CTIP バージョン 2.0.1 仕様

目次

1.概要

CTIPは、アプリケーションからドキュメント変換サーバーCopper PDFの機能を利用するための、ソケット通信プロトコルとして開発されました。 ライブラリのリンクという方法ではなく、ソケット通信により連携することにより、プログラミング言語に依存しない、２つのプログラムが物理的に離れていても連携できるというメリットがあります。

前バージョンのCTIP 1.0は2005年に開発されました。 CTIP 2.0はその後継版で、次の新しい機能を備えています。

- 一度のソケット接続で繰り返しドキュメント変換処理を実行する(KeepAlive)。
- メッセージコードによる詳細な処理経過情報の取得。
- サーバーのバージョンや機能に関する情報の取得。
- サーバーが必要としたリソースをクライアントに要求する。

1.1 バージョン2.0.1の追加機能

新たに、複数の変換結果を結合する機能が追加されました。

2.基本的なデータ型

2.1 数値とバイト列

CTIPはバイナリデータによって情報をやりとりします。 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。 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
: 符号つき4バイト整数

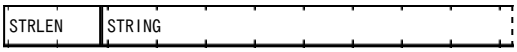
long
: 符号つき8バイト整数

data
: 符号なしバイト列

2.2 文字列

文字列は次の形式で表します。

string :=



ただし

STRLEN(short)
: 文字列のバイト数。

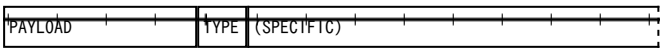
STRING(data)
: 文字列。バイト数はSTRLENの通り。

文字列のキャラクタ・エンコーディングは、接続の際にクライアントが決定します。

2.3 パケット

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パケットを単位として 送られます。各パケットは次の形式のバイト列です。

packet :=



ただし

PAYLOAD(int)
: パケットの大きさで、PAYLOAD自身の4バイトを除いた全体のバイト数です。

TYPE(byte)
: パケットの型です。

(SPECIFIC)
: この部分はパケットの型により異なります。

3.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キャラクタ・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/2.0" " " ENCODING "\n"
```

ENCODINGは以降の通信でstringに使うキャラクタ・エンコーディング名です。

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/2.0 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。

4.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。 認証情報は以下の形式です。

```
"PLAIN:" USER " " PASSWORD "\n"
```

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1行を送ります。

```
PLAIN: user password
```

認証に成功した場合は、サーバーは"OK \n"を返します。 失敗した場合は"NG \n"を返し、直ちに接続を切断します。

認証の完了後、クライアントからバイナリ形式のパケットを送出することにより、サーバー側に各種操作を要求することが出来ます。

5.パケットの一覧

以下は各パケットの概要と、状態遷移表です。 クライアントから送るパケットには"cXX"、 サーバーから送るパケットには"sXX"、 という名前を付けています。 XXはTYPEの部分の16進数表記です。

5.1 クライアントからサーバーに送られるパケット

TYPE	SPECIFIC	処理
c01	NAME(string) VALUE(string)	プロパティの送信
c02	URI(string) MIME_TYPE(string) ENCODING(string) LENGTH(long)	メインドキュメントの開始を通知
c03	URI(string)	メインドキュメントの変換を要求
c04	MODE(byte)	サーバーからリソースを要求するモードに切り替え
c05	MODE(byte)	複数の変換結果を後で結合するモードに切り替え
c11	DATA(data)	データ
c21	URI(string) MIME_TYPE(string) ENCODING(string) LENGTH(long)	リソースの開始を通知
c22	URI(string)	リソースの不存在を通知
c31	なし	データの終了
c32	MODE(byte)	中断
c33	なし	結合
c41	なし	リセット
c42	なし	切断
c51	URI(string)	サーバー情報の問い合わせ

5.2 サーバーからクライアントに送られるパケット

TYPE	SPECIFIC	処理
s01	URI(string) MIME_TYPE(string) ENCODING(string) LENGTH(long)	データの開始
s11	BLOCK_ID(int) DATA(data)	ブロックデータ
s12	なし	ブロックの追加
s13	ANCHOR_ID(int)	ブロックの挿入
s14	CODE(short) MESSAGE(string) ARGn(string)	メッセージ
s15	LENGTH(long)	メインドキュメントのバイト数
s16	READ(long)	メインドキュメントの読み込み済みバイト数
s17	DATA(data)	データ
s18	BLOCK_ID(int)	ブロックのクローズ
s21	URI(string)	リソースの要求
s31	なし	データの終了
s32	MODE(byte) CODE(short) MESSAGE(string) ARGn(string)	処理の中断

6.状態遷移表

横軸はパケットの種類、縦軸は状態、各マス目の数値は遷移先の状態です。 空欄の部分では、パケットが無視され、状態の変化はありません。

↓状態 | パケット→		c01	c02	c03	c04	c05	c11	c21	c22	c31	c32	c33	c41	c42	c51
0	初期状態	0	3/8[*1]	4	0	0		2					0	切断	1
1	サーバー情報取得														
2	リソース送信						2			0					
3	メインドキュメント送信						2			4					
4	サーバーからのリソース要求受信										4				
5	要求されたリソース存否判定							6	4						
6	要求されたリソース送信						6			4					
7	メインドキュメント変換結果受信										7	7			
8	メインドキュメントのパイプライン変換						8			7	7				

↓状態 | パケット→		s01	s11	s12	s13	s14	s15	s16	s17	s18	s21	s31	s32
0	初期状態												
1	サーバー情報取得								1			0	
2	リソース送信												
3	メインドキュメント送信												
4	サーバーからのリソース要求受信	4[*2]	4[*2]	4[*2]	4[*2]	4	4	4	4[*2]	4[*2]	5	0	0
5	要求されたリソース存否判定												
6	要求されたリソース送信												
7	メインドキュメント変換結果受信	7[*2]	7[*2]	7[*2]	7[*2]	7	7	7	7[*2]	7[*2]		0	0
8	メインドキュメントのパイプライン変換	8[*2]	8[*2]	8[*2]	8[*2]	8	8	8	8[*2]	8[*2]		0	0

[*1] c04によりサーバーからリソースを要求するモードになっている場合は3, それ以外は8に遷移します。

[*2] これらのパケットには送られる順序があります。 クライアント側でのデータの構築を参照してください。

7.各状態の詳細

7.1 初期状態

初期状態では、次のいずれかの操作を行うことが出来ます。

- プロパティの送信(c01)
- リソースの開始を通知(c21)
- サーバーからリソースを要求するモードに切り替え(c04)
- メインドキュメントの開始を通知(c02)
- メインドキュメントの変換を要求(c03)
- リセット(c41)
- 切断(c42)
- サーバー情報の問い合わせ(c51)

7.1.1 プロパティの送信

プロパティを送信(c01)することで、メインドキュメントの変換処理等の、以降のサーバーの動作を変える事が出来ます。 プロパティは名前と値の文字列のペアで、名前と値の組み合わせの意味はサーバーに依存します。

7.1.2 リソースの開始を通知

リソースの開始を通知(c21)することで、リソース送信状態に切り替わり、クライアント側からサーバーにデータを送ることが出来ます。 リソースは、メインドキュメントから参照されるデータです。例えば、HTML文書中のIMG要素から参照する画像等です。

7.1.3 サーバーからリソースを要求するモードに切り替え

サーバーからリソースを要求するモードに切り替える(c04)ことで、ドキュメントから参照されているリソースの送信要求をサーバーから受けられるようになります。 このモードは、ドキュメントの内容を解析しなければ必要なリソースが分からないような状況で有効です。

7.1.4 メインドキュメントの開始を通知または変換を要求

変換対象となるメインのドキュメントは、クライアントから送信する方法(c02)と、サーバー側から取得する方法(c03)の２種類があります。 前者の方法では、ドキュメントの送信と変換を並行して行うことが出来、後者の方法ではクライアントが指定したアドレスにサーバーがアクセスします。 後者の方法ではメインのドキュメントをリソースとして事前に送り、後でそのリソースをメインドキュメントとして使用することも出来ます。

ただし、サーバーからリソースを要求するモードでは、必要なリソースの要求とドキュメントの変換処理はクライアントから送られるドキュメントの全体を受信した後に始まります。 従って、ドキュメントの送信と変換は並行しては行われません。

7.1.5 複数の変換結果を結合する

通常は、メインドキュメントの開始(c02)から終了(c31)、あるいはメインドキュメントの変換を要求(c03)した時点で、 変換結果が最後までクライアントに送信されますが、複数の変換結果を結合するモードに切り替える(c05)ことで、 明示的に結合が要求(c33)されるまで最終的な変換結果を出さず、複数の変換があれば、その結果を結合するようにします。

例えば、PDFへの変換であれば通常は別のソースは別のPDFとして返されるのに対し、 結果を結合すると、複数のソースが連結された 1 つのPDFとして返されます。

7.1.6 サーバー側でメインドキュメントを取得する

サーバー側でメインドキュメントを取得する場合、クライアントはドキュメントのURIを送るだけです(c03)。

7.1.7 KeepAliveとリセット

ドキュメントの変換を同じ接続で続けて行うと、設定したプロパティと、送信したリソース（事前に送信したものと、サーバーの要求により送信したものも含む）は再利用されます。 そのため、同じ設定やリソースを共有する複数のドキュメントの変換が高速に行えます。 サーバーからリソースを要求するモードに切り替えていた場合は、そのモードも維持されます。

リセット(c41)を行うと、接続を維持したまま、全ての状態がクリアされ、接続直後と同じ状態に戻ります。

7.1.8 切断

クライアントが切断(c42)を要求すると、サーバー側からソケットが切断されます。

7.2 サーバー情報取得

サーバー情報の問い合わせ(c51)により、サーバー側のソフトウェアの種類やバージョンを知ることができます。

受け取ることができるデータの詳細はサーバー情報のURIと内容¶を参照してください。

サーバー情報の取得を要求した場合、サーバーからのデータは複数のデータパケット(s17)によって送られ、 データの終了パケット(s31)によって終了します。クライアント側でデータパケットのDATAを連結したものがサーバー情報です。

7.3 リソース送信

リソースの送信は、リソースの開始パケット(c21)によって開始し、 クライアントからのデータは複数のデータパケット(c11)によって送られ、 データの終了パケット(c31)によって終了します。 データパケットのDATAを連結したものがサーバー側に記録されるリソースとなります。

7.4 メインドキュメント送信

クライアントからメインドキュメントのデータを送る場合は、メインドキュメントの開始を通知(c02)し、データの送信を開始します。 複数のデータパケット(c11)によって送り、 データの終了パケット(c31)により終了します。 サーバーからリソースを要求するモードでない場合は、クライアントからのメインドキュメントのデータを送信している最中に、 サーバーがクライアントにデータを送ることがあります。

7.4.1 処理の中断

クライアントは、常にサーバー側に処理を中断するように要求することができます。処理の中断(c32)は生成済みのデータを出力して中断する方法と、即時中断する方法の２種類があります。 前者ではサーバー側はなるべくきりのよいところまで処理を続けます。 即時中断する場合は、文字どおり強制的にサーバー側の処理を中断します。 どちらの場合も、途中までのページまでしか読み込めないような中途半端なデータや、そもそも見ることが出来ない不正なデータが生成されてしまう可能性がありますが、その可能性は後者の方がずっと高くなります。 どの状態かは、サーバーからデータの終了(s31)が通知されるか、処理の中断(s32)のパケットのMODEにより判別することができます。

7.5 サーバーからのリソース要求受信

サーバーからリソースを要求するモードでは、クライアントがメインドキュメントのURIを送った後、 またはメインドキュメントのデータを送り終わった後に、サーバーがリソースの送信をクライアントに要求します(s21)。 リソースの要求はサーバー側の処理結果の送信に混じって行われますが、サーバーがリソースを要求して、 クライアントが要求したリソースを送り終えるまでは、サーバーが処理結果を送ることはありません。 クライアント側のリソース送信の手順は、リソース不存在の通知(c22)を除いては、事前にリソースを送る場合と同じです。

7.6 要求されたリソース存否判定

サーバーからリソースが要求された場合は、リソースを送信するか(c21)、リソースが存在しないことを通知(c22)することができます。

7.7 要求されたリソース送信

サーバーから要求されたリソースを送信する場合の手順は、事前にリソースを送信する場合と同じです。

7.8 メインドキュメント変換結果受信

サーバーがクライアントにデータを送信する方法は２通りあります。 １つ目はデータパケット(s17)による単純な方法で、クライアントはデータパケットのDATAを連結するだけです。

２つ目は断片化されたデータを構築する方法で、 サーバーはブロックの追加パケット、ブロックの挿入パケット、メッセージパケット、ブロックデータパケット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、サーバ側からのデータの終了(s31), 処理の中断(s32)のいずれかのパケットの送信をもって終了します。

どちらの方法かは、最初にクライアントに送られるパケットがデータパケット(s17)であるか、 ブロックの追加パケット(s12)であるかどうかで判別できます。

後者の場合、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。 データパケット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連番で生成します。

7.8.1 ブロックの追加(s12)

断片化されたデータをサーバーから送る場合は、最初に必ずブロックの追加パケットが送られます。 追加パケット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。

7.8.2 ブロックの挿入(s13)

ブロックの挿入パケット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。

7.8.3 ブロックデータ(s11)

ブロックデータパケット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DATAを追加します。

7.8.4 メッセージ(s14)

処理中の各種情報やエラーメッセージが送られます。 詳細はメッセージコードの説明を参照してください。

7.8.5 メインドキュメントの読み込み状況(s15, s16)

処理の進行状況を確認しやすくするため、メインドキュメントのサイズ(s15)と読み込み済みバイト数(s16)がクライアントに通知されます。 これらは通知されないこともありますし、通知のタイミングも不定期です。

7.8.6 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の列を順に結合し、完成されたデータとします。

7.9 メインドキュメントのパイプライン変換

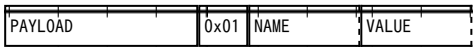
クライアントからメインドキュメントを送信(c02)する場合、サーバーからリソースを要求するモードでなければ、ドキュメントの送信と構築が並行して行われます。 このとき、クライアントはメインドキュメントの送信とメインドキュメント変換結果受信を同時に行う必要があります。 そのため、クライアントはノンブロッキングI/Oか、マルチスレッドのいずれかの手段で実装される必要があります。

8.クライアントからサーバーに送られるパケットの詳細

8.1 c01 プロパティの送信

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。

c01 :=



ただし

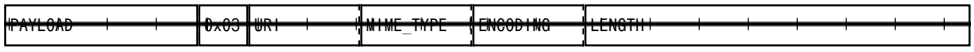
NAME(string)
: プロパティの名前

VALUE(string)
: プロパティの値

8.2 c02 メインドキュメントの開始を通知

メインドキュメントの本体の開始を通知します。

c02 :=



ただし

URI(string)
: メインドキュメントの仮想URI

MIME_TYPE(string)
: メインドキュメントのMIME型(サーバー側で自動判別する場合は空文字)

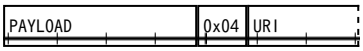
ENCODING(string)
: メインドキュメントのキャラクタ・エンコーディング(サーバー側で自動判別または不確定な場合は空文字)

LENGTH(long)
: メインドキュメントのファイルサイズ(不確定な場合は-1)

8.3 c03 メインドキュメントの変換を要求

サーバー側でメインドキュメントを取得して変換する処理を開始します。

c03 :=



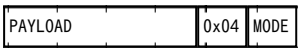
ただし

URI(string)
ドキュメントのURI

8.4 c04 サーバーからリソースを要求する

サーバーからリソースを要求するモードを変更します。

c04 :=



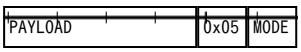
だたし

MODE(byte)
0=サーバーからリソースを要求しない, 1=サーバからリソースを要求する

8.5 c05 複数の出力結果を結合する

複数の出力結果を結合するモードを変更します。

c05 :=



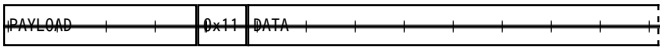
だたし

MODE(byte)
0=結果を返還の都度に返す, 1=結合(c33)が要求されたときだけ全体の結果を結合して返す

8.6 c11 データパケット

クライアントから送るデータの断片です。

c11 :=



だたし

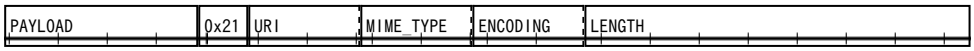
DATA(data)
データの断片です。

DATAの大きさには制限があり、最大で8192バイトです。従ってパケット全体の 大きさは最大8193バイトです。

8.7 c21 要求されたリソースの開始

要求されたリソースの開始を通知します。

c21 :=



ただし

URI(string)
リソースの仮想URI

MIME_TYPE(string)
リソースのMIME型(サーバー側で自動判別する場合は空文字)

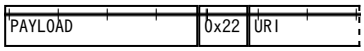
ENCODING(string)
リソースのキャラクタ・エンコーディング(サーバー側で自動判別または不確定な場合は空文字)

LENGTH(long)
リソースのファイルサイズ(不確定な場合は-1)

8.8 c22 要求されたリソースの不存在通知

要求されたリソースのが存在しなかったことを通知します。

c22 :=



ただし

URI(string)
リソースのURI

8.9 c31 データの終了

データの終了です。

c31 :=

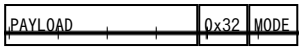
だたし

RESOURCE_ID(short)
リソースのID

8.10 c32 中断

処理を中断することを要求します。

c32 :=



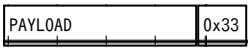
だたし

MODE(byte)
0=生成済みのデータを出力して中断, 1=即時中断

8.11 c33 結合

結果を結合します。

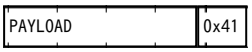
c33 :=



8.12 c41 リセット

設定済みのプロパティ、サーバーに送信済みのリソースをクリアします。

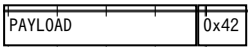
c41 :=



8.13 c42 切断

接続を切断します。

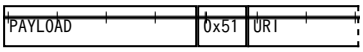
c42 :=



8.14 c51 サーバー情報の問い合わせ

サーバー情報の送信を要求します。

c51 :=



ただし

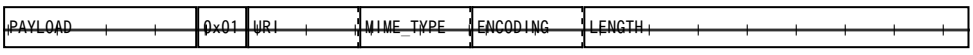
URI(string)
サーバー情報のURI

9.サーバーからクライアントに送られるパケットの詳細

9.1 s01 データの開始

データの開始をクライアントに通知します。

s01 :=



ただし

URI(string)
データの仮想URI

MIME_TYPE(string)
データのMIME型

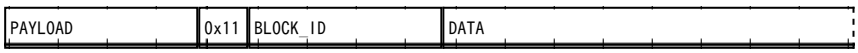
ENCODING(string)
データのキャラクタ・エンコーディング(不確定な場合は空文字)

LENGTH(long)
データのファイルサイズ(不確定な場合は-1)

9.2 s11 ブロックデータ

クライアント側のブロックにデータを追加します。

s11 :=



だたし

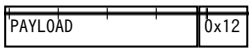
BLOCK_ID(int)
データを追加するブロックのID

DATA(data)
ブロックに追加するデータ

9.3 s12 ブロックの追加

クライアント側にブロックを追加します。

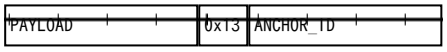
s12 :=



9.4 s13 ブロックの挿入

クライアント側にブロックを挿入します。

s13 :=



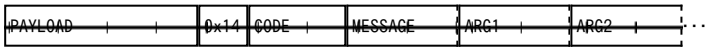
だたし

ANCHOR_ID(int)
挿入位置の直後にあるブロックの番号。

9.5 s14 メッセージ

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。

s14 :=



だたし

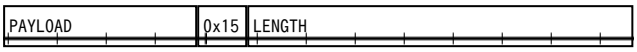
CODE(int)
メッセージコード

MESSAGE(string)
メッセージ

ARGn(string)
引数

9.6 s15 メインドキュメントのバイト数

s15 :=

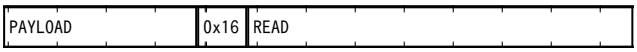


だたし

LENGTH("long")
サーバー側で検出した元文書のサイズ(バイト数)

9.7 s16 メインドキュメントの読み込み済みバイト数

s16 :=



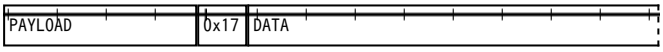
だたし

READ("long")
サーバー側で処理済のデータ(バイト数)

9.8 s17 データパケット

クライアント側にデータを送ります。

s17 :=



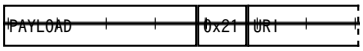
だたし

DATA(data)
送られたデータ

9.9 s21 リソースの要求

クライアントにリソースの送信を要求します。

s21 :=



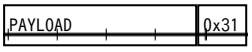
だたし

URI(string)
リソースのURI

9.10 s31 データの終了

結果の送信を終了します。

s31 :=



9.11 s32 中断

処理を中断します。

s32 :=

だたし

MODE(byte)
0=データが利用可能, 1=データを利用不可

CODE(int)
メッセージコード

MESSAGE(string)
メッセージ

ARGn(string)
引数

9.12 メッセージコード

16ビットのメッセージコードは、次の16進数表記によりクラス分けします。 Xの部分には0～Fの任意の値が入ります。

1XXX
処理情報。

2XXX
警告メッセージ。通常の運用でも発生する可能性のある軽微なエラーを示すもので、処理が続行されます。

3XXX
エラーメッセージ。アプリケーション等の問題によるエラーで、処理が中断されます。

4XXX
深刻なエラー。ドキュメント変換サーバー自体の問題によるもので、処理が中断されます。

X00X
CTIPインターフェースのためのコード。

X8XX～XFXX
ドキュメント変換サーバーに依存するコード。

CTIPインターフェースのメッセージコードは次のとおりです。

コード	値	説明
1001		abort等により、正常に処理が中断された。
2001	リソースのURI(string)	ドキュメントから参照されたリソースURIの形式の不正。
2002	ベースURI(string)	文書のベースURIの形式に不正がある。
3001	ドキュメントのURI(string)	メインドキュメントのURIの形式に不正がある。
3002	エラーメッセージ(string)	通信エラー。
4001	エラーメッセージ(string)	予期しないエラー。

10.ライセンス