HI.CommunicationEngine
NFSクライアント
リファレンスマニュアル

ルネサスセミコンダクタ パッケージ＆テストソリューションズ株式会社

ご注意

1. 本製品(ソフトウエア製品及びその関連ソフトウエア製品を含む。以下、同じ。)の使用に際しては、「外国為替及び外国貿易法」等、技術輸出に関する日本及び関連諸国の関係法規の遵守が必要となります。
2. 弊社は、本製品の使用に際しては、弊社もしくは第三者の特許権、著作権、商標権、その他の知的所有権等の権利に関し、別途、個別の契約書等(マニュアルの記載を含む。以下、同じ。)にて弊社による明示的な許諾がある場合を除き、その保証または実施権の許諾を行うものではありません。また本製品を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合、弊社はその責を負いませんので予めご了承ください。
3. 本製品およびその仕様、またはマニュアルに記載されている事柄については、将来、事前の予告なしに変更することがありますので、最終的な設計、ご購入、ご使用に際しましては、事前に最新の製品規格または仕様書(マニュアルを含む)をご確認ください。
4. 本製品の使用(マニュアル記載事項に基づくものも含む)により直接または間接に生ずるいかなる損害についても、弊社は一切の責任を負いません。また、本製品の配布に使用される搭載機器や媒体が原因の損害に対しましても、弊社は一切の責任を負いません。
5. 本製品を、宇宙、航空、原子力、燃焼制御、運輸、交通、各種安全装置、ライフサポート関連の医療機器等のように、特別な品質・信頼性が要求され、その故障や誤動作が直接人命を脅かしたり、人体に危害を及ぼす恐れのある用途向けには使用できません。お客様の用途がこれに該当するかどうか疑問のある場合には、事前に弊社営業担当迄ご相談をお願い致します。
6. 本製品を使用してお客様のシステム製品を設計される際には、通常予測される故障発生率、故障モードをご考慮の上、本製品の動作が原因での事故、その他の拡大損害を生じないようにフェールセーフ等の充分なシステム上の対策を講じて頂きますようお願い致します。
7. 本製品およびマニュアルの著作権は弊社が所有しております。お客様は、弊社から提供された本製品を、別途、個別の契約書等にて定める場合を除き、いかなる場合においても全体的または部分的に複写・解析・改変することはできないものとします。
8. お客様は、別途、個別の契約書等にて定める場合を除き、本製品のマニュアルの一部または全部を無断で使用、複製することはできません。
9. 弊社は、本製品を1台のコンピュータで使用する権利をお客様に対してのみ許諾します。よって、本製品を第三者へ譲渡、貸与、賃借することは許諾しないものとします。但し、別途、個別の契約書等にて定められる場合はその条件に従います。
10. 本製品をはじめ弊社製品およびその関連製品についてのお問い合わせ、ご相談は弊社営業担当迄お願い致します。

はじめに

このマニュアルは、HI.CommunicationEngine TCP/IPマネージャ上で動作するTCP/IPネットワークアプリケーション「NFSクライアント(HI.CommunicationEngine NFS)」について説明します。

本製品ではインタフェース関数としてNFSクライアント機能を実現しています。
NFSクライアントは、TCP/IPマネージャを経由してネットワーク上のRPC/NFSサーバとの通信を行う機能を提供します。

このリファレンスマニュアルではNFSクライアントの使用方法および関連事項を説明します。TCP/IPマネージャについては関連マニュアルを参照してください。

【関連マニュアル】
- HI.CommunicationEngine TCP/IPマネージャ リファレンスマニュアル
- 使用するμITRON のユーザーズマニュアル

目次

図表目次

1. 概要

1.1 NFS の機能概要

ネットワーク上のコンピュータ同士でファイル転送を行うためのプロトコルです。NFS は、RPC（リモートプロシージャコール）の上で実現されます。本製品は RPC version2、および NFS version2 です。また NFS version2 を実現する為に Portmap version2、MOUNT version 1 の機能もサポートします。

1.2 NFS クライアント

NFS クライアントは専用のタスクを持たずに、ユーザタスクのコンテキスト上で動作する「インタフェース関数」として実現します。

本製品では、RPC サーバと通信を行うためのインタフェース関数群と、NFS サーバとのデータ転送を主にしたインタフェース関数群を用意しております。
RPC サーバと通信を行うためのインタフェース関数群を本マニュアルでは RPC クライアント（低水準インタフェース関数）と呼びます。
NFS サーバとのデータ転送を主にしたインタフェース関数群を本マニュアルでは NFS クライアント（高水準インタフェース関数）と呼びます。

NFS クライアントは、RPC クライアントの機能を組合わせて実装しております。

RPC クライアント自体をユーザプログラムから呼出すことで、RPC サーバ上でユーザが作成したプログラムのプロシージャを実行することも可能です。

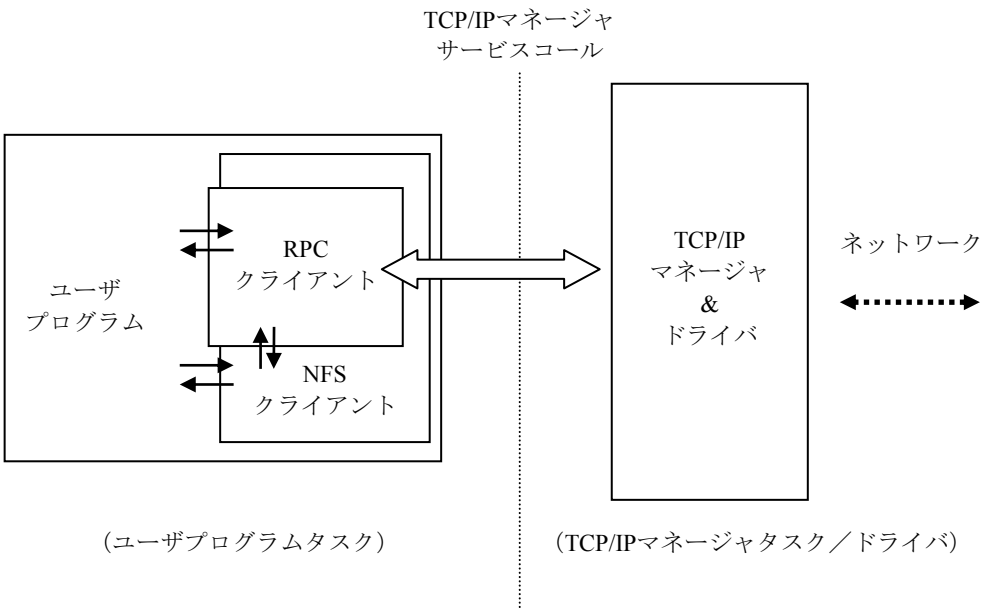


図 1-1 NFSクライアント

1.3　サポートする機能

本製品では、RPC クライアント（低水準インタフェース関数）として、下記の表で示す機能サポートします。NFS クライアント（高水準インタフェース関数）では、低水準インタフェース関数を組合わせることで、NFS サーバとのデータ転送を実現します。

表 1-1　サポートする機能

機能名称	プログラム番号	バージョン番号	プロトコル
Portmapプログラム通信機能	100000	2	TCP / UDP
MOUNTプログラム通信機能	100005	1	TCP / UDP
NFSプログラム通信機能	100003	2	TCP / UDP
指定RPCプログラム通信機能	ユーザ指定可	ユーザ指定可	TCP / UDP

2.　インタフェース関数説明形式

本章以降では、インタフェース関数についての詳細な説明を以下の形式で行っています。

（No.）関数名　　機能

【発行可能なシステム状態*1】

C言語インタフェース

関数の呼出し形式

パラメータ

型	パラメータ	パラメータの意味
・	・	・
・	・	・

リターンパラメータ

型	パラメータ	パラメータの意味
・	・	・
・	・	・

リターン値

リターン値	リターン値の意味
・	・
・	・

パケットの構造

struct {

・・・

・・・

解　説

・・・・・・・・・・・・

*1 発行可能なシステム状態を以下のアルファベットで示します

Ｔ：タスク実行状態

Ｄ：ディスパッチ禁止状態

Ｌ：ＣＰＵロック状態

Ｉ：非タスク部実行状態

なお、各状態の詳細は各ITRONのユーザーズマニュアルを参照してください。

発行可能なシステム状態以外の状態でインタフェース関数をコールした場合、システムの正常な動作は保証されません。

3. 低水準インタフェース関数

3.1 低水準インターフェース関数一覧

表 3-1、表 3-2 に 低水準インターフェース関数一覧表 を示します。

表 3-1 低水準インターフェース関数一覧表（1 / 2）

項番	種別	関数名	機能概要
1	システム	RPC_Init	RPCを初期化する
2		RPC_Version	RPCクライアントのバージョンを取得する
3		RPC_Start	RPCを起動する
4		RPC_Stop	RPCを停止する
5	Portmap 通信制御	RPC_PMAP_Open	Portmapプログラム通信を開始する
6		RPC_PMAP_Close	Portmapプログラム通信を終了する
7		RPC_PMAP_Cancel	Portmap応答待ちをキャンセルする
8		RPC_PMAP_SetAuth	RPC認証パラメータを設定する
9	Portmap プログラム機能呼出し	RPC_PMAP_Null	サーバプログラムの応答をテストする
10		RPC_PMAP_Set	サーバプログラムの登録・起動を行う
11		RPC_PMAP_UnSet	サーバプログラムの終了・登録解除を行う
12		RPC_PMAP_GetPort	サーバプログラムのポート番号を取得する
13		RPC_PMAP_Dump	サーバプログラムの全エントリを取得する
14		RPC_PMAP_CallIt	別プログラムのプロシージャを実行する
15	MOUNT 通信制御	RPC_MNT_Open	MOUNT通信を開始する
16		RPC_MNT_Close	MOUNT通信を終了する
17		RPC_MNT_Cancel	MOUNT応答待ちをキャンセルする
18		RPC_MNT_SetAuth	RPC認証パラメータを設定する
19	MOUNT プログラム機能呼出し	RPC_MNT_Null	サーバプログラムの応答をテストする
20		RPC_MNT_Mnt	ディレクトリをマウントする
21		RPC_MNT_Dump	マウントの全エントリを取得する
22		RPC_MNT_Umnt	ディレクトリをアンマウントする
23		RPC_MNT_UmntAll	全てのディレクトリをアンマウントする
24		RPC_MNT_Export	エクスポートリストを取得する
25	NFSv2 通信制御	RPC_NFS_Open	NFSv2通信を開始する
26		RPC_NFS_Close	NFSv2通信を終了する
27		RPC_NFS_Cancel	NFSv2応答待ちをキャンセルする
28		RPC_NFS_SetAuth	RPC認証パラメータを設定する
29	NFSv2 プログラム機能呼出し	RPC_NFS_Null	サーバプログラムの応答をテストする
30		RPC_NFS_GetAttr	属性を取得する
31		RPC_NFS_SetAttr	属性を設定する
32		RPC_NFS_Root	ファイルシステムのルートを取得する
33		RPC_NFS_LookUp	ファイルを探し出す
34		RPC_NFS_ReadLink	リンクから読出す
35		RPC_NFS_Read	ファイルから読出す
36		RPC_NFS_WriteCache	キャッシュの更新を行う
37		RPC_NFS_Write	ファイルへ書込む
38		RPC_NFS_Create	ファイルを作成する
39		RPC_NFS_Remove	ファイルを削除する
40		RPC_NFS_Rename	ファイル・ディレクトリ名称を変更する

表 3-2　低水準インターフェース関数一覧表（2 / 2）

項番	種別	関数名	機能概要
41	NFSv2 プログラム機能呼出し	RPC_NFS_Link	ハードリンクを作成する
42		RPC_NFS_SymLink	シンボリックリンクを作成する
43		RPC_NFS_MkDir	ディレクトリを作成する
44		RPC_NFS_RmDir	ディレクトリを削除する
45		RPC_NFS_ReadDir	ディレクトリリストを取得する
46		RPC_NFS_StatFs	ファイルシステム状態を取得する
47	直接RPC通信制御・機能呼出し	RPC_Open	指定したプログラムとRPC通信を開始する
48		RPC_Close	指定したプログラムとRPC通信を終了する
49		RPC_Cancel	RPC応答待ちをキャンセルする
50		RPC_SetAuth	RPC認証パラメータを設定する
51		RPC_ExecProcedure	指定したプロシージャを実行する

3.1.1　　システムインターフェース関数

(1)　　RPC_Init　　RPC 環境の初期化

【T／D／L／I】

C言語インタフェース

void RPC_Init(void);

パラメータ

無し

リターンパラメータ

無し

解　説

RPCの環境を初期化します。

本関数は、システム初期化時などで、1度だけコールしてください。

(2)　RPC_Version　　RPC クライアントバージョンの取得

【T／D／L／I】

C言語インタフェース

```
ER rtcd = RPC_Version(T_RPC_VERSION *par);
```

パラメータ

T_RPC_VERSION	*par	パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
UW	par->version	RPCクライアントバージョン
UW	par->status	RPCクライアントステータス

パケットの構造

```
typedef struct {
        UW      version;        RPCクライアントバージョン
        UW      status;         RPCクライアントステータス
                                （RPCSTA_STOP = 0，RPCSTA_START = 1,
                                  RPCSTA_STARTING = 2,
                                  RPCSTA_STOPPING = 3）
} T_RPC_VERSION;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（parが0または4の倍数以外）

解　説

RPCクライアント機能のバージョンを取得します。

正常に実行されると、par->versionにRPCクライアントのバージョンを、par->statusにRPCクライアントの状態を設定します。

本関数は、RPCクライアントがどのような状態でも実行可能です。

(3)　RPC_Start　　RPC の起動

【T／D】

C言語インタフェース

ER rtcd = RPC_Start(T_RPC_START *par);

パラメータ		
T_RPC_START	*par	RPC起動パラメータ

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
W	par->memlen	使用したメモリの長さ

パケットの構造		
typedef struct {		
H	pmap_maxid;	最大Portmapプログラム通信用ID
H	mnt_maxid;	最大MOUNTプログラム通信用ID
H	nfs_maxid;	最大NFSv2プログラム通信用ID
H	rpc_maxid;	最大指定RPCプログラム通信用ID
UW	*memaddr;	使用するメモリの先頭アドレス
W	memlen;	使用可能なメモリの長さ
} T_RPC_START;		

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー（par,par->memaddrが0または4の倍数以外）
E_NOMEM	メモリ不足（必要なメモリが確保できない）
E_OBJ	オブジェクト状態異常（既にRPCクライアントが動作開始している）

解　説

RPCを起動します。

pmap_maxid、mnt_maxid、nfs_maxidおよびrpc_maxidにはそれぞれRPCクライアント機能して使用する最大Portmapプログラム通信用ID、最大MOUNTプログラム通信用ID、最大NFSv2プログラム通信用IDおよび最大指定RPCプログラム通信用IDを指定します。本関数では、必要となるメモリをmemaddrで指定されたメモリから切り出し、使用したメモリの長さをmemlenに返します。

指定したmemlenが必要なメモリサイズに満たない場合には、memlenに必要なメモリの長さを設定し、エラーコードとしてE_NOMEMを返します。

必要なメモリサイズは、『NFSクライアント メモリサイズ計算シート(NFSV1_SZ.xls)』で事前に確認することができます。

(4)　RPC_Stop　RPC の停止

【T／D】

C言語インタフェース

ER rtcd = RPC_Stop(void);

パラメータ

無し

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ILUSE	不正使用（RPCクライアントが停止している）

解　説

RPCを停止します。

本関数の発行後、RPCクライアントの動作を停止し、各機能は無効となります。そのためRPC_Start以外の低水準インターフェース関数の動作は保証されません。

3.1.2 Portmap プログラム通信制御用インターフェース関数

(1) RPC_PMAP_Open RPC サーバの Portmap プログラムと通信を開始する

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_Open(T_RPC_OPEN *par, TMO tmout);

パラメータ		
T_RPC_OPEN	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
ID	par->returnid	Portmapプログラム通信用ID

パケットの構造

```
typedef struct {
        T_IPV4EP        myaddr;         クライアントアドレス
        T_IPV4EP        svaddr;         RPCサーバアドレス
        H               protocol;       Portmapプログラム通信用プロトコル
                                        （IPPROTO_TCP = 6, IPPROTO_UDP = 17）
        ID              reqcepid;       TCP/IPマネージャ通信端点ID
        ID              returnid;       RPCサーバプログラム通信用ID
} T_RPC_OPEN;

typedef struct {
        UW              ipaddr;         IPアドレス
        UH              portno;         ポート番号
} T_IPV4EP;
```

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、myaddrが動作を開始しているアドレスと異なる， svaddr.ipaddrが0または0xffffffff（TCPの場合），svaddr.ipaddrが0（UDPの場合）， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_OBJ	オブジェクト状態エラー （ポート番号既使用，指定したIDのTCPまたはUDP通信端点が使用中）
E_ID	不正ID番号（reqcepid＜0, reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大）
E_NOID	ID番号不足（Portmapプログラム通信用IDに空きが無い）
E_TMOUT	タイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_CLS	接続要求が拒否された（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

解　説

RPCサーバのPortmapプログラムと通信を開始します。

par->svaddrで指定したRPCサーバに対して、par->protocolで指定したトランスポートプロトコルを使用し通信開始の設定を行います。
その際、RPCサーバのトランスポートのポート番号には、必ず $111_{(10)}$（Portmap）を指定してください。

par->reqcepidには、TCP/IPマネージャで使用する通信端点IDを指定します。特にIDを指定する必要がない場合は、par->reqcepidに0を指定することで空いている通信端点を使用します。

トランスポートプロトコルにTCPを指定して本関数を実行する場合は、RPCサーバに対してTCPの接続

までを行います。またトランスポートプロトコルにUDPを指定した場合は、およびRPCサーバとの通信準備（システムテーブル設定等）までを行います。

正常に完了した場合、par->returnidにPortmapプログラム通信用IDを設定し、リターン値としてE_OKを返します。

(2)　RPC_PMAP_Close　　RPC サーバの Portmap プログラムと通信を終了する

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_Close(ID pmapid, TMO tmout);

パラメータ		
ID	pmapid	Portmapプログラム通信用ID
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー（tmout<TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*1）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	ポーリング失敗またはタイムアウト（TCP）
E_RLWAI	処理のキャンセル，待ち状態の強制解除（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

*1 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

解　説

RPCサーバとのPortmapプログラム通信を終了します。

pmapidで指定したPortmapプログラム通信がトランスポートプロトコルがTCPの場合は、TCPの接続を切断し、TCP資源の解放を行います。また、トランスポートプロトコルがUDPの場合は、UDP資源の解放を行います。

正常に完了した場合、リターン値としてE_OKを返します。

(3)　RPC_PMAP_Cancel　　Portmap プログラム応答待ちをキャンセルする

【T／D】

C言語インタフェース

ER rtcd = RPC_PMAP_Cancel(ID pmapid);

パラメータ

ID	pmapid	Portmapプログラム通信用ID

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*1）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_ILUSE	不正使用（RPCクライアントが停止している）

*1 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

解　説

待ちをキャンセルします。

pmapidで示されたRPCサーバ上のPortmapプログラムからの応答待ち処理を解除します。

本関数で待ち解除された関数には、E_RLWAIを返します。

(4)　RPC_PMAP_SetAuth Portmap プログラム用の RPC 認証パラメータを設定する

【T／D】

C言語インタフェース

```
ER rtcd = RPC_PMAP_SetAuth(ID pmapid, T_RPC_AUTH *par);
```

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_AUTH	*par	RPC認証パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        UW              flavor;                 認証方式
                                                  （RPC_AUTH_NULL = 0,
                                                   RPC_AUTH_UNIX = 1）
        T_RPC_NAME      machine;                RPCクライアント名
        UW              uid;                    ユーザーID
        UW              gid;                    グループID
        UW              auxgidcnt;              予備グループID数
        UW              auxgid[RPC_MAXAUXGID];  予備グループID
                                                  （RPC_MAXAUXGID = 16）
} T_RPC_AUTH;

typedef struct name{
        W       namelen;                        文字列の長さ
        UB      namedata[RPC_MAXNAMLEN+1];      文字列のデータ
                                                  （RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，par->flavor＜RPC_AUTH_NULL(0), par->flavor＞RPC_AUTH_UNIX(1), par->machine.namelen＜0, par->machine.namelen＞RPC_MAXNAMLEN(255), par->auxgidcnt＞RPC_MAXAUXGID(16)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*1）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）

*1 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

解　説

pmapidで指定したPortmapプログラム通信用IDにRPC認証パラメータを設定します。
pmapidが有効なIDである間（RPC_PMAP_OpenからRPC_PMAP_Closeの間）のみ設定可能です。

par->flavorをRPC_AUTH_NULL(0)に指定して本関数を呼出した場合、以降のパラメータは全て無効となります（チェックも行いません）。

本関数で認証パラメータを設定していない場合は、各プログラム機能呼出し関数のpar->atrにATR_AUTH(0x00000001)を設定しても、RPCヘッダ部の認証パラメータは認証なし(flavor=AUTH_ NULL(0))で送信します。

本関数が正常終了すると、次のプログラム実行要求の送信データより、RPCヘッダ部の認証パラメータへ設定値を反映します。（各プログラム機能呼出し関数のpar->atrにRPC_ATR_AUTH(0x00000001)を設定した場合）

3.1.3 Portmap プログラム機能呼出し用インターフェース関数

(1) RPC_PMAP_Null **サーバのプログラム応答テスト**

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_Null(ID pmapid, T_RPC_PMAP_NULL *par, TMO tmout);

パラメータ		
ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_NULL	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果
} T_RPC_PMAP_NULL;

typedef struct {
        UW              replystate; RPC応答状態
        UW              acceptstate;RPC受付状態
        UW              status;     機能実行結果
} T_RPC_RESULT;
```

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*1）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*1 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）	
0	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバに対するPortmapプログラムの応答テストを行います。

pmapidで指定したRPCサーバに対し、Portmapプログラム動作中かを確認する為のテストを行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*2をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

応答テストが正常に実行されると、正常終了としてpar->result.statusに 0 を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*2 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(2)　RPC_PMAP_Set　　**サーバのプログラム登録と起動**

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_Set(ID pmapid, T_RPC_PMAP_SET *par, TMO tmout);

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_SET	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_MAPPING   map;        マッピング情報
} T_RPC_PMAP_SET;

typedef struct mapping {
        UW              prog;       プログラム番号
        UW              vers;       プログラムのバージョン番号
        UW              prot;       プログラムのプロトコル番号
                                    （IPPROTO_TCP = 6、IPPROTO_UDP = 17）
        UH              portno;     プログラムのポート番号
        UH              reserve;    予約（未使用）
} T_RPC_MAPPING;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*2）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）

TRUE(1)	登録成功
FALSE(0)	登録失敗
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバに対するプログラムの登録・起動を行います。

pmapidで指定したRPCサーバに対し、プログラム番号par->map.prog、バージョン番号 par->map.vers、トランスポートプロトコル番号 par->map.protとポート番号 par->map.portを登録します。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

RPCサーバに対するマッピング（登録）が成功した場合はpar->result.statusにTRUE(1)を設定します。
RPCサーバに対するマッピングが失敗した場合（既に登録されている場合など）は、登録失敗としてpar->result.statusにFALSE(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(3)　RPC_PMAP_UnSet　　**サーバのプログラム停止と登録解除**

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_UnSet(ID pmapid, T_RPC_PMAP_UNSET *par, TMO tmout);

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_UNSET	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_MAPPING   map;        マッピング情報*2
} T_RPC_PMAP_UNSET;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_MAPPING：RPC_PMAP_SetのT_RPC_MAPPINGを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*3）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）

TRUE(1)	登録解除成功
FALSE(0)	登録解除失敗
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバに対するプログラムの終了・登録解除を行います。

pmapidで指定したRPCサーバに対し、par->map.progにプログラム番号、par->map.versにプログラムのバージョン番号に対応付けされたプログラムの登録を解除します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

RPCサーバに対するマッピング（登録解除）が成功した場合はpar->result.statusにTRUE(1)を設定しま

す。
　RPCサーバに対するマッピングが失敗した場合（指定した内容で登録されていない場合など）は、登録解除失敗としてpar->result.statusにFALSE(0)を設定します。

　本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(4)　RPC_PMAP_GetPort　　**プログラムポート番号の取得**

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_GetPort(ID pmapid, T_RPC_PMAP_GETPORT *par, TMO tmout);

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_GETPORT	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_MAPPING   map;        マッピング情報*2
} T_RPC_PMAP_GETPORT;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_MAPPING：RPC_PMAP_SetのT_RPC_MAPPINGを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*3）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）

0	プログラムが登録されていない
1～65535	プログラムが呼び出し要求を待っているポート番号
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバで実行しているプログラムのポート番号を取得します。

pmapidで指定したRPCサーバに対し、par->map.progにプログラム番号、par->map.versにプログラムのバージョン番号、par->map.protにトランスポートプロトコルの組合せで実行しているプログラムのポート番号の取得を要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

RPCサーバにプログラムが登録され実行中である場合は、正常終了としてpar->result.statusにポート番号を設定します。
RPCサーバにプログラムが登録されていない場合は、ポート番号取得失敗としてpar->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(5)　RPC_PMAP_Dump　　**サーバプログラム・マッピング情報リストの取得**

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_Dump(ID pmapid, T_RPC_PMAP_DUMP *par, TMO tmout);

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_DUMP	*par	パラメータテーブル
H	par->availcount	格納可能なマッピング情報数
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_MAPPING	par->maplist	マッピング情報リスト
H	par->recvcount	受信したマッピング情報数
H	par->storecount	格納したマッピング情報数

パケットの構造

```
typedef struct {
        ATR             atr;            実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;         実行結果*1
        T_RPC_MAPPING   *maplist;       マッピング情報の先頭アドレス*2
        H               availcount;     格納可能なマッピング情報数
        H               recvcount;      受信したマッピング情報数
        H               storecount;     格納したマッピング情報数
} T_RPC_PMAP_DUMP;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_MAPPING：RPC_PMAP_SetのT_RPC_MAPPINGを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par, par->maplistが0または4の倍数以外，par->availcount≦0, tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*3）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）

TRUE(1)	正常終了
FALSE(0)	異常終了
その他	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバで実行しているプログラムのマッピング情報リストを取得します。

pmapidで指定したRPCサーバに対し、動作中プログラムのマッピング情報リストを取得します。
par->availcountには格納可能なマッピング情報数を指定します。
par->maplistにはマッピング情報を格納する領域の先頭アドレスを指定します。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*5をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

マッピング情報リストの取得が行われると、par->recvcountに受信したマッピング情報数を設定します。
受信したマッピング情報数がpar->availcountより小さい、またはpar->availcountと等しい場合は、受信したマッピング情報数分、par->maplistで指定した領域に格納し、par->storecountに格納したマッピング情報数を設定します。
また、受信したマッピング情報数がpar->availcountより大きい場合は、par->availcountに指定した分だけpar-> maplistで指定した領域にマッピング情報を格納し、par->storecountに格納したマッピング情報数を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

UDPでオープンした通信では、RPCサーバが多くのマッピング情報を持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->availcountで指定した分のpar->maplistの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->availcountで格納可能としたマッピング情報分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*5 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(6)　RPC_PMAP_CallIt　　別プログラムのプロシージャ実行

【T】

C言語インタフェース

ER rtcd = RPC_PMAP_CallIt(ID pmapid, T_RPC_PMAP_CALLIT *par, TMO tmout);

パラメータ

ID	pmapid	Portmapプログラム通信用ID
T_RPC_PMAP_CALLIT	*par	パラメータテーブル
W	par->res.availlen	格納可能なプロシージャ実行結果の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_CALL_RES	par->res	出力パラメータ
W	par->res.recvlen	受信したプロシージャ実行結果の長さ
W	par->res.storelen	格納したプロシージャ実行結果の長さ

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_CALL_ARGS         args;       入力パラメータ
        T_RPC_CALL_RES          res;        出力パラメータ
} T_RPC_PMAP_CALLIT;

typedef struct call_args {
        UB                      *args;      プロシージャ引数の先頭アドレス
        W                       arglen;     プロシージャ引数の長さ
        UW                      prog;       プログラム番号
        UW                      vers;       プログラムバージョン
        UW                      proc;       プロシージャ番号
} T_RPC_CALL_ARGS;

typedef struct call_res {
        UB                      *res;       プロシージャ実行結果
        W                       availlen;   格納可能なプロシージャ実行結果の長さ
        W                       recvlen;    受信したプロシージャ実行結果の長さ
        W                       storelen;   格納したプロシージャ実行結果の長さ
        UH                      portno;     トランスポート ポート番号
        UH                      reserve;    予約（未使用）
} T_RPC_CALL_RES;
```

*1 T_RPC_RESULT : RPC_PMAP_Null の T_RPC_RESULT を参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par, par->call_arg.args, par->call_res.resが0または4の倍数以外, tmout＝TMO_POL(0),tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（pmapid≦0，pmapid＞pmap_maxid*2）
E_NOEXS	オブジェクト未生成（pmapidが存在していない）
E_OBJ	オブジェクト状態エラー（pmapidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	処理のキャンセル，待ち状態の強制解除

E_CLS　　RPC サーバとの接続が切断された

*2 pmap_maxid：RPC_Startで指定した最大Portmapプログラム通信用ID

実行結果（par->result.status）

0　　正常終了
その他の値　　『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバ上の別プロシージャを呼出します。

pmapidで指定したRPCサーバに対し、par->argsで指定したプログラムのプロシージャを、par->args.argsのパラメータを使用して直接呼出します。
この関数には、
・　実行したいプログラムのポート番号を知らなくても実行可能
・　ブロードキャストにてプログラムが実行可能
のメリットがあります。（ただし実行可能なプログラムはUDPで実行中のプログラムに限ります。）
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

par->argsで指定したプログラムのプロシージャが正常に実行されると、正常終了としてpar->resにプロシージャの応答を、par->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

本関数で扱われるデータ(par->args.args, par->res.res)は、ネットワークバイトオーダ（ビッグエンディアン）となります。リトルエンディアンで使用する場合は注意が必要です。エンディアンに依存しないコードにする為に、バイトストリームでデータを作成することをお勧めします。

UDPでオープンした通信では、par->args.argslenで指定したデータ長とヘッダとの合計が1472バイトを超える大きなCALLデータを持つ場合、クライアント側からIPでフラグメントしたデータを送信します。TCP/IPマネージャの設定でIPフラグメント送信ができない設定で構築されている場合は、正しく送信できません。また、RPCクライアント内部では、UDPの送信は8760バイトの領域で送信データ作成後、CALLデータを送信する為、IPフラグメント送信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを送信しようとした場合、8760バイト以降は無視されます。

上記同様に、UDPでオープンした通信では、RPCサーバで実行したプロシージャがヘッダとの合計が1472バイトを超える大きなREPLYデータを持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->res.availlenで指定した分のpar->res.resの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->res.availlenで格納可能としたREPLYデータ分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*3 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

3.1.4 MOUNT 通信制御用インターフェース関数

(1) RPC_MNT_Open RPC サーバとの MOUNT 通信を開始する

【T】

C言語インタフェース

ER rtcd = RPC_MNT_Open(T_RPC_OPEN *par, TMO tmout);

パラメータ

T_RPC_OPEN	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
ID	par->returnid	Portmapプログラム通信用ID

パケットの構造

```
typedef struct {
        T_IPV4EP        myaddr;         クライアントアドレス
        T_IPV4EP        svaddr;         RPCサーバアドレス
        H               protocol;       MOUNT通信用プロトコル
                                        （IPPROTO_TCP = 6, IPPROTO_UDP = 17）
        ID              reqcepid;       TCP/IPマネージャ通信端点ID
        ID              returnid;       RPCサーバプログラム通信用ID
} T_RPC_OPEN;

typedef struct {
        UW              ipaddr;         IPアドレス
        UH              portno;         ポート番号
} T_IPV4EP;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、myaddrが動作を開始しているアドレスと異なる， svaddr.ipaddrが0または0xffffffff（TCPの場合），svaddr.ipaddrが0（UDPの場合）， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_OBJ	オブジェクト状態エラー （ポート番号既使用，指定したIDのTCPまたはUDP通信端点が使用中）
E_ID	不正ID番号（reqcepid＜0, reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大）
E_NOID	ID番号不足（MOUNTプログラム通信用IDに空きが無い）
E_TMOUT	タイムアウト（TCP）
E_RLWAI	処理のキャンセル，待ち状態の強制解除（TCP）
E_CLS	接続要求が拒否された（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

解　説

RPCサーバとのMOUNT通信を開始します。

par->svaddrで指定したRPCサーバに対して、par->protocolで指定したトランスポートプロトコルを使用し通信開始の設定を行います。

その際、par->svaddr.portno（トランスポートのポート番号）には、Portmapプログラム通信等で取得したMOUNT用のポート番号を指定してください。

par->reqcepidには、TCP/IPマネージャで使用する通信端点IDを指定します。特にIDを指定する必要がない場合は、par->reqcepidに0を指定することで空いている通信端点を使用します。

トランスポートプロトコルにTCPを指定して本関数を実行する場合は、RPCサーバに対してTCPの接続までを行います。またトランスポートプロトコルにUDPを指定した場合は、RPCサーバとの通信準備（システムテーブル設定等）までを行います。

正常に完了した場合、par->returnidにMOUNTプログラム通信用IDを設定し、リターン値としてE_OKを返します。

(2)　RPC_MNT_Close　　RPC サーバとの MOUNT 通信を終了する

【T】

C言語インタフェース

ER rtcd = RPC_MNT_Close(ID mntid, TMO tmout);

パラメータ

ID	mntid	MOUNTプログラム通信用ID
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（tmout<TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	ポーリング失敗またはタイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

*1 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

解　説

RPCサーバとのMOUNT通信を終了します。

mntidで指定したMOUNT通信がトランスポートプロトコルがTCPの場合は、TCPの接続を切断し、TCP資源の解放を行います。また、トランスポートプロトコルがUDPの場合は、UDP資源の解放を行います。

正常に完了した場合、リターン値としてE_OKを返します。

(3)　RPC_MNT_Cancel　　MOUNT プログラム応答待ちをキャンセルする

【T／D】

C言語インタフェース

ER rtcd = RPC_MNT_Cancel(ID mntid);

パラメータ

ID	mntid	MOUNTプログラム通信用ID

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid[*1]）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_ILUSE	不正使用（RPCクライアントが停止している）

*1 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

解　説

待ちをキャンセルします。

mntidで示されたRPCサーバ上のMOUNTプログラムからの応答待ち処理を解除します。

本関数で待ち解除された関数には、E_RLWAIを返します。

(4)　RPC_MNT_SetAuth　MOUNT プログラム用の RPC 認証パラメータを設定する

【T／D】

C言語インタフェース

```
ER rtcd = RPC_MNT_SetAuth(ID mntid, T_RPC_AUTH *par);
```

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_AUTH	*par	RPC認証パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        UW              flavor;                 認証方式
                                                  （RPC_AUTH_NULL = 0,
                                                   RPC_AUTH_UNIX = 1）
        T_RPC_NAME      machine;                RPCクライアント名
        UW              uid;                    ユーザーID
        UW              gid;                    グループID
        UW              auxgidcnt;              予備グループID数
        UW              auxgid[RPC_MAXAUXGID];  予備グループID
                                                  （RPC_MAXAUXGID = 16）
} T_RPC_AUTH;

typedef struct name {
        W       namelen;                        文字列のデータ長
        UB      namedata[RPC_MAXNAMLEN+1];      文字列のデータ
                                                  （RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，par->flavor＜RPC_AUTH_NULL(0), par->flavor＞RPC_AUTH_UNIX(1), par->machine.namelen＜0, par->machine.namelen＞RPC_MAXNAMLEN(255), par->auxgidcnt＞RPC_MAXAUXGID(16)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）

*1 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

解　説

mntidで指定したMOUNTプログラム通信用IDにRPC認証パラメータを設定します。
mntidが有効なIDである間（RPC_MNT_OpenからRPC_MNT_Closeの間）のみ設定可能です。

par->flavorをRPC_AUTH_NULL(0)に指定して本関数を呼出した場合、以降のパラメータは全て無効となります（チェックも行いません）。

本関数で認証パラメータを設定していない場合は、各プログラム機能呼出し関数のpar->atrにRPC_ATR_AUTH(0x00000001)を設定しても、RPCヘッダ部の認証パラメータは認証なし（flavor=AUTH_ NULL(0)）で送信します。

本関数が正常終了すると、次のプログラム実行要求の送信データより、RPCヘッダ部の認証パラメータへ設定値を反映します。（各プログラム機能呼出し関数のpar->atrにRPC_ATR_AUTH(0x00000001)を設定した場合）

3.1.5 MOUNT プログラム機能呼出し用インターフェース関数

(1) RPC_MNT_Null **サーバのプログラム応答テスト**

【T】

C言語インタフェース

ER rtcd = RPC_MNT_Null(ID mntid, T_RPC_MNT_NULL *par, TMO tmout);

パラメータ		
ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_NULL	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造		
typedef struct {		
ATR	atr;	実行属性 （RPC_ATR_AUTH = 0x00000001）
T_RPC_RESULT	result;	実行結果*1
} T_RPC_MNT_NULL;		

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*2）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）	
OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバに対するMOUNTプログラムの応答テストを行います。

mntidで指定したRPCサーバに対し、MOUNTプログラム動作中かを確認する為のテストを行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

応答テストが正常に実行されると、正常終了としてpar->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が

発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(2)　RPC_MNT_Mnt　　ディレクトリのマウント

【T】

C言語インタフェース

```
ER rtcd = RPC_MNT_Mnt(ID mntid, T_RPC_MNT_MNT *par, TMO tmout);
```

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_MNT	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_FHANDLE	par->directory	ディレクトリハンドル

パケットの構造

```
typedef struct {
        ATR             atr;                    実行属性
                                                （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;                 実行結果*1
        T_RPC_DIRPATH   path;                   マウント対象ディレクトリ名
        T_RPC_FHANDLE   directory;              ディレクトリハンドル
} T_RPC_MNT_MNT;

typedef struct dirpath{
        W       dirlen;                         文字列のデータ長
        UB      dirdata[RPC_MAXPATHLEN];        文字列のデータ
                                                （RPC_MAXPATHLEN = 1024）
} T_RPC_DIRPATH;

typedef struct fhandle {
        UB      handle[RPC_FHSIZE];             ファイルハンドル（RPC_FHSIZE = 32）
} T_RPC_FHANDLE;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->path.dirlen≦0，par->path.dirlen＞RPC_MAXPATHLEN(1024)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*2）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）

OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリのマウントを行います。

mntidで指定したRPCサーバに対し、par->pathで指定したディレクトリのマウントを要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ディレクトリのマウントが正常に実行されると、正常終了としてpar->directoryにディレクトリハンドルを、par->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(3)　RPC_MNT_Dump　　マウントリストの取得

【T】

C言語インタフェース

```
ER rtcd = RPC_MNT_Dump(ID mntid, T_RPC_MNT_DUMP *par, TMO tmout);
```

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_DUMP	*par	パラメータテーブル
H	par->availcount	格納可能なマウントエントリリスト数
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_MOUNTLIST	par->list	マウントエントリリスト
H	par->recvcount	受信したマウントエントリリスト数
H	par->storecount	格納したマウントエントリリスト数

パケットの構造

```
typedef struct {
        ATR                     atr;            実行属性
                                                （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;         実行結果*1
        T_RPC_MOUNTLIST         *list;          マウントリストの先頭アドレス
        H                       availcount;     格納可能なマウントリストエントリ数
        H                       recvcount;      受信したマウントリストエントリ数
        H                       storecount;     格納したマウントリストエントリ数
} T_RPC_MNT_DUMP;


typedef struct mountlist {
        T_RPC_NAME              hostname;       ホスト名
        T_RPC_DIRPATH           directory;      ディレクトリ名*2
} T_RPC_MOUNTLIST;


typedef struct name {
        W       namelen;                        文字列のデータ長
        UB      namedata[RPC_MAXNAMLEN+1];      文字列のデータ
                                                （RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIRPATH：RPC_MNT_MntのT_RPC_DIRPATHを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par,par->listが0または4の倍数以外，tmout＝TMO_POL(0)， tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*3）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	処理のキャンセル，待ち状態の強制解除

E_CLS　　RPC サーバとの接続が切断された

*3 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）	
TRUE(1)	正常終了
FALSE(0)	エントリー無し
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

マウントリストの取得を行います。

mntidで指定したRPCサーバに対し、par->availcountで指定した分、マウントリスト取得の要求を行います。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

マウントリストの取得が正常に実行されると、正常終了としてpar->listにマウントリストを、par->countにリスト数を、par->result.statusにTRUE(1)を設定します。
マウントのエントリーが無く、取得が出来なかった場合は、par->result.statusにFALSE(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

UDPでオープンした通信では、RPCサーバが多くのマウントリストを持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->availcountで指定した分のpar->listの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->availcountで格納可能としたマウントリスト分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(4)　RPC_MNT_Umnt　　ディレクトリのアンマウント

【T】

C言語インタフェース

ER rtcd = RPC_MNT_Umnt(ID mntid, T_RPC_MNT_UMNT *par, TMO tmout);

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_UMNT	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性（RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_DIRPATH   path;       アンマウント対象ディレクトリ名*2
} T_RPC_MNT_UMNT;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIRPATH：RPC_MNT_MntのT_RPC_DIRPATHを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->path.dirlen≦0，par->path.dirlen＞RPC_MAXPATHLEN(1024)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*3）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）

OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリのアンマウントを行います。

mntidで指定したRPCサーバに対し、par->pathで指定したディレクトリのアンマウントを要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ディレクトリのアンマウントが正常に実行されると、正常終了としてpar->result.statusに0を設定します。
本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル

（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(5)　RPC_MNT_UmntAll　　全ディレクトリのアンマウント

【T】

C言語インタフェース

ER rtcd = RPC_MNT_UmntAll(ID mntid, T_RPC_MNT_UMNTALL *par, TMO tmout);

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_UMNTALL	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
} T_RPC_MNT_UMNTALL;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*2）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）

OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

マウント中の全てのディレクトリをアンマウントします。

mntidで指定したRPCサーバに対し、全ディレクトリのアンマウントを要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ディレクトリのアンマウントが正常に実行されると、正常終了としてpar->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が

発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 実行結果 ：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(6)　RPC_MNT_Export　　**エクスポートリストの取得**

【T】

C言語インタフェース

```
ER rtcd = RPC_MNT_Export(ID mntid, T_RPC_MNT_EXPORT *par, TMO tmout);
```

パラメータ

ID	mntid	MOUNTプログラム通信用ID
T_RPC_MNT_EXPORT	*par	パラメータテーブル
H	par->availcount	格納可能なエクスポートリスト数
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
H	par->recvcount	受信したエクスポートリスト数
H	par->storecount	格納したエクスポートリスト数
T_RPC_EXPORTLIST	par->list	エクスポートリスト

パケットの構造

```
typedef struct {
        ATR                     atr;            実行属性
                                                （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;         実行結果*1
        T_RPC_EXPORTLIST        *list;          エクスポートリストの先頭アドレス
        H                       availcount;     格納可能なエクスポートリスト数
        H                       recvcount;      受信したエクスポートリスト数
        H                       storecount;     格納したエクスポートリスト数
} T_RPC_MNT_EXPORT;

typedef struct exportlist {
        T_RPC_DIRPATH   filesys;                ファイルシステム*2
        W               grcount;                グループリスト数
        T_RPC_GROUPS    grlist[MAXGRPLIST];     グループリスト（MAXGRPLIST = 16）
} T_RPC_EXPORTLIST;

typedef struct groups {
        T_RPC_NAME      grname;                 ネットワークグループ情報*3
} T_RPC_GROUPS;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIRPATH：RPC_MNT_MntのT_RPC_DIRPATHを参照
*3 T_RPC_NAME：RPC_MNT_DumpのT_RPC_NAMEを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞mnt_maxid*4）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*4 mnt_maxid：RPC_Startで指定した最大MOUNTプログラム通信用ID

実行結果（par->result.status）

OK(0)　　正常終了
その他の値　　『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

エクスポートリストを取得します。
エクスポートリストとは、マウント可能なファイルシステム（ディレクトリ）名称、およびアクセスを許可しているネットワークグループのリストのことを指します。

mntidで指定したRPCサーバに対し、par->availcountに指定した分、エクスポートリスト取得の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*5をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

エクスポートリストの取得が正常に実行されると、正常終了としてpar->storecountに格納したリスト数を、par->recvcountに受信したリスト数を、par->listにリストの内容を、par->result.statusに0を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_MNT_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

UDPでオープンした通信では、RPCサーバが多くのエクスポートリストを持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->availcountで指定した分のpar->listの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->availcountで格納可能としたエクスポートリスト分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*5 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

3.1.6 NFSv2 通信制御用インターフェース関数

(1) RPC_NFS_Open RPC サーバとの NFSv2 通信を開始する

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Open(T_RPC_OPEN *par, TMO tmout);

パラメータ

T_RPC_OPEN	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
ID	par->returnid	NFSv2プログラム通信用ID

パケットの構造

```
typedef struct {
        T_IPV4EP        myaddr;         クライアントアドレス
        T_IPV4EP        svaddr;         RPCサーバアドレス
        H               protocol;       NFSv2通信用プロトコル
                                        （IPPROTO_TCP = 6, IPPROTO_UDP = 17）
        ID              reqcepid;       TCP/IPマネージャ通信端点ID
        ID              returnid;       RPCサーバプログラム通信用ID
} T_RPC_OPEN;

typedef struct {
        UW              ipaddr;         IPアドレス
        UH              portno;         ポート番号
} T_IPV4EP;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、par->myaddrが動作を開始しているアドレスと異なる， par->svaddr.ipaddrが0または0xffffffff（TCPの場合）， par->svaddr.ipaddrが0（UDPの場合）， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_OBJ	オブジェクト状態エラー （ポート番号既使用，指定したIDのTCPまたはUDP通信端点が使用中）
E_ID	不正ID番号（reqcepid＜0, reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大）
E_NOID	ID番号不足（NFSv2プログラム通信用IDに空きが無い）
E_TMOUT	タイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_CLS	接続要求が拒否された（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

解　説

RPCサーバとのNFSv2通信を開始します。

par->svaddrで指定したRPCサーバに対して、par->protocolで指定したトランスポートプロトコルを使用し通信開始の設定を行います。

その際、par->svaddr.portno（トランスポートのポート番号）には、Portmapプログラム通信等で取得したNFSv2用のポート番号を指定してください。

par->reqcepidには、TCP/IPマネージャで使用する通信端点IDを指定します。特にIDを指定する必要がない場合は、par->reqcepidに0を指定することで空いている通信端点を使用します。

トランスポートプロトコルにTCPを指定して本関数を実行する場合は、RPCサーバに対してTCPの接続までを行います。またトランスポートプロトコルにUDPを指定した場合は、およびRPCサーバとの通信準備（システムテーブル設定等）までを行います。

正常に完了した場合、par->returnidにNFSv2プログラム通信用IDを設定し、リターン値としてE_OKを返します。

(2)　RPC_NFS_Close　　RPC サーバとの NFSv2 通信を終了する

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Close(ID nfsid, TMO tmout);

パラメータ		
ID	nfsid	NFSv2プログラム通信用ID
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー（tmout<TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*1）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	ポーリング失敗またはタイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

*1 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

解　説

RPCサーバとのNFSv2通信を終了します。

nfsidで指定したNFSv2通信がトランスポートプロトコルがTCPの場合は、TCPの接続を切断し、TCP資源の解放を行います。また、トランスポートプロトコルがUDPの場合は、UDP資源の解放を行います。

正常に完了した場合、リターン値としてE_OKを返します。

(3)　RPC_NFS_Cancel　　NFSv2 プログラム応答待ちをキャンセルする

【T／D】

C言語インタフェース

ER rtcd = RPC_NFS_Cancel(ID nfsid);

パラメータ		
ID	nfsid	NFSv2プログラム通信用ID

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*1）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_ILUSE	不正使用（RPCクライアントが停止している）

*1 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

解　説

待ちをキャンセルします。

nfsidで示されたRPCサーバ上のNFSv2プログラムからの応答待ち処理を解除します。

本関数で待ち解除された関数には、E_RLWAIを返します。

(4)　RPC_NFS_SetAuth　　NFSv2 プログラム用の RPC 認証パラメータを設定する

【T／D】

C言語インタフェース

ER rtcd = RPC_NFS_SetAuth(ID nfsid, T_RPC_AUTH *par);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_AUTH	*par	パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        UW              flavor;                 認証方式
                                                  （RPC_AUTH_NULL = 0,
                                                   RPC_AUTH_UNIX = 1）
        T_RPC_NAME      machine;                RPCクライアント名
        UW              uid;                    ユーザーID
        UW              gid;                    グループID
        UW              auxgidcnt;              予備グループID数
        UW              auxgid[RPC_MAXAUXGID];  予備グループID
                                                  （RPC_MAXAUXGID = 16）
} T_RPC_AUTH;

typedef struct name {
        W       namelen;                        データ長
        UB      namedata[RPC_MAXNAMLEN+1];      データ（RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，par->flavor＜RPC_AUTH_NULL(0), par->flavor＞RPC_AUTH_UNIX(1), par->machine.namelen＜0, par->machine.namelen＞RPC_MAXNAMLEN(255), par->auxgidcnt＞RPC_MAXAUXGID(16)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*1）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）

*1 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

解　説

mntidで指定したMOUNTプログラム通信用IDにRPC認証パラメータを設定します。
mntidが有効なIDである間（RPC_MNT_OpenからRPC_MNT_Closeの間）のみ設定可能です。

par->flavorをRPC_AUTH_NULL(0)に指定して本関数を呼出した場合、以降のパラメータは全て無効となります（チェックも行いません）。

本関数で認証パラメータを設定していない場合は、各プログラム機能呼出し関数のpar->atrにRPC_ATR_AUTH(0x00000001)を設定しても、RPCヘッダ部の認証パラメータは認証なし（flavor=AUTH_ NULL(0)）で送信します。

本関数が正常終了すると、次のプログラム実行要求の送信データより、RPCヘッダ部の認証パラメータへ設定値を反映します。（各プログラム機能呼出し関数のpar->atrにRPC_ATR_AUTH(0x00000001)を設定した場合）

3.1.7　NFSv2 プログラム機能呼出し用インターフェース関数

(1)　RPC_NFS_Null　　**サーバのプログラム応答テスト**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Null(ID nfsid, T_RPC_NFS_NULL *par, TMO tmout);

パラメータ		
ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_NULL	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果*1

パケットの構造		
typedef struct {		
ATR	atr;	実行属性（RPC_ATR_AUTH = 0x00000001）
T_RPC_RESULT	result;	実行結果*1
} T_RPC_NFS_NULL;		

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*2）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）	
NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバに対するNFSv2プログラムの応答テストを行います。

nfsidで指定したRPCサーバに対し、NFSv2プログラム動作中かを確認する為のテストを行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

応答テストが正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を返します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が

発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(2)　RPC_NFS_GetAttr　　属性の取得

【T】

C言語インタフェース

ER rtcd = RPC_NFS_GetAttr(ID nfsid, T_RPC_NFS_GETATTR *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_GETATTR	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_FATTR	par->attributes	属性情報

パケットの構造

```
typedef struct {
        ATR             atr;            実行属性（RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;         実行結果*1
        T_RPC_FHANDLE   file;           ファイルハンドル（またはディレクトリハンドル）*2
        T_RPC_FATTR     attributes;     属性情報
} T_RPC_NFS_GETATTR;

typedef struct fattr {
        UW              type;           種別（RPC_NFNON = 0［非ファイル］,
                                                RPC_NFREG = 1［普通のファイル］,
                                                RPC_NFDIR = 2［ディレクトリ］,
                                                RPC_NFBLK = 3[特殊なブロックデバイス],
                                                RPC_NFCHR = 4
                                                        ［特殊なキャラクタデバイス],
                                                RPC_NFLNK = 5［シンボリックリンク］）
        UW              mode;           モード
        UW              nlink;          リンク
        UW              uid;            ユーザID
        UW              gid;            グループID
        UW              size;           サイズ
        UW              blocksize;      ブロックサイズ
        UW              rdev;           特殊デバイス
        UW              blocks;         ブロック数
        UW              fsid;           ファイルシステムID
        UW              fileid;         ファイルID
        T_RPC_TIMEVAL   atime;          アクセス時刻
        T_RPC_TIMEVAL   mtime;          更新時刻
        T_RPC_TIMEVAL   ctime;          作成時刻
} T_RPC_FATTR;


typedef struct timeval {
        UW              seconds;        秒（1970年1月1日を基準とした相対時間）
        UW              useconds;       マイクロ秒
} T_RPC_TIMEVAL;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0), tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）	
NFS_OK(0)	正常終了
NFSERR_STALE(70)	ファイルハンドル（またはディレクトリハンドル）異常
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルやディレクトリの属性を取得します。

nfsidで指定したRPCサーバに対し、par->fileで指定したファイルハンドル（またはディレクトリハンドル）についての属性取得を要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

属性の取得が正常に実行されると、正常終了としてpar->attributesに属性情報を、par->result.statusにNFS_OK(0)を設定します。
RPCサーバ上にpar->fileのハンドルが存在しないなどの場合、ハンドル異常としてpar->result.statusにNFSERR_STALE(70)を返します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(3)　RPC_NFS_SetAttr　　属性の設定

【T】

C言語インタフェース

```
ER rtcd = RPC_NFS_SetAttr(ID nfsid, T_RPC_NFS_SETATTR *par, TMO tmout);
```

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_SETATTR	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_SATTRRES	par->res	出力パラメータ

パケットの構造

```
typedef struct {
        ATR                     atr;            実行属性
                                                （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;         実行結果*1
        T_RPC_SATTRARGS         args;           入力パラメータ
        T_RPC_SATTRRES          res;            出力パラメータ
} T_RPC_NFS_SETATTR;

typedef struct sattrargs {
        T_RPC_FHANDLE           file;           ファイルハンドル*2
                                                （またはディレクトリハンドル）
        T_RPC_SATTR             attributes;     属性情報
} T_RPC_SATTRARGS;

typedef struct sattrres {
        T_RPC_FATTR             attributes;     属性情報*3
} T_RPC_SATTRRES;


typedef struct sattr {
        UW                      mode;           モード
        UW                      uid;            ユーザID
        UW                      gid;            グループID
        UW                      size;           サイズ
        T_RPC_TIMEVAL           atime;          アクセス時刻*4
        T_RPC_TIMEVAL           mtime;          更新時刻*3
} T_RPC_SATTR;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_FATTR：RPC_NFS_GetAttrのT_RPC_FATTRを参照
*4 T_RPC_TIMEVAL：RPC_NFS_GetAttrのT_RPC_TIMEVALを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*5）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）

E_TMOUT	**タイムアウト**
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*5 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）	
NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルやディレクトリの属性を設定します。

nfsidで指定したRPCサーバに対し、par->fileで指定したファイルハンドル（またはディレクトリハンドル）について、par->argsの内容での属性設定を要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

属性の設定が正常に実行されると、正常終了としてpar->res.attributesに属性情報を、par->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(4)　RPC_NFS_Root　　**ファイルシステム・ルートの取得**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Root(ID nfsid, T_RPC_NFS_ROOT *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_ROOT	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性（RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
} T_RPC_NFS_ROOT;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*2）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルシステムのルートを取得します。*3

nfsidで指定したRPCサーバに対し、ファイルシステムのルート取得を要求します。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

応答が正常に返ると、正常終了としてpar->result.statusに0を返します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 RPC_NFS_Root：RFC1094では廃止されています。（MOUNTプログラムのMNT機能が同義の為）
*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(5)　RPC_NFS_LookUp　　ファイルの検索

【T】

C言語インタフェース

ER rtcd = RPC_NFS_LookUp(ID nfsid, T_RPC_NFS_LOOKUP *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_LOOKUP	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_DIROPRES	par->res	出力パラメータ

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_DIROPARGS         args;       入力パラメータ
        T_RPC_DIROPRES          res;        出力パラメータ
} T_RPC_NFS_LOOKUP;

typedef struct diropargs {
        T_RPC_FHANDLE           dir;        ディレクトリハンドル*2
        T_RPC_NAME              name;       ファイル名*3
} T_RPC_DIROPARGS;

typedef struct diropres {
        T_RPC_FHANDLE           file;       ファイルハンドル*4
        T_RPC_FATTR             attributes; 属性情報*3
} T_RPC_DIROPRES;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_NAME：RPC_MNT_DumpのT_RPC_NAMEを参照
*4 T_RPC_FATTR：RPC_NFS_GetAttrのT_RPC_FATTRを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->args.name.namelen≦0， par->args.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*5）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*5 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
NFSERR_NOENT(2)	ファイルが存在しない

NFSERR_STALE(70)	ファイルハンドル（またはディレクトリハンドル）異常
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリにファイルがあるかを検索します。

nfsidで指定したRPCサーバに対し、par->args.dirで指定したハンドルのディレクトリにpar->args.nameで指定したファイル名が存在するかの問合せを行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ファイルの問合せが正常に実行され、指定したファイル名が存在する場合は、正常終了としてpar->res.fileにファイル（またはディレクトリ）のハンドルを、par->res.attributesに属性情報を、par->result.statusにNFS_OK(0)を設定します。
問合せは実行されたものの、指定したファイル名が存在しない場合は、ファイルが存在しないとしてpar->result.statusにNFSERR_NOENT(2)を返します。
RPCサーバ上にpar->args.dirのハンドルが存在しないなどの場合、ハンドル異常としてpar->result.statusにNFSERR_STALE(70)を返します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(6)　RPC_NFS_ReadLink　　**シンボリックリンクの取得**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_ReadLink(ID nfsid, T_RPC_NFS_READLINK *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_READLINK	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_DIRPATH	data	パス情報

パケットの構造

```
typedef struct {
        ATR                 atr;        実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT        result;     実行結果*1
        T_RPC_FHANDLE       link;       シンボリックリンクハンドル*2
        T_RPC_DIRPATH       data;       パス情報
} T_RPC_NFS_READLINK;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0), tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
NFSERR_INVAL(22)	シンボリックリンクのハンドルではない
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

シンボリックリンクのリンクのパス情報を読出します。

nfsidで指定したRPCサーバに対し、par->linkに指定したシンボリックリンクハンドルについて、リンクのパス情報取得要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとして

E_TMOUTを返します。

　情報の読出しが正常に実行されると、正常終了としてpar->dataにリンク元パス情報を、par->result.statusにNFS_OK(0)を設定します。
　par->linkに指定したハンドルがシンボリックリンクのハンドルでない場合は、ハンドル異常としてpar->result.statusにNFS_INVAL(22)を設定します。

　本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(7)　RPC_NFS_Read　　ファイルの読出し

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Read(ID nfsid, T_RPC_NFS_READ *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_READ	*par	パラメータテーブル
TMO	tmout	タイムアウト指定
W	par->availlen;	格納可能なデータ長

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
W	par->recvlen	受信したデータ長
W	par->storelen	格納したデータ長
UB	*par->data	読出しデータ

パケットの構造

```
typedef struct {
        ATR             atr;            実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;         実行結果*1
        T_RPC_FHANDLE   file;           ファイルハンドル*2
        UB              *data;          読出しデータ格納先頭アドレス
        UW              offset;         ファイルオフセット
        W               availlen;       格納可能なデータ長
        W               recvlen;        受信したデータ長
        W               storelen;       格納したデータ長
        T_RPC_FATTR     attributes;     属性情報*3
} T_RPC_NFS_READ;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_FATTR：RPC_NFS_GetAttrのT_RPC_FATTRを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par,par->dataが0または4の倍数以外， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*4）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*4 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルからデータを読出します。

nfsidで指定したRPCサーバに対し、parで指定したファイルハンドル、オフセット、および読出しカウントに従いデータ読出しの要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*5をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

データの読出しが正常に実行されると、正常終了としてpar->dataに読出しデータおよびデータ長を、par->attributesに読出し時のファイル属性を、par->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

UDPでオープンした通信では、（RPCヘッダと合わせて）par->availlenに1472バイトを超える大きいデータ長を指定した場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->dataの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->availlenで格納可能としたデータ長分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。受信できる範囲で本関数の呼出しを実行してください。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上でpar->availlenを設定するようお願いします。

*5 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(8)　RPC_NFS_WriteCache　　キャッシュの更新

【T】

C言語インタフェース

ER rtcd = RPC_NFS_WriteCache(ID nfsid, T_RPC_NFS_WRITECACHE *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_WRITECACHE	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
} T_RPC_NFS_WRITECACHE;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*2）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*2 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

RPCサーバのキャッシュ更新を行います。*3

nfsidで指定したRPCサーバに対し、キャッシュの更新要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

応答が正常に返ると、正常終了としてpar->result.statusに0を返します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル

（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*3 nfs_maxid：RFC1094では次期プロトコルリビジョンで使用するための定義となっています。
*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(9)　RPC_NFS_Write　　**ファイルの書込み**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Write(ID nfsid, T_RPC_NFS_WRITE *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_WRITE	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_FATTR	par->attributes	属性情報

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_FHANDLE   file;       ファイルハンドル*2
        UB              *data;      書込みデータ格納先頭アドレス
        UW              offset;     ファイルオフセット
        W               datalen;    書込みデータ長
        T_RPC_FATTR     attributes; 属性情報*3
} T_RPC_NFS_WRITE;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_FATTR：RPC_NFS_GetAttrのfattrを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*5）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*5 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルへデータを書込みます。

nfsidで指定したRPCサーバに対し、par->argsで指定したファイルハンドル、オフセット、書込みカウント、および書込みデータに従いデータ書込みの要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返し

ます。
　tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

　データの書込みが正常に実行されると、正常終了としてpar->attributesに書込み後のファイル属性を、par->result.statusにNFS_OK(0)を設定します。

　本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(10)　RPC_NFS_Create　　ファイルの作成

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Create(ID nfsid, T_RPC_NFS_CREATE *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_CREATE	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_DIROPRES	par->res	出力パラメータ

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_CREATEARGS        args;       入力パラメータ
        T_RPC_DIROPRES          res;        出力パラメータ*2
} T_RPC_NFS_CREATE;

typedef struct createargs {
        T_RPC_DIROPARGS         where;      作成位置情報（ディレクトリ）*3
        T_RPC_SATTR             attributes  属性情報*4
} T_RPC_CREATEARGS;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIROPRES：RPC_NFS_LookUpのT_RPC_DIROPRESを参照
*3 T_RPC_DIROPARGS：RPC_NFS_LookUpのT_RPC_DIROPARGSを参照
*4 T_RPC_SATTR：RPC_NFS_SetAttrのT_RPC_SATTRを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->args.where.name.namelen≦0, par->args.where.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*5）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*5 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルの新規作成を行います。

nfsidで指定したRPCサーバに対し、par->argsで指定したファイル名称、およびファイル作成位置情報

に従いファイル作成の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ファイルの作成が正常に実行されると、正常終了としてpar->res.fileにファイル（またはディレクトリ）のハンドルを、par->res.attributesに属性情報を、par->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(11)　RPC_NFS_Remove　　**ファイルの削除**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Remove(ID nfsid, T_RPC_NFS_REMOVE *par, TMO tmout);

パラメータ		
ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_REMOVE	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造		
typedef struct {		
ATR	atr;	実行属性 （RPC_ATR_AUTH = 0x00000001）
T_RPC_RESULT	result;	実行結果*1
T_RPC_DIROPARGS	args;	入力パラメータ*2
} T_RPC_NFS_REMOVE;		

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIROPARGS：RPC_NFS_LookUpのT_RPC_DIROPARGSを参照

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->args.name.namelen≦0， par->args.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）	
NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルの削除を行います。

nfsidで指定したRPCサーバに対し、par->args.dirで指定したハンドルのディレクトリ上でのpar->args.nameで指定した名前を持つファイルの削除要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ファイルの削除が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(12)　RPC_NFS_Rename　　**ファイル・ディレクトリの名称変更**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Rename(ID nfsid, T_RPC_NFS_RENAME *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_RENAME	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_RENAMEARGS        args;       入力パラメータ
} T_RPC_NFS_RENAME;

typedef struct renameargs {
        T_RPC_DIROPARGS         from;       変更前ファイル名*2
        T_RPC_DIROPARGS         to;         変更後ファイル名*2
} T_RPC_RENAMEARGS;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIROPARGS：RPC_NFS_LookUpのT_RPC_DIROPARGSを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1), par->args.from.name.namelen≦0, par->args.from.name.namelen＞RPC_MAXNAMLEN(255)， par->args.to.name.namelen≦0, par->args.to.name.namelen＞RPC_MAXNAMLEN(255))
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイル、およびディレクトリ名称の変更を行います。

nfsidで指定したRPCサーバに対し、par->args.from，par->args.toのファイル名情報に従いファイル名変更の要求を行います。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ファイル名の変更が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(13)　RPC_NFS_Link　　**ハードリンクの作成**

【T】

C言語インタフェース

ER rtcd = RPC_NFS_Link(ID nfsid, T_RPC_NFS_LINK *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_LINK	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR             atr;        実行属性
                                    （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;     実行結果*1
        T_RPC_LINKARGS  args;       入力パラメータ
} T_RPC_NFS_LINK;

typedef struct linkargs {
        T_RPC_FHANDLE   from;       リンク作成元ハンドル*2
        T_RPC_DIROPARGS to;         リンク作成先ファイル名*3
} T_RPC_LINKARGS;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_DIROPARGS：RPC_NFS_LookUpのT_RPC_DIROPARGSを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1), par->args.to.name.namelen≦0, par->args.to.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*4）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*4 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ハードリンクの作成を行います。

nfsidで指定したRPCサーバに対し、par->args.from，par->args.toのリンク作成情報に従い、ハードリンク作成の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*5をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ハードリンクの作成が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*5 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

.

(14)　RPC_NFS_SymLink　　シンボリックリンクの作成

【T】

C言語インタフェース

ER rtcd = RPC_NFS_SymLink(ID nfsid, T_RPC_NFS_SYMLINK *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_SYMLINK	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_SYMLINKARGS       args;       入力パラメータ
} T_RPC_NFS_SYMLINK;

typedef struct {
        T_RPC_DIROPARGS         from;       作成するリンクファイル名*2
        T_RPC_DIRPATH           to;         リンク作成元パス*3
        T_RPC_SATTR             attributes; 属性情報*4
} T_RPC_SYMLINKARGS;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIROPARGS: RPC_NFS_LookUpのT_RPC_DIROPARGSを参照
*3 T_RPC_DIRPATH：RPC_NFS_ReadLinkのT_RPC_DIRPATHを参照
*4 T_RPC_SATTR：RPC_NFS_SetAttrのT_RPC_SATTRを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1), par->args.from.name.namelen≦0, par->args.from.name.namelen＞RPC_MAXNAMLEN(255), par->args.to.dirlen≦0，par->args.to.dirlen＞RPC_MAXPATHLEN(1024)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*5）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*5 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

シンボリックリンクの作成を行います。

nfsidで指定したRPCサーバに対し、par->args.from，par->args.to，par->args.attributesのリンク作成

情報に従い、シンボリックリンク作成の要求を行います。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

シンボリックリンクの作成が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(15)　RPC_NFS_MkDir　　ディレクトリの作成

【T】

C言語インタフェース

ER rtcd = RPC_NFS_MkDir(ID nfsid, T_RPC_NFS_MKDIR *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_MKDIR	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_DIROPRES	par->res	出力パラメータ

パケットの構造

```
typedef struct {
        ATR                     atr;        実行属性
                                            （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT            result;     実行結果*1
        T_RPC_CREATEARGS        args;       入力パラメータ*2
        T_RPC_DIROPRES          res;        出力パラメータ*3
} T_RPC_NFS_MKDIR;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_CREATEARGS：RPC_NFS_CreateのT_RPC_CREATEARGSを参照
*3 T_RPC_DIROPRES：RPC_NFS_LookUpのT_RPC_DIROPRESを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->args.where.name.namelen≦0， par->args.where.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*4）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*4 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリの作成を行います。

nfsidで指定したRPCサーバに対し、par->argsで指定したディレクトリハンドル、およびサブディレクトリ名に従いサブディレクトリ作成の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*5をpar->resultに設定し、リターン値としてE_OKを返します。

tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。
ディレクトリの作成が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*5 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(16)　RPC_NFS_RmDir　　ディレクトリの削除

【T】

C言語インタフェース

ER rtcd = RPC_NFS_RmDir(ID nfsid, T_RPC_NFS_RMDIR *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_RMDIR	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果

パケットの構造

```
typedef struct {
        ATR                 atr;        実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT        result;     実行結果*1
        T_RPC_DIROPARGS     args;       入力パラメータ*2
} T_RPC_NFS_RMDIR;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_DIROPARGS：RPC_NFS_LookUpのT_RPC_DIROPARGSを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)， par->args.name.namelen≦0, par->args.name.namelen＞RPC_MAXNAMLEN(255)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリの削除を行います。

nfsidで指定したRPCサーバに対し、par->argsのディレクトリ名情報に従い、ディレクトリ削除の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ディレクトリの削除が正常に実行されると、正常終了としてpar->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(17)　RPC_NFS_ReadDir　　ディレクトリリストの取得

【T】

C言語インタフェース

ER rtcd = RPC_NFS_ReadDir(ID nfsid, T_RPC_NFS_READDIR *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_READDIR	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_ENTRY	par->entries	ディレクトリリスト
H	par->recvcount	受信したディレクトリリストの数
H	par->storecount	格納したディレクトリリストの数

パケットの構造

```
typedef struct {
        ATR             atr;            実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT    result;         実行結果*1
        T_RPC_FHANDLE   dir;            ディレクトリハンドル*2
        UW              cookie;         開始クッキー番号
        T_RPC_ENTRY     *entries;       リストを格納する先頭アドレス
        H               availcount;     格納可能なディレクトリリストの長さ
        H               recvcount;      受信したディレクトリリストの数
        H               storecount;     格納したディレクトリリストの数
        BOOL            eof;            終端フラグ
} T_RPC_NFS_READDIR;

typedef struct entry {
        UW              fileid;         ファイルID
        UW              cookie;         クッキー番号
        T_RPC_NAME      name;           ファイル名*3
} T_RPC_ENTRY;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照
*3 T_RPC_NAME：RPC_NFS_LookUpのT_RPC_NAMEを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par,par->entriesが0または4の倍数以外， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*4）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*4 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)　　正常終了
その他の値　　『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ディレクトリリストの取得を行います。

nfsidで指定したRPCサーバに対し、par->argsで指定したディレクトリハンドル、クッキー情報*5に従い、ディレクトリリスト取得の要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*6をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ディレクトリリストの取得が正常に実行されると、par->resにはディレクトリリストを、par->result.statusにはNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_NFS_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

UDPでオープンした通信では、RPCサーバが多くのディレクトリリストを持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->availcountで指定した分のpar->entriesの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->availcountで格納可能としたディレクトリリスト分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*5 クッキー情報：同一ディレクトリ内でのファイルの位置情報。この位置情報は連番になっており、一度のRPC_NFS_ReadDir実行でディレクトリリストが全て取れない場合でも、クッキー情報を指定することで指定したクッキー情報を持つファイルから先のリストを取得することが出来ます。
*6 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

(18)　RPC_NFS_StatFs　　ファイルシステム状態の取得

【T】

C言語インタフェース

ER rtcd = RPC_NFS_StatFs(ID nfsid, T_RPC_NFS_STATFS *par, TMO tmout);

パラメータ

ID	nfsid	NFSv2プログラム通信用ID
T_RPC_NFS_STATFS	*par	パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
T_RPC_STATFSRES	par->res	出力パラメータ

パケットの構造

```
typedef struct {
        ATR                 atr;        実行属性
                                        （RPC_ATR_AUTH = 0x00000001）
        T_RPC_RESULT        result;     実行結果*1
        T_RPC_FHANDLE       filesys;    ファイルシステム*2
        T_RPC_STATFSRES     res;        出力パラメータ
} T_RPC_NFS_STATFS;

typedef struct statfsres {
        UW                  tsize;      サーバの送信サイズ
        UW                  bsize;      ブロックサイズ
        UW                  blocks;     ブロック数
        UW                  bfree;      空きブロック数
        UW                  bavail      使用可能なブロック数
} T_RPC_STATFSRES;
```

*1 T_RPC_RESULT：RPC_PMAP_NullのT_RPC_RESULTを参照
*2 T_RPC_FHANDLE：RPC_MNT_MntのT_RPC_FHANDLEを参照

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（nfsid≦0，nfsid＞nfs_maxid*3）
E_NOEXS	オブジェクト未生成（nfsidが存在していない）
E_OBJ	オブジェクト状態エラー（nfsidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*3 nfs_maxid：RPC_Startで指定した最大NFSv2プログラム通信用ID

実行結果（par->result.status）

NFS_OK(0)	正常終了
その他の値	『5.1 RPCクライアント機能実行結果一覧』を参照

解　説

ファイルシステムの状態を取得します。

nfsidで指定したRPCサーバに対し、par->filesysで指定したハンドルが所属しているファイルシステムの状態を取得します。

tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*4をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。

ファイルシステム状態の取得が正常に実行されると、正常終了としてpar->resにファイルシステムの状態を、par->result.statusにNFS_OK(0)を設定します。

本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_PMAP_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行された場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

*4 実行結果：詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

3.1.8　直接 RPC 通信用インターフェース関数

(1)　RPC_Open　　RPC サーバとの RPC 通信を開始する

【T】

C言語インタフェース

ER rtcd = RPC_Open(T_RPC_OPEN *par, UW prog, UW vers, TMO tmout);

パラメータ

T_RPC_OPEN	*par	パラメータテーブル
UW	porg	RPCサーバで動作中のプログラム番号
UW	vers	RPCサーバで動作中のプログラムバージョン
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
ID	par->returnid	指定RPCプログラム通信用ID

パケットの構造

```
typedef struct {
        T_IPV4EP        myaddr;         クライアントアドレス
        T_IPV4EP        svaddr;         RPCサーバアドレス
        H               protocol;       RPC通信用プロトコル
                                        （IPPROTO_TCP = 6, IPPROTO_UDP = 17）
        ID              reqcepid;       TCP/IPマネージャ通信端点ID
        ID              returnid;       RPCサーバプログラム通信用ID
} T_RPC_OPEN;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外、myaddrが動作を開始しているアドレスと異なる， svaddr.ipaddrが0または0xffffffff（TCPの場合），svaddr.ipaddrが0（UDPの場合）， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_OBJ	オブジェクト状態エラー （ポート番号既使用，指定したIDのTCPまたはUDP通信端点が使用中）
E_ID	不正ID番号（reqcepid＜0, reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大）
E_NOID	ID番号不足（指定RPCプログラム通信用IDに空きが無い）
E_TMOUT	タイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_CLS	接続要求が拒否された（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

解　説

RPCサーバとのRPC通信を開始します。

par->svaddrで指定したRPCサーバに対して、par->protocolで指定したトランスポートプロトコルを使用し通信開始の設定を行います。

その際、

- par->svaddr.portno（トランスポートのポート番号）には、Portmapプログラム通信等で取得したRPCプログラム用のポート番号を指定してください。
- porg（動作中のプログラム番号）には、RPCサーバで実行中のプログラム番号を指定してください。
- vers（動作中のプログラムバージョン）には、RPCサーバで実行中のプログラムバージョンを指定してください。

par->reqcepidには、TCP/IPマネージャで使用する通信端点IDを指定します。特にIDを指定する必要がない場合は、par->reqcepidに0を指定することで空いている通信端点を使用します。

トランスポートプロトコルにTCPを指定して本関数を実行する場合は、RPCサーバに対してTCPの接続までを行います。またトランスポートプロトコルにUDPを指定した場合は、およびRPCサーバとの通信準備（システムテーブル設定等）までを行います。

正常に完了した場合、par->returnidに指定RPCプログラム通信用IDを設定し、リターン値としてE_OKを返します。

(2)　RPC_Close　　RPC サーバとの RPC 通信を終了する

【T】

C言語インタフェース

ER rtcd = RPC_Close(ID rpcid, TMO tmout);

パラメータ		
ID	rpcid	指定RPCプログラム通信用ID
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー（tmout<TMO_FEVR(-1)）
E_ID	不正ID番号（rpcid≦0，rpcid＞rpc_maxid*1）
E_NOEXS	オブジェクト未生成（rpcidが存在していない）
E_OBJ	オブジェクト状態エラー（rpcidは既に実行中）
E_TMOUT	ポーリング失敗またはタイムアウト（TCP）
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**（TCP）
E_ILUSE	不正使用（TCP/IPマネージャまたはRPCクライアントが停止している）

*1 rpc_maxid：RPC_Startで指定した最大指定RPCプログラム通信用ID

解　説

RPCサーバとのRPC通信を終了します。

rpcidで指定したRPC通信がトランスポートプロトコルがTCPの場合は、TCPの接続を切断し、TCP資源の解放を行います。また、トランスポートプロトコルがUDPの場合は、UDP資源の解放を行います。

正常に完了した場合、リターン値としてE_OKを返します。

(3) RPC_Cancel 指定 RPC プログラム応答待ちをキャンセルする

【T／D】

C言語インタフェース

ER rtcd = RPC_Cancel(ID rpcid);

パラメータ

ID	rpcid	指定RPCプログラム通信用ID

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ID	不正ID番号（rpcid≦0，rpcid＞rpc_maxid*1)
E_NOEXS	オブジェクト未生成（rpcidが存在していない）
E_ILUSE	不正使用（RPCクライアントが停止している）

*1 rpc_maxid：RPC_Startで指定した最大指定RPCプログラム通信用ID

解 説

待ちをキャンセルします。

rpcidで示されたRPCサーバ上の指定プログラムからの応答待ち処理を解除します。

本関数で待ち解除された関数には、E_RLWAIを返します。

(4)　RPC_SetAuth　　**指定 RPC 通信用の RPC 認証パラメータを設定する**

【T／D】

C言語インタフェース

```
ER rtcd = RPC_SetAuth(ID rpcid, T_RPC_AUTH *par);
```

パラメータ

ID	rpcid	指定RPCプログラム通信用ID
T_RPC_AUTH	*par	RPC認証パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        UW              flavor;                 認証方式
                                                （RPC_AUTH_NULL = 0,
                                                 RPC_AUTH_UNIX = 1）
        T_RPC_NAME      machine;                RPCクライアント名
        UW              uid;                    ユーザーID
        UW              gid;                    グループID
        UW              auxgidcnt;              予備グループID数
        UW              auxgid[RPC_MAXAUXGID];  予備グループID
                                                （RPC_MAXAUXGID = 16）
} T_RPC_AUTH;

typedef struct name {
        W       namelen;                        データ長
        UB      namedata[RPC_MAXNAMLEN+1];      データ（RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （parが0または4の倍数以外，par->flavor＜RPC_AUTH_NULL(0), par->flavor＞RPC_AUTH_UNIX(1), par->machine.namelen＜0, par->machine.namelen＞RPC_MAXNAMLEN, par->auxgidcnt＞RPC_MAXAUXGID）
E_ID	不正ID番号（rpcid≦0，rpcid＞rpc_maxid*1）
E_NOEXS	オブジェクト未生成（rpcidが存在していない）
E_OBJ	オブジェクト状態エラー（rpcidは既に実行中）

*1 rpc_maxid：RPC_Startで指定した最大指定RPCプログラム通信用ID

解　説

rpcidで指定したRPC通信用IDにRPC認証パラメータを設定します。
rpcidが有効なIDである間（RPC_OpenからRPC_Closeの間）のみ設定可能です。

par->flavorをRPC_AUTH_NULL(0)に指定して本関数を呼出した場合、以降のパラメータは全て無効となります。（チェックも行いません。）

本関数で認証パラメータを設定していない場合は、RPC_ExecProcedure関数のpar->atrにRPC_ATR_AUTHを設定しても、RPCヘッダ部の認証パラメータは認証なし（flavor=AUTH_ NULL(0)）で送信します。

本関数が正常終了すると、次のプログラム実行要求の送信データより、RPCヘッダ部の認証パラメータへ設定値を反映します。（RPC_ExecProcedureのpar->atrにRPC_ATR_AUTHを設定した場合）

(5)　RPC_ExecProcedure　　**指定したプロシージャを実行する**

【T】

C言語インタフェース

ER_UINT rdatlen = RPC_ExecProcedure(ID rpcid, T_RPC_EXEP *par, TMO tmout);

パラメータ		
ID	rpcid	指定RPCプログラム通信用ID
T_RPC_EXEP	*par	パラメータテーブル

リターンパラメータ		
ER_UINT	rdatlen	リターン値またはエラーコード
T_RPC_RESULT	par->result	実行結果
UB	par->replydata	サーバから受信したREPLYデータ

パケットの構造

```
typedef struct {
        ATR             atr;            実行属性（RPC_ATR_AUTH = 0x00000001）
        UW              procno;         プロシージャ番号
        T_RPC_RESULT    result;         プロシージャ実行結果
        UB              *calldata;      CALLとして送信するデータの先頭アドレス
        W               calllen;        CALLとして送信するデータ長
        UB              *replydata;     REPLYを受信・格納するデータ先頭アドレス
        W               replylen;       REPLYを受信可能なデータ長
} T_RPC_EXEP;
```

*1 T_RPC_RESULT : RPC_PMAP_Null の T_RPC_RESULT を参照

リターン値／エラーコード	
0または正の値	正常終了（受信したREPLYのデータ長）
E_PAR	パラメータエラー （par, par->calldata, par->replydataが0または4の倍数以外, tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（rpcid≦0，rpcid＞rpc_maxid*1）
E_NOEXS	オブジェクト未生成（rpcidが存在していない）
E_OBJ	オブジェクト状態エラー（rpcidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	RPC サーバとの接続が切断された

*1 rpc_maxid：RPC_Startで指定した最大指定RPCプログラム通信用ID

実行結果（par->result.status）	
0	正常終了*2
その他	異常終了*2

*2: RPCサーバで動作中のプログラムに依存します。

解　説

rpcidで指定したIDでRPC通信中のRPCサーバに対し、任意のプロシージャの実行要求を行います。
tmoutには、RPCサーバからの応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

RPCサーバから応答があった場合、実行結果*3をpar->resultに設定し、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもRPCサーバからの応答がない場合は、エラーコードとしてE_TMOUTを返します。
本インターフェース関数内でタイムアウト待ちになっているタスクに対し、応答待ちキャンセル（RPC_Cancel）、起床要求（wup_tsk，iwup_tsk）または強制待ち解除（rel_wai，irel_wai）が発行さ

れた場合は、待ち状態を解除し、エラーコードとしてE_RLWAIを返します。

本関数で扱われるデータ(par->calldata, par->replydata)は、ネットワークバイトオーダ（ビッグエンディアン）となります。リトルエンディアンで使用する場合は注意が必要です。エンディアンに依存しないコードにする為に、バイトストリームでデータを作成することをお勧めします。

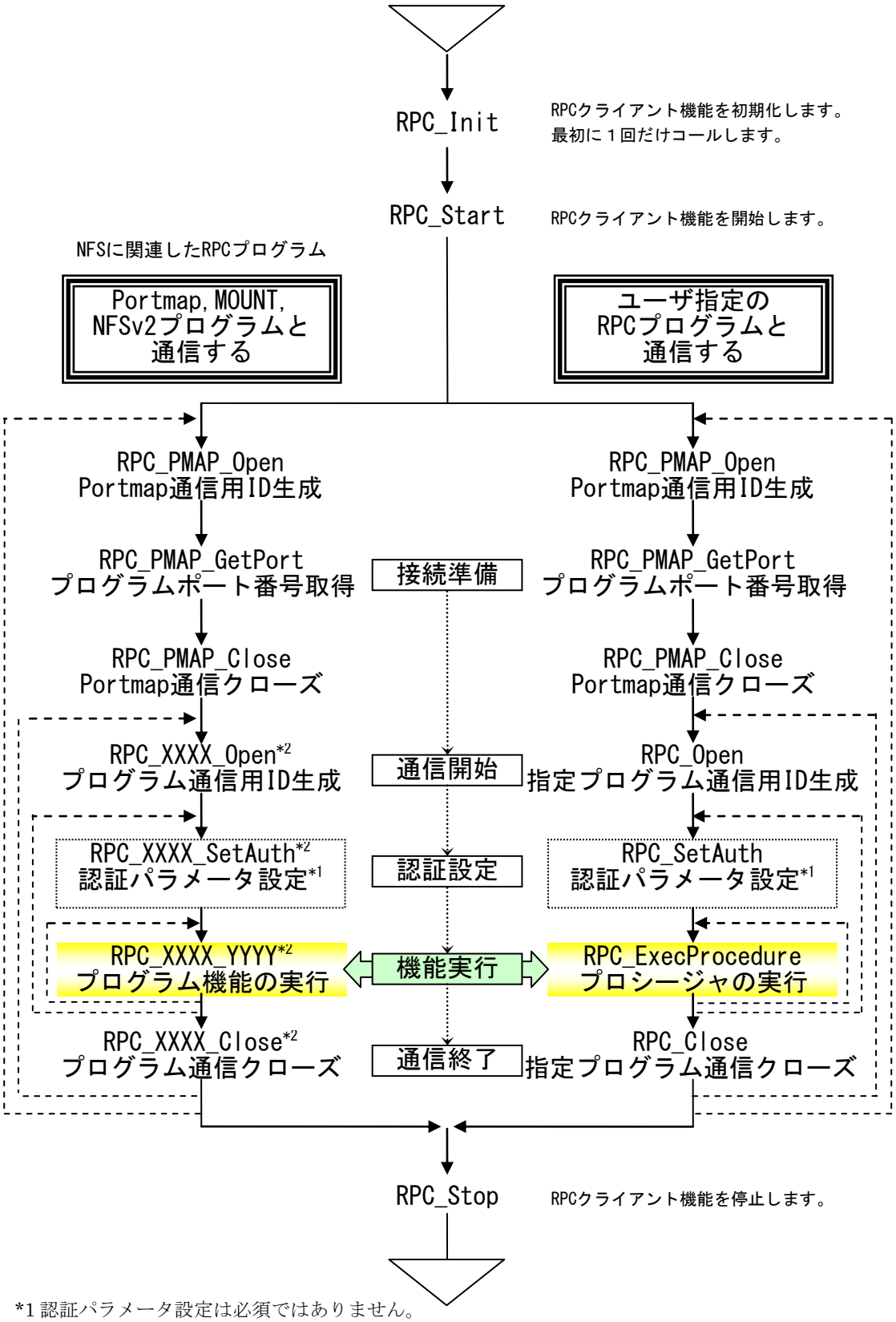
UDPでオープンした通信では、par->calllenで指定したデータ長とヘッダとの合計が1472バイトを超える大きなCALLデータを持つ場合、クライアント側からIPでフラグメントしたデータを送信します。TCP/IPマネージャの設定でIPフラグメント送信ができない設定で構築されている場合は、正しく送信できません。また、RPCクライアント内部では、UDPの送信は8760バイトの領域で送信データ作成後、CALLデータを送信する為、IPフラグメント送信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを送信しようとした場合、8760バイト以降は無視されます。

上記同様に、UDPでオープンした通信では、RPCサーバで実行したプロシージャがヘッダとの合計が1472バイトを超える大きなREPLYデータを持つ場合、RPCサーバは応答としてIPでフラグメントしたデータを送信します。par->replylenで指定した分のpar->replydataの領域を大きく持っていたとしても、TCP/IPマネージャの設定でIPフラグメントが受信できない設定で構築されている場合は、正しく受信できずに失敗します。また、RPCクライアント内部では、UDPの受信は8760バイトの受信領域で取得後、par->replylenで格納可能としたREPLYデータ分をコピーする為、IPフラグメント受信が可能である場合でも、ヘッダとデータの合計が8760バイトを超えるフラグメントデータを受信した場合、8760バイト以降は無視されます。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上で本関数を実行するようお願いします。

*3 実行結果 ： 詳細は『5.1 RPCクライアント機能実行結果一覧』を参照

3.2　低水準インターフェース関数を使用した機能実行手順



*1 認証パラメータ設定は必須ではありません。
*2 XXXX には、PMAP, MNT, NFS が当てはまります。YYYY には各機能名称が当てはまります。

4. 高水準インタフェース関数

4.1 高水準インターフェース関数一覧

表 4-1 に 高水準インターフェース関数一覧表 を示します。

表 4-1 高水準インターフェース関数一覧表

項番	種別	関数名	機能概要
1	NFSクライアント機能	NFS_ClientInit	NFSクライアント機能を初期化する
2		NFS_ClientStart	NFSクライアント機能を起動する
3		NFS_ClientStop	NFSクライアント機能を停止する
4		NFS_ClientCancel	NFSクライアント機能での待ちを解除する
5		NFS_Mount	マウントを行う
6		NFS_Unmount	アンマウントを行う
7		NFS_OpenFile	ファイルをオープンする
8		NFS_CloseFile	ファイルをクローズする
9		NFS_SetFileAttr	ファイルの属性を設定する
10		NFS_GetFileAttr	ファイルの属性を取得する
11		NFS_WriteFile	ファイルに書込む
12		NFS_ReadFile	ファイルから読出す
13		NFS_RenameFile	ファイル名を変更する
14		NFS_RemoveFile	ファイルを削除する
15		NFS_ChangeDir	カレントディレクトリの移動を行う
16		NFS_CreateDir	サブディレクトリを作成する
17		NFS_RenameDir	サブディレクトリ名を変更する
18		NFS_RemoveDir	サブディレクトリを削除する
19		NFS_GetProtError	プロトコルエラーの詳細を取得する
20	低水準関数相互運用機能	NFS_GetMntDirHandle	マウント時のハンドルを取得する
21		NFS_GetCurDirHandle	現在のディレクトリハンドルを取得する
22		NFS_GetFileHandle	ファイルハンドルを取得する

4.1.1 NFS クライアント機能インターフェース関数

(1) NFS_ClientInit NFS クライアント環境の初期化

【T／D／L／I】

C言語インタフェース

void NFS_ClientInit(void);

パラメータ

無し

リターンパラメータ

無し

解　説

NFSクライアント機能の環境を初期化します。

本関数は、システム初期化時などで、1度だけコールしてください。

(2)　NFS_ClientStart　　NFS クライアントの起動

【T／D】

C言語インタフェース

ER rtcd = NFS_ClientStart(T_NFS_CLIENT *par);

パラメータ

T_NFS_CLIENT	*par	NFSクライアント起動用パラメータテーブル

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        H       maxmntcnt;      同時にマウント可能なディレクトリ数
        H       maxopncnt;      同時にオープン可能なファイル数
        UW      *memaddr;       使用するメモリの先頭アドレス
        W       memlen;         使用可能なメモリの長さ
} T_NFS_CLIENT;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （par，par->memaddrが0または4の倍数以外， nfsConfTbl.cTcpSndLmt＜1，nfsConfTbl.cTcpSndLmt＞8192， nfsConfTbl.cUdpSndLmt＜1，nfsConfTbl.cUdpSndLmt＞8192， nfsConfTbl.cTcpRcvLmt＜1，nfsConfTbl.cTcpRcvLmt＞8192， nfsConfTbl.cUdpRcvLmt＜1，nfsConfTbl.cUdpRcvLmt＞8192）
E_NOMEM	メモリ不足（必要なメモリが確保できない）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
E_OBJ	オブジェクト状態異常（NFSクライアントが既に動作開始している）

解　説

NFSクライアント機能を起動します。

maxmntcnt、およびmaxopncntにはそれぞれNFSクライアント機能して使用する同時にマウント可能なディレクトリ数、および同時にオープン可能なファイル数を指定します。本関数では、必要となるメモリをmemaddrで指定されたメモリから切り出し、使用したメモリの長さをmemlenに返します。

指定したmemlenが必要なメモリサイズに満たない場合には、memlenに必要なメモリの長さを設定し、エラーコードとしてE_NOMEMを返します。

必要なメモリサイズは、『NFSクライアント メモリサイズ計算シート(NFSV1_SZ.xls)』で事前に確認することができます。

本関数を実行する前までに、あらかじめ低水準インターフェース関数であるRPC_Startを実行しておく必要があります。また、RPC_Startのパラメータとして、

- 最大Portmapプログラム通信用IDには、低水準インターフェースを単独で使用する数にmaxmntcntを足した数分の値を設定してください。
- 最大MOUNTプログラム通信用IDには、低水準インターフェースを単独で使用する数にmaxmntcntを足した数分の値を設定してください。
- 最大NFSv2プログラム通信用IDには、低水準インターフェースを単独で使用する数にmaxmntcntを足した数分の値を設定してください。

尚、高水準インターフェースであるNFSクライアント機能の内部では、低水準インターフェースの直接RPC通信インターフェースを使用することはありません。

(3)　NFS_ClientStop　　NFS クライアントの停止

【T／D】

C言語インタフェース

ER rtcd = NFS_ClientStop(void);

パラメータ

無し

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ILUSE	不正使用（既に NFS クライアント機能が停止している）

解　説

NFSクライアント機能を停止します。

本関数の発行後、NFSクライアントの動作を停止し、各機能は無効となります。そのため NFS_ClientStart以外の高水準インターフェース関数の動作は保証されません。

(4)　NFS_ClientCancel　　NFS クライアント処理のキャンセル

【T／D】

C言語インタフェース

ER rtcd = NFS_ClientCancel(void);

パラメータ

無し

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

解　説

NFSクライアントでの待ちをキャンセルします。

本関数で待ち解除された関数には、E_RLWAIを返します。

(5)　NFS_Mount　　マウント

【T】

C言語インタフェース

```
ER_ID mntid = NFS_Mount(T_NFS_MOUNT *par, TMO tmout);
```

パラメータ

T_NFS_MOUNT	*par	マウント用パラメータテーブル
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_ID	mntid	リターン値またはエラーコード

パケットの構造

```
typedef struct {
        T_IPV4EP            myaddr;     クライアントアドレス
        T_IPV4EP            svaddr;     サーバアドレス
        UW                  protocol;   使用するプロトコル
        T_RPC_DIRPATH       directory;  マウントディレクトリ名
        T_NFS_RES_EX        *res_ex;    リソース設定用拡張パラメータの
                                        先頭アドレス
        T_NFS_AUTH_EX       *auth_ex;   認証設定用拡張パラメータの
                                        先頭アドレス
} T_NFS_MOUNT;

typedef struct {
        UW                  ipaddr;     IPアドレス
        UH                  portno;     ポート番号
} T_IPV4EP;

typedef struct {
        T_NFS_RES_PROG      pmap;       Portmapプログラム用リソース
                                        パラメータ
        T_NFS_RES_PROG      mnt;        MOUNTプログラム用リソース
                                        パラメータ
        T_NFS_RES_PROG      nfs;        NFSv2プログラム用リソースパラメータ
} T_NFS_RES_EX;

typedef struct {
        ID                  reqcepid;   TCP/IPマネージャ通信端点ID
} T_NFS_RES_RPOG;

typedef struct {
        T_NFS_AUTH_PROG     pmap;       Portmapプログラム用認証パラメータ
        T_NFS_AUTH_PROG     mnt;        MOUNTプログラム用認証パラメータ
        T_NFS_AUTH_PROG     nfs;        NFSv2プログラム用認証パラメータ
} T_NFS_AUTH_EX;

typedef struct {
        ATR                 atr;        実行属性
        T_RPC_AUTH          auth;       認証パラメータ
} T_NFS_AUTH_RPOG;
```

```
typedef struct {
        UW              flavor;         認証方式
                                          （RPC_AUTH_NULL = 0,
                                            RPC_AUTH_UNIX = 1）
        T_RPC_NAME      machine;        RPCクライアント名
        UW              uid;            ユーザーID
        UW              gid;            グループID
        UW              auxgidcnt;      予備グループID数
        UW              auxgid[RPC_MAXAUXGID];
                                        予備グループID（RPC_MAXAUXGID = 16）
} T_RPC_AUTH;

typedef struct name {
        W       namelen;                データ長
        UB      namedata[RPC_MAXNAMLEN+1];
                                        データ（RPC_MAXNAMLEN = 255）
} T_RPC_NAME;
```

リターン値／エラーコード	
正の値	正常終了（マウントID）
E_PAR	パラメータエラー （parが0または4の倍数以外, par->myaddrが動作を開始しているアドレスと異なる, par->svaddr.ipaddrが0または0xffffffff（TCPの場合）, par->svaddr.ipaddrが0（UDPの場合）, par->directory.dirlen≦0，par->directory.dirlen＞RPC_MAXPATHLEN(1024), (par->auth_ex)->pmap.auth.machine.namelen≦0, (par->auth_ex)->pmap.auth.machine.namelen＞RPC_MAXNAMLEN(255), (par->auth_ex)->pmap.auth.auxgidcnt＞RPC_MAXAUXGID(16), (par->auth_ex)->mnt.auth.machine.namelen≦0, (par->auth_ex)->mnt.auth.machine.namelen＞RPC_MAXNAMLEN(255), (par->auth_ex)->mnt.auth.auxgidcnt＞RPC_MAXAUXGID(16), (par->auth_ex)->nfs.auth.machine.namelen≦0, (par->auth_ex)->nfs.auth.machine.namelen＞RPC_MAXNAMLEN(255), (par->auth_ex)->nfs.auth.auxgidcnt＞RPC_MAXAUXGID(16), tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号 （(par->res_ex)->pmap.reqcepid＜0, (par->res_ex)->pmap.reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大, (par->res_ex)->mnt.reqcepid＜0, (par->res_ex)->mnt.reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大, (par->res_ex)->nfs.reqcepid＜0, (par->res_ex)->nfs.reqcepid＞使用可能なTCPまたはUDP通信端点IDの最大）
E_NOID	ID番号不足（同時にマウント可能なディレクトリ数を超えた）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_CLS	接続要求が拒否された（TCP）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコルエラー（マウント失敗、等）

解　説

マウントを行います。

par->myaddr、par->svaddr、par->protocol、par->directoryにそれぞれ、クライアントが使用するIPアドレス、サーバのIPアドレス、NFS通信（Portmap,MOUNT,NFSv2）に使用するトランスポートプロトコル、マウントするディレクトリ名を指定し、本関数を呼出すことでNFSサーバにマウントします。

par->svaddrのポート番号は使用しません（設定値は無視します）。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

マウントの実行結果が正常の場合、リターン値としてマウントIDを返します。また、マウントの実行に失敗した場合、エラーコードとしてEV_PROTを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

par->res_exにはリソース設定用拡張パラメータテーブルの先頭アドレスを設定します。この拡張パラメータテーブルを使用する場合は、それぞれのプログラム通信でのTCP/IPマネージャ通信端点IDの指定が可能です。
par->res_exに0を指定した場合、または4の倍数以外のアドレスを指定した場合は、この拡張パラメータテーブルは使用しません。（その場合、TCP/IPマネージャの通信端点IDは自動生成となります。）

par->auth_exには認証設定用拡張パラメータテーブルの先頭アドレスを設定します。この拡張パラメータテーブルを使用する場合は、それぞれのプログラム通信での実行時の認証機能の有効／無効が設定可能です。
par->auth_exに0を指定した場合、または4の倍数以外のアドレスを指定した場合は、この拡張パラメータテーブルは使用しません。（その場合、認証機能は無効となります。）

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

本関数でリターン値として返すマウントIDは、以降で説明するNFSクライアント高水準インターフェース関数で使用します。

(6)　NFS_UnMount　　**アンマウント**

【T】

C言語インタフェース

ER rtcd = NFS_UnMount(ID mntid, TMO tmout);

パラメータ		
ID	mntid	マウントID
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt[*1]）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

NFSサーバでマウント中のディレクトリをアンマウントします。

アンマウントが正常に実行されると、mntidで指定したマウントIDに結びついたサーバやマウントディレクトリの情報を解放し、リターン値としてE_OKを返します。
その際、mntidは再びマウントにより情報が結びつけられるまで使用不能となります。

尚、本関数のみ、待ち状態を解除した場合は正常終了となり、リターン値としてE_OKを返します。

(7)　NFS_OpenFile　　ファイルオープン

【T】

C言語インタフェース

```
ER_ID fileid = NFS_OpenFile(ID mntid, UB *fname, W len, TMO tmout);
```

パラメータ

ID	mntid	マウントID
UB	*fname	ファイル名
W	len	ファイル名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_ID	fileid	リターン値またはエラーコード

リターン値／エラーコード

正の値	正常終了（ファイルID）
E_PAR	パラメータエラー （fnameが0または4の倍数以外，len≦0，len＞RPC_MAXNAMLEN(255), tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOID	ID番号不足（同時にオープン可能なファイル数を超えた）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコルエラー（ファイルオープン失敗、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidがマウント中であるNFSサーバ上のディレクトリで、fnameで指定したファイル名のファイルをオープンします。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ファイルオープンが正常に実行されると、リターン値としてファイルIDを返します。また、ファイルオープンに失敗した場合には、par->result.statusに実行結果を設定し、リターン値としてEV_PROTを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ファイル名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名とファイル名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

本関数がリターン値として返すファイルIDは、以降で説明するNFSクライアント高水準インターフェース関数で使用します。

(8)　NFS_CloseFile　　**ファイルクローズ**

【T／D】

C言語インタフェース

ER rtcd = NFS_CloseFile(ID fileid);

パラメータ		
ID	fileid	ファイルID

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*1)
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

*1 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

解　説

ファイルオープン中のファイルをクローズします。

ファイルクローズが正常に実行されると、fileidで指定したファイル情報を解放し、リターン値としてE_OKを返します。

その際、fileidは再びファイルオープンにより情報が結びつけられるまで使用不能となります。

(9)　NFS_SetFileAttr　　**ファイル属性設定**

【T】

C言語インタフェース

```
ER rtcd = NFS_SetFileAttr(ID fileid, T_RPC_FATTR *attr, TMO tmout);
```

パラメータ

ID	fileid	ファイルID
T_RPC_FATTR	*attr	ファイル属性
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

パケットの構造

```
typedef struct fattr {
        UW              type;       種別（未使用）
        UW              mode;       モード
        UW              nlink;      リンク（未使用）
        UW              uid;        ユーザID
        UW              gid;        グループID
        UW              size;       サイズ
        UW              blocksize;  ブロックサイズ（未使用）
        UW              rdev;       特殊デバイス（未使用）
        UW              blocks;     ブロック数（未使用）
        UW              fsid;       ファイルシステムID（未使用）
        UW              fileid;     ファイルID（未使用）
        T_RPC_TIMEVAL   atime;      アクセス時刻
        T_RPC_TIMEVAL   mtime;      更新時刻
        T_RPC_TIMEVAL   ctime;      作成時刻（未使用）
} T_RPC_FATTR;

typedef struct timeval {
        UW              seconds;    秒
        UW              useconds;   マイクロ秒
} T_RPC_TIMEVAL;
```

*1 fileid: NFS サーバが管理するファイル ID で、NFS クライアントがパラメータにするものとは別物

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （attrが0または4の倍数以外，tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*2）
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（fileidのファイルが他のユーザによって削除された、等）

*2 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

解　説

fileidで指定したファイルの属性をattrで指定した属性で更新します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ファイル属性の設定が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

尚、本関数において、以下の属性設定値は無視されます（チェックもしません）。
・par->type（種別）
・par->nlink（リンク）
・par->blocksize（ブロックサイズ）
・par->rdev（特殊デバイス）
・par->blocks（ブロック数）
・par->fsid（ファイルシステムID）
・par->fileid（ファイルID）
・par->ctime（作成時刻）

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(10)　NFS_GetFileAttr　　**ファイル属性取得**

【T】

C言語インタフェース

ER rtcd = NFS_GetFileAttr(ID fileid, T_RPC_FATTR *attr, TMO tmout);

パラメータ

ID	fileid	ファイルID
T_RPC_FATTR	*attr	ファイル属性
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_FATTR	*attr	ファイル属性

パケットの構造

```
typedef struct fattr {
        UW              type;           種別（RPC_NFNON = 0［非ファイル］,
                                                RPC_NFREG = 1［普通のファイル］,
                                                RPC_NFDIR = 2［ディレクトリ］,
                                                RPC_NFBLK = 3[特殊なブロックデバイス],
                                                RPC_NFCHR = 4
                                                        ［特殊なキャラクタデバイス],
                                                RPC_NFLNK = 5［シンボリックリンク］）
        UW              mode;           モード
        UW              nlink;          リンク
        UW              uid;            ユーザID
        UW              gid;            グループID
        UW              size;           サイズ
        UW              blocksize;      ブロックサイズ
        UW              rdev;           特殊デバイス
        UW              blocks;         ブロック数
        UW              fsid;           ファイルシステムID
        UW              fileid;         ファイルID
        T_RPC_TIMEVAL   atime;          アクセス時刻
        T_RPC_TIMEVAL   mtime;          更新時刻
        T_RPC_TIMEVAL   ctime;          作成時刻
} T_RPC_FATTR;

typedef struct timeval {
        UW              seconds;        秒
        UW              useconds;       マイクロ秒
} T_RPC_TIMEVAL;
```

*1 fileid: NFS サーバが管理するファイル ID で、NFS クライアントがパラメータにするものとは別物

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（attrが0または4の倍数以外，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*2）
E_TMOUT	タイムアウト
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（fileidのファイルが他のユーザによって削除された、等）

*2 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

解　説

fileidで指定したファイルの最新のファイル属性をattrで指定したアドレスの構造へコピーします。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

tmoutにTMO_POL(0)を指定した場合、NFSクライアント内部のファイル属性テーブルからattrへコピーします。

tmoutに正の値またはTMO_FEVR(-1)を指定した場合、サーバからファイル属性を取得し、NFSクライアント内部のファイル属性テーブルを更新した後、attrへコピーします。

NFSクライアントのファイル属性テーブルはファイルオープン時、ファイル属性設定時、ファイル書込み時、およびファイル読出し時に更新されます。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。

詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(11)　NFS_WriteFile　　**ファイル書込み**

【T】

C言語インタフェース

ER_UINT wdatlen = NFS_WriteFile(ID fileid, W offset, UB *dat, W datlen, TMO tmout);

パラメータ		
ID	fileid	ファイルID
W	offset	ファイル書込み開始オフセット
UB	*dat	ファイル書込みデータ
W	datlen	ファイル書込みデータ長
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER_UINT	wdatlen	リターン値またはエラーコード

リターン値／エラーコード	
正の値	正常終了（書込みを行ったデータ長）
E_PAR	パラメータエラー （datが0または4の倍数以外，datlen≦0， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*1）
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（fileidのファイルが他のユーザによって削除された、等）

*1 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

解　説

fileidで指定したNFSサーバ上ファイルに対して書込みます。

ファイルの先頭からoffset分進んだ位置から、datlenで指定したデータ長の分だけ、datで指定したアドレスの内容を書込みます。

datlenに大きいデータ長を指定した場合、本関数内部では複数回に分けて書込みを実行します。複数回に分けて書込みを行う際の１回の書込みデータ長は構築情報のnfsConfTbl.cTcpSndLmt(TCPの場合)、nfsConfTbl.cUdpSndLmt(UDPの場合)になります。

尚、UDPで１回の書込みデータ長が1472を越える設定の場合、TCP/IPマネージャにてIPフラグメント送信が可能な状態に構築されていないと正しく送信できません。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

また、大きいデータをサーバへ書込む際、tmoutはサーバ書込み毎の待ち時間となるため、待ち時間の合計がtmoutで指定したタイムアウト時間より長くなる場合があります。

ファイル書込みが正常に実行できた場合、ファイル属性を更新し、リターン値として書き込みを行ったデータ長を返します。

tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

尚、ファイル書込み中に異常が発生した場合のファイル属性は不定となります。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。

詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(12)　NFS_ReadFile　　ファイル読出し

【T】

C言語インタフェース

ER_UINT rdatlen= NFS_ReadFile(ID fileid, W offset, UB *dat, W datlen, TMO tmout);

パラメータ

ID	fileid	ファイルID
W	offset	ファイル読出し開始オフセット
UB	*dat	ファイル読出しデータ格納アドレス
W	datlen	ファイル読出しデータ格納可能データ長
TMO	tmout	タイムアウト指定

リターンパラメータ

ER_UINT	rdatlen	リターン値またはエラーコード
UB	*dat	ファイル読出しデータ

リターン値／エラーコード

正の値	正常終了（格納を行ったデータ長）
E_PAR	パラメータエラー （datが0または4の倍数以外，daten≦0， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*1）
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（fileidのファイルが他のユーザによって削除された、等）

*1 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

解　説

fileidで指定したNFSサーバ上ファイルから読出します。

ファイルの先頭からoffset分進んだ位置から、datlenで指定したデータ長の分だけ、ファイル読出しデータをdatで指定したアドレスに格納します。

datlenに大きいデータ長を指定した場合、本関数内部では複数回に分けて読出しを実行します。複数回に分けて読出しを行う際の１回の読出しデータ長は構築情報のnfsConfTbl.cTcpRcvLmt(TCPの場合)、nfsConfTbl.cUdpRcvLmt(UDPの場合)になります。

尚、UDPで１回の読出しデータ長が1472を越える設定の場合、TCP/IPマネージャにてIPフラグメント受信が可能な状態に構築されていないと正しく受信できません。

また、御使用になるLANコントローラによっては、受信FIFOや受信バッファ数に制限があるため、フラグメントデータが揃わずに受信失敗(タイムアウト)する場合があります。LANコントローラの性能をしっかりと把握した上でnfsConfTbl.cUdpRcvLmtを設定するようお願いします。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。

tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

また、大きいデータをサーバから読出す際、tmoutはサーバ読出し毎の待ち時間となるため、待ち時間の合計がtmoutで指定したタイムアウト時間より長くなる場合があります。

ファイル読出しが正常に実行されると、ファイル属性を更新し、リターン値として格納したデータ長を返します。

tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

尚、ファイル読出し中に異常が発生した場合のファイル属性は不定となります。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(13)　NFS_RenameFile　　**ファイル名変更**

【T】

C言語インタフェース

ER rtcd = NFS_RenameFile(ID mntid, UB *fnfrom, W fromlen, UB *fnto, W tolen, TMO tmout);

パラメータ		
ID	mntid	マウントID
UB	*fnfrom	変更前のファイル名
W	fromlen	変更前のファイル名の長さ
UB	*fnto	変更後のファイル名
W	tolen	変更後のファイル名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （fnfrom,fntoが0または4の倍数以外， fromlen≦0，fromlen＞RPC_MAXNAMLEN(255)， tolen≦0，tolen＞RPC_MAXNAMLEN(255)， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常 （fnfromのファイルが存在しない，fntoのファイルが既に存在する、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリで、fnfromで指定したファイルの名称をfntoで指定した名称に変更します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ファイル名変更が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ファイル名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名とファイル名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(14)　NFS_RemoveFile　　**ファイル削除**

【T】

C言語インタフェース

ER rtcd = NFS_RemoveFile(ID mntid, UB *fname, W len, TMO tmout);

パラメータ

ID	mntid	マウントID
UB	*fname	削除するファイル名
W	len	削除するファイル名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー （fnameが0または4の倍数以外，len≦0，len＞RPC_MAXNAMLEN(255), tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（fnameのファイルが存在しない、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリで、fnameで指定したファイルを削除します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ファイル削除が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ファイル名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名とファイル名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(15)　NFS_ChangeDir　　**カレントディレクトリ移動**

【T】

C言語インタフェース

ER rtcd = NFS_ChangeDir(ID mntid, UB *directory, W len, TMO tmout);

パラメータ		
ID	mntid	マウントID
UB	*directory	移動先のディレクトリ名
W	len	移動先のディレクトリ名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （directoryが0または4の倍数以外，len≦0，len＞RPC_MAXNAMLEN(255)， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（directoryのディレクトリが存在しない、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリから、directoryで指定したディレクトリへカレントディレクトリを移動します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ディレクトリの移動が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ディレクトリ名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名と末端のディレクトリ名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(16)　NFS_CreateDir　　**サブディレクトリ作成**

【T】

C言語インタフェース

ER rtcd = NFS_CreateDir(ID mntid, UB *directory, W len, TMO tmout);

パラメータ		
ID	mntid	マウントID
UB	*directory	作成するディレクトリ名
W	len	作成するディレクトリ名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （directoryが0または4の倍数以外，len≦0，len＞RPC_MAXNAMLEN(255)， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（directoryのディレクトリが既に存在する、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリで、directoryで指定したディレクトリを新規に作成します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ディレクトリの作成が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ディレクトリ名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名と末端のディレクトリ名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(17)　NFS_RenameDir　　**サブディレクトリ名変更**

【T】

C言語インタフェース

ER rtcd = NFS_RenameDir(ID mntid, UB *dirfrom, W fromlen, UB *dirto, W tolen, TMO tmout);

パラメータ		
ID	mntid	マウントID
UB	*dirfrom	変更前のディレクトリ名
W	fromlen	変更前のディレクトリ名の長さ
UB	*dirto	変更後のディレクトリ名
W	tolen	変更後のディレクトリ名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （dirfrom,dirtoが0または4の倍数以外， fromlen≦0，fromlen＞RPC_MAXNAMLEN(255)， tolen≦0，tolen＞RPC_MAXNAMLEN(255)， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常 （dirfromのディレクトリが存在しない，dirtoのディレクトリが既に存在する、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリで、dirfromで指定したディレクトリの名称をdirtoで指定した名称へ変更します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ディレクトリ名の変更が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ディレクトリ名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名と末端のディレクトリ名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(18)　NFS_RemoveDir　　**サブディレクトリ削除**

【T】

C言語インタフェース

ER rtcd = NFS_RemoveDir(ID mntid, UB *directory, W len, TMO tmout);

パラメータ		
ID	mntid	マウントID
UB	*directory	削除するディレクトリ名
W	len	削除するディレクトリ名の長さ
TMO	tmout	タイムアウト指定

リターンパラメータ		
ER	rtcd	リターン値またはエラーコード

リターン値／エラーコード	
E_OK	正常終了
E_PAR	パラメータエラー （directoryが0または4の倍数以外，len≦0，len＞RPC_MAXNAMLEN(255)， tmout＝TMO_POL(0)，tmout＜TMO_FEVR(-1)）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_TMOUT	タイムアウト
E_RLWAI	**処理のキャンセル，待ち状態の強制解除**
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）
EV_PROT	プロトコル異常（directoryのディレクトリが存在しない、等）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidのNFSサーバ上ディレクトリの、directoryで指定したディレクトリを削除します。

tmoutには、クライアント内部で実行する各プロシージャの実行応答を受信するまでの待ち時間を設定します。
tmoutに正の値を指定した場合は待ち時間、TMO_FEVR(-1)を指定した場合は永久待ちとなります。

ディレクトリの削除が正常に実行されると、リターン値としてE_OKを返します。
tmoutに指定した待ち時間を過ぎてもプロシージャの実行応答がない場合は、エラーコードとしてE_TMOUTを返します。

本関数では、ディレクトリ名について、現在のディレクトリからの相対パス指定が可能です。ただし、ディレクトリパス名と末端のディレクトリ名の長さの合計がRPC_MAXNAMLEN(255)を超える指定は出来ません。

プロトコル異常が発生した場合（エラーコードとしてEV_PROTが返る場合）は、詳細コードをGetProtError()関数で取得することができます。
詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(19)　NFS_GetProtError　　**プロトコルエラー詳細の取得**

【T／D】

C言語インタフェース

UW rtcd = NFS_GetProtError(void);

パラメータ

無し

リターンパラメータ

UW	rtcd	プロトコルエラー詳細コード

解　説

プロトコルエラー(EV_PROT)が発生した場合の詳細コードを取得します。

本関数で詳細コードを取得する場合は、EV_PROT発生直後に実行してください。連続してプロトコルエラーが発生するような場合は、後に発生したコードによって上書きされます。

尚、プロトコルエラー詳細コードについては、『5.2 NFSクライアント機能プロトコルエラー詳細コード一覧』を参照してください。

(20)　NFS_GetMntDirHandle　　**マウント時のディレクトリハンドル取得**

【T】

C言語インタフェース

ER rtcd = NFS_GetMntDirHandle(ID mntid, T_RPC_FHANDLE *handle);

パラメータ

ID	mntid	マウントID
T_RPC_FHANDLE	*handle	ディレクトリハンドル格納領域の先頭アドレス

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_FHANDLE	*handle	ディレクトリハンドル

パケットの構造

```
typedef struct fhandle {
        UB      handle[RPC_FHSIZE];         ファイルハンドル（RPC_FHSIZE = 32）
} T_RPC_FHANDLE;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（handleが0または4の倍数以外）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt*1）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidでマウント中のマウント時のディレクトリハンドルを取得します。

本関数を使用することで、低水準インターフェース関数とのディレクトリハンドルの相互運用が可能です。

NFS_Mountでマウントしたディレクトリのハンドルが取得できます。mntidにはNFS_Mountで取得したmntidを指定してください。

(21)　NFS_GetCurDirHandle　　**現在のディレクトリハンドル取得**

【T】

C言語インタフェース

ER rtcd = NFS_GetCurDirHandle(ID mntid, T_RPC_FHANDLE *handle);

パラメータ

ID	mntid	マウントID
T_RPC_FHANDLE	*handle	ディレクトリハンドル格納領域の先頭アドレス

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_FHANDLE	*handle	ディレクトリハンドル

パケットの構造

```
typedef struct fhandle {
        UB      handle[RPC_FHSIZE];         ファイルハンドル（RPC_FHSIZE = 32）
} T_RPC_FHANDLE;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（handleが0または4の倍数以外）
E_ID	不正ID番号（mntid≦0，mntid＞maxmntcnt[*1]）
E_NOEXS	オブジェクト未生成（mntidが存在していない）
E_OBJ	オブジェクト状態エラー（mntidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

*1 maxmntcnt: NFS_ClientStartで指定した同時にマウント可能なディレクトリ数

解　説

mntidでマウント中の現在のディレクトリハンドルを取得します。

本関数を使用することで、低水準インターフェース関数とのディレクトリハンドルの相互運用が可能です。

NFS_Mountでマウント後、NFS_ChangeDirでサブディレクトリに移動したディレクトリのハンドルが取得できます。NFS_ChangeDirを実行していない状態の場合は、マウント時のディレクトリハンドルが取得できます。mntidにはNFS_Mountで取得したmntidを指定してください。

(22)　NFS_GetFileHandle　　**ファイルハンドル取得**

【T】

C言語インタフェース

ER rtcd = NFS_GetFileHandle(ID fileid, T_RPC_FHANDLE *handle);

パラメータ

ID	fileid	ファイルID
T_RPC_FHANDLE	*handle	ファイルハンドル格納領域の先頭アドレス

リターンパラメータ

ER	rtcd	リターン値またはエラーコード
T_RPC_FHANDLE	*handle	ファイルハンドル

パケットの構造

```
typedef struct fhandle {
        UB      handle[RPC_FHSIZE];     ファイルハンドル（RPC_FHSIZE = 32）
} T_RPC_FHANDLE;
```

リターン値／エラーコード

E_OK	正常終了
E_PAR	パラメータエラー（handleが0または4の倍数以外）
E_ID	不正ID番号（fileid≦0，fileid＞maxopncnt*2)
E_NOEXS	オブジェクト未生成（fileidが存在していない）
E_OBJ	オブジェクト状態エラー（fileidは既に実行中）
E_ILUSE	不正使用（NFS または RPC のクライアント機能が停止している）

*2 maxopncnt: NFS_ClientStartで指定した同時にオープン可能なファイル数

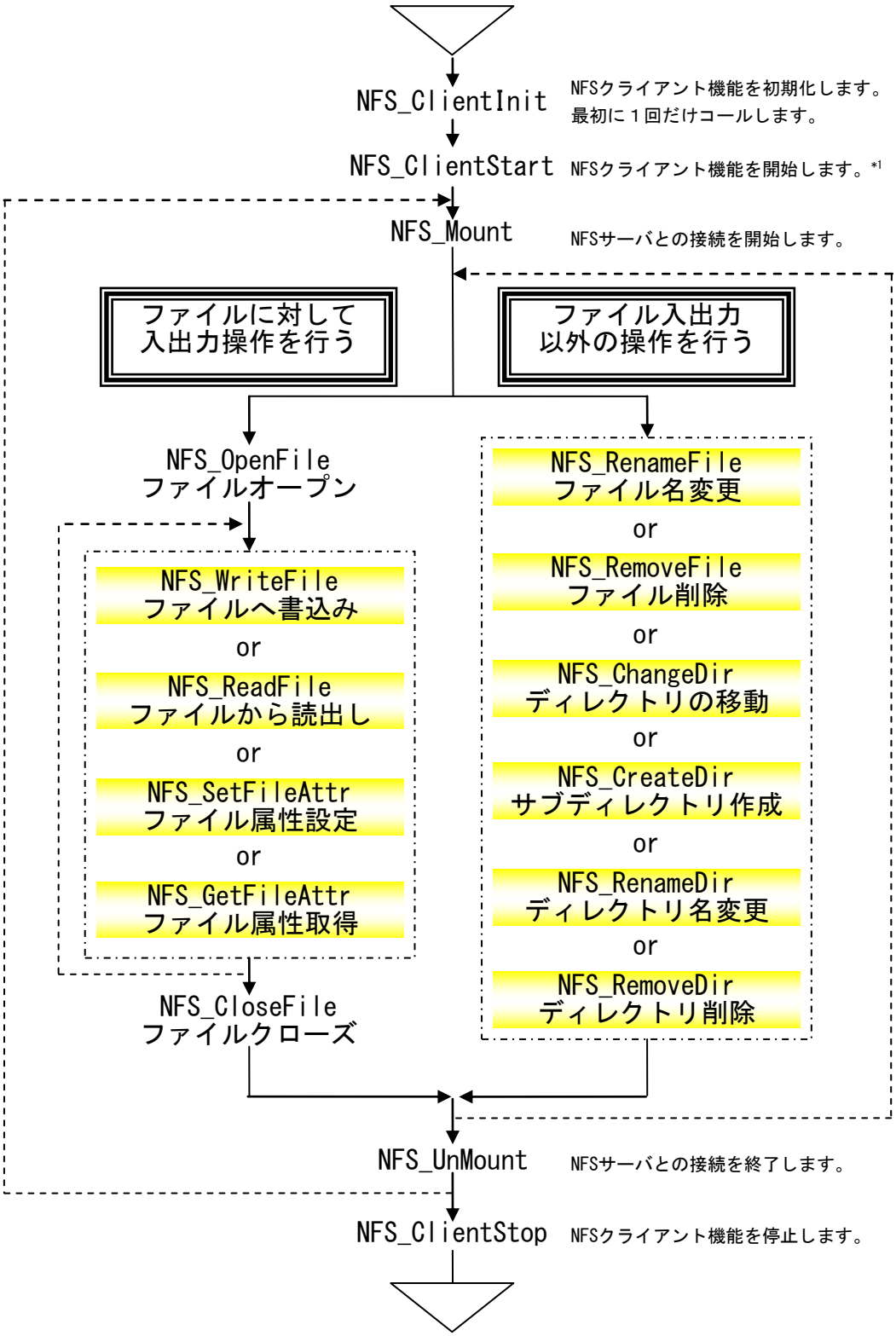
解　説

fileidで指定したファイルのファイルハンドルを取得します。

本関数を使用することで、低水準インターフェース関数とのファイルハンドルの相互運用が可能です。

NFS_OpenFileでオープンしたファイルのハンドルが取得できます。fileidにはNFS_OpenFileで取得したfileidを指定してください。

4.2　高水準インターフェース関数の使用手順



*1 NFS_ClientStart を実行する前に、NFS_ClientInit、および低水準インターフェース関数の RPC_Start を実行しておく必要があります。

5. 付録

5.1 RPC クライアント機能実行結果一覧

表 5-1にRPC応答状態を、表 5-2、表 5-3にRPC受付状態を、表 5-4、表 5-5、表 5-6に各機能の実行結果をそれぞれ示します。

※ ここに記載した値および定義名はRFCで定義されている内容であり、RPCサーバによっては独自実装等の理由により、記載がない値を使用している場合があります。値の記載がない場合はRPCサーバ、およびRPCサーバ上で動作中のプログラムの仕様を確認ください。

表 5-1 RPC応答状態（par->result.replystate）

値	定義名	意味
0	MSG_ACCEPTED	メッセージ受付
1	MSG_DENIED	メッセージ拒否

表 5-2 メッセージ受付時のRPC受付状態（par->result.acceptstate）

値	定義名	意味
0	SUCCESS	RPC実行正常終了
1	PROG_UNAVAIL	プログラム番号異常
2	PROG_MISMATCH	プログラムバージョン異常
3	PROC_UNAVAIL	プロシージャ番号異常
4	GARBAGE_ARGS	プロシージャパラメータ異常
5	SYSTEM_ERR	システム異常

表 5-3 メッセージ拒否時のRPC受付状態（par->result.acceptstate）

値	定義名	意味
0	RPC_MISMATCH	RPCバージョン異常
1	AUTH_ERROR	認証異常

表 5-4 認証異常時の機能実行結果（par->result.status）

値	定義名	意味
0	AUTH_OK	正常終了
1	AUTH_BADCRED	Credential情報異常
2	AUTH_REJECTEDCRED	Credential拒否
3	AUTH_BADVERF	Verifier情報異常
4	AUTH_REJECTEDVERF	Verifier拒否
5	AUTH_TOOWEAK	セキュリティが弱い
6	AUTH_INVALIDRESP	レスポンス異常
7	AUTH_FAILED	未定義な異常

表 5-5　Portmapプログラム実行時の機能実行結果（par->result.status）

値	定義名	意味
0	FALSE	失敗
1	TRUE	成功

表 5-6　NFSv2プログラム実行時の機能実行結果（par->result.status）

値	定義名	意味
0	NFS_OK	正常終了
1	NFSERR_PERM	所有者権限異常
2	NFSERR_NOENT	ファイルまたはディレクトリが存在しない
5	NFSERR_IO	入出力異常
6	NFSERR_NXIO	デバイスまたはアドレスが存在しない
13	NFSERR_ACCESS	アクセス拒否
17	NFSERR_EXIST	ファイルまたはディレクトリが既に存在する
19	NFSERR_NODEV	デバイスが存在しない
20	NFSERR_NOTDIR	ディレクトリではない
21	NFSERR_ISDIR	ディレクトリである
27	NFSERR_FBIG	ファイルが大きすぎる
28	NFSERR_NOSPC	デバイス残容量不足
30	NFSERR_ROFS	読込み専用
63	NFSERR_NAMETOOLONG	ファイル名が長すぎる
66	NFSERR_NOTEMPTY	ディレクトリが空ではない
69	NFSERR_DQUOT	ディスク割当て超過
70	NFSERR_STALE	ハンドル異常
99	NFSERR_WFLUSH	キャッシュフラッシュ異常

5.2　NFS クライアント機能プロトコルエラー詳細コード一覧

表 5-7にNFSクライアントプロトコルエラー詳細コードを示します。

※　ここに記載した一部の値および定義名はRFCで定義されている内容であり、RPCサーバによっては独自実装等の理由により、記載がない値を使用している場合があります。値の記載がない場合はRPCサーバ、およびRPCサーバ上で動作中のプログラムの仕様を確認ください。

表 5-7　NFSクライアントプロトコルエラー詳細コード

値	定義名	意味
0	NFS_OK	正常終了
1	NFSERR_PERM	所有者権限異常
2	NFSERR_NOENT	ファイルまたはディレクトリが存在しない
5	NFSERR_IO	入出力異常
6	NFSERR_NXIO	デバイスまたはアドレスが存在しない
13	NFSERR_ACCESS	アクセス拒否
17	NFSERR_EXIST	ファイルまたはディレクトリが既に存在する
19	NFSERR_NODEV	デバイスが存在しない
20	NFSERR_NOTDIR	ディレクトリではない
21	NFSERR_ISDIR	ディレクトリである
27	NFSERR_FBIG	ファイルが大きすぎる
28	NFSERR_NOSPC	デバイス残容量不足
30	NFSERR_ROFS	読込み専用
63	NFSERR_NAMETOOLONG	ファイル名が長すぎる
66	NFSERR_NOTEMPTY	ディレクトリが空ではない
69	NFSERR_DQUOT	ディスク割当て超過
70	NFSERR_STALE	ハンドル異常
99	NFSERR_WFLUSH	キャッシュフラッシュ異常
10004	NFSERR_NOTSUPP	未サポート
10006	NFSERR_SERVERFAULT	サーバプログラムが動作していない、または失敗した

HI.CommunicationEngine
NFSクライアント リファレンスマニュアル
CM7000NFS01J-3

発行年月　　2014年 9月　　第3版
発　　行　　ルネサスセミコンダクタ
　　パッケージ＆テストソリューションズ株式会社
編　　集　　ルネサスセミコンダクタ
　　パッケージ＆テストソリューションズ株式会社