Prime®

X.400 API Development Kit Programmer's Guide

Release 1.0

DOC11234-1LA

X.400 API Development Kit Programmer's Guide

First Edition

Peter Hassall and Liz Parsons

This book documents the use of
The Prime X.400 API Development Kit
at Release 1.0.

Prime Computer, Inc., Prime Park, Natick, MA 01760

PRINTING HISTORY

First Edition (DOC11234-1LA) October, 1988 for Release 1.0

CREDITS

Design: UK Technical Publications
Editorial: John Wells
Project Support: Brian Etherington
Illustration: Kevin Maguire
Document Preparation: Kevin Maguire
Production: Prime Technical Publications production unit

HOW TO ORDER TECHNICAL DOCUMENTS

To order copies of documents, or to obtain a catalog and price list:

United States Customers

Call Prime Telemarketing,
toll free, at 1-800-343-2533,
Monday through Friday,
8:30 a.m. to 5:00 p.m. (EST).

International

Contact your local Prime
subsidiary or distributor.

CUSTOMER SUPPORT

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (Massachusetts)
1-800-541-8888 (Alaska and Hawaii)
1-800-343-2320 (within other states)

For other locations, contact your Prime representative.

SURVEYS AND CORRESPONDENCE

Please comment on this manual using the Reader Response Form provided in the back of this book. Address any additional comments on this or other Prime documents to:

Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, MA 01701

CONTENTS

ABOUT THIS BOOK

The X.400 API Development Kit Programmer's Guide is a reference and guide to the Prime X.400 Application Programming Interface (API), and is for programmers of mail applications using Prime X.400 library routines. The book begins with a brief overview of Prime X.400, describes the function and usage of the Prime X.400 application programming subroutines, and lists the internal data structures.

Chapter Contents

Chapter 1	Introduction, provides an introduction to X.400 protocols, OSI architecture, and Prime X.400 concepts.
Chapter 2	Prime X.400 API Library, contains details of Prime X.400 library subroutines, in easy reference format.

Related Documentation

Other Prime manuals which may be useful are:

- *X.400 API Development Kit Administrator's Guide* (*DOC11232-1LA*)

Other manuals which you may find useful are:

- *CCITT Red Book Volume VIII Fascicle VIII.7, Recommendations X.400 - X.430*

Prime Documentation Conventions

The following conventions are used in command formats, statement formats, and in examples throughout this document. Examples illustrate how you use these commands and statements in typical applications.

Convention	*Explanation*	*Example*
UPPERCASE	In command formats, words in uppercase indicate the names of commands, options, statements, and keywords. Enter them in either uppercase or lowercase.	DISPLAY-USER
lowercase	In command formats, words in lowercase indicate variables for which you must substitute a suitable value.	CONFIG_X400 filename
Abbreviations in option descriptions	If an uppercase word in a command format has an abbreviation, the name and abbreviation are placed within braces.	$\left\{\begin{array}{l}\textbf{-HELP}\\ \textbf{-H}\end{array}\right\}$
Underscore in examples	In examples, user input is underscored but system prompts and output are not.	`OK,display-user user=all`
Angle brackets in messages < >	In messages, text enclosed within angle brackets indicates a variable for which the program substitutes the appropriate value.	`<filename> not found.`
Boldface	When they first appear in text, new terms are entered in boldface.	**applications**
Italics	In text, italics indicate variable user input or emphasis. Where Prime documentation is referred to in text, the title of the manual is entered in italics.	*pathname* the *default* file *Programmer's Guide*
`Monospace`	User examples and program listings are displayed in monospace.	

1 INTRODUCTION

OSI Communications

Open Systems Interconnection (OSI) is a set of internationally recognised recommendations from ISO and CCITT to allow intervendor communication based upon a seven layer architectural model. This model is illustrated in Figure 1-1.

Each layer in the model is defined in terms of the service it provides to the layer above, the service it expects from the layer below, and the protocol used to communicate between equivalent entities within the same layer at different points within a network.

X.400 is a series of protocols that define a store-and-forward Message Handling System (MHS) for the exchange of messages between computer network users. It addresses primarily the requirements of electronic mail applications. X.400 is a Message Handling System application implemented in layer 7 of the OSI Seven Layer Model.

The X.400 series of definitions and protocols define a network logical model to which all X.400-compatible message handling systems must conform. The network logical model consists of two types of software processes, known as Message Transfer Agents (MTAs) and User Agents (UAs). MTAs act as servers for the exchange of messages between X.400 over the network, and UAs interface with users, to provide the message transfer service. In Prime X.400, UAs are implemented by Prime X.400 Applications that use the Prime X.400 services provided by the Application Programming Interface (API).

The Prime X.400 Application Program Interface (API) is a set of library calls that

- Establish a communications path to a Prime X.400 Server process.
- Establish a Prime X.400 session.
- Allocate storage and initialize a Message Envelope or Message Header Data Structure.
- Release storage for Message Envelope, or Message Header Data Structures.
- Add fields to the Message Envelope, or Message Header Data Structures.
- Send an Interpersonal Message using these Message Envelopes, and Message Header Data Structures.

- Request that incoming Message, Delivery and Receipt Indications be read.
- Fetch individual fields from the Message Envelope and Message Header data structures.
- Positively or negatively acknowledge receipt of Mail.
- Terminate a Prime X.400 session.
- Terminate a communication path to a Prime X.400 Server process.
- Decode/encode body files.

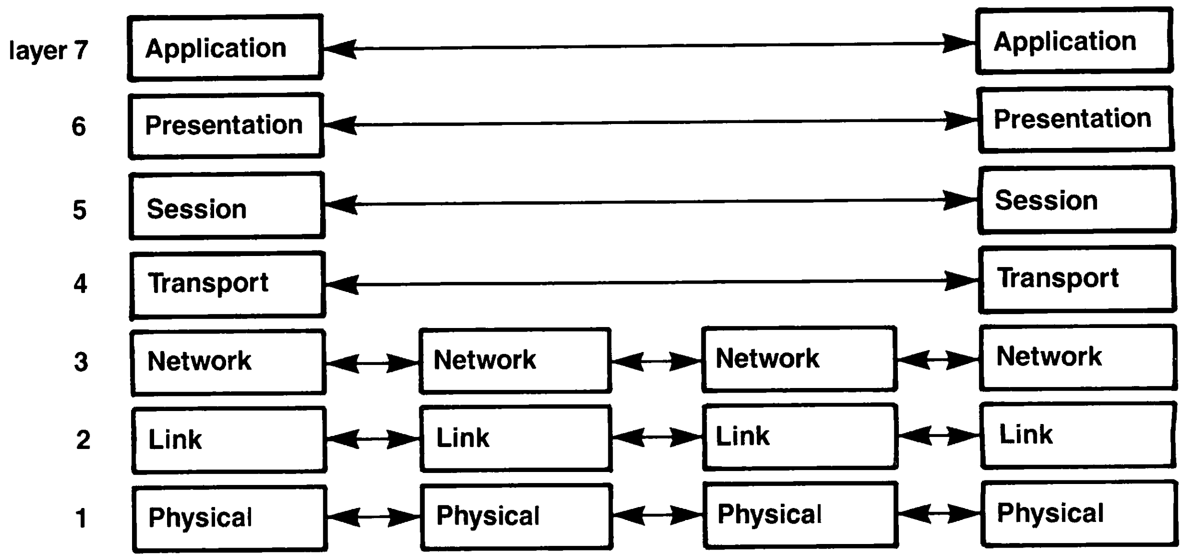

FIGURE 1-1. The OSI Seven Layer Model

X.400

CCITT recommendations X.400-X.430 formed the basis for the design of the Message Handling System which is implemented in layer 7 of the OSI model. Figure 1-2 illustrates the architectural layers of X.400.

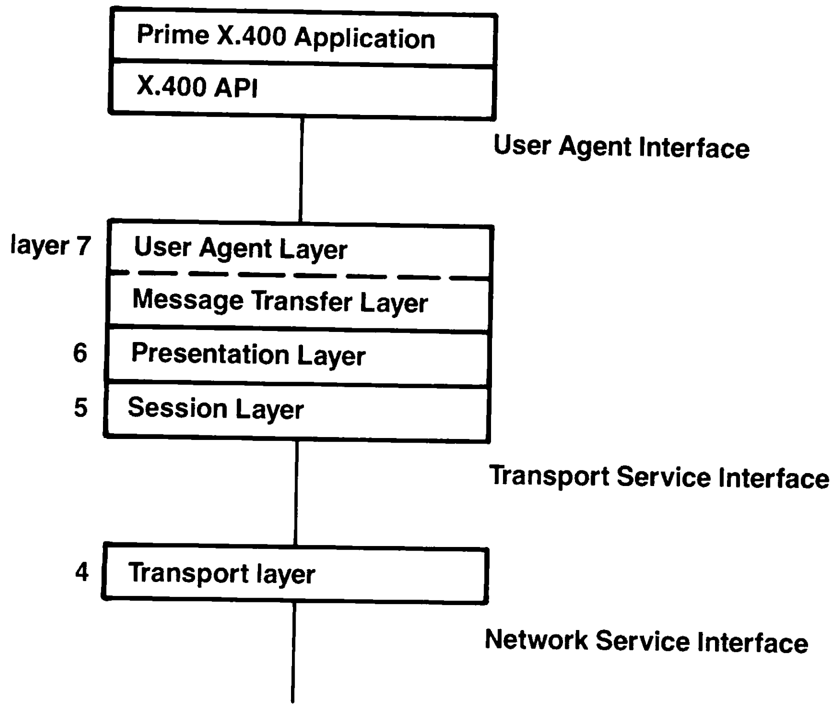

FIGURE 1-2. The Architectural Layers of X.400

The User Agent Interface (UAI) is a message based interface which allows a Prime X.400 Application to send and receive messages via the User Agent layer. X.400 implements the protocol aspects of the User Agent (UA), while the Prime X.400 Application implements local aspects, for example, wordprocessing facilities.

The X.400 protocol provides for the following Interpersonal Message Types:

- Messages are sent with a **Data Request.**
- Incoming messages are received with a **Data Indication.**
- If an incoming message requested a reply, then one should be returned by the Prime X.400 Application with a **Reply Request.**
- Replies from Recipient Prime X.400 Applications to previously transmitted Messages are notified by receipt of a **Reply Indication.**
- Delivery to the Recipient Message Transfer Agent is confirmed by receipt of a **Delivery Indication.**

Prime X.400

Prime X.400 is the name of Prime's X.400 product. Lower levels of the OSI model are provided by an integral Class 0 transport layer, that interfaces to X.25 via PRIMENET™. The flow of data between Prime X.400 Applications is illustrated in Figure 1-3.

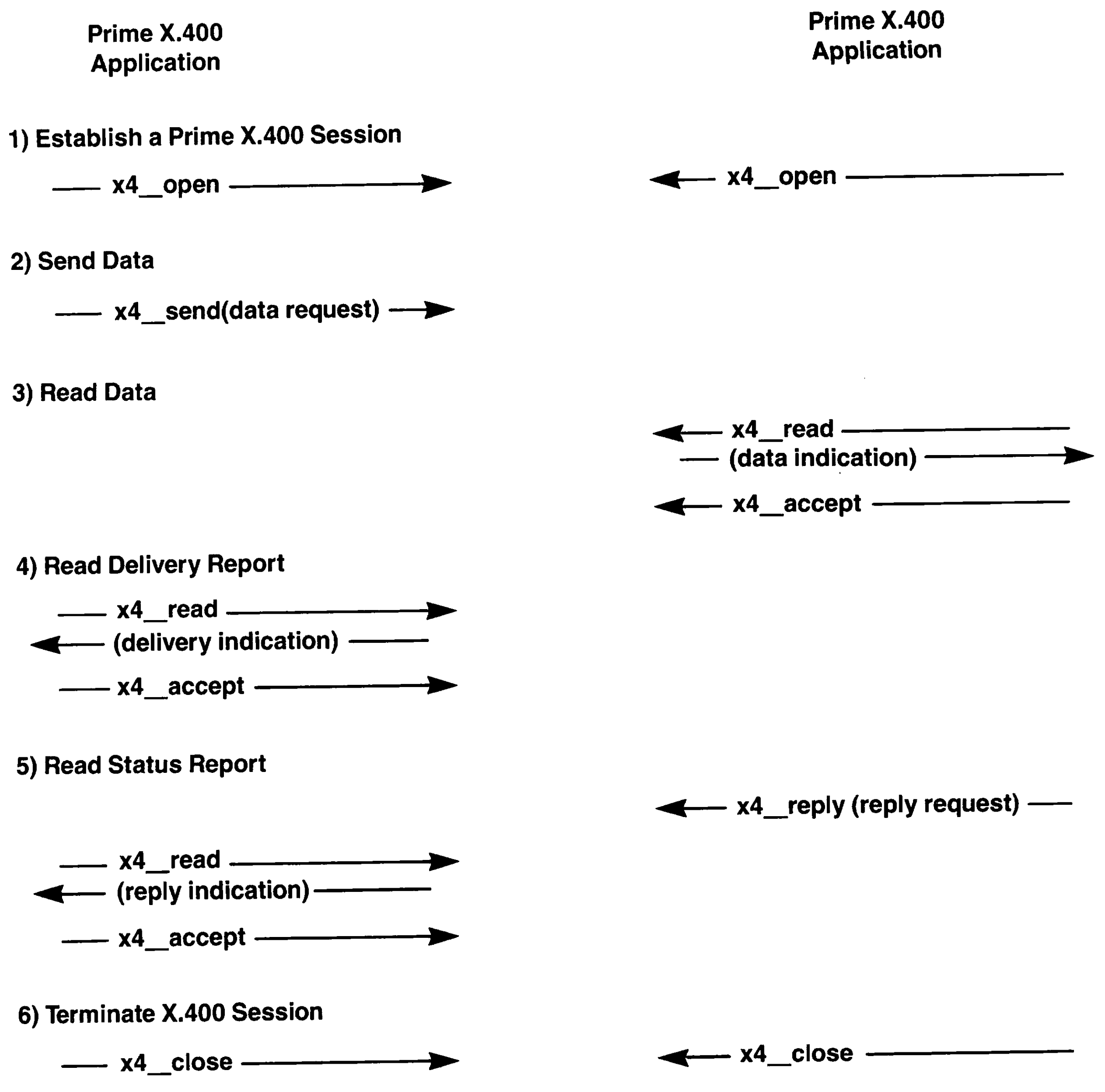

FIGURE 1-3. Data Flow Between Prime X.400 Applications

Message Structure

A message comprises an envelope, header and body.

Message Envelope

The Message Envelope is used to route the body of the message from Originator to Recipients. The X.400 defined P1 protocol governs the flow of data between Message Transfer Agents, and conveys information which is an envelope for the Message.

The data items applicable to the Message Envelope data structure, for each of the possible Interpersonal Message Types, are shown in Figure 1-4.

	Data Request	Data Indication	Reply Request	Reply Indication	Delivery Indication
X4_K_MPDU_ID	-	Y	-	Y	Y
X4_K_CONTENT_TYPE	Y	Y	Y	Y	-
X4_K_CONTENT_ID	Y	Y	Y	Y	Y
X4_K_ENCODED	Y	Y	Y	Y	-
X4_K_ORIGINATOR	-	Y	-	Y	Y
X4_K_PRIORITY	Y	Y	Y	Y	-
X4_K_PER_MESSAGE_FLAG	Y	Y	Y	Y	-
X4_K_DEFERRED_DELIVER	Y	Y	Y	Y	-
X4_K_RECIPIENT	Y	Y	Y	Y	-
X4_K_TRACE	-	Y	-	Y	Y
X4_K_REPORTED_TRACE	-	-	-	-	Y
X4_K_REPORTED_MESSAGE_ID	-	-	-	-	Y

FIGURE 1-4. Message Envelope Data Items for Interpersonal Message Types

Message Header

The Message Header is largely informative, and is not used by Prime X.400 for routing the Message Body. The X.400 defined P2 protocol governs the flow of data between the User Agent and the Message Transfer Agent, and conveys information which is a Header for the Message.

The data items applicable to the Message Header data structure, for each of the possible Interpersonal Message Types, are shown in Figure 1-5.

	Data Request	Data Indication	Reply Request	Reply Indication	Delivery Indication
X4_K_REF	Y	Y	Y	Y	-
X4_K_FROM	Y	Y	Y	Y	-
X4_K_AUTHORISE	Y	Y	-	-	-
X4_K_TO	Y	Y	Y	Y	-
X4_K_CC	Y	Y	-	-	-
X4_K_BCC	Y	Y	-	-	-
X4_K_IN_REPLY_TO	Y	Y	-	-	-
X4_K_OBSOLETES	Y	Y	-	-	-
X4_K_XREF	Y	Y	-	-	-
X4_K_SUBJECT	Y	Y	-	-	-
X4_K_EXPIRES	Y	Y	-	-	-
X4_K_REPLY_BY	Y	Y	-	-	-
X4_K_REPLY_TO	Y	Y	-	-	-
X4_K_IMPORTANCE	Y	Y	-	-	-
X4_K_SENSITIVITY	Y	Y	-	-	-
X4_K_AUTO_FORWARD	-	Y	-	-	-
X4_K_ACTUAL_RECIPIENT	-	-	Y	Y	-
X4_K_INTENDED_RECIPIENT	-	-	Y	Y	-
X4_K_CONVERTED	-	-	Y	Y	-
X4_K_RECEIPT_INFO	-	-	Y	Y	-
X4_K_NON-RECEIPT_INFO	-	-	Y	Y	-
X4_K_BODY	Y	Y	-	-	-
X4-K-DELIVERY-TIME	Y	Y	-	-	-

FIGURE 1-5. Message Header Data Items for Interpersonal Message Types

Message Body

The Message Body is the basic text of the message encoded in X.409 format. The CCITT X.400 recommendations allow the following Body Types:

Body Type	*Description*
IA5Text	ASCII
G3Fax	A sequence of bit strings, each representing a page of Group 3 facsimile information encoded according to Recommendation T.4.
TIF0	A document whose structure is defined in Recommendation T.73 and which conforms to TIF (Text Interchange Format) 0 application rules.
TTX	Teletex.
NationallyDefined	Anything at all.
ForwardedIPMessage	A Message contained within the body of another Message for the purpose of distribution to a further set of recipients. It optionally includes the original Message Header information.

SFD	A simple formatable Document.
TIF1	A document whose structure is defined in Recommendation T.73 and which conforms to TIF1 application rules.

Prime X.400 User Agent

Each Prime X.400 Application can establish multiple Prime X.400 Sessions. Alternatively, Prime X.400's configuration file can be set up so that mail to a number of individual users can be collected by a single Prime X.400 Session. Similarly, mail can be sent by a single Prime X.400 Application on behalf of a number of individual users. Where mail is sent by a single Prime X.400 Application on behalf of a number of individual users, the P1 Originator field in the Message Envelope contains the Prime X.400 Application's O/R Address. The Recipient receives the P2 Originator O/R Address from the Message Header intact (the Prime X.400 Application must ensure that this field is present).

2 PRIME X.400 API LIBRARY

Introduction

This section lists all the Prime X.400 library subroutines. Each subroutine provides details of function, C syntax, description, and returns.

Summary of Routines

x4_accept	Accept responsibility for the incoming message.
x4_alloc	Allocate and initialize a Prime X.400 Data Structure.
x4_clear	Clear a Prime X.400 error condition.
x4_close	Terminate a communication path to a Prime X.400 Server process.
x4_copy	Copy a Prime X.400 Data Structure.
x4_decia5	Convert an X.409-encoded IA5 text file to Prime ECS.
x4_dump	Formatted diagnostic print of a Prime X.400 Data Structure.
x4_enchain	Add a record to the end of a Prime X.400 Linked List.
x4_encia5	Convert a Prime ECS text file to X409-encoded IA5 text.
x4_error	Return the current error status code and qualifier.
x4_find	Locate items within a list data structure.
x4_get	Return the address of a data item from a nominated data structure.
x4_init	Initialize a Prime X.400 data structure.

x4_kill	Release storage for all items in a list data structure.
x4_logoff	Terminate a Prime X.400 session.
x4_logon	Establish a Prime X.400 session.
x4_open_uai	Establish a communication path to a Prime X.400 Server.
x4_put	Add a data item to a Prime X.400 data structure.
x4_read	Initiate a read of an awaiting Message, which may be of type **Data Indication**, **Reply Indication** or **Delivery Indication.**
x4_reject	Do not accept responsibility for the incoming Message.
x4_release	Release storage for a Prime X.400 Data Structure.
x4_reply	Send a Message Reply of type **Reply Request.**
x4_send	Send a Message of type **Data Request.**

X4_ACCEPT

Function

Accept responsibility for the last message read with **x4_read** for a specified Prime X.400 Logon Session.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_accept(logon_ptr);

char *logon_ptr;
```

Description

This is an acknowledgement that the Prime X.400 Application has successfully handled the last incoming Message for the specified Prime X.400 Logon Session (*logon_ptr*), and that the Message can be deleted from Prime X.400's Reliable Transfer Store. Prime X.400 deletes the stored message.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_NOT_OPEN	The user does not have a session open to Prime X.400.
X4_ERR_NO_READ	The user does not have an unanswered x4_read request.
X4_ERR_ISC_ERR	An Inter Server Communication (ISC) Error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_SYN_ERR	An ISC Synchroniser Error has occurred. The error Qualifier contains the Synchroniser Error code.

X4_ALLOC

Function

Allocate and initialize a Prime X.400 Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

char *x4_alloc(struc_id, version);

int struc_id;
int version;
```

Description

The routine returns a pointer to an initialized Prime X.400 Data Structure of the required type and version. The version number must be 1.

Returns

If the routine returns a Null Pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_NO_RESOURCE	There were insufficient resources to allocate the Data Structure. The error Qualifier contains the Data Structure ID.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_CLEAR

Function

Clear a Prime X.400 Error Condition.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_clear();
```

Description

The routine resets a Prime X.400 error condition. This routine must be called after an X.400 error, otherwise all succeeding X.400 calls return the same error.

Returns

The routine returns X4_OK to indicate successful operation.

X4_CLOSE

Function

Prime X.400 Communication Session disconnect request

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_close();
```

Description

This routine terminates a previously opened Communication Session between the Prime X.400 Application and the Prime X.400 Server.

Any logged on Prime X.400 sessions are automatically logged off by this routine.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_NOT_OPEN	The user does not have a Communication Session open to Prime X.400.
X4_ERR_ISC_ERR	An Inter Server Communication (ISC) Error has occurred. The error Qualifier contains the ISC error code.

X4_COPY

Function

Copy a Prime X.400 Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_copy(struc1, struc2);

char *struc1;
char *struc2;
```

Description

The routine copies the contents of *struc2* to *struc1.*

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_BAD_COPY	*Struc2* is not the same type of Data Structure as *Struc1.*
X4_ERR_BAD_REV	*Struc2* and/or *Struc1* contains an invalid Version ID.
X4_ERR_NO_DATA	*Struc2* does not contain data.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_DECIA5

Function

Decodes an X409 encoded IA5 Text body file into Prime ECS.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_decia5(dest,src);

FILE *dest, *src;
```

Description

This routine reads the file accessed by the file pointer *src*. It strips out the X409 encoding, and converts the contents to Prime ECS. The result is written to the file accessed by the file pointer *dest*.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_EXSEQ	X.409 Sequence expected. The error Qualifier contains the type found.
X4_ERR_EXSET	X.409 Set expected. The error Qualifier contains the type found.
X4_ERR_EXTAG_INT	X.409 tagged integer expected. The error Qualifier contains the type found.
X4_ERR_UXSIZE	Unexpected X.409 size. The error Qualifier contains the type found.
X4_ERR_EXOCT_STR	X.409 octet string expected. The error Qualifier contains the type found.
X4_ERR_EXIA5_STR	X.409 IA5 string expected. The error Qualifier contains the type found.

X4_DUMP

Function

Formatted Diagnostic Print of a Prime X.400 Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_dump(fp, struc);

FILE *fp;
char *struc;
```

Description

The routine produces a formatted listing of the specified Prime X.400 Data Structure on the nominated open file unit.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_ENCHAIN

Function

Add a record to the end of a Prime X.400 Linked List.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_enchain(root, list);

char *root;
char *list;
```

Description

The routine adds the Prime X.400 List Data Structure to the end of the linked list indicated by the Prime X.400 Root Data Structure.

Returns

The routine returns the following value:

Value	*Meaning*
X4_OK	The operation was successful.

X4_ENCIA5

Function

Encodes a Prime ECS text file as an X409 encoded IA5 Text body part.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_encia5(dest,src);

FILE *dest, *src;
```

Description

This routine reads the file accessed by the file pointer *src*. It then writes an X409 encoded IA5 Text body, to the file accessed by the file pointer *dest*.

Returns

The routine returns the following value:

Value	*Meaning*
X4_OK	The operation was successful.

X4_ERROR

Function

Return the current error status code and qualifier.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_error(error, qualifier);

int *error;
int *qualifier;
```

Description

Call this routine to return the current error status code and qualifier.

Returns

The routine returns a non_zero (logical true) value if an error has occurred, or zero (logical false) if no error has occurred.

X4_FIND

Function

To locate items within a list data structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

char *x4_find(root, key);

char *root;
int key;
```

Description

The routine returns a pointer to an item within the list data structure indicated by root.

Key may take the following values:

X4_K_NEXT	Returns a pointer to the next item in the list
X4_K_FIRST	Returns a pointer to the first item in the list
X4_K_LAST	Returns a pointer to the last item in the list
X4_K_PREVIOUS	Returns a pointer to the previous item in the list

Returns

If the routine returns a Null Pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_END_OF_LIST	There are no more items in this list.
X4_ERR_BAD_KEY	The user has specified an invalid key.
X4_ERR_BAD_STRUC	The user has specified an invalid data structure ID.
X4_ERR_BAD_REV	An invalid data structure version was provided.
X4_ERR_LIST_EMPTY	There is no data item present for this list.

X4_GET

Function

Return a data item from a Message Envelope or Message Header Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_keys.h>

#include <x4_error.h>

char *x4_get(struct, key);

char *struct;
int key;
```

Description

The routine returns a pointer to an individual field within the Message Header, or Message Envelope Data Structure, depending upon the type indicated by *struct*.

The following data items are found in the Message Envelope data structure, depending on the value of *key* :

Item	*Description*
X4_K_MPDU_ID	*Message Protocol Data Unit Identifier*, assigned by the originator User Agent. It consists of a data structure of type *X4_MPDUID.*
X4_K_CONTENT_TYPE	A *Content Type* parameter is supplied by the originating UA and identifies the convention which governs the structure of the contents. It consists of a data type of structure *X4_CONTENTTYPE.* The only defined value is *X4_CT_P2* which identifies the P2 protocol for Interpersonal Messaging, as specified in CCITT Recommendation X.420.
X4_K_CONTENT_ID	The *UA Content Identifier* is provided by the UA and carried back to the originator (in a delivery indication) by the Message Transfer Layer. It consists of a data structure of type *X4_UACONTENTID.* This parameter is limited to 16 characters in length.
X4_K_ENCODED	The encoding format used in the body of the message. It consists of a data structure of type *X4_ENCODED.*

X4_K_ORIGINATOR	The P1 originator name. It consists of a data structure of type *X4_ORNAME.*
X4_K_PRIORITY	The P1 priority field. It consists of a data structure of type *X4_PRIORITY*, and may be *X4_P_NORMAL*, *X4_P_NONURGENT* or *X4_P_URGENT.*
X4_K_PER_MESSAGE_FLAG	P1 options field which apply to all recipients of the message. It consists of a data structure of type *X4_PERMESSAGEFLAG*, and may be *X4_PMF_DISCLOSERECIPIENTS* (indicates whether the O/R names of all recipients should be indicated to each recipient UA when the message is delivered), *X4_PMF_CONVERSIONPROHIBITED* (if conversion is to be inhibited), *X4_PMF_ALTERNATERECIPIENTALLOWED* (indicates whether the Alternate Recipient Allowed service is requested), and *X4_PMF_CONTENTRETURNREQUEST* (indicates whether the content of the message is to be returned with any Non-delivery Notification).
X4_K_DEFERRED_DELIVERY	This P1 field specifies the earliest time this message may be delivered to the recipient. It consists of a data structure of type *X4_TIME.*
X4_K_RECIPIENT	This P1 field specifies the names of recipients for the message. This information is used for routing the message. It may occur one or more times and consists of a data structure of type *X4_P1RECIPIENT.* If the envelope is a delivery report, then this field describes the reported recipients of the original message and consists of a list of data structures of type *X4_REPORTEDP1RECIPIENT.*
X4_K_TRACE	Information about the Messages passage through the Message Transfer Layer. It consists of a data structure of type *X4_TRACE.*
X4_K_REPORTED_TRACE	Trace information regarding the Message which is the subject of a *Delivery Indication.* It consists of a data structure of type *X4_TRACE.*
X4_K_REPORTED_~~MESSAGE~~MPDU_ID	The Message Protocol Data Unit Identifier of the Message which is the subject of a *Delivery Indication.* It consists of a data structure of type *X4_MPDUID.*

The following data items are found in the Message Header Data Structure depending on the value of *key* :

X4_K_REF	This P2 field contains the IP Message Identifier supplied by the originating Prime X.400 Application. It consists of a data structure of type *X4_REF*.
X4_K_FROM	This P2 field identifies the User who submitted the Prime X.400 message. It consists of a data structure of type *X4_ORDESCRIPTOR* and is for information only. Prime X.400 does not validate this field.
X4_K_AUTHORISE	This optional P2 field describes the user who authorized the sending of the Message. There may be more than one authorizing user specified. It consists of a data structure of type *X4_ORDESCRIPTOR* and is not validated by Prime X.400.
X4_K_TO	This P2 descriptor identifies a primary recipient of the Message. It may occur one or more times. It consists of a data structure of type *X4_RECIPIENT*, which comprises an *X4_ORDESCRIPTOR*, an *X4_REPORT_REQUEST* and an *X4_REPLY_REQUEST*. The *X4_REPORT_REQUEST* allows the user to select receipt notification or non-receipt notification. The *X4_REPLY_REQUEST* allows the user to request the recipient to acknowledge receipt by sending a reply.
X4_K_CC	This P2 descriptor identifies a copy recipient of the Prime X.400 message. It may occur zero or more times. It consists of the same fields as the primary recipient.
X4_K_BCC	This P2 descriptor identifies a blind copy recipient. That is, a recipient whose name is not disclosed to primary or copy recipients. It may occur zero or more times, and consists of the same fields as the primary recipient.
X4_K_IN_REPLY_TO	This P2 field Identifies a previous Message to which this is a reply. It is optional and consists of a data structure of type *X4_REF*.
X4_K_OBSOLETES	This P2 descriptor identifies any previous Messages that are made obsolete by this Message. It may occur zero or more times, and consists of a data structure of type *X4_REF*.
X4_K_XREF	This P2 descriptor identifies any previous Prime X.400 messages that are cross referenced by this Prime X.400 message. It may occur zero or more times, and consists of a data structure of type *X4_REF*.
X4_K_SUBJECT	This P2 descriptor describes the subject of the Prime X.400 message being sent. It may occur zero or more times, and consists of a data structure of type *X4_SUBJECT*.

X4_K_EXPIRES	This P2 field indicates the date and time by which the originator considers the Message to be no longer valid and useful. It is optional and consists of a data structure of type *X4_TIME.*
X4_K_REPLY_BY	This P2 descriptor give the date and time by which a reply to this Message should be sent. It is optional and consists of a data structure of type *X4_TIME.*
X4_K_REPLY_TO	This P2 descriptor gives the names of users to whom the reply should be sent. It may occur zero or more times. It consists of an *X4_ORDESCRIPTOR* which must contain an *X4_ORNAME.*
X4_K_IMPORTANCE	This P2 descriptor gives an indication of the importance of the Message being sent. It consists of a data structure of type *X4_IMPORTANCE.* Allowable values are *X4_IMP_LOW*, *X4_IMP_NORMAL* or *X4_IMP_HIGH.* If not present a default value of *X4_IMP_NORMAL* is supplied.
X4_K_SENSITIVITY	This P2 field gives an indication of the sensitivity of the Message being sent. It consists of a data structure of type *X4_SENSITIVITY.* Allowable values are *X4_SEN_PERSONAL*, *X4_SEN_PRIVATE* or *X4_SEN_COMPANYCONFIDENTIAL.* If not present a value of *X4_SEN_PERSONAL* will be supplied.
X4_K_AUTO_FORWARD	An indication that the Message has been redirected by the original Recipient Message Transfer Agent. It consists of a data structure of type *X4_AUTOFORWARD.*
X4_K_ACTUAL_RECIPIENT	This field is returned in a *Reply Indication* and indicates the actual recipient who received the Message. It consists of a data structure of type *X4_ORDESCRIPTOR.*
X4_K_INTENDED_RECIPIENT	This field is returned in a *Reply Indication* and indicates the intended recipient for the Message (where this is different to the actual recipient). It consists of a data structure of type *X4_ORDESCRIPTOR.*
X4_K_ENCODED	This field indicates the converted Encoded Information Types of the Message. It consists of a data structure of type *X4_ENCODED.*
X4_K_RECEIPT_INFO	This field provides information regarding receipt of the Message by the recipient User Agent. It consists of a data structure of type *X4_RECEIPTINFO.*

X4_K_NON_RECEIPT_INFO
This field provides information regarding non-receipt of the Message by the recipient User Agent. It consists of a data structure of type *X4_NONRECEIPTINFO.*

X4_K_BODY
This field describes the type of each body part within the body file. If the body part is of type *ForwardedIPMessage,* then it contains a reference to a separate Message Header Data Structure for the forwarded message. Such enclosures may be repeated.

X4-K-DELIVERY-TIME
~~This field provides~~ See Programmer Guide (1.1) p 3-22)

The following keys are provided to allow access to the root structures in the Header and Envelope structure:

X4_K_ROOT_AUTHORISE
This key accesses the root to the list of P2 Authorise fields.

X4_K_ROOT_TO
This key accesses the root to the list of P2 primary recipient list.

X4_K_ROOT_CC
This key accesses the root to the list of P2 CC recipients.

X4_K_ROOT_BCC
This key accesses the root to the list of P2 BCC recipients.

X4_K_ROOT_IN_REPLY_TO
This key accesses ~~the root to~~ the list of *X4_REF* structures, ~~that identify~~ to which ~~previous message~~ this ~~message~~ applies.

X4_K_ROOT_OBSOLETES
This key accesses the root to the list of *X4_REF* structures, that identify which previous messages have been made obsolete by this message.

X4_K_ROOT_XREF
This key accesses the root to the list of *X4_REF* structures that identify which previous messages are cross referenced by this one.

X4_K_ROOT_REPLY_TO
This key accesses the root to the list of *X4_ORDESCRIPTOR* structures that identify which users a reply should be sent to.

X4_K_ROOT_BODY
This key accesses the root to the list of *X4_BODY* structures that describe the type of each body part within the message.

X4_K_ROOT_RECIPIENT
This key accesses the root to the list of *X4_TRACE* structures that make up the intermediate trace list.

X4_K_ROOT_TRACE	This key accesses the root to the list of *X4_TRACE* structures.
X4_K_ROOT_REPORTED_TRACE	This key accesses the root to the list of *X4_TRACE* structures that make up the intermediate trace list.

Returns

If the routine returns a Null Pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_NO_DATA	There is no data item present of the type requested.
X4_ERR_LIST_EMPTY	There is no data item present for the list type requested.
X4_ERR_END_OF_LIST	There are no further data items of the list type requested.
X4_ERR_BAD_KEY	The user has specified an invalid key.
X4_ERR_BAD_REV	An invalid Data Structure Version ID was provided.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_INIT

Function

Initialize a Prime X.400 Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_init(struc, key, version);

char *struc;
int key;
int version;
```

Description

This routine initializes a Prime X.400 Data Structure of type *key*. The content of individual fields may then be added using the x4_put library call prior to submitting a SEND DATA REQUEST, to send a message, or a SEND REPLY REQUEST to acknowledge receipt of a message to the Originator Prime X.400 Application.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The Operation was successful.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_KILL

Function

Release storage for all items in a list data structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_kill(root);

char *root;
```

Description

This routine releases storage for all items in the list data structure indicated by root. The root data structure, itself, is updated to indicate an empty list.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_BAD_STRUC	The data structure provided was not a 'root' structure.

X4_LOGOFF

Function

Prime X.400 session disconnect request.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_logoff(logon_ptr);
char *logon_ptr;
```

Description

This routine terminates a previously opened session between the Prime X.400 Application and Prime X.400.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_SYN_ERR	An ISC Synchroniser Error has occurred. The error Qualifier contains the Synchroniser Error code.
X4_ERR_ISC_ERR	An ISC Error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_NOT_OPEN	An ISC Session was not open.

X4_LOGON

Function

Prime X.400 session connect request

C Syntax

```
#include <x4_struc.h>

#include <x4_keys.h>

#include <x4_error.h>

char *x4_logon(user_name, directory, mode);

char *user_name;
char *directory;
int mode;
```

Description

This routine establishes a session between the Prime X.400 Application, and a Prime X.400 User Agent. Prime X.400 searches the configuration file for a match against the logical user name. If the match is successful, the configuration file contains the Prime X.400 *ORName* for this logical user name. This *ORName* is used as the P1 Originator field for all transmitted messages from this user.

The routine returns a pointer to a Prime X.400 logon record. This pointer is used as an argument in subsequent API calls, to identify this Prime X.400 session.

The directory name provided is the destination location for incoming mail body parts, and/or the default source location for outgoing mail body parts. If NULL, it defaults to a sub-ufd called X400_MAIL in the users' origin directory.

Mode may be X4_K_RECEIVE, X4_K_SEND, or both.

Returns

If the routine returns a NULL value, then x4_error returns one of the following error codes:

Code	*Meaning*
X4_ERR_UNKNOWN_USER	The user name was not present in the Configuration file being used by Prime X.400.
X4_ERR_NO_RESOURCE	Prime X.400 had no resource available to support this user.

X4_ERR_LOGGED_ON	The user is already Logged on.
X4_ERR_RECONFIGURING	The Prime X.400 Server is reconfiguring.
X4_ERR_TERMINATED	The Prime X.400 has closed down.
X4_ERR_BAD_RESPONSE	An Unrecognised Message Type from the Prime X.400 Server. The error qualifier contains the message type in question.
X4_ERR_SYN_ERR	An ISC synchroniser error has occurred. The error qualifier contains the synchroniser error code.
X4_ERR_ISC_ERR	An ISC Error has occurred. The error qualifier contains the ISC error code.
X4_ERR_NOT_OPEN	An ISC session is not open.
X4_ERR_NAC	The user does not have access rights to this Prime X.400 user name.
X4_ERR_MAX_LOGON	The maximum number of Prime X.400 logons has been exceeded. The error qualifier contains the maximum number of sessions allowed.

X4_OPEN_UAI

Function

Prime X.400 Communication Session connect request

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

char *x4_open_uai(server_node, max_logon);

char *server_node;
int max_logon;
```

Description

This routine establishes a Communication Session between the Prime X.400 Application and the Prime X.400 Server on the specified processor node.

max_logon specifies the maximum number of simultaneous logon sessions allowed.

Returns

If the routine returns a NULL pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_OPEN	The user already has a session open to Prime X.400.
X4_ERR_ISC_ERR	An ISC error has occurred. The error qualifier contains the ISC error code.
X4_ERR_SYN_ERR	An ISC synchroniser error has occurred. The error qualifier contains the synchroniser error code.

X4_PUT

Function

Add a data element to a Message Envelope or Message Header Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_keys.h>

#include <x4_error.h>

int x4_put(struct, key, arg);

char *struct;
int key;
char *arg;
```

Description

This routine adds the data item referenced by *arg*, to the data structure of type Message Envelope, or Message Header, referenced by *struct*.

The following data items are placed in the Message Envelope data structure according to the value of *key* :

X4_K_CONTENT_TYPE	A *Content Type* parameter is supplied by the originating UA, and identifies the convention which governs the structure of the contents. It consists of a data structure of type *X4_CONTENTTYPE*. The only defined value is *X4_CT_P2* which identifies the P2 protocol for Interpersonal Messaging, as specified in CCITT Recommendation X.420.
X4_K_CONTENT_ID	The *UA Content Identifier* is provided by the UA and carried back to the originator (in a delivery indication) by the Message Transfer Layer. It consists of a data structure of type *X4_UACONTENTID*. This parameter is limited to 16 characters in length.
X4_K_ENCODED	The encoding formats used in the body of the message. It consists of a data structure of type *X4_ENCODED*.
X4_K_ORIGINATOR	The P1 originator name. It consists of a data structure of type *X4_ORNAME*.
X4_K_PRIORITY	The P1 priority field. It consists of a data structure of type

X4_PRIORITY, and may be *X4_P_NORMAL*, *X4_P_NONURGENT* or *X4_P_URGENT*.

X4_K_PER_MESSAGE_FLAG

P1 options field that applies to all Recipients of the message. It consists of a data structure of type *X4_PERMESSAGEFLAG*, and may be *X4_PMF_DISCLOSERECIPIENTS* (indicates whether the O/R names of all recipients should be indicated to each recipient UA when the message is delivered), *X4_PMF_CONVERSIONPROHIBITED* (if conversion is to be inhibited), *X4_PMF_ALTERNATERECIPIENTALLOWED* (indicates whether the Alternate Recipient Allowed service is requested), and *X4_PMF_CONTENTRETURNREQUEST* (indicates whether the content of the message is to be returned with any Non-Delivery Notification).

X4_K_DEFERRED_DELIVERY

This P1 field specifies the earliest time this message may be delivered to the recipient. It consists of a data structure of type *X4_TIME*.

X4_K_RECIPIENT

This P1 field specifies the names of recipients for the message. This information is used for routing the message. It may occur one or more times and consists of a data structure of type *X4_P1RECIPIENT*.

The following data items are placed in the Message Header data structure according to the value of *key* :

X4_K_REF

This field is provided by the originating Prime X.400 Application to identify this Message. It consists of a data structure of type *X4_REF*.

X4_K_FROM

This field identifies the User who submitted the Prime X.400 message. It consists of a data structure of type *X4_ORDESCRIPTOR* and is for information only. Prime X.400 does not validate this field.

X4_K_AUTHORISE

This optional P2 field describes the user who authorized the sending of the Message. There may be more than one authorizing user specified. It consists of a data structure of type *X4_ORDESCRIPTOR* and is not validated by Prime X.400.

X4_K_TO

This P2 descriptor identifies a primary recipient of the Message. It may occur one or more times. It consists of a data structure of type *X4_RECIPIENT* and comprises an *X4_ORDESCRIPTOR*, a *X4_REPORT_REQUEST* and a *X4_REPLY_REQUEST*. The *X4_REPORT_REQUEST*

	allows the user to select receipt notification or non-receipt notification. The *X4_REPLY_REQUEST* allows the user to request the recipient to acknowledge receipt by sending a reply.
X4_K_CC	This P2 descriptor identifies a copy recipient of the Prime X.400 message. It may occur zero or more times. It consists of the same fields as the Primary Recipient.
X4_K_BCC	This P2 descriptor identifies a blind copy recipient, that is, a recipient whose name is not disclosed to primary or copy recipients. It may occur zero or more times, and consists of the same fields as the Primary Recipient.
X4_K_IN_REPLY_TO	Identifies a previous Message to which this is a reply. It is optional, and consists of a data structure of type *X4_REF*.
X4_K_OBSOLETES	This P2 descriptor identifies any previous Messages that are made obsolete by this Message. It may occur zero or more times, and consists of a data structure of type *X4_REF*.
X4_K_XREF	This P2 descriptor identifies any previous Prime X.400 messages that are cross referenced by this Prime X.400 message. It may occur zero or more times, and consists of a data structure of type *X4_REF*.
X4_K_SUBJECT	This P2 descriptor describes the subject of the Prime X.400 message being sent. It is optional and consists of a data structure of type *X4_SUBJECT*.
X4_K_EXPIRES	Indicates the date and time by which the originator considers the Message to be no longer valid and useful. It is optional, and consists of a data structure of type *X4_TIME*.
X4_K_REPLY_BY	This P2 descriptor give the date and time by which a reply to this Message should be sent. It is optional and consists of a data structure of type *X4_TIME*.
X4_K_REPLY_TO	This P2 descriptor gives the names of users to whom the reply should be sent. It may occur zero or more times. It consists of a data structure of type *X4_ORDESCRIPTOR* which must contain an *X4_ORNAME*.
X4_K_IMPORTANCE	This P2 descriptor gives an indication of the importance of the Message being sent. It consists of a data structure of type *X4_IMPORTANCE*. Allowable values are *X4_IMP_LOW*, *X4_IMP_NORMAL* or *X4_IMP_HIGH*. If not present a default value of *X4_IMP_NORMAL* is supplied.
X4_K_SENSITIVITY	This P2 field gives an indication of the sensitivity of the

Message being sent. It consists of a data structure of type *X4_SENSITIVITY*. Allowable values are *X4_SEN_PERSONAL*, *X4_SEN_PRIVATE* or *X4_SEN_COMPANYCONFIDENTIAL*. If not present a value of *X4_SEN_PERSONAL* is supplied.

X4_K_ACTUAL_RECIPIENT

This field is provided in a *Reply Request* and indicates the actual Recipient who received the Message. It consists of a data structure of type *X4_ORDESCRIPTOR*.

X4_K_INTENDED_RECIPIENT

This field is provided in a *Reply Request* and indicates the intended Recipient for the Message, (where this is different to the actual Recipient). It consists of a data structure of type *X4_ORDESCRIPTOR*.

X4_K_CONVERTED This field indicates the final encoded information types of the message. It consists of a data structure of type *X4_ENCODED*.

X4_K_RECEIPT_INFO This field provides information regarding receipt of the message by the recipient user agent. It consists of a data structure of type *X4_RECEIPTINFO*.

X4_K_NON-RECEIPT_INFO

This field provides information regarding non-receipt of the message by the recipient user agent. It consists of a data structure of type *X4_NONRECEIPTINFO*.

X4_K_BODY This field describes the type of each body part within the body file. If the body part is of type *ForwardedIPMessage*, then it contains a reference to a separate Message Header Data Structure for the forwarded message. Such enclosures may be repeated.

X4_K_DELIVERY_TIME
See X400 Programmers Guide (1.1) p 3-38

The following keys are provided to allow access to the root structures in the Header and Envelope structure.

X4_K_ROOT_AUTHORISE

This key accesses the root to the list of P2 authorize fields.

X4_K_ROOT_TO This key accesses the root to the list of P2 primary recipient list.

X4_K_ROOT_CC This key accesses the root to the list of P2 CC recipients.

X4_K_ROOT_BCC This key accesses the root to the list of P2 BCC recipients.

~~X4_K_ROOT~~_IN_REPLY_TO

This ~~key~~ accesses ~~the root to the list of X4~~*_REF* structures, that identify to which previous message this message applies.

X4_K_ROOT_OBSOLETES	This key accesses the root to the list of *X4_REF* structures, that identify which previous messages have been made obsolete by this message.
X4_K_ROOT_XREF	This key accesses the root to the list of *X4_REF* structures that identify which previous messages are cross referenced by this one.
X4_K_ROOT_REPLY_TO	This key accesses the root to the list of *X4_ORDESCRIPTOR* structures that identify which users a reply should be sent to.
X4_K_ROOT_BODY	This key accesses the root to the list of *X4_BODY* structures that describe the type of each body part within the message.
X4_K_ROOT_RECIPIENT	This key accesses the root to the list of *X4_TRACE* structures that make up the intermediate trace list.
X4_K_ROOT_TRACE	This key accesses the root to the list of *X4_TRACE* structures.
X4_K_ROOT_REPORTED_TRACE	This key accesses the root to the list of *X4_TRACE* structures that make up the intermediate trace list.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_BAD_KEY	The user has specified an invalid key.
X4_ERR_BAD_REV	An Invalid Data Structure Version ID was provided.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_READ

Function

Initiate a read of the waiting message.

C Syntax

#include <x4_struc.h>

#include <x4_error.h>

char *x4_read(wait);

int wait;

Description

This routine polls Prime X.400 for incoming messages, and returns a pointer to an X4_Msg data structure. *Wait*, is the maximum poll period, and is specified in milliseconds. It should be zero for an immediate return, or negative to wait for an indefinite period. Individual data fields may be retrieved by subsequent calls to x4_get. When users have finished processing this mail, they must call x4_accept (in which case Prime X.400 will delete the stored message), or x4_logoff (in which case Prime X.400 attempts to deliver it the next time the user establishes a Prime X.400 session).

Returns

If the routine returns a Null Pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_NOT_OPEN	The user does not have a session open to Prime X.400.
X4_ERR_NO_MESSAGE	There is no message waiting.
X4_ERR_SYN_ERR	An ISC Synchroniser Error has occurred. The error Qualifier contains the Synchroniser Error code.
X4_ERR_ISC_ERR	An ISC Error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_BAD_MESSAGE	An invalid message format has been received.

X4_REJECT

Function

Do not accept responsibility for a received message.

C Syntax

#include <x4_struc.h>

#include <x4_error.h>

int x4_reject(logon_ptr);

char *logon_ptr;

Description

This routine informs Prime X.400 that the Prime X.400 Application is unable to handle the incoming Message. Prime X.400 will delete the stored message.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_NOT_OPEN	The user does not have a session open to Prime X.400.
X4_ERR_NO_READ	The user does not have an unanswered x4_read request.
X4_ERR_ISC_ERR	An Inter Server Communication (ISC) Error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_SYN_ERR	An ISC Synchroniser Error has occurred. The error Qualifier contains the Synchroniser Error code.

X4_RELEASE

Function

Release storage for a Prime X.400 Data Structure.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

int x4_release(struct);

char *struct;
```

Description

This routine releases storage for an initialized Prime X.400 Data Structure.

Returns

The routine returns one of the following values:

Value	*Meaning*
X4_OK	The operation was successful.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.

X4_REPLY

Function

Send a Message Reply Request.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

char *x4_reply(logon_ptr, envelope, header);

char *logon_ptr;
char *envelope;
char *header;
```

Description

An acknowledgement is sent in response to a previously read message, using the information previously stored to the Message Envelope data structure.

This routine returns a pointer to a static X4_MPDUSTRING Data Structure that contains the MPDU identifier assigned by PRIME_X400.

Returns

If the routine returns a NULL pointer, then X4_error returns one of the following:

Value	*Meaning*
X4_ERR_NOT_OPEN	The user does not have a session open to Prime X.400.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.
X4_ERR_ISC_ERR	An ISC Error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_SYN_ERR	An ISC Synchroniser Error has occurred. The error Qualifier contains the Synchroniser Error code.
X4_ERR_NO_RESOURCE	Prime X.400 is unable to accept this request. The error Qualifier contains the reason for rejection: 1 = Server reconfiguring, 2 = Invalid header or envelope, 3 = Queue manager error.
X4_ERR_MDNP	A mandatory descriptor is missing from the envelope or header structure provided. The error Qualifier contains the structure ID of the missing descriptor.

X4_SEND

Function

Submit a send message request.

C Syntax

```
#include <x4_struc.h>

#include <x4_error.h>

char *x4_send(logon_ptr, envelope, header);

char *logon_ptr;
char *envelope;
char *header;
```

Description

A message is submitted to Prime X.400 using the information previously stored in the nominated data structures.

The routine returns a pointer to a static X4_MPDUSTRING Data structure containing the MPDU identifier assigned by Prime X.400.

Returns

If the routine returns a NULL Pointer, then x4_error returns one of the following values:

Value	*Meaning*
X4_ERR_NOT_OPEN	The user does not have a session open to Prime X.400.
X4_ERR_BAD_STRUC	An invalid Data Structure ID was provided.
X4_ERR_ISC_ERR	An ISC error has occurred. The error Qualifier contains the ISC error code.
X4_ERR_SYN_ERR	An ISC synchroniser error has occurred. The error qualifier contains the synchroniser error code.
X4_ERR_NO_RESOURCE	Prime X.400 is unable to accept this request. The error Qualifier contains the reason for rejection: 1 = Server reconfiguring, 2 = Invalid header or envelope, 3 = Queue manager error.

X4_ERR_MDNP	A mandatory descriptor is missing from the envelope or header structure provided. The error Qualifier contains the structure ID of the missing descriptor.

INDEX

INDEX

SURVEYS

READER RESPONSE FORM

Your feedback will help us continue to improve the quality, accuracy, and organization of our publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

   ____
   ____
   ____
   ____
   ____
   ____

3. What faults or errors in this manual gave you problems?

   ____
   ____
   ____
   ____
   ____
   ____

4. How does this manual compare to equivalent manuals produced by other computer companies?

   ☐ *Much better* ☐ *Slightly better* ☐ *About the same*
   ☐ *Much worse* ☐ *Slightly worse* ☐ *Can't judge*

5. Which other companies' manuals have you read?

   ____
   ____

Name:______________________ Position:______________________
Company:____
Address:____
____
______________________________ Postal Code:______________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760