

GridFTP : グリッド向け FTP のプロトコル拡張

本書の状態

この文書はグローバルグリッドフォーラムドラフトであり、GFD-C.1 のすべての内容に合致する。

本書で使用する慣用語

この文書で使われている「しなければならない (MUST)」、「してはいけない (MUST NOT)」、「要求される (REQUIRED)」、「することになる (SHALL)」、「することはない (SHALL NOT)」、「する必要がある (SHOULD)」、「しないほうがよい (SHOULD NOT)」、「推奨する (RECOMMENDED)」、「してもよい (することもできる) (MAY)」、「選択できる (OPTIONAL)」は、RFC-2119 [5] の語義と同一のものと解釈する。

概要

この文書は GridFTP プロトコルの全容を述べたものである。このプロトコルは、RFC959「ファイル転送プロトコル (FTP)」、RFC2228「FTP セキュリティの拡張」、RFC2389「ファイル転送プロトコルの機能ネゴシエーション方式」、IETF ドラフト draft-ietf-ftpext-mlst-16「FTP 拡張」をもとにしたもので、いずれも参考として各所に引用されている。また、以下の拡張機能を定義している。

SPAS Striped Passive

このコマンドは PASV コマンドと似ているが、一連のホスト/ポート接続を返すことができる。これによりストライピング、つまり複数のネットワークエンドポイント (マルチホームホストまたは複数のホスト) を転送に関わらせることができる。

SPOR Striped Port

このコマンドは PORT コマンドと似ているが、一連のホスト/ポート接続を送信することができる。これによりストライピング、つまり複数のネットワークエンドポイント (マルチホームホストまたは複数のホスト) を転送に関わらせることができる。

ERET Extended Retrieve

このコマンドは RETR コマンドと似ているが、転送前にデータを操作 (一般にはサイズの縮小) することができる。

ESTO Extended Store

このコマンドは STOR コマンドと似ているが、格納前にデータを操作（一般にはサイズの縮小）することができる。

SBUF TCP バッファサイズ設定

TCP バッファサイズを明示的に設定する。

ABUF オートネゴシエート TCP バッファサイズ

アルゴリズムの選択により、自動的に適切な TCP バッファサイズを決定することができる。

DCAU データチャネル認証

RFC 2228 はコントロールチャネルで gss を使用する方法を確立したが、データチャネルについては手が付けられていない。我々はデータチャネルで認証を行うことによりデータの傍受を防止できる拡張機能を追加した。

以下のようなオプションが追加されている。

RETR：複数のネットワークエンドポイント（ストライピング）に記述される際、TCP ストリーム数の仕様を、一組のネットワークエンドポイント（PARALLELISM）とデータのレイアウトとの間に使用できるようにするオプションが追加されている。

実行時にこれらのコマンドをサポートするかどうかをクライアントが決定できるように、適切な機能応答（FEAT）が定義されている。ERET、ESTO、ABUF のようなアルゴリズム選択コマンドは、利用可能なアルゴリズムの一覧を表示する HELP コマンドへ応答できるように実装する必要がある。

次のような新しいモードが定義されている。

EBLOCK（拡張ブロック）は、この仕様における機能の多くを実現するための鍵となる。データをブロックで送信するが、1つのブロックは8ビットのフラグ、64ビット長、64ビットオフセット、そしてデータの「長さ」分のバイト数で構成される。これにより、パラレルおよびストライプ型データ転送に必要なアウトオブオーダー受信が可能となる。

この機能の組み合わせにより、安全、高速、効率的で柔軟な、拡張性に富むデータ転送およびデータアクセスが可能となる。

目次

1. 執筆者
2. はじめに
 - 2.1. 背景
 - 2.2. 動機
 - 2.3. 用語解説
3. 拡張機能
 - 3.1. 要約
 - 3.2. コマンド
 - 3.2.1. Striped Passive (SPAS)
 - 3.2.2. Striped Data Port (SPOR)
 - 3.2.3. Extended Retrieve (ERET)
 - 3.2.4. Extended Store (ESTO)
 - 3.2.5. Set Buffer Size (SBUF)
 - 3.2.6. オートネゴシエートバッファサイズ (ABUF)
 - 3.2.7. データチャンネル認証 (DCAU)
 - 3.3. 機能
 - 3.4. 拡張ブロックモード
 - 3.4.1. 拡張ブロックヘッダ
 - 3.4.2. 拡張ブロック EODC ヘッダ (記述子でコード 64 セット)
 - 3.4.3. 拡張ブロックモードにおける EOF の処理
 - 3.5. オプション
 - 3.5.1. RETR に対するオプション
 - 3.5.1.1. レイアウトオプション
 - 3.5.1.2. パラレルリズムオプション
4. 宣言的仕様
 - 4.1. 最小実装
 - 4.2. 推奨実装
5. セキュリティ要件
6. 既知の問題

6. 1. EBLOCK モードにおける一方向性のデータ転送
6. 2. PASV/SPAS および STOR/RETR 間の配列依存
6. 3. コマンドのパイプライン化と E ブロックデータチャネルの再利用
6. 4. ディスクリソース管理のサポート
7. 付録 I : リスタート
8. 付録 II : パフォーマンスのモニタリング
9. 付録 III : 開発中に検討された RFC
10. GGF 著作権表示
11. GGF 知的所有権表示
12. 参考

1. 執筆者

この仕様の完成にあたっては、多くの方々にご協力いただいた。執筆者はアルファベット順に記載している。執筆者のリストに漏れている方があれば、一報を乞う。

W. Allcock

アルゴンヌ国立研究所

エディタ。初期のアーキテクチャの審議およびプロトコルの初回ドラフトの作成を担当。

J. Bester

アルゴンヌ国立研究所

初期のアーキテクチャの審議およびプロトコルの初回ドラフトの作成を担当。Globus インプリメンテーションの主要開発メンバーの一人。

J. Bresnahan

アルゴンヌ国立研究所

初期のアーキテクチャの審議およびプロトコルの初回ドラフトの作成を担当。Globus インプリメンテーションの主要開発メンバーの一人。

S. Meder

アルゴンヌ国立研究所

初期のアーキテクチャの審議およびプロトコルの初回ドラフトの作成を担当。Globus イン

プリメンテーションの主要開発メンバーの一人。

P. Plaszczak

アルゴンヌ国立研究所

2 回目以降のドラフトのレビューを担当。Globus Java クライアントインプリメンテーションの共同執筆者。

S. Tuecke

アルゴンヌ国立研究所

主任設計者、プロトコルの初回ドラフトの作成を担当。

上記のリストには記載されていないが、本書の各節のさまざまな問題点について見解を述べ、非公式のレビューに参加された方々も多数いる。このような多くの方々からの意見や指摘がなければ、本書は十分な成果を得ることができなかつた。貴重なコメントを下さった皆様に心より感謝する。

2. はじめに

2.1. 背景

一般的にグリッド環境では、分散データへのアクセスは分散コンピューティング資源へのアクセスと同じように重要である。分散型の科学およびエンジニアリング用アプリケーションでは、次の2つの要件を満たさなければならない。

- ・ストレージシステム間で大量のデータ（テラバイトやペタバイト）を送信できること。
- ・地理的に分散した多くのアプリケーションやユーザーが、分析や視覚化のために大量のデータ（ギガバイトやテラバイト）にアクセスできること。

インターネットで最も広く利用されている標準データ転送およびアクセスプロトコル、つまり FTP や HTTP には、グリッドアプリケーションに必要とされる重要な機能が欠落している。また、マルチプル・ストレージ・アーカイブ・システム（HPSS、DPSS、SRB など）は、追加的な機能を提供するために専用のインターフェイスを実装しているが、利用できるのは API だけであり、基本的なプロトコルでは利用できない。したがって、こうしたシステム間に相互運用性があるかどうかは疑問である。このために、グリッドストレージコミュニティは散在的なものとなった。異なるストレージシステムにアクセスしようとするユーザーは、複数のプロトコ

ルや API の使用を余儀なくされる上、こうしたストレージシステム間でデータを効率的に転送するのは容易ではない。

そこで、グリッド環境で安全かつ効率的なデータ転送を可能にする GridFTP と呼ばれる共通のデータ転送およびアクセスプロトコルを提案する。標準 FTP プロトコルの拡張版であるこのプロトコルは、現在使用されている各種のグリッドストレージシステムが備えている機能を含んだ包括的な機能を提供する。我々が FTP プロトコルを選択した理由は、それがインターネットでバルクデータ転送に使用されている最も一般的なプロトコルであり、http、DPSS、HPSS、SRB などの既存のデータ転送手法の中で、グリッドアプリケーションのニーズにもっとも見合うものと考えたからである。GridFTP プロトコルには次のような機能がある。

- ・グリッドセキュリティインフラストラクチャ (GSI) および Kerberos のサポート
- ・データ転送のサードパーティによる管理
- ・並行データ転送 (2つのネットワークエンドポイント間の複数の TCP ストリーム)
- ・ストライプ型データ転送 (送信側では m 個のネットワークエンドポイント、受信側では n 個のネットワークエンドポイント間の 1 つまたは複数の TCP ストリーム。m と n の数が異なる場合も含む)
- ・部分ファイル転送
- ・TCP バッファ/ウィンドウサイズの手動/自動コントロール
- ・信頼性の高い再スタート可能なデータ転送
- ・統合型の装置

2.2. 動機

グリッドコミュニティでは、すでいくつかのストレージシステムが使用されている。これらのストレージシステムは、大規模なデータセットの格納やアクセスに固有のニーズを満たすために開発された。その一つ一つは個別の要件を解決し、個別のサービスをクライアントに提供している。

たとえば、ある種のストレージシステム (DPSS や HPSS) は、データへの高度なアクセスや並行データ転送ストリームの活用、パフォーマンス向上のための複数のサーバへのストライピングを実現している。また他のシステム (DFS) では、大規模な処理量に対応し、サーバの負荷をバランス良く分散するためにデータセットの複製およびローカルキャッシング機能を利用している。SRB システムは異種のデータ集合を接続し、一様なクライアントインターフェイス

をこれらのリポジトリに提供している。また、ストレージシステム内のデータの識別や指定に使用するメタデータを提供している。その他のシステム (HDF5) には、データの構造に着目し、多種多様なストレージシステムから構造化データへのアクセスをサポートするクライアントを提供するものもある。

残念ながら、これらのストレージシステムの大半は互換性がなく、またデータにアクセスするためのプロトコルもほとんど発行されていない。このためデータにアクセスするには個々のクライアントライブラリを使用する必要がある。これが事実上グリッドアプリケーションで利用可能なデータセット間の仕切りとなっている。アプリケーションが異なるストレージシステムに格納されているデータにアクセスするには、ストレージシステムのサブセットのみを使用するか、各種ストレージシステムからデータを取得する複数の方式を用いるかのいずれかを選ぶ必要がある。FTP は一般にオプションとして利用できるが、拡張機能をもたない多くのグリッドアプリケーションにとっては実行可能な選択肢とはいえない。

このような相互に互換性のないストレージシステムプロトコルによる仕切りを取り去る1つの解決策として、階層化されたクライアントまたはゲートウェイを構築する方法が考えられる。1つのインターフェイスをユーザーに提供しつつ、要求を各種のストレージシステムプロトコルやクライアントライブラリコールに変換できるというものである。この方式は新しいプロトコルを採用するためのサポートを必要としないため、既存のストレージシステムプロバイダにとっては魅力的である。しかし次のような大きなデメリットがあることも事実だ。

- ・パフォーマンス：階層化クライアントと、ストレージシステム固有のクライアントライブラリやプロトコルとの間で損失の大きい変換が必要になること。また、1つのストレージシステムから他方へデータセットを効率よく転送するのは容易ではないこと。
- ・複雑性：多数のストレージシステムをサポートするクライアントやゲートウェイを構築し、維持管理するのは手間のかかる作業であること。個々のストレージシステムの発展に合わせて最新の状態に維持するのは困難であること。また、C/C++、Java、Perl、Python、Shellなどの複数のクライアント言語への対応ニーズとあいまって、いっそう作業が煩雑になること。

そこで、ストレージプロバイダとユーザーの双方が、これらの異なるすべてのシステム間に共通レベルの相互運用性、つまり共通だが拡張性のあるデータ転送プロトコルを確保することによって、相互にメリットを得ることができるようにする。ストレージプロバイダは、データがどのクライアントでも利用できるため、幅広いユーザー層を獲得できるであろう。ストレージ

ジューザーは多種多様な幅広いストレージシステムやデータにアクセスできるようになる。またこれらのメリットは、階層化クライアントやゲートウェイ方式特有のパフォーマンスや複雑性の問題を発生させることなく享受することが可能だ。

2.3. 用語解説

- ・ 並行転送：複数の TCP ストリームを使用する 2 つのネットワークエンドポイント間におけるデータ転送。
- ・ ストライプ型転送：送信側では m 個のネットワークエンドポイント、受信側では n 個のネットワークエンドポイント間のデータ転送。マルチホームホストや複数のホスト（クラスター）の形をとる。
- ・ データノード：ストライプ型転送で、データノードが SPAS コマンドで返された、または SPOR コマンドで送信されたネットワークエンドポイントの 1 つ。
- ・ DTP（データ転送プロセス）：データ転送プロセスはデータ接続の確立と管理を実行。DTP はパッシブまたはアクティブとすることが可能。
- ・ PI：プロトコルインタープリタ。ユーザーとプロトコルのサーバ側は、ユーザーPI とサーバPI に異なる機能を実装。
- ・ 機能：指定された機能一式をサポートしていることを示すサーバからの応答。RFC 2389 に準拠。
- ・ オプション：サーバに対して代替動作を定義するコマンド。RFC 2389 に準拠。

3. 拡張機能

3.1. 要約

ここでは RFC 959 に対する機能拡張について述べる。これらの拡張は、コマンド、オプション、機能、新規モードで構成されている。

3.2. コマンド

3.2.1. Striped Passive (SPAS)

この拡張機能は、データの各ストライプについてデータソケットリスナーのベクトルを確立するために使用される。並行データ転送拡張との対話を簡素化するため、SPAS は、データがコントロール接続によって用意されたファイルスペース上に格納されるとき、そのコントロール

接続でのみ実行されなければならない。SPAS コマンドは、転送コマンドを受信したらすぐに接続を開始するのではなく、FTP サーバにデータポート（デフォルトのデータポートではない）を「聴き取」らせ、1 つまたは複数のデータ接続を待つように要求する。このコマンドに対する応答には、サーバが聴き取っているホストやポートアドレスのリストが含まれる。このコマンドは、つねに拡張ブロックモードとともに使用しなければならない。

構文

SPAS コマンドの構文は次のとおり。

```
spas = "SPAS" <CRLF>
```

応答

サーバ PI は、229 応答を使って SPAS コマンドに応答し、リモートサーバ DTP またはユーザー DTP が接続するためのホストポートストリングのリストを提供する。

```
spas-response = "229-Entering Striped Passive Mode" CRLF
```

```
1*(<SP> host-port CRLF)
```

```
229 End
```

ホストポートのフォーマットは次のとおり。

```
h1, h2, h3, h4, p1, p2
```

これは 32 ビットの IP アドレスと、16 ビットの TCP ポートアドレスの結合を表す。h1~h4 までは文字列の代わりに数字を表すドットで区切られ送信される IPv4 IP アドレスの 4 つのフィールドを意味する。h1 は IP アドレスの上位 8 ビット。p1 は TCP ポートの上位 8 ビット。IP アドレスを形成するには、h1.h2.h3.h4 とする。また TCP ポートを決定するには、 $p1*256 + p2$ とする。

コマンドが正常に構文解析されてもサーバ DTP が SPAS 要求を処理できない場合は、PASV コマンドと同じエラー応答を返すものとする。

SPAS 用の OPTS

この SPAS 仕様にはオプションがないため、OPTS コマンドは定義されていない。

3.2.2. Striped Data Port (SPOR)

この拡張機能は、ストライプされたサードパーティの転送実行用 SPAS コマンドを補完するために使用される。並行データ転送拡張との対話を簡素化するため、SPOR は、データがサードパーティ転送のためにコントロール接続によって用意されたファイルスペース上から取得される時、そのコントロール接続でのみ実行しなければならない。このコマンドは、つねに拡張ブロックモードとともに使用しなければならない。

構文

SPOR コマンドの構文は次のとおり。

```
SPOR 1*(<SP> <host-port>) <CRLF>
```

コマンド構造におけるホストポートのシーケンスは、SPAS コマンドに回答するホストポートに一致しなければならない。

応答

サーバ PI は、ftp 仕様で述べた PORT コマンドと同じ応答セットを使って SPOR コマンドに回答する。

SPOR 用の OPTS

この SPOR 仕様にはオプションがないため、OPTS コマンドは定義されていない。

実装の注意点：

FTP プロトコルはマルチラインの応答を定義するが、マルチラインのコマンドは定義しない。我々はこのモデルを維持しようと試みたが、その結果、SPOR コマンドは、そのコマンドを表すために極めて長い文字列が必要であることが分かった。実装ではこの点に着目し、対処方法を準備しなくてはならない。

3.2.3. Extended Retrieve (ERET)

Extended Retrieve 拡張は、サーバ上の追加的処理として取得を行う要求に対して使用される。このコマンドは、RETR コマンドでサーバサイドのデータ圧縮その他の変更を行うための拡張機能である。このコマンドは RETR コマンドに対する OPTS の代わりに使用されるもので、レイテンシーを重視するアプリケーションでサーバサイド処理を 1 往復（サーバには 2 つではなく

1つのコマンドのみ送信)で実行することができる。

構文

ERET コマンドの構文は次のとおり。

```
ERET <SP> <module-name>="<module-params>" <SP> <resource-name><CRLF>
```

```
module-name ::= <unique string identifying the module>
```

```
module=params ::= <module specific opaque string>
```

モジュール引数は、二重引用符で囲む。引数に二重引用符が含まれる場合は、それらをバックスラッシュ (\) でエスケープしなければならない。

また、引数にバックスラッシュが含まれる場合は、それらを (二重) バックスラッシュ (\) でエスケープしなければならない。

リソース名は、モジュールで必要としない場合は省略可能である。

コマンドは構文解析され、モジュール名トークンとの一致にもとづいてモジュールが選択される。このモジュールには逐一モジュール引数が渡され、解析と実行が行われる。

この仕様のすべてのインプリメンテーションでは、次の部分ファイル転送 ERET モジュールを実装する必要がある。

```
ERET <SP> <retrieve-mode> <SP> <filename>
```

```
retrieve-mode ::= P <SP> <offset> <SP> <size>
```

```
offset ::= 64 bit integer
```

```
size ::= 64 bit integer
```

ここで指定するオフセットはファイルのオフセットであり、MODE E ヘッダに指定されたオフセット (ワイヤでの転送におけるオフセット) に関連するものではない。

応答

ERET コマンドに対する応答は、RETR コマンドについて RFC 959 ごとく実行するものとする。

また、指定されたモジュールが認識されない場合、501 が返されなければならない。テキストには利用可能なモジュールの一覧が記載される必要がある。また選択したモジュールが引数を解析できない場合、502 が返されなければならない。テキストには適切な構文が記載される必要がある。

オプション

この ERET 仕様にはオプションがないため、OPTS コマンドは定義されていない。

ERET 用の代替構文

この仕様の前バージョンでは、ERET コマンド用の代替構文が提案されている。この構文は広く実装されている。このプロトコルを実装しているサーバでは、移行を支援するために旧構文を選択することもできる。しかし、新しいモジュールの場合は、いずれも上に述べたモジュールを使って実装しなければならない。

ERET コマンド用の代替フォーマットは次のとおり。

```
ERET <SP> <retrieve-mode> <SP> <filename>
retrieve-mode ::= P <SP> <offset> <SP> <size>
offset ::= 64 bit
size ::= 64 bit integer
```

Extended Retrieve モード

部分取得モード (P) : ファイルのセクションは、データサーバから取得される。セクションは開始オフセットとエクステントサイズ引数によって定義される。

3.2.4. Extended Store (ESTO)

Extended Store 拡張は、サーバ上の追加的処理として格納を行う要求に対して使用される。

構文

ESTO コマンドの構文は次のとおり。

```
ESTO <SP> <module-name>="<module-params>" <SP> <resourcename><
CRLF>
module-name ::= <unique string identifying the module>
module=params ::= <module specific opaque string>
```

モジュール引数は、二重引用符で囲む。引数に二重引用符が含まれる場合は、それらをバックスラッシュ (¥) でエスケープしなければならない。また、引数にバックスラッシュが含まれる場合は、それらを (二重) バックスラッシュ (¥¥) でエスケープしなければならない。

リソース名は、モジュールで必要としない場合は省略することもできる。

コマンドは構文解析され、モジュール名トークンとの一致にもとづいてモジュールを選択する。このモジュールには逐一モジュール引数が渡され、解析と実行が行われる。

この仕様のすべてのインプリメンテーションでは、次の部分ファイル転送 EST0 モジュールを実装する必要がある。

```
EST0 <SP> PFT="<offset>,<length>" <filename>
```

```
offset ::= string representation of a positive 64 bit integer
```

```
length ::= string representation of a positive 64 bit integer
```

ここで指定するオフセットはファイルのオフセットであり、MODE E ヘッダに指定されたオフセット（ワイヤでの転送におけるオフセット）に関連するものではない。

応答

EST0 コマンドに対する応答は、STOR コマンドについて RFC 959 ごとに実行するものとする。また、指定されたモジュールが認識されない場合、501 が返されなければならない。テキストには利用可能なモジュールの一覧が記載される必要がある。また選択したモジュールが引数を解析できない場合、502 が返されなければならない。テキストには適切な構文が記載される必要がある。

オプション

この EST0 仕様にはオプションがないため、OPTS コマンドは定義されていない。

EST0 用の代替構文

この仕様の前バージョンでは、EST0 コマンド用の代替構文が提案されている。この構文は広く実装されている。このプロトコルを実装しているサーバでは、移行を支援するために旧構文を選択することもできる。しかし、新しいモジュールの場合は、いずれも上に述べたモジュールを使って実装しなければならない。

EST0 コマンド用の代替フォーマットは次のとおり。

```
EST0 <SP> <store-mode> <filename> <CRLF>
```

```
store-mode ::= A <
```

```
offset ::= 64 bit Integer
```

格納モード

調整化格納(A) : ファイルのデータは、ファイルのブロックを格納する前にファイルポインタに追加されたオフセットとともに格納される。拡張ブロックモードでは、この値は拡張ブロックヘッダにあるオフセットに追加され、正または負の値をとる。ブロック、圧縮、ストリームの各モードでは、オフセットはデータの開始のために0の暗黙のオフセットに追加される。

3.2.5. Set Buffer Size (SBUF)

この拡張は、その後のデータ接続に必要な TCP バッファサイズを値にセットするためにクライアント機能を追加する。これはサーバ固有のコマンド (SITE RBUFSIZE、SITE RETRBUFSIZE、SITE RBUFSZ、SITE SBUFSIZE、SITE SBUFSZ、SITE BUFSIZE) に取って代わるものである。

構文

SBUF コマンドの構文は次のとおり。

```
sbuf = SBUF <SP> <buffer-size>  
buffer-size ::= <number>
```

バッファサイズの値は、TCP バッファサイズをバイト数で表したものである。TCP ウィンドウサイズはサーバごとに設定する必要がある。

応答コード

サーバPI がバッファサイズステータスを要求通りの値にセットできる場合、サーバPI は 200 を返す。注 : サーバが SBUF を受け入れたとしても、実際にデータ接続が確立したときにはエラーが発生する可能性がある。

3.2.6. オートネゴシエートバッファサイズ (ABUF)

この拡張は、アルゴリズムを引用して TCP バッファサイズを決定およびセットするものである。ここでは特定のアルゴリズムは定義されていないが、プロトコルには任意にアルゴリズムを追加するためのサポート機能が用意されている。追加されたどのアルゴリズムも、利用可能なモジュールを表示するように定義された関連のある FEAT 応答と、モジュールごとに引数構文を示す関連のある HELP 応答をもつ必要がある。

構文

ABUF コマンドの構文は次のとおりです。

```
ABUF <SP> <module-name>=<module-params><CRLF>
module-name ::= <unique string identifying the module>
module=params ::= <module specific opaque string>
```

モジュール引数は、二重引用符で囲みます。引数に二重引用符が含まれる場合は、それらをバックスラッシュ (\) でエスケープしなければならない。また、引数にバックスラッシュが含まれる場合は、それらを (二重) バックスラッシュ (\) でエスケープしなければならない。

コマンドは構文解析され、モジュール名トークンとの一致にもとづいてモジュールが選択される。このモジュールには逐一モジュール引数が渡され、解析と実行が行われる。

応答コード

サーバPI がバッファサイズステータスを計算した値にセットできる場合、サーバPI は 200 を返し、そのバッファサイズセットをテキストに含める必要がある。アルゴリズムが特定されない場合、501 が返されなければならない。テキストには利用可能なアルゴリズムの一覧が記載される必要がある。また選択したアルゴリズムが引数を解析できない場合、502 が返されなければならない。注：サーバが ABUF を受け入れたとしても、実際にデータ接続が確立したときにはエラーが発生する可能性がある。

3.2.7. データチャネル認証 (DCAU)

この拡張は、FTP データチャネルで実行される認証タイプを指定する方法を提供するものである。この拡張は、コントロール接続が RFC 2228 セキュリティ拡張を使用して認証された場合のみ使用することができる。

構文

DCAU コマンドの構文は次のとおりです。

```
DCAU <SP> <authentication-mode> <CRLF>
authentication-mode ::= <no-authentication>
| <authenticate-with-self>
```

```
| <authenticate-with-subject>
no-authentication ::= N
authenticate-with-self ::= A
authenticate-with-subject ::= S <subject-name>
subject-name ::= string
```

認証モード

- No authentication (N)

データ接続の確立に関して認証ハンドシェイクは実行されない。

- Self authentication (A)

データチャンネルにセキュリティプロトコル固有の認証が使用される。リモートデータ接続の ID はコントロール接続に対して認証されたユーザーの ID と同じになる。

- Subject-name authentication (S)

データチャンネルにセキュリティプロトコル固有の認証が使用される。リモートデータ接続の ID は、供給されたサブジェクト名の文字列と一致しなければならない。

デフォルトのデータチャンネルの認証モードは、RFC 2228 で承認されている FTP セッションについては A を使用する。セキュリティハンドシェイクが失敗した場合、サーバはエラー応答 432 (データチャンネル認証の失敗を表す) を返すものとする。

3.3. 機能

RFC 2389 は、FEAT および OPTS コマンドの追加機能を提供することで、機能セットのネゴシエーションに対応している。以下の新しい機能名は、ここに示した機能セットを実装する場合、FTP サーバの FEAT への応答に追加されることになるものである。

- PARALLEL : サーバは MODE E をサポート。また、転送のために複数の TCP 接続を受け入れ、開始することが可能。
- MODE-E-RESTART : サーバは本書で述べている MODE E、リスタートマーカー、リスタートセマンティクスをサポート。
- MODE-E-PERF : サーバは本書で述べている MODE E、パフォーマンスマーカーをサポート。

- ・ STRIPING : サーバは本書で述べている SPOR および SPAS コマンド、RETR オプション、拡張ブロックモードをサポート。
- ・ ESTO : サーバは本書で述べている ESTO コマンドを実装。
- ・ ERET : サーバは本書で述べている ERET コマンドを実装。
- ・ SBUF : サーバは本書で述べている SBUF コマンドを実装。
- ・ ABUF : サーバは本書で述べている ABUF コマンドを実装。
- ・ DCAU : サーバは本書で述べている ABUF コマンドを実装。コントロールチャネルの確立に RFC 2228 認証が使用されている場合、データチャネルはデフォルトで認証されるという要件を含む。

ESTO、ERET、ABUF のようなモジュール選択を可能にする機能は、利用可能なモジュールを一覧表示する HELP 応答を実装する必要がある。

3.4. 拡張ブロックモード

上に述べたようなストライプ化された並行データ転送方式では、アウトオブシーケンスなデータ配信やデータ接続ごとの部分データ転送のサポートに、拡張転送モードを必要とする。ここで説明する拡張ブロックモードとは、ブロックモードヘッダを拡張することによって、これらのブロックだけでなく大きなブロックやデータ終端の同期化をサポートできるようにするものである。クライアントは転送コマンドが送信される前にコントロールチャネル上で次のコマンドを送信することにより、拡張ブロックモードを使用したいという要求を出す。

```
MODE <SP> E <CRLF>
```

拡張ブロックヘッダの構造については次節で述べる。

3.4.1. 拡張ブロックヘッダ

```
+-----+-----+-----+
| Descriptor | Byte Count | Offset Count |
|
+-----+-----+-----+
```

記述子コードは記述子バイトにビットフラグで示される。6 個のコードが割り当てられており、

各コード数はバイトにおいて対応するビットの十進値で表される。これらのビットの詳しい使用方法については、セクション 2.4.2 を参照。

コード：意味

128：このブロックはレコードの終端（EOR）を表す（レガシー）。

64：このブロックはデータ数の終端（EODC）を含む。EODC は受信されなければならないデータ終端（EOD）マーカー数（下のビット 8 を参照）を表す。

32：データブロックにおける疑わしいエラーを表す。

16：データブロックはリスタートマーカーを表す（レガシー）。

8：このブロックは、このリンクのデータ終端（EOD）マーカーを表す。

4：送信側はデータ接続を終了する。

これはブロックモード記述子に類似した形でモデル化されているが、ブロックモードと拡張ブロックモードは完全に独立型のモード（データチャンネルプロトコル）であり、相互の関連はない。このエンコーディングでは、特定のブロックについて複数の記述子コード条件が存在する可能性がある。たとえば、64（EODC）、8（EOD）、4（CLOSE）は、すべて同時にセットできる。しかし、意味のないエンコーディングもある。たとえば、コード 16 は 128 にとっては意味がない。これは、リスタートマーカーがレコードの終端であるはずがないためである。何が適切な組み合わせであるかは実装者に判断を委ねることにする。インプリメンテーションでは、解釈できないとのフラグが立った場合に、エラーを生成しなければならない。拡張ブロックモードデータチャンネルには、データストリームの未知の番号が現れた場合、ファイル終端の検出を適切に処理するいくつかの追加的なプロトコルが加えられている。

- ・ 所定のデータチャンネルにこれ以上データが送信されないとき、送信側は最後のブロックをマークするか、または最後のブロックの後に、そのデータチャンネルの拡張ブロックヘッダに EOD ビット（8）セットを含めて長さが 0 のブロックを送信する。
- ・ EOD を受信した後は、その後の転送に使用するためにデータ接続をキャッシュすることができる。データ接続が終了することを通知するために、送信側は送信された最後のメッセージのヘッダに、終了ビット（4）をセットする。
- ・ 送信側は、データを受信するすべてのサーバに EOF メッセージを送信することにより、ファイルの終端を通知する。EOF メッセージフォーマットは以下のとおり。

1 つの例を挙げてこの仕組みを説明する。送信側は複数のファイルを同じ受信側に送信し、接続の再確立で発生するオーバーヘッドを回避しようとする。受信側はポートを聴き取り、送信側は A、B、C の 3 つのデータ接続を開く。最初の転送では 3 つのすべてのデータチャンネルが使用され、EOD はそれぞれについて送信され、3 つの内の 1 つの EODC が送信される。CLOSE は送信されないため、3 つのすべてのチャンネルを開いたままにしておくことができる。いずれかの端末はそれらを任意に終了するように選択できるが、この例ではキャッシュされる（開いた状態を維持する）ものとする。二番目の転送では、A と B の 2 つのチャンネルのみ使用される。2 つの内の一方の EODC が送信される（2 つのチャンネルについてだが 1 つのみ送信可能）。EOD は A と B について送信される。CLOSE はチャンネル B について送信し、接続を終了する必要がある。これは、競合状態を避けるためである。A と B について EOD と 2 つの内の EODC を送信し、終了しなかったとする。このとき B は著しい輻輳状態となり、EOD はしばらくの間破棄される。ここで、A と C を用いた新しい転送を開始する。極めて小容量のデータを送信し、C に関して EOD を送信すると、これが B の前に到達する。CLOSE はこの競合状態を回避する。

3.4.2. 拡張ブロック EODC ヘッダ（記述子でコード 64 セット）

```
+-----+-----/-----+-----/-----+
| Descriptor | unused | EOD count expected |
| 8 bits | 64 bits | 64 bits |
+-----+-----/-----+-----/-----+
```

EOF 記述子。EOF ヘッダ記述子は、上に述べた正規のデータメッセージヘッダと同じ定義をもっている。

予定 EOD 数。この 64 ビットのフィールドは、ファイルを受信するサーバで確立されるデータ接続の総数を表す。この数値は、データのすべてを受信したことを確認するために、受信側で使用される。受信した EOD メッセージ数が「予定 EOD 数」で示した数値と等しい場合、受信側はファイルの終端を検出する。

開いているすべてのデータ接続について、EOD を待つだけでは十分ではない。受信側は、追加的なデータ接続の転送中に、開いているすべてのデータ接続に関する EOD メッセージを読むことが可能である。受信側がファイルの終端に達したと仮定することになっていた場合、転送

中のデータ接続を受信し損なう場合もある。マルチノード転送では、それぞれの受信ノードは、任意に選択されたデータチャネルについて、正確に 1 個の EODC 応答を受信しなければならない。EODC 数をどのように統合するかは、実装の詳細として残されている。

3.4.3. 拡張ブロックモードにおける EOF の処理

ストライプまたは並行モードにある場合、それぞれの SPAS 固有のポート（ストライプ）について正確に 1 個の EOF を受信する。EOF ブロックを送信する前に、各 SPOR ポートについて任意の接続数を受け入れるためには、拡張ブロックモードのホストを準備しなければならない。

3.5. オプション

3.5.1. RETR に対するオプション

ここで述べるオプションは、ストライプおよび転送パラレルズム情報をサーバ DTP に伝える手段を提供するものである。RETR コマンドでは、クライアント FTP が、サーバ DTP での使用を希望するパラレルズムおよびストライピングモードを指定する場合がある。取得動作が拡張ブロックモードで行われる場合、サーバ DTP のみがこれらのオプションを使用する。これらのオプションは RFC 2389 拡張として実装される。

RETR OPTS のフォーマットは次によって指定される。

```
retr-opts = "OPTS" <SP> "RETR" [<SP> option-list] CRLF
option-list = [ layout-opts ";" ] [ parallel-opts ";" ]
layout-opts = "StripeLayout=Partitioned"
| "StripeLayout=Blocked;BlockSize=" <block-size>
parallel-opts = "Parallelism=" <starting-parallelism> ","
<minimum-parallelism> ","
<maximum-parallelism>
block-size ::= <number>
starting-parallelism ::= <number>
minimum-parallelism ::= <number>
maximum-parallelism ::= <number>
```

3.5.1.1. レイアウトオプション

データファイルのセクションを適切な宛先ストライプに送信する場合、ソースデータノードはレイアウトオプションを使用する。実装には次のようなさまざまなストライプレイアウト引数を使用される。

パーティション化

パーティション化されたデータレイアウトは、データが宛先データノードに均等に分配されたものを指す。それぞれのデータノードに関してはデータの連続的な1つのセクションのみ格納される。データノードは、ここでは SPOR コマンドに記述された1つのホストポートとして定義されている。

ブロック化

ブロック化されたデータレイアウトは、データがラウンドロビン方式で宛先データノードに分配されたものを指す。データの分配は SPOR コマンドに記述されたホストポート仕様の順序にもとづいて配列される。ブロックサイズとは分配されるブロックのサイズを定義したものである。

3.5.1.2. パラレルizmオプション

ソースデータノードで、それぞれの宛先データノードに対して確立できる並列データ接続数を管理するために使用するのがパラレルizmオプションである。この拡張オプションを使うことで、固定レベルのパラレルizmを提供し、ホスト/ネットワーク接続に対するパラレルizmをある範囲内に適合させることができる。開始パラレルizmオプションが設定されると、サーバDTP はそれぞれの宛先データノードに対して開始パラレルizm接続を確立する。最小パラレルizmオプションが設定されると、サーバは宛先データノードごとの並列接続数を、この値まで減少させることができる。また、最大パラレルizmオプションが設定されると、サーバは宛先データノードごとの並列接続数を、最大でこの値まで拡張することができる。

応答

OPT コマンドに対する応答は、RFC 2389 に定義されている。通常の正常なコマンドには 200 という応答が戻される。永続的なエラーには 501 が、また一時的なエラーまたは構成にもとづく

エラーについては 451 が戻される。

コマンドが認識されない場合は、500 または 502 が返される。要求したソケット数が多すぎる場合、サーバは 501 による応答を選択することができる。

4. 宣言的仕様

4.1. 最小実装

本書で述べている拡張機能は、安全かつ高速で、信頼性、柔軟性、拡張性に優れたデータアクセスおよび伝送方法を提供することを目的としている。しかし、すべてのアプリケーションがこうしたすべての機能を必要としているわけではないため、「グリッドの一部」として利用できることが望ましいと考える。つまり、最小実装では、ここに述べた拡張をまったく必要としないことも事実である。最小実装に関しては、我々は RFC959 に RFC 2228 セキュリティ拡張を追加したものを提案している。クリアテキストによるパスワードはもはや受け入れられていない。詳細は以下のとおり。

Per RFC 959:

TYPE: ASCII Non-print

MODE: Stream

STRUCTURE: File, Record

COMMANDS: USER, QUIT, PORT, TYPE, MODE, STRU,

COMMANDS: RETR, STOR, NOOP (these commands with default values

転送引数のデフォルト値は次のとおり。

TYPE: ASCII Non-print

MODE: Stream

STRU: File

すべてのホストは、標準のデフォルトとして上の値を受け入れなければならない。

Per RFC 2228:

COMMANDS: AUTH , ADAT, MIC, CONF, ENC

4.2. 推奨実装

グリッドのあらゆる利点を取り入れ、十分に活用するために、完全な機能実装に役立つ以下の項目を推薦する。なおこれらのコマンド、モード、機能等の中には、実装に活かされたことが

ほとんどない、単に適用には向かないという理由で異議が唱えられているものもある。

RFC 959、ファイル転送プロトコル (FTP) 、J. Postel、R. Reynolds (1985 年 10 月)

GridFTP で使用するコマンド

USER	PASS	ACCT	CWD	CDUP	QUIT
REIN	PORT	PASV	TYPE	MODE	RETR
STOR	STOU	APPE	ALLO	REST	RNFR
RNTO	ABOR	DELE	RMD	MKD	PWD
LIST	NLST	SITE	SYST	STAT	HELP
NOOP					

GridFTP で使用する機能

タイプ : ASCII、イメージ

モード : ストリーム、EBlock

構造 : ファイル構造

RFC 2228、FTP セキュリティ拡張、Horowitz、M. and S. Lunt (1997 年 10 月)

GridFTP で使用するコマンド

AUTH

ADAT

MIC

CONF

ENC

GridFTP で使用する機能

GSSAPI 認証

RFC 2389、ファイル転送プロトコルの機能ネゴシエーション方式 P. Hethmon、R. Elz (1998 年 8 月)

GridFTP で使用するコマンド

FEAT

OPTS

FTP 拡張、R. Elz, P. Hethmon (2000 年 9 月)

GridFTP で使用するコマンド

SIZE

GridFTP で使用する機能

ストリームモード転送のリスタート

5. セキュリティ要件

セキュリティはグリッドの重要な要件の 1 つであり、また RFC 959 で定義されているように、FTP が今日受け入れられない根拠にもなっている。GridFTP は当初より安全性を意識して設計され、事実その努力のもとに研究が進められてきた。anonymous (アノニマス) FTP を維持し、USER や PASS コマンドをサポートしながらも、その使用、とくに PASS の使用をやめるよう極力勧告してきた。ここでは、GSI および Kerberos バインディングとともに、RFC 2228 に定義されている GSS API 拡張を組み込んでいる。

6. 既知の問題

6.1. EBLOCK モードにおける一方向性のデータ転送

現在 MODE E (EBLOCK) モードにある場合、PASV は STOR と、また PORT は RETR と対になっていなければならない。つまり、データチャンネルの接続方向は送信側 (RETR) から受信側 (STOR) になっている必要がある。ところがこれが機能しているとき、次の問題が発生する。

- 1) 現行の FTP プロトコルにはこの制限がない。
- 2) ファイアウォール：一部のファイアウォールをより簡単に経由して、ファイアウォールの後ろから接続できるような方向に変えることができる。
- 3) 部分的な get/put の組み合わせでは、現在 2 つのコントロールチャンネルの接続が必要。これは理想的とはいえない。

上記の制限が必要な理由は、逆の状況の場合ファイルの終端をはっきりと確認できないのでデータの消失につながる可能性があるためである。上に述べたように、送信側はそれぞれの接続と EOD カウントごとに EOD を送信する必要がある。受信側が接続している場合、n 個の接続を確立し、すべてのデータをこの n 個の接続について送信し、それぞれの接続と n 個の送信数

をもつ EOD ごとに EOD を送信しながら、受信側は別の接続を転送する。この接続ではまったく EOD は受信されず、このため EOF が正しく判定されることはない。

一看するとこの問題はすぐに解決できそうに見える。しかし、ストリームの任意の数とタイミング、マルチノード転送、キャッシュされたデータ接続、受信側のデータレイアウト、受信側ではどのようなホストあるいはいくつのホストがポートに接続しているかは分からないという事実を考えると、極めて難しい問題といえる。

6.2. PASV/SPAS および STOR/RETR 間の配列依存

ひと言でいえば、RFC 959 に定義されているように、GET または PUT 動作における次のようなイベントの順序を意味する。

1. 転送の一端でポートを受信
2. 他の一端でそのポートへの接続を確立
3. 他の一端にファイルを STOR（書き込み）するように指示。ここではファイル名は STOR コマンドに対する引数
4. 一端にファイルを RETR（読み込み）するように指示。ここではファイル名は RETR に対する引数

このイベントの順序は、不要なこともあるが、ある例では重要な制限となる。この制限とは、接続は転送されるファイルが識別される前に接続を確立しなければならないという点である。

この制限によって、ファイル名をもとに判定されるいかなる接続も回避される。ファイル名をもとにした判定機能を導入するには、負荷分散とファイルの内部移動を透過的に行うために、大規模なインストールを必要とすることになる。データ接続が確立される前に URL をもつフロントエンドサーバがある場合、フロントエンドはデータ接続を提供する最適なホストを決定するだろう。たとえば次のような場合である。

URL に指定されたホストがメンテナンスのために停止し、そのファイルが別のホストに移されている場合。データ接続を確立するために、フロントエンドは新しいホストに転送する。

複数のバックエンドサーバで、ファイルへの集中的なアクセスが発生している場合。フロントエンドは現在の負荷をもとに、この要求をどのサーバで処理するか選択する。

この問題に対しては、次のようないくつかの解決策が提案されている。

PASV コマンドに新しい応答を定義する必要がある。この応答は「遅延 IP/ポート」となる。

この応答は、STOR/RETR コマンドの最初の間答として、IP/ポート情報が返されることを意味する。次に、STOR/RETR コマンドを受信すると、どのホストでデータ接続を確立するかを、提供されたファイル名/URL にもとづいて決定することができる。この情報が利用可能となったことで、接続が確立し、この後の通常の STOR/RETR 動作が続行可能となる。OPT や SITE コマンドなどを用いて、この機能のオン、オフを切り替える。

どの組み合わせにおいても、PORT/PASV および STOR/RETR となるようなステートフルマシンを再定義する。ただし順番はない。現在スタートフルマシンは、直ちに転送を開始できることを（データ接続はすでに存在していなければならない）STOR/RETR コマンドが認知している、という状態にある。しかしステートフルマシンで、「転送開始 OK」の状態が定義され、PORT/PASV および STOR/RETR それぞれを受信することによってその状態に到達するように再定義された場合、もはや順番による制限はなくなる。

Pre Transfer (PRET) コマンドと呼ばれる新しいコマンドを導入する（検討中の新しいコマンドの詳細については本節のこの後の説明を参照してください）。このコマンドは、他のコマンドが発行される前に、1 つの転送について任意の状態情報を設定できるというものである。このオプションには、ファイル名以外の状態情報が提供されるというメリットがある。ファイルサイズなどはオプションの一例である。しかし、このシステムは今日利用可能とは考えられていない情報ももたらす。一方、これには追加的な状態を必要とし、RFC 959 状態マシンを複雑にするというデメリットがある。PRET コマンドがなければ、標準の RFC 959 ステートフルマシンを使用する。

6.3. コマンドのパイプライン化と E ブロックデータチャネルの再利用

データチャネルの再利用に加え、複数の RETR/ERET/STOR/ESTO コマンドを発行する際の効率を最大限に高める方法として、コマンド発行のパイプライン化がある。たとえば、1 つの RETR のデータが引き続きサーバから送信されている場合、クライアントは別の RETR コマンドを発行できるというものである。理論的にはサーバはデータチャネルのパイプを完全に維持することが可能である。最初の RETR のデータ送信を完了すると、直ちに次の RETR のデータ送信を開始する。またこのとき同時に、最初の RETR に対するコントロールチャネル応答を送信する。

ここでの課題は、受信側で最初の RETR のデータがどこで終端しどこで二番目が開始されるかを判断できるかどうか、確認することである。複数のデータチャネルがある場合は、複雑に

なることは言うまでもない。

6.4. ディスクリソース管理のサポート

利用可能なディスクスペースの確認と確保、高度な予約方法、指定した最小期間ファイルを維持する要求 (pinning) といった、ディスクリソース管理のニーズに対応するプロトコルのサポートは必要であろうか。今のところこれらは不要と考えている。もしあるとすれば、よりハイレベルなサービスがディスクリソース管理サービスと対話し、スペースの割り当てやファイルのピン、ステージ化などを行えるようになるだろう。そして転送サービス (GridFTP) は、これが1つに集約されることで、ファイルを移動する方向に進むと考えられる。

7. 付録 1: リスタート

一般に、ブロックヘッダ経由で渡される不透明リスタートマーカは、拡張ブロックモードでは使用すべきではない。この機能はブロックモードとの後方互換のために用意されているものである。これに代わって次のようなフォームを使い、宛先サーバからコントロール接続経由で拡張データマーカ応答を送信する必要がある。

```
extended-mark-response = "111" <SP> "Range Marker" <SP>
<byte-ranges-list>
byte-ranges-list = <byte-range> [ *(", " <byte-range>) ]
byte-range = <start-offset> "-" <end-offset>
start-offset ::= <number>
end-offset ::= <number>
```

マーカのバイトレンジは、データサーバによってディスクに格納されたバイト範囲の差分セットである。完全なリスタートマーカは、111 応答でクライアントが受信したすべてのバイトレンジを連結したものである。

クライアントは、転送再開のために REST コマンドをサーバに送信するとき、いくつかのレンジ応答で受信した近接するレンジをいくつも組み合わせることもできる。

たとえば、クライアントは応答を受信すると、

```
111 Range Marker 0-29
111 Range Marker 30-89
```

を送信する可能性がある。同じように、

REST 0-29, 30-89

REST 0-89

REST 30-59, 0-29, 60-89

これらの 90 バイトを受信した後、転送再開のために上記を送信する。

サーバでは、複数の後続のレンジマーカーで所定のデータレンジが受信されたことを示すこともでき、クライアントではこれを処理できなければならない。

111 Range Marker 30-59

111 Range Marker 0-89

上記は次の内容と同じである。

111 Range Marker 30-59

111 Range Marker 0-29, 60-89

同じように、クライアントがリスタートマーカーをまったく処理していない場合は、リスタートで冗長情報を送信することもできる。

8. 付録 II : パフォーマンスのモニタリング

拡張ブロックモード転送のパフォーマンスを監視するために、コントロールチャンネルに追加的な予備応答を送信することもできる。この応答のフォームは次のとおり。

extended-perf-response =

"112-Perf Marker" CRLF

<SP> "Timestamp:" <SP> <timestamp> CRLF

<SP> "Stripe Index:" <SP> <stripe-index> CRLF

<SP> "Stripe Bytes Transferred:" <SP> <byte count> CRLF

<SP> "Total Stripe Count:" <SP> <stripe count> CRLF

"112 End" CRLF

timestamp = <number> ["." <digit>]

stripe-index = <number>

byte count = <number>

stripe count = <number>

すべての perf-line ファクトは、所定のタイムスタンプにおける転送の瞬間の状態を表す。ファクトの意味は以下のとおり。

タイムスタンプ - サーバがパフォーマンス情報を計算したときの時間。エポックデート (00:00:00 協定世界時、1970年1月1日) 以降、秒で表される。

ストライプインデックス - このマーカが属するインデックス (レンジ 0~n、n は転送の STOR 側にあるストライプ数を表す)。

転送されたストライプ数 - このストライプで受信されているバイト数。

総ストライプ数 - この転送に含まれているストライプ (ネットワークエンドポイントのペア) の総数。

転送開始時間は、perf マーカーで「転送されたストライプバイト」を 0 にセットすることにより指定できる。ストライプごとの最初のマーカのみ、そのストライプの開始時間の指定に使用することができる。その後続く「転送されたストライプバイト」を 0 にセットしたマーカは、いずれもその間にデータ転送がないことを表す。

サーバは、それぞれのストライプごとに「開始」マーカと、最終 perf マーカーを送信しなければならない。これは、「転送されたストライプバイト」を、そのストライプの総転送サイズにセットしたマーカを意味する。

9. 付録 III : 開発中に検討された RFC

以下に掲載した RFC は、GridFTP プロトコルに機能を搭載する上で検討されたものである。ある意味では FTP プロトコルに関連した他の RFC もあるが、それらは単に情報として役立ったか、あるいはファイル転送プロトコルの一形態ではあっても RFC 959 に定義されたプロトコルとはまったく異なるものであった。我々はグリッドアプリケーションのニーズに最適な機能セットを選択し、必要に応じて拡張に追加していく方針である。

RFC0959 ファイル転送プロトコル Oct-85

コメント : GridFTP プロトコルの最も基本とするプロトコル。一般的な機能 (STOR、RETR、PORT、PASV) や、定義されていてもあまり実装には使われていない機能を選択した (第三者の転送)。

RFC2773 KEA と SKIPJACK を使用した暗号化技術 Feb-00

コメント：SSL レベルで Globus 内の既存の暗号化方式を採用した。

RFC2640 FTP の国際化 Jul-99

コメント：FTP における UniCode 文字列の使用方法。これは優れたアイデアであり、将来の追加に備え検討中であるが、今のところ決定的な要件ではない。

RFC2428 IPv6 および NAT のための FTP 拡張 Sep-98

コメント：新しいコマンド（EPRT と EPSV）を定義することで、任意のアドレッシングスキームに対応。IPV4 と IPV6 が定義されているが、他のスキームの追加も容易である。これは優れたアイデアであり、将来の追加に備え検討中であるが、今のところ決定的な要件ではない。実装されれば PORT、PASV、SPAS、SPOR などの他のコマンドを、EPRT と EPSV に透過的にマッピングする可能性がある。

RFC2389 FTP の機能ネゴシエーション方式 Aug-98

コメント：RFC に定義されているような FEAT および OPTS 拡張が、GridFTP プロトコルに組み込まれている。

RFC2228 FTP セキュリティ拡張 Oct-97

コメント：RFC に定義されているような AUTH、ADATA、PROT、PBSZ、CCC、MIC、CONF、ENC、6yz 応答が、GridFTP プロトコルに組み込まれている。

RFC1639 FTP Operation Over Big Address Records (FOOBAR) Jun-94

RFC1545 FTP Operation Over Big Address Records (FOOBAR) Nov-93

コメント：RFC 2428 に定義されている方式は、IPV4 以外のアドレッシングスキームの処理に適していると考えられていた。

RFC1068 バックグラウンドファイル転送プログラム (BFTP) Aug-88

コメント：グリッドの主要なサービスとなる、GridFTP プロトコルにもとづく信頼性の高いフ

ファイル転送サービスを検討している。この RFC の資料は、そうしたサービスの設計に有益な情報を提供する。

10. GGF 著作権表示

“Copyright (C) Global Grid Forum (2004). All Rights Reserved.

上記著作権表示とこの段落が全ての複製文書や派生的な研究に含まれている限りにおいて、いかなる種類の制限を課すことなく、一部または全部を対象に、この文書およびこの文書の翻訳物を複製し他者に供することができる。また、その内容に対するコメントや説明、あるいはその実装を支援する派生的な研究物を作成、複製、発行、配布することができる。

しかしながら、この文書自体は、GGF 文書プロセスに定義された著作権の手続きに従わなければならない。あるいはそれを英語以外の言語に翻訳する必要がある場合において、グリッドの推薦内容を発展させる上で必要とされる場合を除き、著作権表示や GGF その他組織への照会情報を削除するなど、いかなる方法によっても変更することはできない。

上記に規定する限定的な許可は永続的であり、GGF またはその後継者や譲受人によって無効となることはない。

この文書およびこの中に記載された情報は「無保証」で提供される。グローバルグリッドフォーラムは、ここに含まれる情報の利用が、商業利用や特定の目的に合致することのいかなる権利や黙示的な保証も侵害しないという保証を含め、明示的、黙示的にすべての保証を拒否する。

11. GGF 知的所有権表示

GGF は、いかなる知的所有権、実装に関わるものとして主張され得るその他の権利、本書に述べられている技術の使用、あるいはこのような権利の行使が可能または不可能とされるライセンスの限度についても、その有効性や適用範囲に関してはいかなる立場もとらない。また、そうした権利を特定するために取り組んできた事実はない。出版のために利用できる権利の主張のコピーや利用されるライセンスのいかなる保証、実装者またはこの仕様のユーザーによるそうした所有権の使用に対して、一般的なライセンスまたは許可を得るための手続きの結果は、GGF 事務局より入手することができる。

GGF は、すべての利害関係者に対して、この推薦内容を研究するために必要な技術をカバーできる著作権や特許、特許の応用、その他の所有権に対し、注意を向けるよう勧める。詳細は

GGF エグゼクティブディレクタ宛に送付ください。

12. 参考

- [1] Postel, J. and Reynolds, J. 『ファイル転送プロトコル (FTP) 』、STD 9、RFC 959 (1985 年 10 月)
- [2] Hethmon, P. and Elz, R. 『ファイル転送プロトコルの機能ネゴシエーション方式』、RFC 2389 (1998 年 8 月)
- [3] Horowitz, M. and Lunt, S. 『FTP セキュリティの拡張』、RFC 2228 (1997 年 10 月)
- [4] Elz, R. and Hethom, P. 『FTP 拡張』、IETF ドラフト (2000 年 9 月)