEDI_ETRN_START_ COUNT	The number of EDIFACT Translator processes to start on each system.
DECEDI_ETFS_START_ COUNT	The number of EDIFACT Transmission File Splitter processes to start on each system.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_TCNV_START_COUNT	The number of TRADACOMS Converter processes to start on each system.
DECEDI_TTRN_START_COUNT	The number of TRADACOMS Translator processes to start on each system.
DECEDI_XTFB_START_COUNT	The number of X12 TFB[2] processes to start on each system.
DECEDI_XCNV_START_COUNT	The number of X12 Converter processes to start on each system.
DECEDI_XTRN_START_COUNT	The number of X12 Translator processes to start on each system.
DECEDI_TCP_TRACE	Causes the post/fetch, track and CommandCenter servers to begin logging information to the /var/adm/decedi/logs directory. Information is very detailed and will have a negative impact on both performance and disk space usage. This environment variable should only be used user the direction of Compaq or Digital Globalsoft. The value set in this environment variable can be one or more of the following flags C – trace 'child' server interaction P – trace TCP/IP protocol messages S – trace TCP/IP service events T – trace TCP/IP traffic A – trace all of the above
DECEDI_SAP_START	Directs DEC/EDI to start the SAP Status Generator process. See the appendix on the SAP Status Generator for further details.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
MEMQUEUE_TRACE	Causes all components in DEC/EDI that interact with the DEC/EDI Memory Queues to log that information to trace files in the /var/adm/decedi/logs directory. Information is very detailed and will have a negative impact on both performance and disk space usage. This environment variable should only be used user the direction of Compaq or Digital Global-soft.
ANA_VERSION	By default, DEC/EDI supports only syn-tax version 1 of the TRADACOMS stand-ard and places this identifier in the STX segment. However, if you wish to override this value, you can set this environment variable with the required value.
DECEDI_CSF_TIMEOUT	The number of minutes that the port/fetch server should wait for a client to respond before assuming that the client is stalled. The value should be between 60 (1 hour) and 720 (12 hours).
DECEDI_CSF_PRESTART	The number of post/fetch servers to start at DEC/EDI startup time. By default, DEC/EDI starts 1 post/fetch server at star-tup time and others as and when required. The value must be between 1 and the value of DECEDI_CSF_MAX.
DECEDI_CSF_MAX	The maximum number of post/fetch serv-ers that are allowed to run concurrently. DEC/EDI imposes no limit on this value but you must remember to allow for avail-able system resources when setting this value. Default is 3.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_CST_PRESTART	The number of track servers to start at DEC/EDI startup time. By default, DEC/EDI starts 1 track server at startup time and others as and when required. The value must be between 1 and the value of DECEDI_CST_MAX.
DECEDI_CST_MAX	The maximum number of track servers that are allowed to run concurrently. DEC/EDI imposes no limit on this value but you must remember to allow for available system resources when setting this value. Default is 3.
DECEDI_CSG_PRESTART	The number of CommandCenter/Cockpit servers to start at DEC/EDI startup time. By default, DEC/EDI starts 1 server at startup time and others as and when required. The value must be between 1 and the value of DECEDI_CSG_MAX.
DECEDI_CSG_MAX	The maximum number of CommandCenter/Cockpit servers that are allowed to run concurrently. DEC/EDI imposes no limit on this value but you must remember to allow for available system resources when setting this value. Default is 3.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI__NO_TP_AGREE_CONTINUE	The presence of this environment variable tells the DEC/EDI Mapper not to abort a trade post session if a document does not have Trading Partner Profile set up for the requested Application, Partner and Document identifiers. The document concerned is NOT posted to DEC/EDI but a soft error is generated and the next document in the application file is attempted. The missing document can be regenerated by using the –restart_from=<document's position in file> and the –L=<document's position in file> qualifiers to the trade post command.
FBR_NO_STR_WARN	The presence of this environment variable prevents the DEC/EDI Mapper from reporting 'string comparison' informational messages in the trade post output and debug files.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_IPM_P2_SUBJECT	DEC/EDI allows you to control the subject of a message in one of 3 ways. If the '-connection_data' qualifier is specified when using the 'trade post' command then the value associated with the qualifier is used. Otherwise, if a value is specified in the 'Connection Specific Data' field of the Trading Partner Profile then that value will be used for all associated messages. Otherwise, if this environment variable is defined then the value associated with this value is used as the subject in all P2 bodypart messages. The default is to use a blank subject.

Table C-1 (continued) Environment Variables

Environment Variable Name	Effect
DECEDI_TRAX_TYPE	Used to identify Test documents when using the TRADANET or BROKERNET Value Added Networks un the United Kingdom. If the value associated with this environment variable is B and the Trading Partner Profile for this partner has placed the text $MESSAGE-TYPE in the Application Reference field and the document is a test document, then the fourth character is changed to a 'T' to indicate a test document making INVFIL into INVTIL. If the value associated with this environment variable is not B or the variable is not defined then if the Trading Partner Profile for this partner has placed the text $MESSAGE-TYPE in the Application Reference field and the document is a test document, the final 3 characters of the Application Reference are changed to 'TES' making INVFIL into INVTES.
DECEDI_X12_BYPASS_SECURITY	In previous versions of DEC/EDI, if the Security and Authentication qualifiers and values defined in the Trading Partner Profiles must always match incoming interchanges. For DEC/EDI V4 and later, if this environment variable is present then DEC/EDI will ignore the values set in the Security Code and Authentication Code fields if the qualifiers associated with these codes are set to '00'.

Specifying Mapper Audit Levels

It is possible to configure the level of auditing information created by the mapper.

By limiting the amount of information audited you can help to improve the runtime performance of mapping. By setting the environment variable `FBR_RUNTIME_AUDIT_LEVEL`, the mapper can be configured to perform none, some, or maximum auditing of runtime information. The following table shows the auditing levels that can be set.

Table C-2 Mapper Audit Levels

Audit Level	Auditing Performed
0	None
1	Commit Document, Soft Error, Hard Error, Hook, User Event, History File, Start Recovery, End Recovery
2	Maximum

Audit level 0 prevents the mapper from writing any audit information at runtime to the audit database. This provides a considerable performance enhancement at runtime.

Audit level 1, the default, provides better performance over maximum auditing, but not as much as gained at level 0. For most normal mapping tables the only runtime audit point written to the audit database at level 1 is *Commit Document*. The information audited for the Commit Document event provides sufficient information for tracking the document.

Level 1 is the minimum level required by Mapping Tables that utilise hook routines to access information provided by runtime auditing.

Audit level 2 enables maximum auditing.

Appendix D Files and Processes Used by Digital DEC/EDI

This appendix provides information on the main directories, files and processes used, or created, by the Digital DEC/EDI Server and Application Client.

Table D-1 Files Used by Digital DEC/EDI

File or Directory	Purpose
/sbin/init.d/dedi	Scripts for starting and stopping the Server.
/usr/sbin/decedi*	Main Digital DEC/EDI images and scripts.
/usr/bin/trade	User accessible Digital DEC/EDI images i.e. trade.
/usr/examples/decedi/client	Example API and CLI client code and scripts.
/usr/shlib/libdecedi*.so	Digital DEC/EDI shareable libraries.
/usr/man/man1/trade.1	Trade manual help.
/usr/man/man3/decedi*.3	Digital DEC/EDI API manual help.
/usr/man/man8/decedi*.8	Digital DEC/EDI Administrator manual help.
/usr/include/decedi*.h	Include files for Digital DEC/EDI API.
/var/adm/decedi...	Root for all Digital DEC/EDI configuration and operational data.
../3780Plus/	† 3780 gateway working directory.
.../arch_reports/	Secondary Archive reports.

Table D-1 (continued) Files Used by Digital DEC/EDI

File or Directory	Purpose
.../backup/	† Secondary Archive & Retrieve working directory.
.../data/	Configuration data and Profile Cache Files.
../impdef/	Message definitions used by the Message Update Service.
.../ipc/	† ‡ Inter Process Communications working directory.
.../locks/	† ‡ Inter Process Lock working directory.
.../logs/	† Error and Operator Logs.
.../maps/	Mapping Table Repository.
../smtp/	† SMTP/MIME gateway working directory.
../smtp_store/	† SMTP/MIME gateway temporary message store.
.../store_1/	Digital DEC/EDI file store # 1.
.../store_<n>/	Digital DEC/EDI file store #<n>.
.../temp/	† Temporary working directory.
../vanreports/	† VAN reports file store.

† Directories marked thus can be cleared periodically of all files. With the Server shut down, you may delete the contents of these directories to save or recover disk space. The files in these directories are temporary files or log files. You may like to preserve copies of some of these files prior to deletion.

‡ Directories marked thus must not be located on NFS mounted disks.

The following table shows the names of the different process daemons used by Digital DEC/EDI:

Table D-2 Process Daemons Used by Digital DEC/EDI

Process Name	Description
decedi_guid	The GUI daemon. One of these is started for each concurrent request made from a PC running CommandCenter or Cockpit. These processes are not stopped when the Digital DEC/EDI Server is stopped. They must be stopped manually if necessary.
decedi_ecnvd	The EDIFACT Converter daemon. One of these is started if EDIFACT Translation Services are required. It remains active until the Server is stopped.
decedi_etfbd	The EDIFACT Transmission File Builder daemon. One of these is started if EDIFACT Translation Services are required. It remains active until the Server is stopped.
decedi_etfsd	The EDIFACT Transmission File Splitter daemon. One of these is started if EDIFACT Translation Services are required. It remains active until the Server is stopped.
decedi_etrnd	The EDIFACT Translator daemon. One of these is started if EDIFACT Translation Services are required. It remains active until the Server is stopped.
decedi_impexpd	The Import/Export gateway daemon. One of these is started when the Server is started and configured to start the Import/Export gateway. It remains active until the Server is stopped or until the gateway is stopped by using CommandCenter.
decedi_pedid	The Pedi gateway daemon. One of these is started when the Server is started and configured to start the Pedi gateway. It remains active until the Server is stopped or until the gateway is stopped by using CommandCenter.
decedi_oftpd	The OFTP gateway daemon. One of these is started when the Server is started and configured to start the OFTP gateway. It remains active until the Server is stopped or until the gateway is stopped by using CommandCenter.
decedi_asd	The Archive Server daemon. One of these is started when the Server is started. It remains active until the Server is stopped.

Table D-2 (continued) Process Daemons Used by Digital DEC/EDI

Process Name	Description
decedi_xcnvd	The X12 Converter daemon. One of these is started if the X12 Translation Services are required. It remains active until the Server is stopped.
decedi_tcnvd	The Tradacoms Converter daemon. One of these is started if the Tradacoms Translation Services are required. It remains active until the Server is stopped.
decedi_xtfbd	The X12 Transmission File Builder daemon. One of these is started if the X12 Translation Services are required. It remains active until the Server is stopped.
decedi_xtrnd	The X12 Translator daemon. One of these is started if the X12 Translation Services are required. It remains active until the Server is stopped.
decedi_ttrnd	The Tradacoms Translator daemon. One of these is started if the Tradacoms Translation Services are required. It remains active until the Server is stopped.
decedi_3780d	The 3780 gateway daemon. One of these is started if the 3780 communications is required. It remains active until the Server is stopped or until the gateway is stopped by using the CommandCenter.
decedi_smtpd	The SMTP/MIME gateway daemon. One of these is started if the SMTP/MIME communications is required. It remains active until the Server is stopped or until the gateway is stopped by using the CommandCenter.

Appendix E Sample OFTP Logs and Traces

This appendix contains some sample log and trace files that the OFTP gateway may produce.

Typical OFTP Error Log Messages

The following is an example of some typical messages logged by the OFTP gateway into the Digital DEC/EDI Error Log file. They are displayed here as you would see them if you were using `decedi_look`.

Each OFTP session is handled by a separate OFTP process. When a new outbound or inbound connection is required a new child process is created by the main parent OFTP process. This is denoted in the Error Log file by the `DECEDI__SRVSTART` messages. The messages also indicates the current number of OFTP sessions currently running, including the one that has just started.

Messages logged in the Error Log by different child servers can be differentiated by the `NAME` element of the logged message. Each child server has an identification of the format `OFTP Server nnnnn`, where nnnnn is the actual PID of the process. In the example below, two child servers are started, one with identification `OFTP Server 30447`, that was created to handle an outbound session for connection id `OUT` and another, with identification `OFTP Server 30535`, that was created to handle a new inbound X.25 session.

```
Tue May 30 14:44:49 1995 PID = 30306 NAME = Outbound Comms
Controller
DECEDI__JOBSUB (i), job submitted to gateway
DECEDI__GATEWAY (i), gateway = OFTP
DECEDI__CONNID (i), connection id = OUT
```

```
Tue May 30 14:44:49 1995 PID = 30447 NAME = OFTP Server 30447
DECEDI__SRVSTART (i), child server started for new outbound
connection OUT
DECEDI__OFTPCHAN (i), channels currently running = 1

Tue May 30 14:44:52 1995 PID = 30535 NAME = OFTP Server 30535
DECEDI__SRVSTART (i), child server started for new inbound
X.25 call
DECEDI__OFTPCHAN (i), channels currently running = 2

Tue May 30 14:44:57 1995 PID = 30447 NAME = OFTP Server 30447
DECEDI__SRVSTOP (i), child server stopped
DECEDI__CONNID (i), connection id = OUT
DECEDI__OFTPCHAN (i), channels currently running = 1

Tue May 30 14:44:57 1995 PID = 30535 NAME = OFTP Server 30535
DECEDI__SRVSTOP (i), child server stopped
DECEDI__CONNID (i), connection id = IN
DECEDI__OFTPCHAN (i), channels currently running = 0
```

Sample Outbound Trace

The following is a typical trace of an outbound OFTP session. A single file is sent from Digital DEC/EDI to the trading partner system. The session includes the trace of a trading partner acknowledging this file with an End-to-End response message (EERP).

```
Successfully read connection record by connection id OUT
Successfully locked connection record
X25: opened X.25 port
X25: encoded parameters...
     DTE Class       = X25-OUT
     DTE Address     = 87654321
     DTE Sub-Address =
     User Data (hex) =
X25: encoded parameter list for making call
X25: X.25 call established using template Default
X25: got X.25 port status
     port status = X25S_RUNNING
STC: State Changed to I_WF_RM (Initiator waiting for SSRM)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 19 byte(s) of data

MSG: <--- IN
     SSRM - Start Session Ready Message
```

```
     SSRMCMD (Command) = I
     SSRMMSG (Message) = ODETTE FTP READY
     SSRMCR  (Carriage Return)

MSG: ---> OUT
     SSID - Start Session ID
     SSIDCMD  (Command)                 = X
     SSIDLEV  (Protocol Version Level)  = 1
     SSIDCODE (Initiators ID code)      = OUT-ID
     SSIDPSWD (Initiators Password)     = OUTBOUND
     SSIDSDEB (Exchange Buffer Size)    = 02048
     SSIDSR   ((S)end (R)eceive (B)oth) = S
     SSIDCMPR (Compression)             = Y
     SSIDREST (Restarts)                = Y
     SSIDSPEC (Special Logic)           = N
     SSIDCRED (Exchange Buffer Credits) = 003
     SSIDRSV1 (Reserved)                = Resv
     SSIDUSER (User Field)              = User
     SSIDCR   (Carriage Return)

X25: sent 61 byte(s) of data
STC: State Changed to I_WF_SSID (Initiator waiting for SSID)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 61 byte(s) of data

MSG: <--- IN
     SSID - Start Session ID
     SSIDCMD  (Command)                 = X
     SSIDLEV  (Protocol Version Level)  = 1
     SSIDCODE (Initiators ID code)      = IN-ID
     SSIDPSWD (Initiators Password)     = INBOUND
     SSIDSDEB (Exchange Buffer Size)    = 02048
     SSIDSR   ((S)end (R)eceive (B)oth) = R
     SSIDCMPR (Compression)             = Y
     SSIDREST (Restarts)                = Y
     SSIDSPEC (Special Logic)           = N
     SSIDCRED (Exchange Buffer Credits) = 003
     SSIDRSV1 (Reserved)                = Resv
     SSIDUSER (User Field)              = User
     SSIDCR   (Carriage Return)

STC: State changed to IDLESP (Idle Speaker)
Looking for files to send...
   - finding next victim
   - loading DATA cursor
   - found next DATA victim
```

```
FSU: updated file status to SENDING, file =
26APR199514113943_OUT

MSG: ---> OUT
     SFID - Start File ID
     SFIDCMD  (Command)                = H
     SFIDDSN  (File Dataset Name)      = 26APR199514113943
     SFIDRSV1 (Reserved)               =
     SFIDDATE (Date YYMMDD)            = 950426
     SFIDTIME (Time HHMMSS)            = 141139
     SFIDUSER (User Field)             =
     SFIDDEST (Destination ID)         = IN-ID
     SFIDORIG (Originator ID)          = OUT-ID
     SFIDFMT  (File Format F/V/U/T)    = V
     SFIDRECL (Maximum Record Size)    = 00132
     SFIDSIZ  (File Size in 1K Blocks) = 0000001
     SFIDREST (Restart Position)       = 000000000

X25: sent 128 byte(s) of data
Opened transmission file for read
/var/adm/decedi/store_1/26APR199514113943_OUT.TRANSMISSION
STC: State Changed to OPOP (Waiting for SFPA/SFNA after
sending SFID)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 10 byte(s) of data

MSG: <--- IN
     SFPA - Start File Positive Answer
     SFPACMD  (Command)      = 2
     SFPAACNT (Answer Count) = 000000000

STC: State changed to OPENO (Sending Data)
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 132 byte(s) from transmission file
Read 51 byte(s) from transmission file
```

```
Read EOF from transmission file

MSG: ---> OUT
     DATA - Data Exchange Buffer (Length = 1943)
     Command = D
     SR (63)       :
UNB+UNOA:1+RTS_AP_INT_ID+RTS_PT_INT_ID+950519:0943+000003557+R
T
     SR (63)       :
S_AUTH_CODE+MIXXED'UNG+PAYDUC+RTS_AP_GRP_ID+RTS_PT_GRP_ID+9505
1
     SR (06)   EOR : 9:0943
     SR (63)       :
+000003557+UN+1:921'UNH+00000355700001+PAYDUC:1:921:UN+RTS-
PAYD
     SR (63)       :
UC-100'BGM+1:100:1:Document/message name+Document/message
numbe
     SR (06)   EOR : r+1+AA
     SR (63)       :
'PAI+1:10:1:100:1:1'FII+AA+Account holder nu:Account holder
nam
     SR (63)       :
e:Account holder name:Cur+Institution:100:1:Institution
branc:1
     SR (06)   EOR : 00:1:I
     SR (63)       :
nstitution name:Institution branch
place+Cou'DTM+10:Date/time/p
     SR (63)       :
eriod:101'CUX+1:Cur:1:1288+1:Cur:1:1288+128899+AAA'FTX+AAA+1+F
r
     SR (06)   EOR : e:100:
     SR (63)       :
1+Free text:Free text:Free text:Free text:Free
text+Lan'RFF+AAA
     SR (63)       : :
Reference number:Line n:Reference version number'DTM+10:Date/t
     SR (06)   EOR : ime/pe
     SR (63)       :
riod:101'NAD+AA+Party id identifi:100:1+Name and address
line:N
     SR (63)       :
ame and address line:Name and address line:Name and address
lin
     SR (06)   EOR : e:Name
     SR (63)       :
and address line+Party name:Party name:Party name:Party name:P
     SR (63)       :
arty name:1+Street and number/P.O. Box:Street and number/P.O.
B
     SR (06)   EOR : ox:Str
```

```
     SR (63)       :
eet and number/P.O. Box+City name+Country
s+Postcode+Cou'CTA+AA
     SR (63)       :
+Department or emp:Department or employee'COM+Communication
num
     SR (06)   EOR : ber:AA
     SR (63)       :
'GIS+1:100:1:1'RFF+AAA:Reference number:Line n:Reference
versio
     SR (63)       : n
number'MOA+1:128.99:Cur:1:1'BUS+1:ADV:100:1:Business descript
     SR (06)   EOR : ion+DO
     SR (63)       :
+1+ABX:100:1+1'CUX+1:Cur:1:1288+1:Cur:1:1288+128899+AAA'DTM+10
:
     SR (63)       :
Date/time/period:101'NAD+AA+Party id identifi:100:1+Name and
ad
     SR (06)   EOR : dress
     SR (63)       :
line:Name and address line:Name and address line:Name and
addre
     SR (63)       :
ss line:Name and address line+Party name:Party name:Party
name:
     SR (06)   EOR : Party
     SR (63)       :
name:Party name:1+Street and number/P.O. Box:Street and
number/
     SR (63)       :
P.O. Box:Street and number/P.O. Box+City name+Country
s+Postcod
     SR (06)   EOR : e+Cou'
     SR (63)       :
RFF+AAA:Reference number:Line n:Reference version
number'MOA+1:
     SR (63)       :
128.99:Cur:1:1'AJT+1+128899'FTX+AAA+1+Fre:100:1+Free text:Free
     SR (06)   EOR : text:F
     SR (63)       :
ree text:Free text:Free
text+Lan'UNS+S'MOA+1:128.99:Cur:1:1'CNT
     SR (63)       :
+1:128899:Mea'AUT+Validation result+Validation key
identificati
     SR (06)   EOR : on'UNT
     SR (51)   EOR :
+28+00000355700001'UNE+1+000003557'UNZ+1+000003557'

X25: sent 1943 byte(s) of data

MSG: ---> OUT
```

```
     EFID - End File ID
     EFIDCMD  (Command)         = T
     EFIDRCNT (Record Count)    = 000000015
     EFIDUCNT (Byte Unit Count) = 000000001899

X25: sent 22 byte(s) of data
STC: State changed to CLOP (Waiting for EFPA/EFNA after
sending EFID)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 2 byte(s) of data

MSG: <--- IN
     EFPA - End File Positive Answer
     EFPACMD (Command)          = 4
     EFPACD  (Change Direction) = Y

Closed transmission file
/var/adm/decedi/store_1/26APR199514113943_OUT.TRANSMISSION
FSU: updated file status to SENT, file = 26APR199514113943_OUT

MSG: ---> OUT
     CD - Change Direction
     CDCMD (Command) = R

X25: sent 1 byte(s) of data
STC: State changed to IDLELI (Idle listener)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 106 byte(s) of data

MSG: <--- IN
     EERP - End-to-End Response
     EERPCMD  (Command)           = E
     EERPDSN  (File Dataset Name) = 26APR199514113943
     EERPRSV1 (Reserved)          =
     EERPDATE (Date YYMMDD)       = 950426
     EERPTIME (Time HHMMSS)       = 141139
     EERPUSER (User Field)        =
     EERPDEST (Destination ID)    = OUT-ID
     EERPORIG (Originator ID)     = IN-ID

FSU: updated file status to DELIVERED, file =
26APR199514113943_OUT
FSU: updated file status to PURGEABLE, file =
26APR199514113943_OUT

MSG: ---> OUT
```

```
     RTR - Ready to Receive
     RTRCMD (Command) = P

X25: sent 1 byte(s) of data
STC: State changed to IDLELI (Idle Listener)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 1 byte(s) of data

MSG: <--- IN
     CD - Change Direction
     CDCMD (Command) = R

Looking for files to send...
   - finding next victim
   - loading DATA cursor
   - no DATA victim found

MSG: ---> OUT
     ESID - End Session ID
     ESIDCMD (Command)      = F
     ESIDREAS (Reason Code) = 00 (Normal Session Termination)

X25: sent 3 byte(s) of data
STC: State Changed to WFCLEAR (Waiting for X.25 call to be
cleared)
X25: waited for an X.25 event...
     - X25EV_CALLCLEARED event triggered

Executing cleanup routine...
   - clearing error count to zero for connection OUT
   - connection terminated normally

Executing shutdown routine...
   - successfully written back connection record OUT to
database
   - successfully unlocked connection record OUT

X25: closing X.25 port
STC: State Changed to Ending
LOG: DECEDI__OFTPCHAN (i), channels currently running = 1
LOG: DECEDI__CONNID (i), connection id = OUT
LOG: DECEDI__SRVSTOP (i), child server stopped
```

Sample Inbound Trace

The following is a typical trace of an inbound OFTP session. A single file is received from the trading partner system. The session includes the trace of Digital DEC/EDI acknowledging the file just received with an End-to-End response message (EERP).

```
Successfully read connection record by remote OFTP id OUT-ID
   - matched to connection id IN
Successfully locked connection record

MSG: <--- IN
     SSID - Start Session ID
     SSIDCMD  (Command)                  = X
     SSIDLEV  (Protocol Version Level)   = 1
     SSIDCODE (Initiators ID code)       = OUT-ID
     SSIDPSWD (Initiators Password)      = OUTBOUND
     SSIDSDEB (Exchange Buffer Size)     = 02048
     SSIDSR   ((S)end (R)eceive (B)oth)  = S
     SSIDCMPR (Compression)              = Y
     SSIDREST (Restarts)                 = Y
     SSIDSPEC (Special Logic)            = N
     SSIDCRED (Exchange Buffer Credits)  = 003
     SSIDRSV1 (Reserved)                 = Resv
     SSIDUSER (User Field)               = User
     SSIDCR   (Carriage Return)

MSG: ---> OUT
     SSID - Start Session ID
     SSIDCMD  (Command)                  = X
     SSIDLEV  (Protocol Version Level)   = 1
     SSIDCODE (Initiators ID code)       = IN-ID
     SSIDPSWD (Initiators Password)      = INBOUND
     SSIDSDEB (Exchange Buffer Size)     = 02048
     SSIDSR   ((S)end (R)eceive (B)oth)  = R
     SSIDCMPR (Compression)              = Y
     SSIDREST (Restarts)                 = Y
     SSIDSPEC (Special Logic)            = N
     SSIDCRED (Exchange Buffer Credits)  = 003
     SSIDRSV1 (Reserved)                 = Resv
     SSIDUSER (User Field)               = User
     SSIDCR   (Carriage Return)

X25: sent 61 byte(s) of data
STC: State Changed to IDLELI (Idle listener)
X25: waited for an X.25 event...
```

```
     - X25EV_DATA event triggered
X25: read 128 byte(s) of data

MSG: <--- IN
     SFID - Start File ID
     SFIDCMD  (Command)                = H
     SFIDDSN  (File Dataset Name)      = 26APR199514113943
     SFIDRSV1 (Reserved)               =
     SFIDDATE (Date YYMMDD)            = 950426
     SFIDTIME (Time HHMMSS)            = 141139
     SFIDUSER (User Field)             =
     SFIDDEST (Destination ID)         = IN-ID
     SFIDORIG (Originator ID)          = OUT-ID
     SFIDFMT  (File Format F/V/U/T)    = V
     SFIDRECL (Maximum Record Size)    = 00132
     SFIDSIZ  (File Size in 1K Blocks) = 0000001
     SFIDREST (Restart Position)       = 000000000

Receiving new file
FSU: created new communication audit record, file =
30MAY199514445516_IN
Created new transmission file
/var/adm/decedi/store_1/30MAY199514445516_IN.TRANSMISSION

MSG: ---> OUT
     SFPA - Start File Positive Answer
     SFPACMD  (Command)      = 2
     SFPAACNT (Answer Count) = 000000000

X25: sent 10 byte(s) of data
STC: State changed to OPENI (Receiving Data)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 1943 byte(s) of data

MSG: <--- IN
     DATA - Data Exchange Buffer (Length = 1943)
     Command = D
     SR (63)       :
UNB+UNOA:1+RTS_AP_INT_ID+RTS_PT_INT_ID+950519:0943+000003557+R
T
     SR (63)       :
S_AUTH_CODE+MIXXED'UNG+PAYDUC+RTS_AP_GRP_ID+RTS_PT_GRP_ID+9505
1
     SR (06)   EOR : 9:0943
     SR (63)       :
+000003557+UN+1:921'UNH+00000355700001+PAYDUC:1:921:UN+RTS-
PAYD
```

```
     SR (63)       :
UC-100'BGM+1:100:1:Document/message name+Document/message
numbe
     SR (06)   EOR : r+1+AA
     SR (63)       :
'PAI+1:10:1:100:1:1'FII+AA+Account holder nu:Account holder
nam
     SR (63)       :
e:Account holder name:Cur+Institution:100:1:Institution
branc:1
     SR (06)   EOR : 00:1:I
     SR (63)       :
nstitution name:Institution branch
place+Cou'DTM+10:Date/time/p
     SR (63)       :
eriod:101'CUX+1:Cur:1:1288+1:Cur:1:1288+128899+AAA'FTX+AAA+1+F
r
     SR (06)   EOR : e:100:
     SR (63)       :
1+Free text:Free text:Free text:Free text:Free
text+Lan'RFF+AAA
     SR (63)       : :
Reference number:Line n:Reference version number'DTM+10:Date/t
     SR (06)   EOR : ime/pe
     SR (63)       :
riod:101'NAD+AA+Party id identifi:100:1+Name and address
line:N
     SR (63)       :
ame and address line:Name and address line:Name and address
lin
     SR (06)   EOR : e:Name
     SR (63)       :
and address line+Party name:Party name:Party name:Party name:P
     SR (63)       :
arty name:1+Street and number/P.O. Box:Street and number/P.O.
B
     SR (06)   EOR : ox:Str
     SR (63)       :
eet and number/P.O. Box+City name+Country
s+Postcode+Cou'CTA+AA
     SR (63)       :
+Department or emp:Department or employee'COM+Communication
num
     SR (06)   EOR : ber:AA
     SR (63)       :
'GIS+1:100:1:1'RFF+AAA:Reference number:Line n:Reference
versio
     SR (63)       :
n number'MOA+1:128.99:Cur:1:1'BUS+1:ADV:100:1:Business
descript
     SR (06)   EOR : ion+DO
```

```
     SR (63)       :
+1+ABX:100:1+1'CUX+1:Cur:1:1288+1:Cur:1:1288+128899+AAA'DTM+10
:
     SR (63)       :
Date/time/period:101'NAD+AA+Party id identifi:100:1+Name and
ad
     SR (06)   EOR : dress
     SR (63)       :
line:Name and address line:Name and address line:Name and
addre
     SR (63)       :
ss line:Name and address line+Party name:Party name:Party
name:
     SR (06)   EOR : Party
     SR (63)       :
name:Party name:1+Street and number/P.O. Box:Street and
number/
     SR (63)       :
P.O. Box:Street and number/P.O. Box+City name+Country
s+Postcod
     SR (06)   EOR : e+Cou'
     SR (63)       :
RFF+AAA:Reference number:Line n:Reference version
number'MOA+1:
     SR (63)       :
128.99:Cur:1:1'AJT+1+128899'FTX+AAA+1+Fre:100:1+Free text:Free
     SR (06)   EOR : text:F
     SR (63)       :
ree text:Free text:Free
text+Lan'UNS+S'MOA+1:128.99:Cur:1:1'CNT
     SR (63)       :
+1:128899:Mea'AUT+Validation result+Validation key
identificati
     SR (06)   EOR : on'UNT
     SR (51)   EOR :
+28+00000355700001'UNE+1+000003557'UNZ+1+000003557'

Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
Wrote 132 bytes to transmission file
```

```
Wrote 51 bytes to transmission file
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 22 byte(s) of data

MSG: <--- IN
     EFID - End File ID
     EFIDCMD  (Command)         = T
     EFIDRCNT (Record Count)    = 000000015
     EFIDUCNT (Byte Unit Count) = 000000001899

Byte count OK in EFID received
FSU: updated file status to RECEIVED, file =
30MAY199514445516_IN
Closed transmission file
/var/adm/decedi/store_1/30MAY199514445516_IN.TRANSMISSION
Looking for files to acknowledge...
   - finding next victim
   - loading EERP cursor
   - found next EERP victim

MSG: ---> OUT
     EFPA - End File Positive Answer
     EFPACMD (Command)          = 4
     EFPACD  (Change Direction) = Y

X25: sent 2 byte(s) of data
STC: State changed to WFCD (Waiting for change direction)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 1 byte(s) of data

MSG: <--- IN
     CD - Change Direction
     CDCMD (Command) = R

Looking for files to acknowledge...
   - finding next victim
   - loading EERP cursor
   - found next EERP victim

MSG: ---> OUT
     EERP - End-to-End Response
     EERPCMD  (Command)           = E
     EERPDSN  (File Dataset Name) = 26APR199514113943
     EERPRSV1 (Reserved)          =
     EERPDATE (Date YYMMDD)       = 950426
     EERPTIME (Time HHMMSS)       = 141139
```

```
     EERPUSER (User Field)        =
     EERPDEST (Destination ID)    = OUT-ID
     EERPORIG (Originator ID)     = IN-ID

X25: sent 106 byte(s) of data
STC: State Changed to WFRTR (Waiting for RTR after sending
EERP)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 1 byte(s) of data

MSG: <--- IN
     RTR - Ready to Receive
     RTRCMD (Command) = P

FSU: updated file status to ACKED, file = 30MAY199514445516_IN
Notifying CC of new inbound file, file = 30MAY199514445516_IN

Looking for files to acknowledge...
   - finding next victim
   - loading EERP cursor
   - no EERP victim found

MSG: ---> OUT
     CD - Change Direction
     CDCMD (Command) = R

X25: sent 1 byte(s) of data
STC: State Changed to IDLELI (Idle Listener)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 3 byte(s) of data

MSG: <--- IN
     ESID - End Session ID
     ESIDCMD (Command)      = F
     ESIDREAS (Reason Code) = 00 (Normal Session Termination)

End Session Request (ESID) Received - normal termination

X25: X.25 call cleared
STC: State Changed to WFCLEAR (Waiting for X.25 call to be
cleared)
X25: waited for an X.25 event...
     - X25EV_CALLCLEARED event triggered

Executing cleanup routine...
   - clearing error count to zero for connection IN
```

```
   - connection terminated normally

Executing shutdown routine...
   - successfully written back connection record IN to database
   - successfully unlocked connection record IN

X25: closing X.25 port
STC: State Changed to Ending
LOG: DECEDI__OFTPCHAN (i), channels currently running = 0
LOG: DECEDI__CONNID (i), connection id = IN
LOG: DECEDI__SRVSTOP (i), child server stopped
```

Extract of an OFTP Error Trace

The trace below shows a SFID (Start File Request) being rejected with a SFNA (Start File Negative Answer) with a reason code of 13 (Duplicate File). At the end you can also see that this error to send the file has resulted in the file retry count being incremented by one. This has exceeded the specified retry limit and the transmission file is failed as a result.

```
...
STC: State changed to IDLESP (Idle Speaker)
Looking for files to send...
   - finding next victim
   - loading DATA cursor
   - found next DATA victim

FSU: updated file status to SENDING, file =
26APR199514113943_OUT

MSG: ---> OUT
     SFID - Start File ID
     SFIDCMD  (Command)               = H
     SFIDDSN  (File Dataset Name)     = 26APR199514113943
     SFIDRSV1 (Reserved)              =
     SFIDDATE (Date YYMMDD)           = 950426
     SFIDTIME (Time HHMMSS)           = 141139
     SFIDUSER (User Field)            =
     SFIDDEST (Destination ID)        = IN-ID
     SFIDORIG (Originator ID)         = OUT-ID
     SFIDFMT  (File Format F/V/U/T)   = V
     SFIDRECL (Maximum Record Size)   = 32767
     SFIDSIZ  (File Size in 1K Blocks) = 0000001
     SFIDREST (Restart Position)      = 000000000

X25: sent 128 byte(s) of data
```

```
Opened transmission file for read
/usr/users/mrfat1/store_2/26APR199514113943_OUT.TRANSMISSION
STC: State Changed to OPOP (Waiting for SFPA/SFNA after
sending SFID)
X25: waited for an X.25 event...
     - X25EV_DATA event triggered
X25: read 4 byte(s) of data

MSG: <--- IN
     SFNA - Start File Negative Answer
     SFNACMD  (Command)       = 3
     SFNAREAS (Answer Reason) = 13 (Duplicate File)
     SFNARRTR (Answer Retry)  = N

LOG: DECEDI__FNAR13 (i), duplicate file
LOG: DECEDI__TFILENAME (i), transmission file :
26APR199514113943_OUT
LOG: DECEDI__CONNID (i), connection id = OUT
LOG: DECEDI__OFTPSTATE (i), OFTP state = Waiting for SFPA/SFNA
after sending SFID
LOG: DECEDI__STRTREF (w), start file refused

Executing cleanup routine (error condition)...
   - closed transmission file
/usr/users/mrfat1/store_2/26APR199514113943_OUT.TRANSMISSION
LOG: DECEDI__TFILENAME (i), transmission file :
26APR199514113943_OUT
LOG: DECEDI__RETRYEXC (w), file retry count exceeded
   - setting transmission to Failed to Send for
26APR199514113943_OUT
   - file retry count has been exceeded
   - incrementing error count for connection OUT
Looking for files to send...
   - finding next victim
   - loading DATA cursor
   - no DATA victim found
...
```

Extract of an X.25 Error Trace

The following trace shows an error while attempting to make an X.25 call to a trading partner.

The X.25 call attempted to use a DTE Class of `X25-OUT`, a DTE Address of `99999999`, a DTE Sub-Address of `99` and Call User Data of `72000000`. The attempted call was using an X.25 template named `Does_not_exist`.

The attempt to make an X.25 call failed because the specified X.25 template does not exist.

```
...
Successfully read connection record by connection id OUT2
Successfully locked connection record
X25: opened X.25 port
X25: encoded parameters...
     DTE Class       = X25-OUT
     DTE Address     = 99999999
     DTE Sub-Address = 99
     User Data (hex) = 72 0 0 0
X25: encoded parameter list for making call
X25: error making X.25 call using template Does_not_exist
LOG: DECEDI__X25ERROR (e), Specified template does not exist
LOG: DECEDI__SYSSRVERR (e), error calling the system service
routine X25MakeCall
LOG: DECEDI__MAKECALLERR (e), error establishing X.25 call
LOG: DECEDI__CONNID (i), connection id = OUT2
LOG: DECEDI__OFTPSTATE (i), OFTP state = Starting
LOG: DECEDI__OFTPOUTERR (e), error processing new outbound
connection

Executing shutdown routine...
   - successfully written back connection record OUT2 to
database
   - successfully unlocked connection record OUT2

X25: closing X.25 port
STC: State Changed to Ending
LOG: DECEDI__OFTPCHAN (i), channels currently running = 0
LOG: DECEDI__CONNID (i), connection id = OUT2
LOG: DECEDI__SRVSTOP (i), child server stopped
```

Appendix F Sample SMTP/MIME Logs and Traces

This appendix contains some sample trace files that the SMTP gateway can produce. Trace files are produced by the SMTP gateway for all connections that have trace enabled.

Trace files are created in the directory: `/var/adm/decedi/logs/`, and have the following name format:

`SMTP_ABBBBBBBCCCC_<connid>.TRACE`

where `ABBBBBBBCCCC` is a date/time stamp.

Sample SMTP Gateway Outbound Trace (Security Type: None)

The following is a typical trace of an outbound session. A single file is sent in a loopback fashion from Digital DEC/EDI back to Digital DEC/EDI.

```
Successfully read connection record by connection id SMTP
Successfully locked connection record
Looking for files to send...
    - loading transmission file cursor
Processing transmission : 11APR199612350241_SMTP
Updated file status to Sending
Parsing the Connection Specific Data (CSD)
    - 6 found
    - CSD1 =
    - CSD2 = EDI System
    - CSD3 =
    - CSD4 =
    - CSD5 =
    - CSD6 =
```

F-2 *Sample SMTP Gateway Outbound Trace (Security Type: None)*

```
        - CSD7 =
        - CSD8 =
        - CSD9 =
    Building standard RFC1767 message
        - opened .ENVELOPE file
          /var/adm/decedi/smtp/11APR199612350241_SMTP.EVELOPE
        - processing Date: header
          added header (Date: Thu, 11 Apr 1996 12:35:06+0000)
        - processing From: header ()
          From: header in SCF is blank
          using From: header in SPA (edimail@widgets.couk)
          added header (From: edimail@widgets.co.uk)
        - processing Subject: header (#TRF)
          added header (Subject: 11APR199612350241_SMTP
        - processing Sender: header (edimail@widgets.couk)
          added header (Sender: edimail@widgets.co.uk)
        - reply field is empty; not adding Reply-To: heder
        - processing To: header (#CSD2 <edimail@widgetsco.uk>)
          added header (To: EDI System <edimail@widgetsco.uk>)
        - processing cc: header (user@audit.org)
          added header (cc: user@audit.org)
        - bcc field is empty; not adding bcc: header
        - processing MIME-Version: header (1.0)
          added header (MIME-Version: 1.0)
        - header field 1 is empty; not adding header 1
        - header field 2 is empty; not adding header 2
        - header field 3 is empty; not adding header 3
        - header field 4 is empty; not adding header 4
        - header field 5 is empty; not adding header 5
        - processing Message-ID: header (<#TRF@widgets.o.uk>)
          added header
          (Message-ID: <11APR199612350241_SMTP@widgets.co.uk>)
        - closed .ENVELOPE file
          /var/adm/decedi/smtp/11APR199612350241_SMTP.ENVELOPE
        - opened .HEADER file
          /var/adm/decedi/smtp/11APR199612350241_SMTP.HEADER
        - processing content type ()
          using content type ()
          processing content-type parameters ()
          no content-type parameters specified; leaving blank
          content type is blank, not adding Content-Type: header
        - processing content encoding (Quoted-Printable)
          using content encoding (Quoted-Printable)
          added header (Content-Transfer-Encoding: Quoted
                                                  -Printable)
        - added blank line to .HEADER file
        - closed .HEADER file
          /var/adm/decedi/smtp/11APR199612350241_SMTP.HEADER
```

```
    - created .RAW file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.RAW
Checking if external processing required...
    - external processing not required
    - created .DATA file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.DATA
Encoding data into required format
     - encoding to Quoted-Printable
    - opened input file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.DATA
    - opened output file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.CONTENT
    - file encoded to Quoted Printable
      old char count = 2375
      new char count = 2445
    - created .CONTENT file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.CONTENT
Building complete .MESSAGE file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
    - copied .ENVELOPE to .MESSAGE file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
    - appended .HEADER to .MESSAGE file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
    - appended .CONTENT to .MESSAGE file
      /var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
Posting message using sendmail (Interactive)
    - message file =
      /var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
    - command =
      /usr/sbin/sendmail -v -odi -f edimail -t
      </var/adm/decedi/smtp/11APR199612350241_SMTP.MESSAGE
      >>/var/adm/decedi/logs/SMTP_11APR199612350571SMTP.TRACE
    - first check addresses, then post if OK
EDI System <edimail@widgets.co.uk>... deliverable
user@audit.org... deliverable
EDI System <edimail@widgets.co.uk>... Connecting t  (local)...
EDI System <edimail@widgets.co.uk>... Sent
user@audit.org... Connecting to audit.org (smtp)..
220 audit.org ESMTP Sendmail 8.7.3/8.7; Thu, 11 Apr 1996
14:37:10 +0200
>>> HELO widgets.co.uk
250 audit.org Hello widgets.co.uk [16.36.112.184],pleased to
meet you
>>> MAIL From:<edimail@widgets.co.uk>
250 <edimail@widgets.co.uk>... Sender ok
>>> RCPT To:<user@audit.org>
250 Recipient ok
>>> DATA
```

```
354 Enter mail, end with "." on a line by itself
>>> .
250 OAA06761 Message accepted for delivery
>>> QUIT
221 audit.org closing connection
user@audit.org... Sent
    - posted message successfully
Updated file status to Sent
Saving message file into the decedi store directory
    - message file saved
Updated file status to Delivered
Updated file status to Purgeable
Rippled transmission status to documents
Notified Archive Server of completed file
Transmission processed successfully
    - clearing error count to zero for connection SMTP
Cleaning up, ready for next file
    - closing all files
    - deleting temporary work files in smtp directory
About to update connection record SMTP
    - successfully written back connection record
                                        SMTP to database
    - successfully unlocked connection record SMTP
Trace file closed
```

Sample SMTP Gateway Inbound Trace (Security Type: None)

The following is a typical trace of an inbound session. A single file is received from Digital DEC/EDI in a loopback fashion. Notice that the trace file starts by listing all headers found in the inbound message. The inbound trace file starts at the point when the `From:` address is successfully matched to a connection id.

```
Date: Thu, 11 Apr 1996 12:35:06 +0000
From: edimail@widgets.co.uk
Subject: 11APR199612350241_SMTP
Sender: edimail@widgets.co.uk
To: EDI System <edimail@widgets.co.uk>
Cc: user@audit.org
Mime-Version: 1.0
Message-Id: <11APR199612350241_SMTP@widgets.co.uk>
Content-Transfer-Encoding: Quoted-Printable
  Successfully read connection record by remote alias
     - matched to connection id SMTP
```

```
  Successfully locked connection record
  Initializing new audit record
  Found no content type
     - using default SCF value EDIF
  Created new communication audit record,
     file = 11APR199612351288_SMTP
  Decoding data
     - decoding from Quoted-Printable
     - opened input file
       /var/adm/decedi/smtp/11APR199612351288_CONNID.CONTENT
     - opened output file
       /var/adm/decedi/smtp/11APR199612351288_CONNID.DATA
     - file decoded from Quoted Printable
       old char count = 2445
       new char count = 2375
     - created .DATA file
       /var/adm/decedi/smtp/11APR199612351288_CONNID.DATA
  Checking if external processing required...
     - external processing not required
     - created .RAW file
       /var/adm/decedi/smtp/11APR199612351288_CONNID.RAW
     - created .TRANSMISSION

/var/adm/decedi/store_1/11APR199612351288_SMTP.TRANSMISSION
  Updated file status to Received
  Saving message file into the decedi store directories
     - message file saved
  Deleting inbound message

/var/adm/decedi/smtp_store/11APR199612351288_CONNID.MESSAGE
     - deleted processed file
  Transmission processed successfully
     - clearing error count to zero for connection SMTP
  About to update connection record SMTP
     - successfully written back connection record
                                              SMTP to database
     - successfully unlocked connection record SMTP
  Cleaning up...
     - closing all other files
     - deleting temporary work files in smtp directory
  Trace file closed
```

Sample SMTP Gateway Outbound Trace (Security Type: External)

The following is an extract of a typical outbound trace when *Security Type* has been set to `External`.

```
.
.
.
Checking if external processing required...
   - external processing required
   - processing outbound substitution tags
   - initial command = cp #IF #OF
   - final command =
     cp /var/adm/decedi/smtp/16APR199612513845_SMTP.RAW
 /var/adm/decedi/smtp/16APR199612513845_SMTP.DATA
   - external command executed
   - checking if .DATA file produced
   - .DATA file produced
Encoding data into required format
   - encoding to Base64
   - opened input file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.DATA
   - opened output file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.CONTENT
   - file encoded to Base64
     old char count = 2375
     new char count = 3260
   - created .CONTENT file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.CONTENT
Building complete .MESSAGE file
/var/adm/decedi/smtp/16APR199612513845_SMTP.MESSAGE
   - copied .ENVELOPE to .MESSAGE file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.MESSAGE
   - appended .HEADER to .MESSAGE file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.MESSAGE
   - appended .CONTENT to .MESSAGE file
     /var/adm/decedi/smtp/16APR199612513845_SMTP.MESSAGE
Posting message using sendmail (Interactive)
.
.
.
```

Sample SMTP Gateway Inbound Trace (Security Type: External)

The following is an extract of a typical inbound trace when *Security Type* has been set to `External`.

```
.
.
.
Created new communication audit record,
   file = 16APR199612514622_SMTP
Decoding data
   - decoding from Base64
   - opened input file
     /var/adm/decedi/smtp/16APR199612514622_CONNID.CONTENT
   - opened output file
     /var/adm/decedi/smtp/16APR199612514622_CONNID.DATA
   - file decoded from Base64
     old char count = 3214
     new char count = 2375
   - created .DATA file
     /var/adm/decedi/smtp/16APR199612514622_CONNID.DATA
Checking if external processing required...
   - external processing required
   - processing inbound substitution tags
   - initial command = cp #IF #OF
   - final command = cp
 /var/adm/decedi/smtp/16APR199612514622_CONNID.DATA
 /var/adm/decedi/smtp/16APR199612514622_CONNID.RAW
   - external command executed
   - checking if .RAW file produced
   - .RAW file produced
   - created .TRANSMISSION

/var/adm/decedi/store_1/16APR199612514622_SMTP.TRANSMISSION
                   Updated file status to Received
```

Appendix G Sample 3780 Logs and Traces

This appendix contains some sample trace files that the 3780 gateway can produce. Trace files are produced by the 3780 gateway for all connections that appropriate trace options enabled.

Example of the 3780Plus Log File

```
# cat /var/adm/decedi/3780Plus/GEIS_3780_12-JAN-1996-10-51-
23.LOG

Copyright 1983, 84, 87, 89, 90, 92 CLEO Communications, Inc.

13:19:21  **********  3780  LOGON  **********  Fri Feb  2, 1996
    3780Plus (R. 12044) under UNIX with the IAPI available.
    ASCII/EBCDIC Table changed to:
           /huge/users/cleo_kit/asciiebc.ovr
    Running job file  '/var/adm/decedi/3780Plus/geis_job.dat'
13:19:21  Command> EX /var/adm/decedi/3780Plus/geis_job.dat
13:19:39  Command> ##
13:19:39  Command> ## 3780Plus Job File for connecting GEIS
13:19:39  Command> ##
13:19:39  Command> CONFIG
           /var/adm/decedi/3780Plus/decedi_GEIS_config.dat
    Configuration changed to:
           '/var/adm/decedi/3780Plus/decedi_GEIS_config.dat'
13:20:06  Command> TABLE /huge/users/cleo_kit/asciiebc.ovr
           /huge/users/cleo_kit/ebcascii.ovr
    ASCII/EBCDIC Table changed to:
           /huge/users/cleo_kit/asciiebc.ovr
    EBCDIC/ASCII Table changed to:
           /huge/users/cleo_kit/ebcascii.ovr
13:20:06  Command> MONITOR
           /var/adm/decedi/3780Plus/geis_monitor.log
    Monitor ON - Saving protocol to:
           /var/adm/decedi/3780Plus/geis_monitor.log
```

```
13:20:06  Command> ##
13:20:06  Command> AUTODIAL T 9 T5 031819656776 R005
    Tone  Dialing  B
    Tone  Dialing  B
    Tone  Dialing  B
    Tone  Dialing  B
    Tone  Dialing  B
      Unable to Detect Answer Tone
13:23:16         DIAL  : BUSY               : Connect Code = 7
    Awaiting Telephone Connection
13:23:18  Command> BRANCH ON FAIL TO 101
    Executing Branch . . .
13:23:18  Command> 101 LOG %%% 3780Plus Autodial Failed %%%
13:23:18  Command> VOICE
13:23:30  Command> QUIT 101
13:23:30  **********  3780 LOGOFF  **********  Fri Feb  2, 1996
```

Example Extract from a Monitor Log File

```
# cat \
     /var/adm/decedi/3780Plus/geis_mon_2_feb_1996_09_46_32.log
>A T S 0 = 0
<0
>A T S 7 = 3 0
<0
>A T   D T 9 , 0 1 8 1 9 6 5 6 7 7 6
<
>
<
>A T   D T 9 , 0 1 8 1 9 6 5 6 7 7 6
<1 2
<
>SY SY SY SY EQ PD PD PD
<
>SY SY SY SY EQ PD PD PD
<
>SY SY SY SY EQ PD PD PD
<
>SY SY SY SY EQ PD PD PD
<
>SY SY SY SY EQ PD PD PD
<SY DL A0
<
>SY SY SY SY SX C1 D1 C1 F8 F3 F8 F0 F3 6B D1 C1 C3 D2 D7 D6 E3
6B D4 C1 C9 D3
C1 GS 79 5C RS EB CC D4 FF FF FF
<SY DL WK
```

```
>SY SY SY SY EQ PD PD PD
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL A1
>SY SY SY SY SX 5C D3 E3 C9 C4 40 D4 C1 C9 D3 C2 D6 E7 C1 6B C3
D7 E4 D5 C3 C8
6B D4 C9 D5 C9 RS 5C D4 D6 C4 C5 40 C9 D5 D7 E4 E3 4D D6 E4 E3
D7 E4 E3 4D C8
C9 E2 E3 F8 F0 F3 C1 5D 6B D3 C9 E2 E3 5D 6B E6 C1 C9 E3 RS EB
36 C5 FF FF FF
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL A0
>SY SY SY SY SX 5C C4 C1 E3 C1 40 C4 D6 C3 E2 F8 F0 F3 4D D7 E4
D9 C5 6B C1 E2
C3 C9 C9 5D RS EB F8 4E FF FF FF
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL A1
>SY SY SY SY SX E4 D5 C2 4E E4 D5 D6 C1 i7A F1 4E C4 C5 C3 E2
C5 D5 C4 4E C4 C5
C3 D9 C5 C3 C5 C9 E5 C5 4E F9 F6 F0 F2 F0 F2 7A F0 F9 F3 F9 4E
F0 F0 F0 F0 F0
F0 F0 F3 F8 7D E4 D5 C8 4E F0 F0 F0 F0 F0 F0 F0 F0 F3 F8 F0 F0 F0
F0 F1 4E C3 D9
C5 C1 C4 E5 7A F1 7A RS EB 15 2E FF FF FF
<SY DL WK
>SY SY SY SY EQ PD PD PD
<SY DL A0
>SY SY SY SY SX F9 F2 F1 7A E4 D5 4E D9 E3 E2 60 C3 D9 C5 C1 C4
E5 60 F1 F0 F0
7D C2 C7 D4 4E F1 7A F1 F0 F0 7A F1 7A C4 96 83 A4 94 85 95 A3
61 94 85 A2 A2
81 87 85 40 95 81 94 85 4E C4 96 83 A4 94 85 95 A3 61 94 85 A2
A2 81 87 85 40
95 A4 94 82 85 99 4E RS EB 82 20 FF FF FF
<SY DL A1
>SY SY SY SY SX F1 4E C1 C1 7D C4 E3 D4 4E F1 F0 7A C4 81 A3 85
61 A3 89 94 85
61 97 85 99 89 96 84 7A F1 F0 F1 7D D9 C6 C6 4E C1 C1 C1 7A D9
85 86 85 99 85
95 83 85 40 95 A4 94 82 85 99 7A D3 89 95 85 40 95 7A D9 85 86
85 99 85 95 83
85 40 A5 85 99 A2 89 RS EB A4 5D FF FF FF
```

```
<SY DL A0
     .
     .
     .
>SY SY SY SY SX 7A C6 99 85 85 40 A3 85 A7 A3 7A C6 99 85 85 40
A3 85 A7 A3 4E
D3 81 95 7D C1 E4 E3 4E E5 81 93 89 84 81 A3 89 96 95 40 99 85
A2 A4 93 A3 4E
E5 81 93 89 84 81 A3 89 96 95 40 92 85 A8 40 89 84 85 95 A3 89
86 89 83 81 A3
89 96 95 7D C4 E3 D4 RS EB 27 5A FF FF FF
<SY DL A0
>SY SY SY SY SX 4E F1 F0 7A C4 81 A3 85 61 A3 89 94 85 61 97 85
99 89 96 84 7A
F1 F0 F1 7D E4 D5 E3 4E F3 F8 4E F0 F0 F0 F0 F0 F0 F0 F3 F8 F0
F0 F0 F0 F1 7D
E4 D5 E9 4E F1 4E F0 F0 F0 F0 F0 F0 F0 F3 F8 7D RS 5C C5 D6 C6
RS EB CF 17 FF
FF FF
<SY DL A1
>SY SY SY SY SX 61 C5 C4 E7 E2 D5 C4 40 C4 D6 C3 E2 F8 F0 F3 RS
61 C5 C4 E7 D9
C3 E5 RS 5C C5 D6 E2 RS EX 63 73 FF FF FF
<SY DL A0
<
TT  : /var/adm/decedi/3780Plus/geis_login.dat : NRMEOF :    34
blocks
>SY SY SY SY ET PD PD PD
<SY EQ
>SY SY SY SY DL A0 PD PD PD
<SY SX 11 RS EB D8 7F
>SY SY SY SY DL A1 PD PD PD
<SY SX 27 C1 D7 D9 D6 C3 C5 E2 E2 C9 D5 C7 40 C9 D5 D7 E4 E3 40
40 RS EX 16 2A
>SY SY SY SY DL A0 PD PD PD
<
PT  : PRINT.000            : NRMEOF :      2 blocks
SY SX 2D
>SY SY SY SY NK PD PD PD
<SY SX 2D
>SY SY SY SY NK PD PD PD
<SY SX 11 RS EB D8 7F
>SY SY SY SY DL A1 PD PD PD
<SY SX 27 C1 RS 27 E3 RS 27 61 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7
E7 E7 E7 E7 E7
E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7
E7 E7 E7 E7 E7
E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7
E7 E7 E7 RS 27
```

```
61 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7 E7
E7 E7 E7 E7 E7
     .
     .
     .
```

Appendix H **SAP Status Generator**

The DEC/EDI SAP Status Generator enables DEC/EDI to provide feedback to an SAP environment on the status of documents sent by SAP through the DEC/EDI environment. As each document makes its way through DEC/EDI, messages are sent to the SAP Status Generator using the *SAP Status Generator's Queue* Memory Queue. The SAP Status Generator captures these messages and, using the configuration information explained below, generates SAP Status IDOC files to update the SAP environment with the status of the documents.

Note that the SAP Status Generator is a separately licensed utility and you should have a valid license before starting the generator.

Starting the Generator

To start the SAP Status Generator, add the following line to the end of the /usr/sbin/decedi_sysetup file after the 'set –a' command.

```
DECEDI_SAP_START=1
```

Now shutdown (using /usr/sbin/decedi_stop) and restart (using /usr/sbin/decedi_start) DEC/EDI and the generator will start.

Setting up the Environment

The SAP Status IDOCs are made available through DEC/EDI's Bypass Translation facility. For each SAP instance that you want IDOCs created for, you should create an appropriately named Import/Export connection and set the Syntax associated with the connection to BYPASS.

For example, if for my first Production SAP Instance whose client number is 001, I would create a new Import/Export connection called PI1001 and give it the following values…

- Syntax : BYPASS
- Service Description : SAP Statuses for production 001
- Import Directory : /var/adm/decedi/temp
- Export Directory : /var/adm/decedi/temp
- Delete on Import
- Leave the Post-Export and Pre-Import flags as unchecked.

You do not have to add any scheduled import or export jobs. This connection is simply used as a reference by the SAP Status Generator and it never actually performs any Import or Export actions.

There is a set of rules that must be followed before DEC/EDI will generate status messages that can be imported into SAP (using either the DEC/EDI SAP Integrator Client or the startrfc command).

Rules that specify entries in the ‘Advanced Options’ refer to the ‘Advanced’ button that is available in any of the following places…

- The Application Definition screens under the Management Services Editor. All documents posted to the system using this Application ID will inherit these advanced attributes.
- The Application-to-Application List screen under the Management Services Editor. Note that the options apply to the highlighted line in the list only. All documents posted to the system using the Application ID, addressed to the target Application ID of the specified Document Type will inherit these advanced attributes.
- The Trading Partner Definition screen in the Trading Partner Editor. All documents (except EDIFACT CONTRL, X12 997 or TDCC 999 documents) sent to this trading partner will inherit these advanced attributes.
- The Trading Partner Documents Definition screen in the Trading Partner Editor. All documents of this type sent to this trading partner will inherit these advanced attributes.

The rules are as follows. Replace nnn and xxxxxxxxxxxx with the appropriate values.

- Under the ‘Advanced’ options, the SAP Client Number must be specified using the format

  ```
  R3_CLIENT_NUMBER=nnn
  ```

- Under the ‘Advanced’ options, the SAP User Name must be specified using the format

  ```
  R3_USER_NAME=xxxxxxxxxxxx
  ```

- Under the ‘Advanced’ options, the SAP Version must be specified using the format

  ```
  R3_VERSION=nnn
  ```

  SAP Versions 2, 3 and 4 and all subversions at the time of writing are currently supported. For example, to specify V4.0F, enter R3_VERSION=40F.

  Note that for SAP V4 status IDOC records, the first 2 characters of the version are appended to the ‘EDI_DS’ identifier in the table name. Make sure the version you specify allows status IDOCs with that table name.

- Under the ‘Advanced’ options, the DEC/EDI Connection ID that should be used when making the Status IDOCs available (see previous section) should be specified using the following format

  ```
  R3_CONNECTION_ID=xxxxxx
  ```

- The SAP IDOC number (a 16 digit reference number) from the EDI_DC record must be saved as the document’s User Reference using the $USERREF global variable. This rule can be overridden however by adding the following line to the /usr/sbin/decedi_sysetup file

  ```
  DECEDI_R3_IGN_USRREF=1
  ```

  Note that disabling this check may result in status messages being delivered for documents not generated by SAP.

Note that if you have multiple definitions, the most unique definition will always override the others (i.e. Trading Partner Documents Definition options will override Trading Partner Definition which will override Application Definitions).

Also note that not all definitions must appear on the same screen; you can have 2 at the Partner level and the other 2 at the Document level.

Customizing the Generator

The actions of the SAP Status Generator can be controlled using the following settings. Each setting should be placed in the /usr/sbin/decedi_sysetup file in the format shown.

Setting	Description
DECEDI_R3_POLL_TIME=nn	The number of minutes between status file generation passes. The default is every 10 minutes.
DECEDI_R3_STATUS_LEVEL=x	The level of status reporting required. Valid values are MINIMUM – SAP status code 12, 16 and 17 are reported to SAP. MINIMUM_OVERDUE - SAP status code 12, 16, 17 and 22 are reported to SAP. NO_OVERDUE - SAP status code 06 -> 12, 16, 17 and 23 are reported to SAP. MAXIMUM- SAP status code 06 -> 12, 16, 17, 22 and 23 are reported to SAP. This is the default. REPEAT_OVERDUE - SAP status code 06 -> 12, 16, 17, 22 and 23 are reported to SAP. Each time a file is generated, status 22 is repeated for any document that has not yet received a function acknowledgment (997 in X12, 999 in TDCC or CONTRL in EDIFACT).

Getting the Status IDOCs

Once the Status IDOCs are generated, you can fetch them using the ‘trade fetch’ command with the ‘link_id=<your-connection-id>’ and ‘-type=transmission_file’ qualifiers and then pass them to SAP using SAP’s startrfc interface.

Alternatively, DEC/EDI has a separately licensed utility called the DEC/EDI SAP Integrator Client which manages the interface with SAP. Please contact your Compaq or Digital Globalsoft representative for details of this utility.

Appendix I **Monitoring DEC/EDI from Tru64**

DEC/EDI includes a utility called the DEC/EDI Memory Queue Viewer that allows you to monitor the running DEC/EDI system from the Tru64 UNIX command line.

Note that some options of this utility use the curses screen library and require that your terminal be a recognized and supported type. If you are running remote terminal emulation software, please remember to set your emulation mode to at least VT100 and to ensure that the TERM environment variable is set to at least ‘vt100’.

To invoke the utility, enter the command /usr/sbin/decedi_memqueue_monitor with one of the following options…

/usr/sbin/decedi_memqueue_viewer –c

Provides a one-shot view of the number of documents and transmission files in each Memory Queue. Each line contains 3 fields as shown below.

```
# decedi_memqueue_viewer -c

    [00] X12 Docs to be Converted........0
    [01] EDIFACT Docs to be Converted....0
    [02] TRADACOMS Docs to be Converted..0
    [03] Building X12 Docs...............0
    [04] Building X12 Docs (express).....0
    [05] Building EDIFACT Docs...........0
    [06] Building EDIFACT Docs (express).0
    [07] Docs to complete communications.0
    [08] TRFs to be Sent.................0
    [09] TRFs to be Acknowledged.........0
    [10] SAP statuses to be generated....0
    [11] Docs to be Acknowledged.........4
    [12] TRFs partly Received............0
```

```
    [13] EDIFACT TRFs to be split........0
    [14] X12 TRFs to be translated.......0
    [15] EDIFACT Docs to be translated...0
    [16] TRADACOMS TRFs to be translated.0
    [17] Docs/TRFs Available.............4
    [18] Docs/TRFs to be Archived........0
    [19] TRFs to be Identified...........0
```

The first column contains the Queue Number in square brackets.

The second column is a description of the relevant queue.

The third column contains the number of documents or transmission files currently in that queue.

```
/usr/sbin/decedi_memqueue_viewer –l <queue number>
```

Provides a short one-shot view of the documents and transmission files in the specified Memory Queue.

```
# decedi_memqueue_viewer -l 11

Listing of queue 11 - Docs to be Acknowledged.........

    Document            SRIRAM-OUT_O_A41HQGKIGJ

    Document            SRIRAM-OUT_O_AVVKQG0715

    Document            SRIRAM-OUT_O_A1OLQGS4B6

    Document            SRIRAM-OUT_O_A2RLQG8L6D
```

```
/usr/sbin/decedi_memqueue_viewer –f <queue number>
```

Provides a long one-shot view of the documents and transmission files in the specified Memory Queue.

```
# decedi_memqueue_viewer -f 11
Full Listing of queue 11 - Docs to be Acknowledged.........
    Document            SRIRAM-OUT_O_A41HQGKIGJ
        Transmission File           :
20011122080400A4QJ30_IMPEXP
        Partner                     : SRIRAM-IN
        Document Type               : MINVOICE
        Test Indicator              : Live
        Character Count             :        680
        Segment Count               :         19
        EDI Standard                : EDIF
        Format Table Version        : 901
        EDI Document Type           : INVOIC
        Document Control Number     : 00000000100001
```

```
        Group Type                  : INVOIC
        Group Control Number        :
        Our Interchange Qual.       :
        Our Interchange ID          : SRI
        Partner's Interchange Qual. :
        Partner's Interchange ID    : SRI
        Interchange Control Number  : 000000001
        Priority                    : Low
        Tracking Document ID        : A41HQGKIGJ
        Tracking Reference          : UNSPECIFIED
        Business Reference 1        :
        Business Reference 2        :
        Business Reference 3        :
        Business Reference 4        :
        Business Reference 5        :
        Storage Directory           : /var/adm/decedi/store_1/
    Document            SRIRAM-OUT_O_AVVKQG0715

    …
```

/usr/sbin/decedi_memqueue_viewer –m

Allows you to monitor the number of documents in each of the Memory Queues. The Queue Description and the number of documents in each queue is divided into 2 columns and refreshed approximately every 2 seconds. Press the q key to quit the monitor.

```
            Compaq DEC/EDI Memory Queue Viewer - Monitor Queue
Mode

                              Fri Nov 30 14:02:09 2001

X12 Docs to be Converted........        TRADACOMS TRFs to be
translated.
EDIFACT Docs to be Converted....        Docs/TRFs
Available.............4
TRADACOMS Docs to be Converted..        Docs/TRFs to be
Archived........
Building X12 Docs...............        TRFs to be
Identified...........
Building X12 Docs (express).....
Building EDIFACT Docs...........
Building EDIFACT Docs (express).
Docs to complete communications.
TRFs to be Sent.................
TRFs to be Acknowledged.........
SAP statuses to be generated....
Docs to be Acknowledged.........4
TRFs partly Received............
EDIFACT TRFs to be split........
```

```
X12 TRFs to be translated.......
EDIFACT Docs to be translated...


Press q to quit
```

```
/usr/sbin/decedi_memqueue_viewer –a
```

Allows you to monitor the DEC/EDI processes that are connected to the Memory Queues. In a TruCluster Server configuration, all processes on all nodes can be viewed. However only the first 128 processes are captured.

The display is split over multiple screens which can be paged through using the n (for next) and p (for previous) keys. The q (for quit) key allows you to exit the monitor.

4 pieces of information are displayed for each process over 2 lines.

The first line lists the processes' UNIX Process ID and the type of process. The second line lists the number of seconds since the process changed its current status and the status of the process.

```
            Compaq DEC/EDI Memory Queue Viewer - Monitor
Process Mode

     85334 PEDI Gateway
                 56 - Idle
     85307 Port Server
                 60 - Idle
     85315 Post/Fetch Server (TCP)
                 60 - idle
     91982 Queue Counter
                150 - Monitoring processes
     85342 SMTP Gateway
                  1 - Reading the mail
     85322 TRADACOMS Converter
               87961 - Idle
     85324 TRADACOMS Translator
               87961 - Idle
     85314 Track Server (TCP)
               87970 - idle
     85328 X12 Converter
               87961 - idle
     85326 X12 TFB
                  1 - Idle
```

The above example shows that the SMTP Gateway, process ID 85342, is reading the mail and has been doing so for 1 second. All of the other processes are idle (except for the monitor process itself, process id 91982).

Index

Numerics

A

B

C

D

G

I

J

M

N

O

P

R

S

T

U

V

W

X