HITACHI
Inspire the Next

AIX

通信管理

XNF/AS　プログラマーズガイド HDLC 編

解説・文法書

3000-3-B47-10

マニュアルの購入方法

このマニュアル，および関連するマニュアルをご購入の際は，巻末の「ソフトウェアマニュアルのサービス ご案内」をご参照ください。

■対象製品

適用 OS：AIX 5L
P-F1M14-5119 XNF/AS/HDLC 01-01
適用 OS：AIX
P-F1M14-51219 XNF/AS/HDLC 02-01

■輸出時の注意

本製品を輸出される場合には，外国為替および外国貿易法ならびに米国の輸出管理関連法規などの規制をご確認の上，必要な手続きをお取りください。
なお，ご不明な場合は，弊社担当営業にお問い合わせください。

■商標類

AIX は，米国における International Business Machines Corporation の登録商標です。
AIX 5L は，米国およびその他の国における International Business Machines Corporation の商標です。
UNIX は，X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標です。

■発行

2002 年 3 月（第 1 版）3000-3-B47
2009 年 12 月（第 2 版）3000-3-B47-10

■著作権

変更内容

変更内容（3000-3-B47-10） XNF/AS/HDLC 01-01（**適用** OS：AIX 5L），XNF/AS/HDLC 02-01（**適用** OS：AIX）

追加・変更内容	変更個所
64 ビットカーネルで動作できるようにしました。	―

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，XNF/AS/HDLC の API について説明したものです。
このマニュアルの対象となるプログラムプロダクトを次に示します。

適用 OS：AIX 5L
　P-F1M14-5119　XNF/AS/HDLC

適用 OS：AIX
　P-F1M14-51219　XNF/AS/HDLC

■対象読者

XNF/AS/HDLC を利用してアプリケーションプログラムを設計・作成する方を対象としています。
また，マニュアル「通信管理　XNF/AS　解説・運用編」および「通信管理　XNF/AS　構成定義編」の内容を理解していることを前提とします。

■マニュアルの構成

このマニュアルは，次に示す編，章と付録から構成されています。

第 1 編　解説

第 1 章　通信管理の概要
XNF/AS の特長について説明しています。また，XNF/AS/HDLC でアプリケーションプログラム間の通信をするための HDLC パススルー機能について説明しています。

第 2 章　HDLC パススルー API の概要
HDLC ユーザ，同期方式と非同期方式などの概念について説明しています。また，HDLC パススルー API で使用するライブラリ関数について説明しています。

第 2 編　文法

第 3 章　ライブラリ関数の文法
アプリケーションプログラムの作成手順，およびライブラリ関数の記述方法について説明しています。また，エラー時にアプリケーションプログラムが実施しなければならない処置について説明しています。

第 4 章　ライブラリ関数の使用手順
ライブラリ関数の使用手順，および通信管理での処理の流れについて説明しています。

付録 A　用語解説
このマニュアルで使用している用語の意味について説明しています。

付録 B　ライブラリ関数とエラー情報
ライブラリ関数が異常終了したときに設定されるエラー情報について説明しています。

付録 C　従来製品との差異

従来製品（HI-UX/WE2 上で動作する XNF/S-E2）との差異について説明しています。

■関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

- 通信管理　XNF/AS　解説・運用編（3000-3-B41）（AIX 5L 用）
- 通信管理　XNF/AS　構成定義編（3000-3-B42）（AIX 5L 用）
- 通信管理　XNF/AS　解説・運用編（3000-3-B61）（AIX 用）
- 通信管理　XNF/AS　構成定義編（3000-3-B62）（AIX 用）
- AIX マニュアル（AIX に付属する CD-ROM マニュアルなど）

■読書手順

このマニュアルは，次の表に従ってお読みいただくことをお勧めします。

目的	記載個所
XNF/AS および XNF/AS/HDLC の特長について知りたい	1 章
XNF/AS/HDLC の API の機能について知りたい	2 章
ライブラリ関数の記述方法，およびエラー情報について知りたい	3 章，付録 B
ライブラリ関数の使用例，および通信管理での処理の流れについて知りたい	4 章
XNF/AS の用語について知りたい	付録 A
従来製品（HI-UX/WE2 上で動作する XNF/S-E2）との差異について知りたい	付録 C

■このマニュアルでの表記

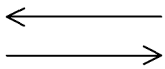
このマニュアルでは，製品名称を次に示す略称で表記しています。

製品名称		略称
AIX 5L V5.1	AIX 5L	AIX
AIX 5L V5.2		
AIX 5L V5.3		
AIX V6.1		
Extended HNA based Communication Networking Facility/for Advanced Server		XNF/AS
Extended HNA Based Communication Networking Facility/for Advanced Server/High Level Data Link Control		XNF/AS/HDLC

■図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。

●制御の流れ



■常用漢字以外の漢字の使用について

このマニュアルでは，常用漢字を使用することを基本としていますが，次に示す用語については，常用漢字以外の漢字を使用しています。
個所（かしょ） 溜まる（たまる）

■ KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

目次

第 1 編　解説

第 2 編　文法

図目次

表目次

1 通信管理の概要

この章では，XNF/AS の特長について説明します。また，XNF/AS/HDLC でアプリケーションプログラム間の通信をするための HDLC パススルー機能について説明します。

1.1 XNF/AS の特長

XNF/AS は，多数の回線をサーバ（EP8000 シリーズ）に接続できるようにする通信管理プログラムです。

XNF/AS には次のような特長があります。

- 大規模なネットワークの構築
- 多様な通信プロトコルへの対応
- 構成情報の作成と管理
- 連続運転への対応
- 保守機能の強化
- ハードウェア障害に対する機能の強化

このマニュアルでは，通信管理 XNF/AS を構成するプログラムプロダクトのうち，XNF/AS/HDLC について説明します。XNF/AS/HDLC は，HDLC 手順でアプリケーションプログラム間の通信をするための API を提供するプログラムプロダクトです。

1.2 HDLC パススルー機能

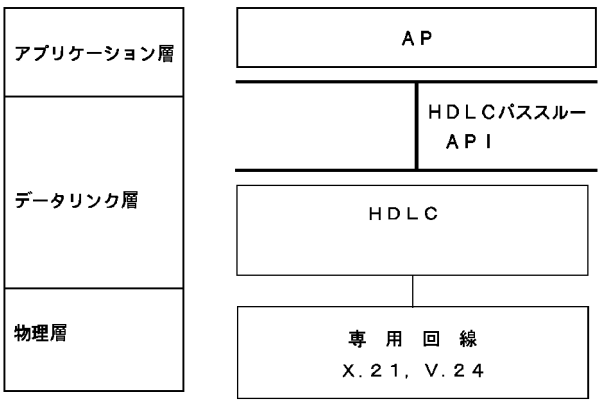
今日のコンピュータネットワークでは，OSI のように国際的に標準化されたプロトコルによる通信が主流になってきています。しかし，一方で，そのほかの独自のプロトコルによる通信形態も多く残されています。

このような独自のプロトコルでの通信のため，XNF/AS/HDLC では，HDLC 手順によるパススルー機能（HDLC **パススルー機能**）を提供しています。HDLC パススルー機能とは，データリンク層に，アプリケーションプログラムに対するインタフェースを設け，データリンク層のサービスを直接アプリケーションプログラムに提供する機能です。これによって，通信相手となるアプリケーションプログラムのプロトコルを容易にサポートできます。

各システムのネットワークソフトウェアは，HDLC パススルー機能の API（HDLC **パススルー** API）を使用することで，HDLC 手順で互いに通信できます。HDLC パススルー API は，C 言語のライブラリ関数として提供されています。

HDLC パススルー機能のサービスの範囲を図 1-1 に示します。HDLC 手順については，マニュアル「通信管理　XNF/AS　解説・運用編」を参照してください。

図 1-1　HDLC **パススルー機能のサービス範囲**



HDLC パススルー機能で使用できる通信回線は，X.21 または V.24 のインタフェースの専用回線です。

2 HDLC パススルー API の概要

HDLC パススルー API について理解するためには，幾つかの概念や用語を知る必要があります。この章では，HDLC ユーザ，同期方式と非同期方式などの概念について説明します。また，HDLC パススルー API で使用するライブラリ関数について説明します。

2.1 アプリケーションプログラム間の通信の流れ

アプリケーションプログラムは，通常，一つまたは複数のプロセスから成ります。さらに，各プロセスは，一つまたは複数のデータの通信路を持つことができます。この，データの通信路一つ一つに対応するものとして，XNF/AS/HDLC では，**データリンクユーザ**という概念を用います。つまり，各アプリケーションプログラムは，一つまたは複数のデータリンクユーザを持つことができるといえます。また，各データリンクユーザを識別するための番号を，**仮想スロット番号**と呼びます。

ここでは，アプリケーションプログラム間の通信の流れを，各データリンクユーザ単位に，四つの段階に分けて説明します。アプリケーションプログラム間の通信は，ライブラリ関数を発行することで実現できます。

(1) データリンク端点の確立・解放

データリンクユーザは，通信管理を識別する UNIX ファイルをオープンして，データリンクユーザと通信管理との間の固有の通信路（**データリンク端点**）を確立します。これをデータリンク端点の確立と呼びます。逆に，通信管理を識別する UNIX ファイルをクローズすることをデータリンク端点の解放と呼びます。データリンクユーザとデータリンク端点とは 1 対 1 に対応します。

データリンク端点を識別するための番号を**データリンク端点識別子**と呼びます。通信管理を識別する UNIX ファイルをオープンしたときにリターン値として返されるファイル記述子がこれに当たります。

(2) データリンク端点と仮想スロット番号の結合

仮想スロット番号をデータリンク端点に関連づけることによって，データリンクユーザと通信管理とを論理的に結び付けます。これを，データリンク端点と仮想スロット番号の結合と呼びます。

(3) コネクションの確立・解放

データリンクユーザは，相手データリンクユーザとの接続を要求します。相手データリンクユーザが接続要求を受け入れることによって，データリンクユーザ同士の接続が成立します。これをコネクションの確立と呼びます。

逆に，この接続を解除することをコネクションの解放と呼びます。

(4) データの転送

確立したコネクションを介して，両方向にデータを転送できます。

2.2 ライブラリ関数

HDLC パススルー API で使用するライブラリ関数の一覧を表 2-1 に示します。

表 2-1 ライブラリ関数一覧

名称	機能
d_bind( )	データリンク端点と仮想スロット番号を結合します。
d_close( )	データリンク端点を解放します。
d_connect( )	コネクションの確立を要求します。
d_getstate( )	通信管理の状態を確認します。
d_look( )	データリンク端点上のイベントを確認します。
d_open( )	データリンク端点を確立します。
d_rcv( )	データを受信します。
d_rcvconnect( )	コネクション確立要求の結果を確認します。
d_rcvdis( )	コネクション切断の原因を取得します。
d_retryck( )	コネクション切断時の対処方法を取得します。
d_snd( )	データを送信します。
d_snddis( )	コネクションを解放します。
d_unbind( )	データリンク端点と仮想スロット番号の結合を解除します。

2.3 通信管理の状態

通信管理の状態を表 2-2 に示します。通信管理の状態は，ライブラリ関数 d_getstate( ) で確認できます。

表 2-2 通信管理の状態

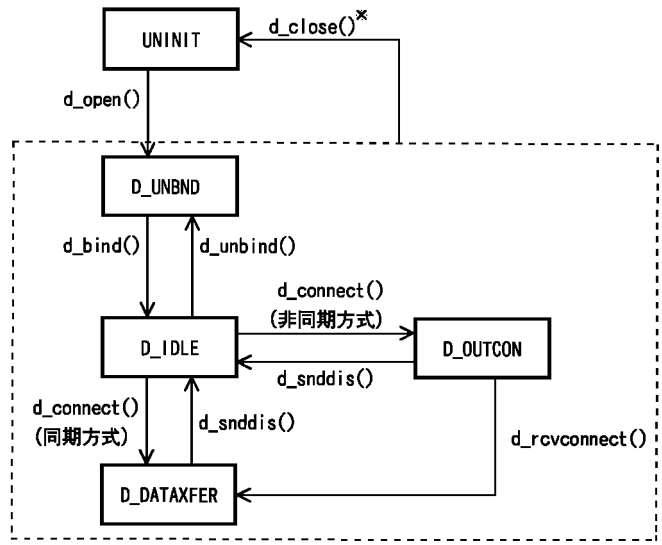
状態	意味
UNINIT ※	データリンク端点が確立していない状態を示します。
D_UNBND	データリンク端点と仮想スロット番号が結合していない状態を示します。
D_IDLE	データリンク端点と仮想スロット番号が結合した状態を示します。
D_OUTCON	コネクションの確立保留中の状態を示します。
D_DATAXFER	コネクションが確立し，データの転送が可能な状態を示します。

注※

ライブラリ関数 d_getstate( ) では確認できません。

ライブラリ関数の発行によって，通信管理の状態がどのように変化するかを図 2-1 に示します。なお，同期方式と非同期方式の違いについては，「2.4 同期方式と非同期方式」を参照してください。

図 2-1 ライブラリ関数の発行による通信管理の状態の変化



注※ UNINIT以外のすべての状態から発行できます。

2.4 同期方式と非同期方式

通信管理がライブラリ関数を処理する方法には，**同期方式**と**非同期方式**があります。非同期方式を使用したい場合は，ライブラリ関数 d_open( ) 発行時に指定します。

特に何も指定しなければ，すべてのライブラリ関数が同期方式で処理されます。

（1） 同期方式

同期方式とは，ライブラリ関数を呼び出したあと，通信管理側の処理がすべて完了した時点でアプリケーションプログラムに制御が戻る方式です。

（2） 非同期方式

非同期方式とは，ライブラリ関数を呼び出したあと，通信管理側の処理が完了してもしなくても，すぐにアプリケーションプログラムに制御が戻る方式です。非同期方式で動作するライブラリ関数は，d_connect( )，d_rcv( )，d_rcvconnect( )，および d_snd( ) の 4 種類です。d_snd( ) を非同期方式で実行すると，システムでビジー状態になった場合にデータの送信を抑止する**フロー制御機能**が動作します。

アプリケーションプログラムは，select **システムコール**を使用して，処理の完了およびビジー状態の解除を監視できます。処理の完了を監視する場合は *readfds を，ビジー状態の解除を監視する場合は *writefds を，それぞれ引数として指定します。select システムコールの使用方法については，AIX マニュアルを参照してください。

3 ライブラリ関数の文法

この章では，アプリケーションプログラムの作成手順，およびライブラリ関数の記述方法について説明します。また，エラー時にアプリケーションプログラムが実施しなければならない処置について説明します。

3.16 詳細エラーコードと切断時の対処

3.1　アプリケーションプログラムの作成手順

アプリケーションプログラムを作成・実行するときに使用する，ヘッダファイルおよびライブラリについて説明します。

（1）ヘッダファイル

HDLC パススルー API を使用するアプリケーションプログラムで取り込む必要のあるヘッダファイルを表 3-1 に示します。

表 3-1　ヘッダファイル

ファイル名	記述形式	内容
diuser.h	#include <xnfs/diuser.h>	各種マクロ，および構造体が定義されています。
errno.h	#include <errno.h>	詳細エラーコードが定義されています。グローバル変数 errno を参照するために使用します。
fcntl.h	#include <fcntl.h>	open システムコールおよび fcntl システムコールのフラグが定義されています。ライブラリ関数 d_open( ) の oflag 引数を指定するために使用します。

（2）ライブラリ

作成したアプリケーションプログラムをコンパイルするとき，表 3-2 に示すライブラリとリンクする必要があります。

表 3-2　ライブラリの種類と指定方法

ライブラリの種類（ファイル名）	コンパイル時のオプションの指定方法※	特徴
共用ライブラリ（/lib/libHDLC.so）	xlc ファイル名 -brtl -lHDLC	アプリケーションプログラム実行時に，ライブラリとリンクされます。

注※

コンパイル時のオプションについては，各種コンパイラのマニュアルを参照してください。

3.2 ライブラリ関数の記述方法

各ライブラリ関数の記述方法を次の形式で説明します。

(1) 名称

ライブラリ関数の名称と，機能の概要について説明しています。

(2) 形式

ライブラリ関数の使用方法をコーディングレベルで記述しています。

(3) 機能

ライブラリ関数の詳細な機能，および引数を説明しています。また，構造体を設定・参照する必要がある場合は，その形式および内容を説明しています。

(4) リターン情報

(a) リターン値

HDLC パススルー API が，ライブラリ関数の実行結果を報告するために，アプリケーションプログラムに対して返す値を説明しています。一般に，0 または正の値は関数が正しく終了したことを示し，-1 は関数が異常終了したことを示します。

(b) エラー情報

ライブラリ関数が異常終了した場合に，グローバル変数 d_errno にセットされるエラーコード（10 進）を説明しています。なお，各エラーコードに対応するエラー名称がヘッダファイル diuser.h に設定されているので，参照してください。

(5) 注意事項

ライブラリ関数を使用するときに注意しなければならない点について説明しています。

3.3 d_bind( )

(1) 名称

d_bind( )

データリンク端点と仮想スロット番号を結合します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_bind(fd,bind);
int fd;
struct d_bind *bind;
```

(3) 機能

データリンク端点に仮想スロット番号を関連づけ，データリンクユーザと通信管理とを論理的に結び付けます。これ以降，このデータリンク端点に対するコネクションの確立要求を開始できます。

指定した仮想スロット番号が使用できない場合，およびすでに使用されている場合は，異常終了します。

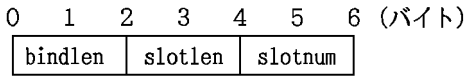
fd：

データリンク端点識別子を指定します。

bind：

構造体 d_bind のアドレスを指定します。構造体 d_bind のテーブル形式および内容は次のとおりです。

テーブル形式



テーブル内容

テーブル要素	内容	指定値
bindlen	構造体 d_bind で設定する情報のバイト数	0x0006
slotlen	仮想スロット番号のバイト数	0x0002
slotnum	仮想スロット番号	任意

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DBADADDR	1	指定された仮想スロット番号が，間違った形式であるか，または存在しません。
DNOADDR	5	指定された仮想スロット番号はすでに使用されています。
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.4 d_close( )

(1) 名称

d_close( )

データリンク端点を解放します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_close(fd);
int fd;
```

(3) 機能

通信管理にデータリンク端点の使用を終了したことを通知し，通信管理を識別するUNIX ファイルをクローズします。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.5 d_connect( )

(1) 名称

d_connect( )

コネクションの確立を要求します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_connect(fd);
int fd;
```

(3) 機能

通信先データリンクユーザに対し，コネクションの確立（接続）を要求します。

同期方式の場合は，相手データリンクユーザからの応答を待ってリターンします。

非同期方式の場合は，相手データリンクユーザからの応答を待たないで異常終了し，エラー情報として DNODATA を設定します。このことから，コネクションがまだ確立していないことを知ることができます。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。
DLOOK	9	データリンク端点上でイベントが発生しました。ライブラリ関数 d_look( ) による確認が必要です。
DNODATA	13	コネクションの確立を要求していますが，相手データリンクユーザからの応答がありません（非同期方式の場合）。

3.6 d_getstate( )

(1) 名称

```
d_getstate( )
```

通信管理の状態を確認します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_getstate(fd);
int fd;
```

(3) 機能

データリンク端点についての通信管理の状態を，リターン値で確認します。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：通信管理の状態を返します。

リターン値	値	状態
D_UNBND	1	データリンク端点と仮想スロット番号が結合していない状態を示します。
D_IDLE	2	データリンク端点と仮想スロット番号が結合した状態を示します。
D_OUTCON	3	コネクションの確立保留中の状態を示します。
D_DATAXFER	5	コネクションが確立し，データの転送ができる状態を示します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.7 d_look( )

(1) 名称

d_look( )

データリンク端点上のイベントを確認します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_look(fd);
int fd;
```

(3) 機能

データリンク端点上のイベントの有無，および種類をリターン値として取得します。ほかのライブラリ関数の実行時に，エラー情報として DLOOK が設定された場合は，この関数を発行して，発生したイベントの種類を確認する必要があります。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：イベントがない場合は 0 を，ある場合はイベントの種類を返します。

リターン値	値	イベントの種類
D_CONNECT	0x0002	コネクション確立要求に対する応答を受信しました。
D_DATA	0x0004	データを受信しました。
D_DISCONNECT	0x0010	コネクションが切断されました。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.8 d_open( )

(1) 名称

d_open( )

データリンク端点を確立します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_open(path,oflag);
char *path;
int oflag;
```

(3) 機能

通信管理を識別する UNIX ファイルをオープンし，ファイル記述子（データリンク端点識別子）をリターン値として取得します。

path：

オープンするファイルのパス名（/dev/xnfs/hdlc）へのポインタです。

oflag：

これ以降のライブラリ関数を，同期方式または非同期方式のどちらで実行するかを指定します。指定方法は次のとおりです。

oflag の指定	ライブラリ関数の処理方式	
	d_snd( )	d_connect( )，d_rcvconnect( )，d_rcv( )
NULL(0)	同期方式	同期方式
O_NDELAY だけ	同期方式	非同期方式
O_DFLOW だけ	非同期方式	非同期方式
O_NDELAY と O_DFLOW	非同期方式	非同期方式

(4) リターン情報

(a) リターン値

正常終了時：ファイル記述子（データリンク端点識別子）を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については,「3.16.1　詳細エラーコード」を参照してください。

3.9 d_rcv( )

(1) 名称

d_rcv( )

データを受信します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_rcv(fd,buf,nbytes,flags);
int fd;
char *buf;
unsigned nbytes;
int *flags;
```

(3) 機能

ユーザデータを受信します。同期方式の場合は，データが到着するのを待ってリターンします。非同期方式の場合は，データが到着していなければ異常終了し，エラー情報として DNODATA を設定します。

fd：

データリンク端点識別子を指定します。

buf：

受信したデータを格納するバッファ（受信バッファ）のアドレスを指定します。

nbytes：

受信バッファのサイズを指定します。

flags：

オプションのフラグ D_MORE が設定されるフラグ領域のアドレスを指定します。リターンした場合に，このフラグが設定されたときは，その時点でデータが終了していないため，再度 d_rcv( ) を発行してデータの続きを受信する必要があることを示します。また，このフラグがクリアされることでデータの終了を示します。

(4) リターン情報

(a) リターン値

正常終了時：受信バイト数を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。
DLOOK	9	データリンク端点上でイベントが発生しました。ライブラリ関数 d_look( ) による確認が必要です。
DNODATA	13	受信できるデータがありません（非同期方式の場合）。

(5) 注意事項

受信バッファにデータが溜まり過ぎると，バッファがビジー状態となるため，一定の間隔で d_rcv( ) を発行することをお勧めします。なお，このような例については，「図 4-7　同期方式でのデータの受信（ビジー状態になった場合）」を参照してください。非同期方式の場合でも運用方法は同じです。

3.10 d_rcvconnect( )

(1) 名称

```
d_rcvconnect( )
```

コネクション確立要求の結果を確認します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_rcvconnect(fd);
int fd;
```

(3) 機能

以前に実行したコネクション確立要求の結果を確認します。

同期方式の場合は，コネクションの確立を確認してリターンします。

非同期方式の場合は，コネクションの確立が確認できなければ，コネクションの確立を待たないで異常終了し，エラー情報として DNODATA を設定します。この場合は，再度 d_rcvconnect( ) を発行する必要があります。d_rcvconnect( ) が正常終了した時点で，コネクションが確立します。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。
DLOOK	9	データリンク端点上でイベントが発生しました。ライブラリ関数 d_look( ) による確認が必要です。
DNODATA	13	コネクションの確立が確認できません（非同期方式の場合）。

3.11 d_rcvdis( )

(1) 名称

d_rcvdis( )

コネクション切断の原因を取得します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_rcvdis(fd,discon);
int fd;
struct d_discon *discon;
```

(3) 機能

コネクションが切断された原因を，切断理由コードとして取得します。

切断理由コードは，構造体 d_discon 中の領域 reason に設定されます。アプリケーションプログラムは，この切断理由コードをメッセージに出力しておくと，運用時の障害調査に便利です。

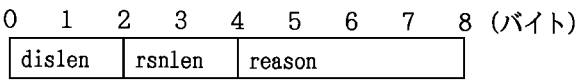
fd：

データリンク端点識別子を指定します。

discon：

構造体 d_discon のアドレスを指定します。構造体 d_discon のテーブル形式および内容は次のとおりです。

テーブル形式



テーブル内容

テーブル要素	内容	指定値
dislen	構造体 d_discon で設定する情報のバイト数	0x0008
rsnlen	切断理由コードのバイト数	0x0004
reason	切断理由コード	－

（凡例）

－：該当しません。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。
DLOOK	9	データリンク端点上でイベントが発生しました。ライブラリ関数 d_look( ) による確認が必要です。
DNODIS	14	指定されたデータリンク端点は，コネクションの切断を通知されていません。

(5) 注意事項

切断理由コードは，ライブラリ関数 d_retryck( ) で使用するためのものです。そのままエラー判定などには使用しないでください。

3.12 d_retryck( )

(1) 名称

```
d_retryck( )
```

コネクション切断時の対処方法を取得します。

(2) 形式

```
#include <xnfs/diuser.h>
unsigned long d_retryck(reason);
int reason;
```

(3) 機能

指定した切断理由コードから，d_connect( ) を再発行する必要があるかどうかの情報を，リターン値として取得します。

reason：

切断理由コード（構造体 d_discon 中の領域 reason に設定されたもの）を指定します。

(4) リターン情報

(a) リターン値

名称	値	再試行の必要性の有無
D_EROLDF	0x00000000	ユーザ独自の基準で判断してください。
D_ERBUSY	0x00200000	一定の間隔で何度か再試行してください。
D_ERPDST	0x00400000	
D_ERNRTY	0x00600000	
D_ERHARD	0x00800000	ハードウェア障害のため，再試行する必要はありません。
D_ERDEST	0x00a00000	保守員に連絡してください。
D_ERPSRC	0x00c00000	
D_ERHALT	0x00e00000	
D_ERER0R	0xffffffff	ユーザ独自の基準で判断してください。

(b) エラー情報

なし。

3.13 d_snd( )

(1) 名称

d_snd( )

データを送信します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_snd(fd,buf,nbytes);
int fd;
char *buf;
unsigned nbytes;
```

(3) 機能

ユーザデータを送信します。通信管理の状態が D_DATAXFER のときだけ発行できます。

同期方式の場合は，ユーザデータが通信管理に取り込まれた時点で正常終了します。システムでビジー状態になると，ビジー状態が解除されるまでリターンしません。このため，業務や運用を考慮して，送信するデータの量を調整する必要があります。

非同期方式の場合は，システムでビジー状態になると異常終了し，エラー情報として DFLOW を設定します（**フロー制御機能**）。このような場合は，次に示すどちらかの方法で d_snd( ) を再発行してください。

- select システムコールでビジー状態の解除を監視し，ビジー状態解除後に d_snd( ) を再発行してください。
- ビジー状態が解除されるまで，一定時間ごとに d_snd( ) を再発行してください。

fd：

データリンク端点識別子を指定します。

buf：

送信するデータを格納しておくバッファ（送信バッファ）のアドレスを指定します。

nbytes：

送信するデータのサイズを指定します。0 を指定すると，データを送信しないで正常終了します。なお，送信できるデータサイズの上限値は，あらかじめ構成定義文で指定しておきます。

(4) リターン情報

(a) リターン値

正常終了時：送信できたバイト数を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。
DLOOK	9	データリンク端点上でイベントが発生しました。ライブラリ関数 d_look( ) による確認が必要です。
DBADDATA	10	指定されたデータサイズがシステムによって許可されている限界を超えました。または，送信されたデータのサイズが送信バッファのサイズを超えました。
DFLOW	12	フロー制御機能によって送信要求が失敗しました（非同期方式の場合）。

3.14 d_snddis( )

(1) 名称

d_snddis( )

コネクションを解放します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_snddis(fd);
int fd;
```

(3) 機能

すでに確立しているコネクションを解放します。または，コネクションの確立要求を取り消します。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.15 d_unbind( )

(1) 名称

d_unbind( )

データリンク端点と仮想スロット番号の結合を解除します。

(2) 形式

```
#include <xnfs/diuser.h>
int d_unbind(fd);
int fd;
```

(3) 機能

データリンク端点と仮想スロット番号の結合を解除します。これ以降，このデータリンク端点に対するコネクション確立やデータ送信などの要求は受け付けません。

fd：

データリンク端点識別子を指定します。

(4) リターン情報

(a) リターン値

正常終了時：0 を返します。

異常終了時：-1 を返します。

(b) エラー情報

エラー名称	値	要因
DOUTSTATE	6	この関数が間違った順序で発行されました。
DSYSERR	8	この関数の実行中にシステムエラーが発生しました。詳細については，「3.16.1　詳細エラーコード」を参照してください。

3.16 詳細エラーコードと切断時の対処

3.16.1 詳細エラーコード

ライブラリ関数の実行中，エラー情報として DSYSERR が設定されると，さらに詳細な情報（詳細エラーコード）がグローバル変数 errno に設定されます。

詳細エラーコードに対応するエラー名称は，ヘッダファイル errno.h に設定されています。詳細エラーコードの意味と対処方法を表 3-3 に示します。

表 3-3 詳細エラーコードの意味と対処方法

名称	値	意味	対処
ENOENT	2	通信管理を識別するファイルがありません。	正しいパス名を指定してください。
EINTR	4	シグナルを受信しました。	アプリケーションプロセスを終了してください。
ENXIO	6	通信管理が動作不能になりました。	アプリケーションプロセスを終了してください。
EBADF	9	データリンク端点識別子が不正です。	正しいデータリンク端点識別子を指定してください。
EACCES	13	データリンク端点を確立する際に，通信管理を識別するファイルのアクセス権がありません。	ファイルの属性を修正してください。
EFAULT	14	不正な領域が指定されているため，メモリ異常が発生しました。	正しい領域アドレスを指定してください。
ENOTDIR	20	通信管理を識別するファイルのパスプレフィックス（パス名の最後を除いた部分）がディレクトリではありません。	正しいパス名を指定してください。
EISDIR	21	通信管理を識別するファイルがディレクトリです。	正しいパス名を指定してください。
EINVAL	22	引数が無効です。	正しい引数を指定してください。
ENFILE	23	システムでオープンできるファイルの上限値を超えました。	システムのファイル数を見直してください。
		データリンク端点の数が，システムで確立できる上限値を超えました。	構成定義の configuration 文の max_HDLCpass_link の値を見直してください。
EMFILE	24	1 プロセスで同時にオープンできるファイルの上限値を超えました。	1 プロセスで接続するコネクション数を見直してください。
ENOTTY	25	通信管理を識別するファイルがキャラクタスペシャルファイルではありません。	正しいパス名を指定してください。

3.16.2 コネクション切断時のプログラムの対処

コネクションが切断された場合は，まず，ライブラリ関数 d_rcvdis( ) で切断理由コードを取得します。次に，ライブラリ関数 d_retryck( ) を発行し，切断理由コードから，再度コネクションの確立を要求する必要があるかどうかを判定します。

詳細については，「3.11　d_rcvdis( )」および「3.12　d_retryck( )」を参照してください。

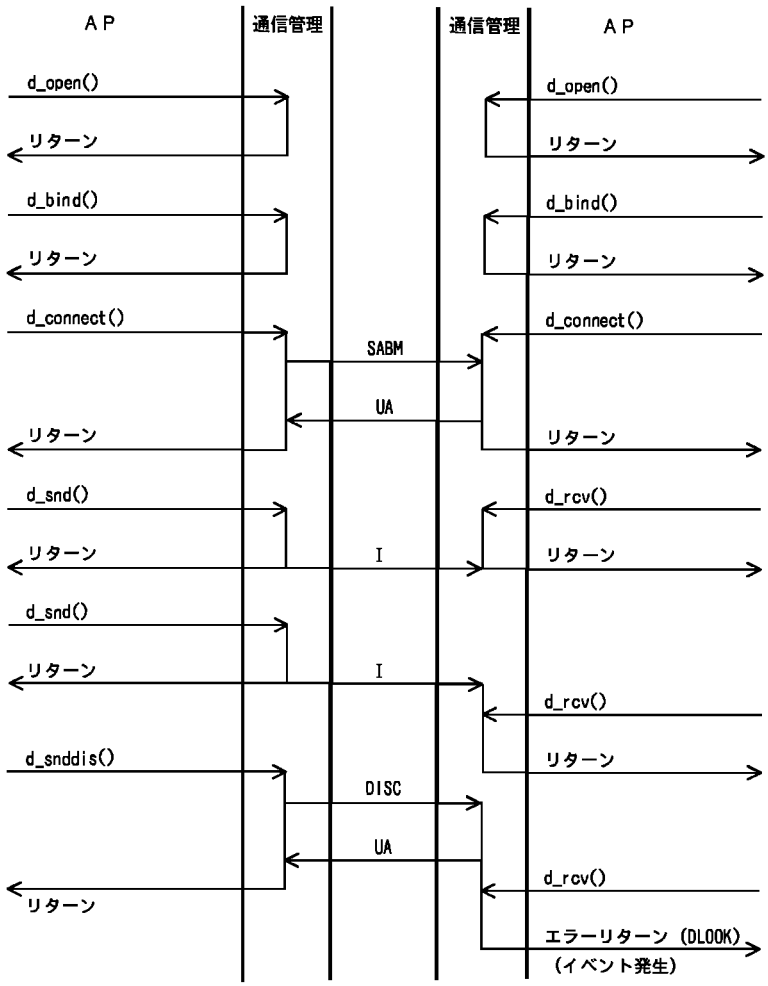
4 ライブラリ関数の使用手順

この章では，ライブラリ関数の使用手順，および通信管理での処理の流れについて説明します。

4.1 同期方式での手順

同期方式で，正常に処理を終了した場合，データリンク端点と仮想スロット番号の結合に失敗した場合，およびコネクションの確立に失敗した場合の一連の処理の流れを，図 4-1 ～図 4-3 に示します。なお，この節で示す例は，すべて HDLC-ABM 手順のものです。

図 4-1 同期方式での処理の流れ（正常終了した場合）



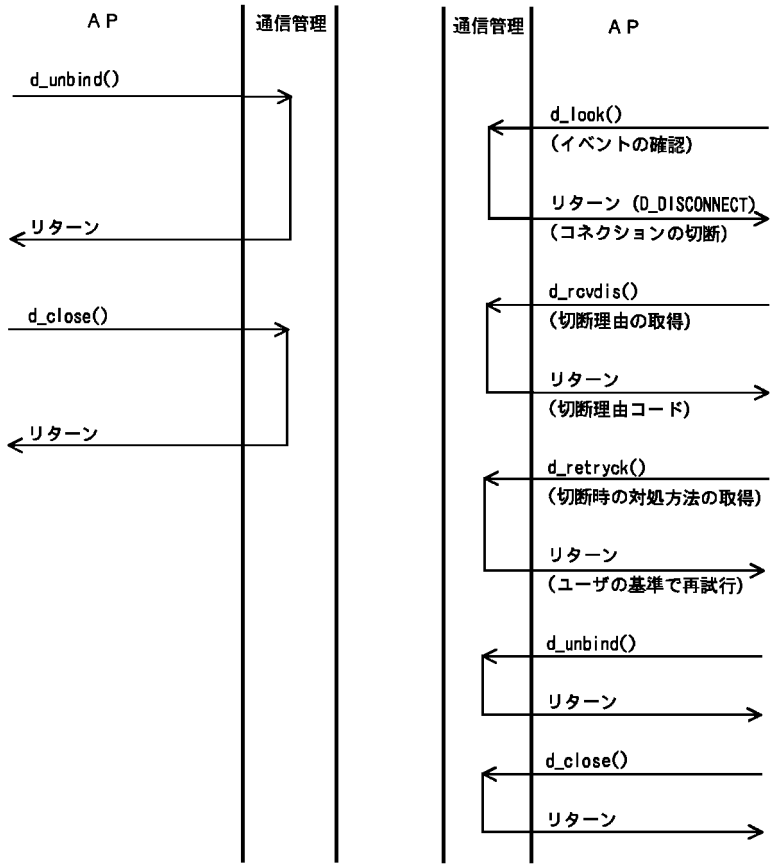
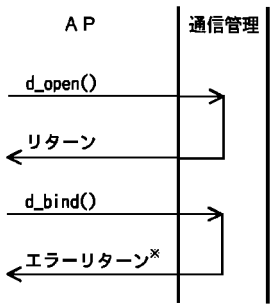
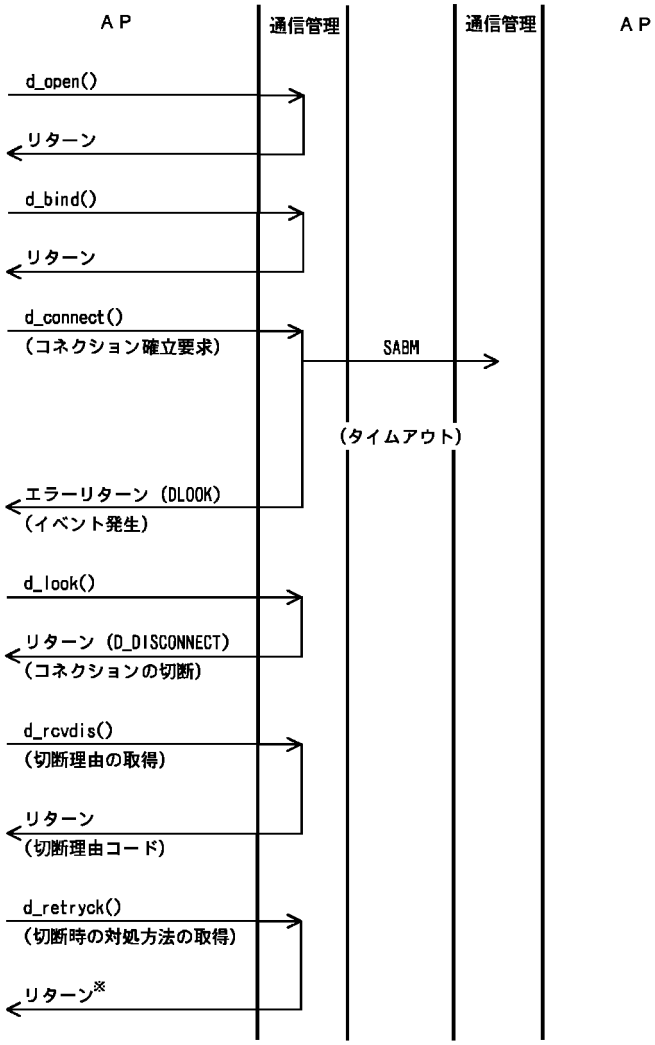


図 4-2　同期方式での処理の流れ（データリンク端点と仮想スロット番号の結合に失敗した場合）



注※　不正な仮想スロット番号を指定した場合は DBADADDR，すでに使用されている仮想スロット番号を重複して指定した場合は DNOADDR が設定されます。

図 4-3　同期方式での処理の流れ（コネクションの確立に失敗した場合）



注※　リターン値によって，d_connect()の再発行が有効な場合とそうでない場合があります。

同期方式で，通信バッファのサイズを超えるデータを送信，または受信した場合の処理の流れを，図 4-4 および図 4-5 に示します。

図 4-4　同期方式でのデータの送信（データ長がバッファサイズを超える場合）

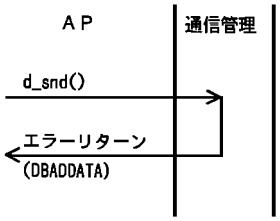
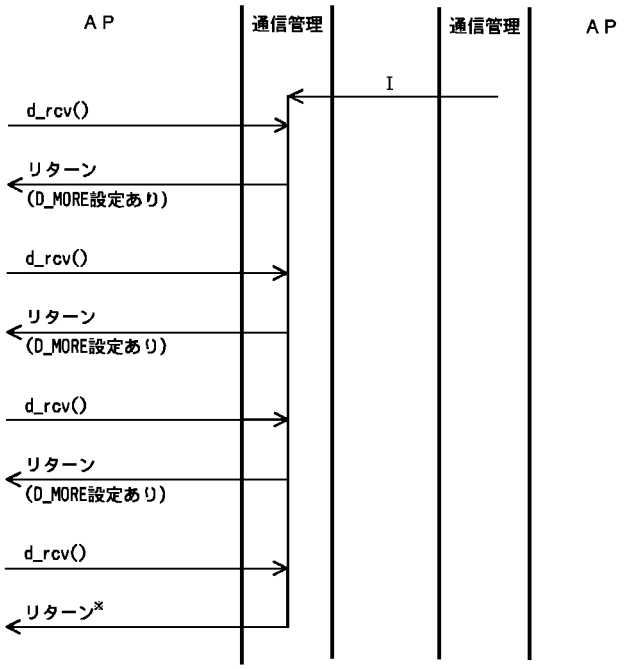


図 4-5　同期方式でのデータの受信（データ長がバッファサイズを超える場合）



注※　データの続きがあることを示すフラグD_MOREをクリアすることで，データの終了を示します。

同期方式でデータを送信，または受信し，システムでビジー状態になった場合の処理の流れを，図 4-6 および図 4-7 に示します。

図 4-6　同期方式でのデータの送信（ビジー状態になった場合）

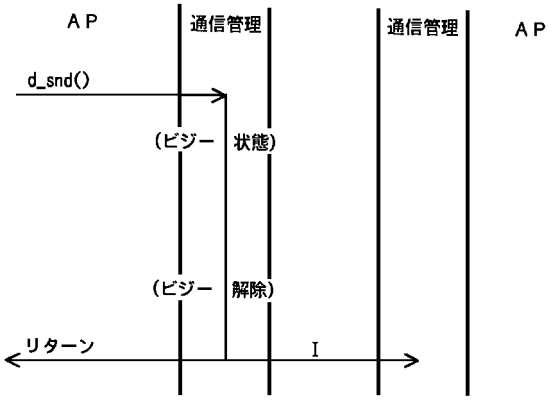
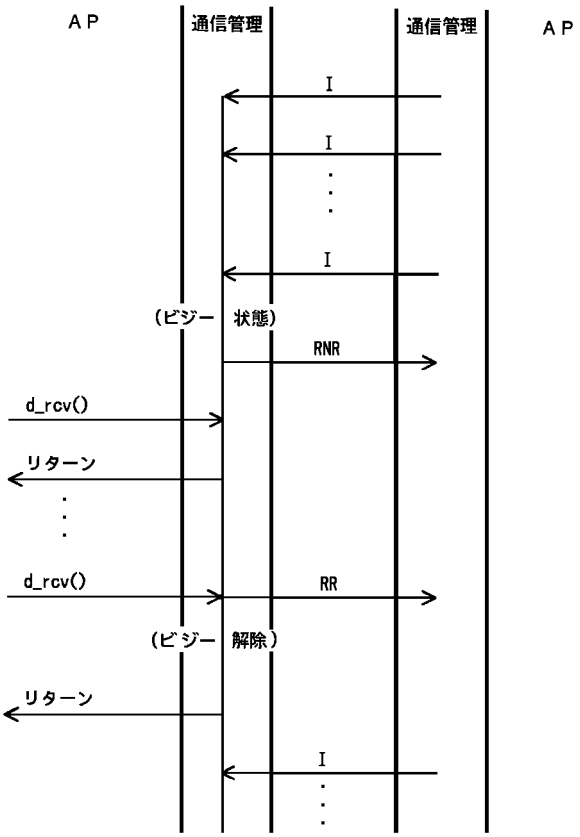


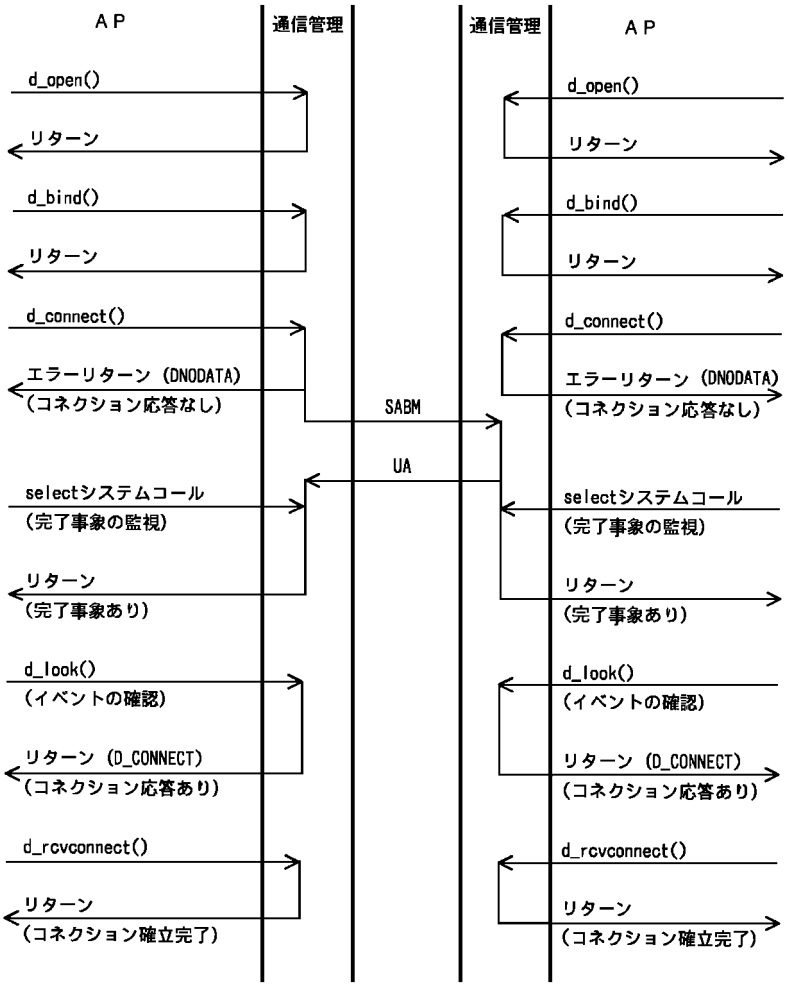
図 4-7　同期方式でのデータの受信（ビジー状態になった場合）



4.2 非同期方式での手順

非同期方式で，正常に処理を終了した場合，およびコネクションの確立に失敗した場合の一連の処理の流れを，図 4-8 および図 4-9 に示します。なお，この節で示す例は，すべて HDLC-ABM 手順のものです。

図 4-8 非同期方式での処理の流れ（正常終了した場合）



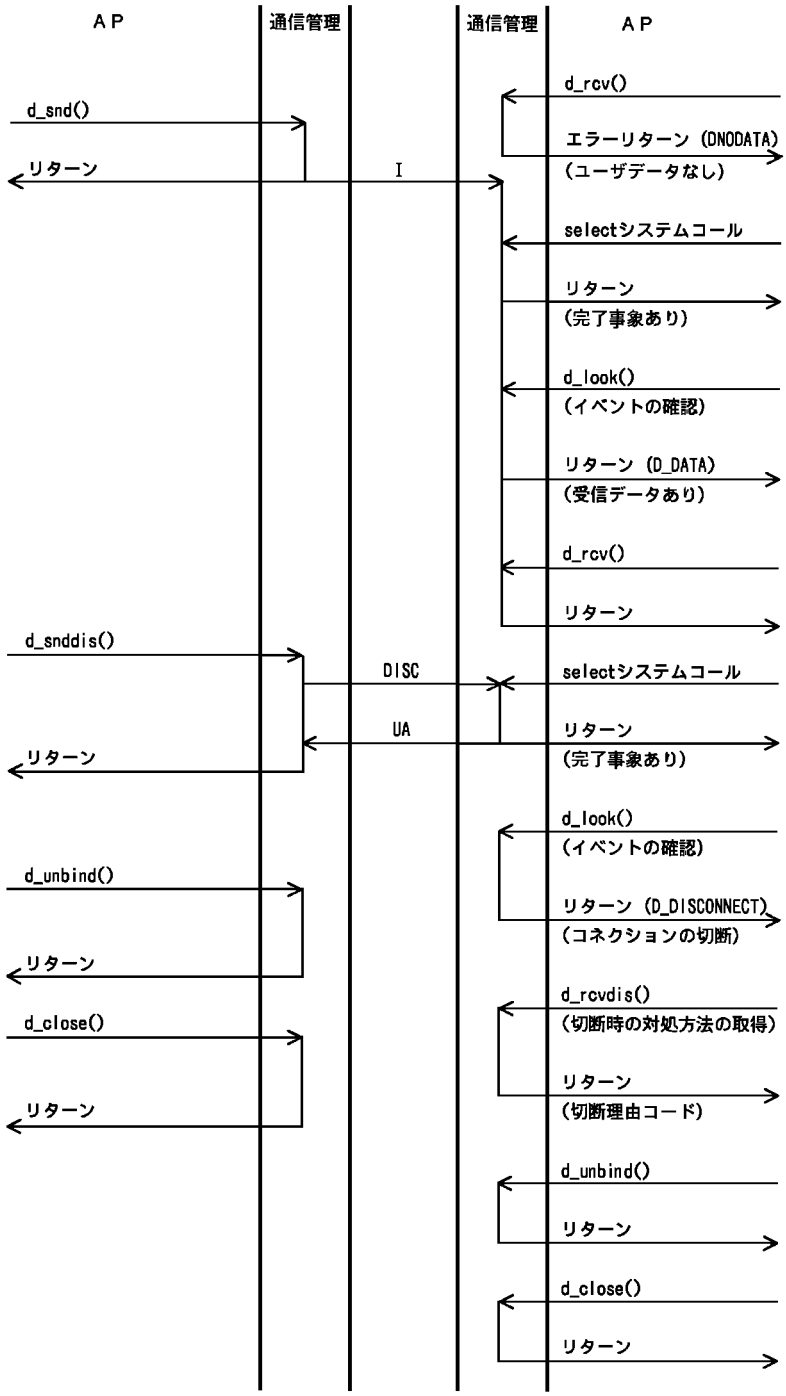
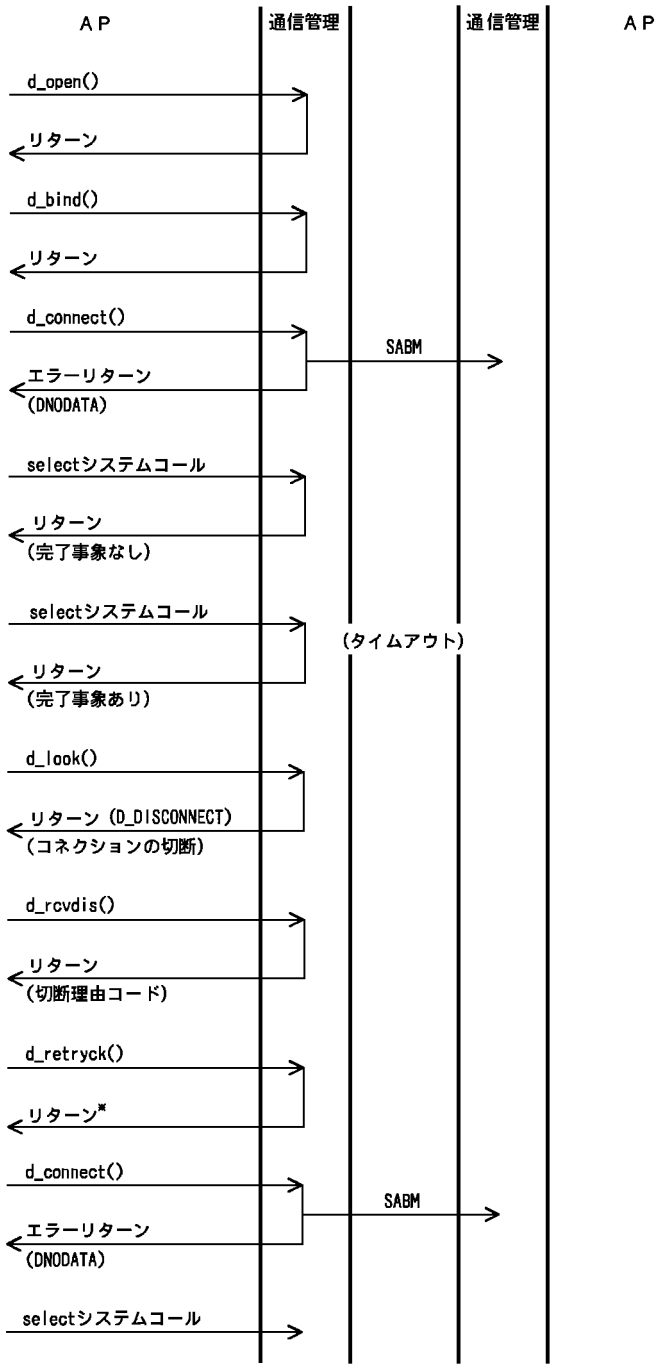

ＡＰ
通信管理
通信管理
ＡＰ
d_rcv()
d_snd()
エラーリターン（DNODATA）
リターン
I
（ユーザデータなし）
selectシステムコール
リターン
（完了事象あり）
d_look()
（イベントの確認）
リターン（D_DATA）
（受信データあり）
d_rcv()
リターン
d_snddis()
DISC
selectシステムコール
UA
リターン
リターン
（完了事象あり）
d_look()
（イベントの確認）
d_unbind()
リターン（D_DISCONNECT）
（コネクションの切断）
リターン
d_rcvdis()
（切断時の対処方法の取得）
d_close()
リターン
（切断理由コード）
リターン
d_unbind()
リターン
d_close()
リターン

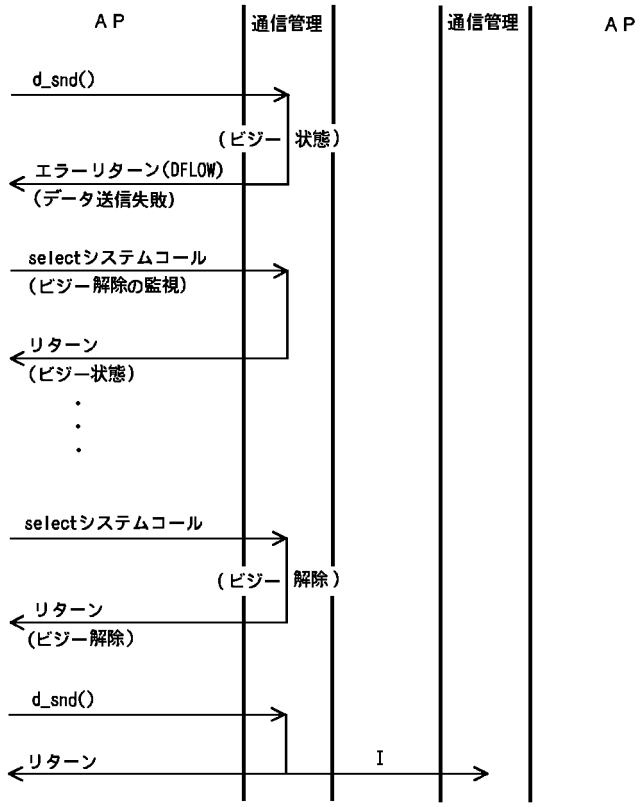
図 4-9　非同期方式での処理の流れ（コネクションの確立に失敗した場合）



注※　リターン値によって，d_connect()の再発行が有効な場合とそうでない場合があります。

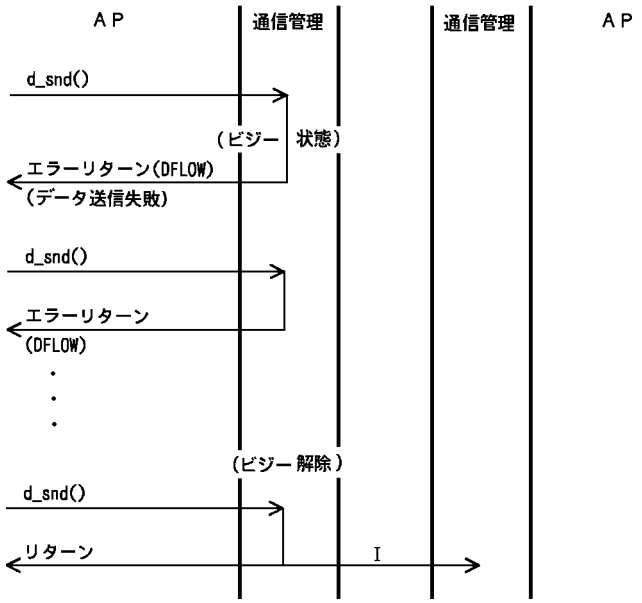
非同期方式でデータの送信を要求し，システムでビジー状態になった場合のフロー制御の処理の流れを，図 4-10 および図 4-11 に示します。

図 4-10　非同期方式でのフロー制御（select システムコールを使用する場合）



注　ライブラリ関数d_open()発行時，引数oflagにO_DFLOWを設定した場合の例です。

図 4-11 非同期方式でのフロー制御（d_snd( ) を連続発行する場合）



注 ライブラリ関数d_open()発行時，引数oflagにO_DFLOWを設定した場合の例です。

付録

付録 A　用語解説

（英字）

HDLC パススルー API

HDLC パススルー機能を実現するために，ライブラリ関数として提供されている API です。

HDLC パススルー機能

HDLC 手順でアプリケーションプログラム間の通信をするための機能です。

select システムコール

UNIX のシステムコールです。ライブラリ関数を非同期方式で実行した場合に，通信管理側の処理の完了，およびビジー状態の解除を監視するために使用します。

（カ行）

仮想スロット番号

データリンクユーザを識別するための番号です。あらかじめ構造体 d_bind に設定しておきます。

（タ行）

データリンク端点

データリンクユーザと通信管理との間の固有な通信端点です。

データリンク端点識別子

データリンク端点を識別するための番号です。ライブラリ関数 d_open( ) が正常終了したときに返されるリターン値がこれに当たります。

データリンクユーザ

アプリケーションプログラムが HDLC パススルー API を使用してデータの転送をするときの，最小単位を表す概念です。

同期方式

ライブラリ関数の処理方式の一つです。ライブラリ関数を呼び出したあと，通信管理側の処理がすべて完了した時点でアプリケーションプログラムに制御が戻ります。

（ハ行）

非同期方式

ライブラリ関数の処理方式の一つです。ライブラリ関数を呼び出したあと，通信管理側の処理が完了してもしなくても，すぐにアプリケーションプログラムに処理が戻ります。

フロー制御機能

ライブラリ関数 d_snd( ) を非同期方式で発行した場合に動作する機能です。システムでビジー状態になったときに，d_snd( ) を異常終了させて送信を抑止します。

付録 B　ライブラリ関数とエラー情報

ライブラリ関数が異常終了したときに設定されるエラー情報を表 B-1 に示します。

表 B-1　ライブラリ関数が異常終了したときに設定されるエラー情報

ライブラリ関数	エラー情報								
	DBAD ADDR	DNO ADDR	DOUT STAT E	DSYS ERR	DLOO K	DFLO W	DBAD DATA	DNO DATA	DNO DIS
d_bind( )	○	○	○	○					
d_close( )				○					
d_connect( )			○	○	○			○	
d_getstate( )				○					
d_look( )				○					
d_open( )				○					
d_rcv( )			○	○	○			○	
d_rcvconnect( )			○	○	○			○	
d_rcvdis( )				○	○				○
d_snd( )			○	○	○	○	○		
d_snddis( )			○	○					
d_unbind( )			○	○					

（凡例）

○：エラー情報が設定されることがあります。

空白：エラー情報は設定されません。

付録 C　従来製品との差異

従来製品（HI-UX/WE2 上で動作する XNF/S-E2）との差異を次に示します。

(1) ライブラリ

ライブラリは，共用ライブラリだけ提供します。アーカイブライブラリは，提供しません。ライブラリの使用方法については，「3.1　アプリケーションプログラムの作成手順」を参照してください。

(2) 詳細エラーコード

通信管理がメモリにロードされていない場合，詳細エラーコードは ENODEV ではなく，ENXIO になります。

索引

記号

B

D

E

F

H

N

O

P

R

S

U

あ

い

え

か

き

こ

し

せ

そ

つ

て

と

は

ひ

ふ

へ

ら

り

ソフトウェアマニュアルのサービス ご案内

１．マニュアル情報ホームページ

ソフトウェアマニュアルの情報をインターネットで公開しています。

URL　http://www.hitachi.co.jp/soft/manual/

ホームページのメニューは次のとおりです。

■マニュアル一覧	日立コンピュータ製品マニュアルを製品カテゴリ，マニュアル名称，資料番号のいずれかから検索できます。
■CD-ROMマニュアル	日立ソフトウェアマニュアルと製品群別CD-ROMマニュアルの仕様について記載しています。
■マニュアルのご購入	マニュアルご購入時のお申し込み方法を記載しています。
■オンラインマニュアル	一部製品のマニュアルをインターネットで公開しています。
■サポートサービス	ソフトウェアサポートサービスお客様向けページでのマニュアル公開サービスを記載しています。
■ご意見・お問い合わせ	マニュアルに関するご意見，ご要望をお寄せください。

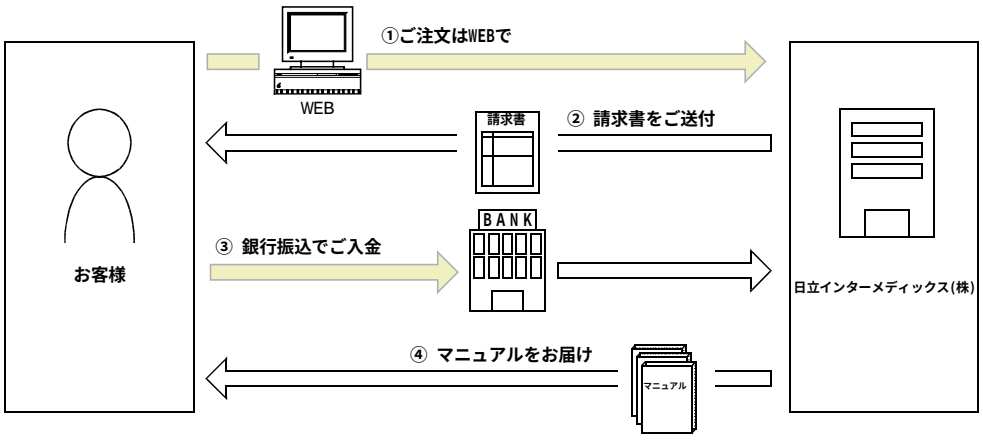
２．インターネットでのマニュアル公開

2 種類のマニュアル公開サービスを実施しています。

(1) マニュアル情報ホームページ「オンラインマニュアル」での公開
　製品をよりご理解いただくためのご参考として，一部製品のマニュアルを公開しています。

(2) ソフトウェアサポートサービスお客様向けページでのマニュアル公開
　ソフトウェアサポートサービスご契約のお客様向けにマニュアルを公開しています。公開しているマニュアルの一覧，本サービスの対象となる契約の種別などはマニュアル情報ホームページの「サポートサービス」をご参照ください。

３．マニュアルのご注文



① マニュアル情報ホームページの「マニュアルのご購入」にアクセスし，お申し込み方法をご確認のうえ WEB からご注文ください。ご注文先は日立インターメディックス(株)となります。

② ご注文いただいたマニュアルについて請求書をお送りします。

③ 請求書の金額を指定銀行へ振り込んでください。

④ 入金確認後 7 日以内にお届けします。在庫切れの場合は，納期を別途ご案内いたします。