

1.CSSJ ドライバプロトコル (CTIP 1.0)

1.1 基本的なデータ型

1.1.1 数値とバイト列

CSSJドライバプロトコルはバイナリ形式を用います。2バイト以上のデータの固まりは、必ず上位のバイトから送られます(ビッグエンディアン)。データの基本単位は次の4通りです。

byte	符号つき1バイト整数
short	符号つき2バイト整数
int	符号付4バイト整数
data	符号なしバイト列

1.1.2 文字列

文字列は次の形式で表します。

```
string := STRLEN STRING
```

ただし

STRLEN(short)
文字列のバイト数。

STRING(data)
文字列。バイト数はSTRLENの通り。

文字列のエンコーディングは、最初の接続の際にクライアントが決定します (4.を参照してください)。

1.1.3 チャンク

クライアント-サーバー間でやり取りされるデータは適当な大きさの「チャンク」を単位として送られます。各チャンクは次の形式のバイト列です。

```
chunk := PAYLOAD TYPE [TYPE_SPECIFIC]
```

ただし

PAYLOAD(int)

チャンクの大きさで、PAYLOAD 自身の4バイトを除いた全体のバイト数です。

TYPE(byte)

チャンクの種類です。

[TYPE_SPECIFIC]

この部分はデータ型により異なります。

1.2 クライアントからサーバーに送られるチャンク

1.2.1 プロパティチャンク (TYPE=1)

サーバーの処理方法を指定するために、名前と値のペアで表現されるプロパティです。

```
property := PAYLOAD (byte)1 NAME VALUE
```

ただし

NAME(string)

プロパティの名前

VALUE(string)

プロパティの値

1.2.2 リソースチャンク (TYPE=2)

画像などのドキュメントの付随するデータの開始を示します。

```
resource := PAYLOAD (byte)2 URI TYPE ENCODING
```

ただし

URI(string)

リソースの仮想URI

TYPE(string)

リソースのMIME型

ENCODING(string)

リソースのキャラクタ・エンコーディング

URI,TYPE,ENCODINGの指定しないものについては空の文字列を送信できます。

1.2.3 メインチャンク (TYPE=3)

ドキュメントの本体の開始を示します。

```
main := PAYLOAD (byte)3 URI TYPE ENCODING
```

ただし

URI(string)

ドキュメントの仮想URI

TYPE(string)

ドキュメントのMIME型

ENCODING(string)

ドキュメントのキャラクタ・エンコーディング

URI,TYPE,ENCODINGの指定しないものについては空の文字列を送信できます。

1.2.4 データチャンク (TYPE=4)

resourceまたはmainチャンクに続くデータで、複数回送ることができます。

```
client_data := PAYLOAD (byte)4 DATA
```

ただし

DATA(data)

DATA の大きさには制限があり、最大で1024バイトです。従ってチャンク全体の大きさは最大1029バイトです。

1.2.5 終了

クライアントからのデータの送信の終了を表します(これはチャンクではありません)。

```
end := (int)0
```

1.3 サーバーからクライアントに送られるチャンク

1.3.1 追加チャンク (TYPE=1)

クライアント側にブロックを追加します。

```
add := PAYLOAD (byte)1
```

1.3.2 挿入チャンク (TYPE=2)

クライアント側にブロックを挿入します。

```
insert := PAYLOAD (byte)2 ANCHOR_ID
```

ただし

ANCHOR_ID(int)

挿入位置の直後にあるブロックの番号。

1.3.3 メッセージチャンク (TYPE=3)

サーバーからクライアントに送られるエラー情報等のメッセージです。

```
message := PAYLOAD (byte)3 MESSAGE_TYPE MESSAGE
```

ただし

MESSAGE_TYPE(byte)

メッセージのタイプ

MESSAGE(string)

メッセージの内容

メッセージのタイプには以下の4種類があります。

1

警告

2

エラー

3

致命的なエラー

4

情報

1.3.4 データ (TYPE=4)

クライアント側のブロックにデータを追加します。

```
server_data := PAYLOAD (byte)4 BLOCK_ID PROGRESS DATA
```

ただし

BLOCK_ID(int)

データを追加するブロックのID

PROGRESS(int)

サーバー側で処理済のデータ(バイト数)

DATA(data)

1.4 接続

ドライバがサーバーにTCP接続した後、プロトコルのバージョンとエンコーディングをサーバーに知らせるために、次のメッセージを送ります。

```
"CTIP/1.0" " " ENCODING "\n"
```

ENCODINGはパラメーターあるいはメッセージの名前、値に使うキャラクタ・エンコーディング名です。CSSJサーバーを実行するJava環境がサポートするキャラクタ・エンコーディング名を使用できます。

例) UTF-8キャラクタ・エンコーディングを用いる場合、ドライバは次のメッセージを送ります。

```
"CTIP/1.0 UTF-8\n"
```

サーバーがプロトコルをサポートしない場合は、直ちに接続が切断されます。プロトコルをサポートする場合は、認証状態になります。

1.5 認証

接続後、クライアントはサーバーに認証情報を送る必要があります。認証情報はプロパティチャンクと同じ形式で、NAMEは"ctip.auth"です。VALUEは以下の形式です。

```
"PLAIN:" USER " " PASSWORD
```

USERはユーザー名で、PASSWORDはパスワードです。例えばユーザー名"user"、パスワード"password"で認証する場合は次のVALUEを用います。

```
"PLAIN: user password"
```

認証に成功した場合は、サーバーはメッセージチャンクと同じ形式で、MESSAGE_TYPEは4、MESSAGEは"OK"のデータを返します。それ以外の結果が返された場合は認証は失敗です。

1.6 プロパティとリソースの送信

認証に成功した後、クライアントはプロパティチャンクをサーバーに送ることができます。プロパティの一覧はCSSJサーバーのドキュメントを参照してください。

サーバーがアクセスできるリソースを指定するために、以下の特殊なNAMEを持つプロパティチャンクを送ることができます。

```
"ctip.include"
```

サーバーがアクセスできるリソースのURIパターン

"ctip.exclude"

サーバーのアクセスを除外するリソースのURIパターン

また、認証に成功した後はリソースチャンクを送ることができます。リソースチャンクの後にはそのリソースの内容のデータチャンクを複数送ることができ、次のリソースチャンクかプロパティチャンクまたはメインチャンクを送ることで前のリソースのデータを終了します。

1.7 ドキュメントの変換

ドキュメントの変換を開始する方法には2通りあり、メインチャンクが"ctip.main"というNAMEを持つプロパティチャンクを送る方法があります。

前者の方法では、メインチャンクを送った後、ドキュメントのデータを複数のデータチャンクとして送ります。一度メインチャンクを送った後は、データチャンクかセッションの終了を示す0(2.5参照)しか送ることはできません。

"ctip.main"プロパティチャンクの値はドキュメントのURIです。ドキュメントへはCSSJサーバーが直接アクセスします。"ctip.main"プロパティチャンクを送った後はセッションの終了を示す0(2.5参照)しか送ることはできません。

1.8 クライアント側でのデータの構築

クライアントがメインチャンクまたは"ctip.main"プロパティチャンクを送った直後からサーバーはドキュメントを変換した後のデータを送り返してきます。クライアントはメインチャンクを送った後にサーバーから変換後のデータを受け取るのと並行して(あるいはノンブロッキングI/Oを用いて反復して)、残りのドキュメントの内容をサーバーに送信できます。

サーバーは追加チャンク、挿入チャンク、メッセージチャンク、データチャンクを繰り返しクライアントに送ります。サーバー側からのデータの送信はサーバー側からのソケットの切断をもって終了します。

CSSJドライバプロトコルは、先頭から順に生成されないPDFのようなデータを連続して送ることができるように設計されています。クライアントはサーバーの指示に従って、クライアント側のディスク上あるいはメモリ空間にIDが割り付けられた「ブロック」の列を生成します。データチャンクはデータと一緒にブロックのID(BLOCK_ID)を含んでおり、クライアントはそのブロックにデータを追加します。

ブロックのIDは最初の値が0であるカウンタをクライアント側に持つことにより生成します。ブロックはIDをファイル名としたファイルで実現できますが、実際のCSSJドライバはもっと効率的な実装をしています。

1.8.1 ブロックの追加

サーバーから送られるデータのサイズが0でない場合は、最初に必ず追加チャンクが送られます。追加チャンクはクライアント側で現在のカウンタのIDを値とするブロックを列の末尾に生成します。その後、カウンタの値を1つ増加させます。

1.8.2 ブロックの挿入

挿入チャンクはクライアント側で現在のカウンタのIDを値とするブロックを、ANCHOR_IDで示されるブロックの直前に生成します。その後、カウンタの値を1つ増加させます。

1.8.3 データの追加

データチャンクを受け取ったクライアントはBLOCK_IDで示されるブロックの末尾に受け取ったデータを追加します。

1.8.4 データの構築

サーバーとの通信が終了した後、クライアントはブロックを先頭から順に結合し、完成されたデータとします。

1.9 ライセンス

Copyright (c) 2009 株式会社GNN

Apache License Version 2.0に基づいてライセンスされます。
あなたがこのファイルを使用するためには、本ライセンスに従わなければなりません。
本ライセンスのコピーは下記の場所から入手できます。

<http://www.apache.org/licenses/LICENSE-2.0>

適用される法律または書面での同意によって命じられない限り、
本ライセンスに基づいて頒布されるソフトウェアは、明示黙示を問わず、
いかなる保証も条件もなしに「現状のまま」頒布されます。
本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してください。

Copyright (c) 2009 GNN & Co.,Ltd.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.