HI.CommunicationEngine
HTTPサーバ
リファレンスマニュアル

株式会社 ルネサス北日本セミコンダクタ

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。

μITRON は、Micro Industrial TRON の略称です。TRON は、The Realtime Operating system Nucleus の略称です。

その他、本書で登場するシステム名、製品名は各社の登録商標または商標です。

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「HTTPサーバ」について説明します。
　HI.CommunicationEngine HTTPサーバは、TCP/IPマネージャを経由してネットワーク上のブラウザからの要求に答えるHTTPサーバとしての機能を提供します。
　このリファレンスマニュアルではHTTPサーバのサービスコールとその使い方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
- HI.CommunicationEngine 時刻／カレンダＡＰＩ リファレンスマニュアル
- 使用するファイルシステムのユーザーズマニュアル
- 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 HTTP サーバの機能概要

ハイパーテキスト転送プロトコル(HTTP ： Hyper Text Transfer Protocol)はネットワーク上につながれるパーソナルコンピュータ、携帯端末、携帯電話などの上で動作するブラウザからの要求（コンテンツの取得、送信など）と、HTTPサーバの応答に関するプロトコルです。

本製品はHTTP/1.1仕様に準拠したHTTPサーバです。HTTPの詳細は、RFC（：Request for Comment ）や市販の書籍などをご参照ください。本製品の基本機能は以下のとおりです。

- ファイルシステム API 仕様の提供により、様々なファイルシステムに対応することができます。サンプルとして、簡易 ROM ファイルシステムを提供します。
- クライアント（ブラウザソフト）からのリクエストである GET、POST、HEAD、OPTION、TRACE、PUT、DELETE メソッドをサポートします。使用するメソッドのみを選択して構築することができます。**だたし、PUT、DELETE メソッドはファイルシステムに合わせて、ユーザによるカスタマイズが必要です。**
- CGI（：Common Gateway Interface ）。
- システム CGI によるシステム情報の参照、設定。
- ユーザ認証機能。**サポートする認証方法は基本認証（ Base64 ）のみ。**

本HTTPサーバでは、一部のヘッダ情報に制限があります。詳細は「付録B」を参照してください。

1.2 HTTP サーバ構成

ユーザタスクでは、TCP/IP マネージャ、ファイルシステム、時刻・カレンダの初期化・サービスの開始を実施したあと、HTTP サーバの初期化・開始を行います。

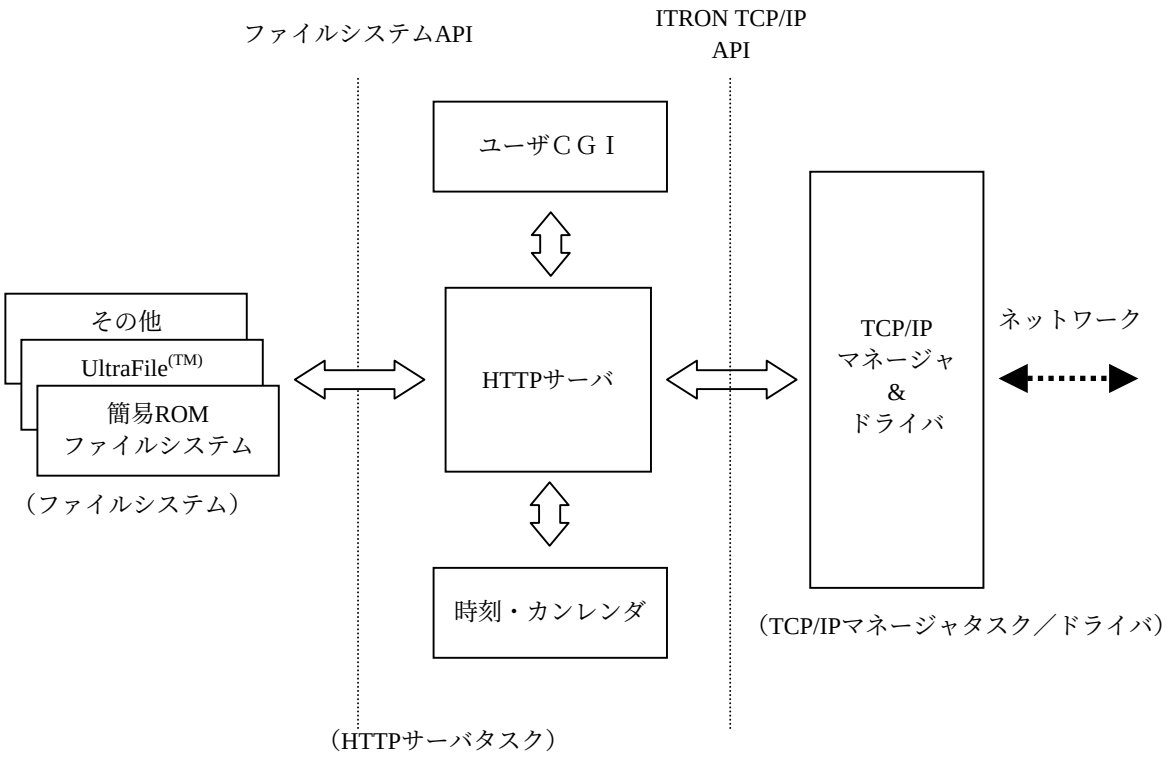

図1-1　HTTPサーバの構成

1.3 前提プログラム

本製品では、HI.CommunicationEngine基本セット、μITRON 準拠のＯＳの他に、以下の前提プログラムが必要となります。

・HI.CommunicationEngine 時刻／カンレンダAPI
・ファイルシステム（サンプルとして簡易ROMファイルシステムを提供します）

1.4 対応範囲

本製品は、HTTP/1.1 に準拠した HTTP サーバです。構築情報の変更により、使用しない機能の削除が可能です。

2. HTTP サーバ使用方法

2.1 HTTP サーバの準備

2.1.1 OS 情報の設定

HTTP サーバはタスクとして実行されます。また、HTTP サーバタスクの内部では、OS のイベントフラグと可変長メモリプールを使用します。

そのためタスク情報テーブル（httpOsConf）に「タスクの優先度」、「タスク ID」、「スタックサイズ」、「イベントフラグの ID」、「可変長メモリプール ID」、「メモリプールサイズ」といった OS 資源の生成情報を登録しなければなりません。タスク情報の登録についての詳細は、「HTTP サーバの構築マニュアル」を参照してください。

・ HTTPサーバタスクの優先度

HTTPサーバの優先度を設定します。

TCP/IPマネージャのタスク優先度より低く設定してください。

HTTPサーバタスクの優先度 ＜ TCP/IPマネージャタスクの優先度

TCP/IPマネージャの優先度より低い優先度に設定してください。TCP/IPマネージャの優先度より高い場合は、TCP/IPマネージャの動作は保証されません。また、HTTPサーバタスクより高い優先度のタスクが割り込んで実行されるとHTTPサーバの性能が低下するおそれがありますので、ご注意願います。

・ 可変長メモリプールのサイズ

同時に使用するメモリの最大値は

最大接続数 × 1028 バイト ＋ POSTメソッドのエンティティボディの最大値

となります。POSTメソッドのエンティティボディとは、<form>タグ内に記載されたテキストボックスやボタンの情報（<input type=xxxx>)です。作成したユーザCGIのコンテンツにより変化します。使用形態に応じて十分なメモリプール領域を確保してください。

メモリプールが不足した場合、そのリクエストに対するレスポンスは、"HTTP/1.1 503 Service Unavailable"（過負荷などの一時的なサーバーエラー）となります。

ご使用のOSが HI2000/3 の場合、タスク・可変長メモリプールの生成はOS構築にて行なわれます。そのため、タスク情報テーブル（httpOsConf）の「タスクID」、「イベントフラグID」、「可変長メモリプールID」以外の情報は使用されません。

2.1.2 TCP/IP マネージャの起動

HTTPサーバを起動する前に、TCP/IPマネージャのサービスを開始してください。詳細はTCP/IPマネージャのリファレンスマニュアルを参照してください。

HTTPサーバは、複数のクライアント（ブラウザ）と同時に接続されます。リクエストの処理が集中する場合、TCP通信端点の数が少ないとHTTPサーバの性能に影響するおそれがあります。推奨するTCP通信端点の数は、

同時にリクエストされる最大値[*1]×コンテンツあたりの接続数[*2] <= TCP通信端点

[*1] 想定されるご使用形態で動作保証したい最大リクエスト数

[*2] フレームや画像等を含むコンテンツでは、一般的に複数の接続を同時に使用します（ブラウザ側の仕様）。

となります。なお、サンプルプログラムのTCP通信端点の数は5個です。

HTTPサーバはTCP通信端点IDをTCP_CEPIDANY(0)指定で確保します。同一システム上で通信を行なうプログラムを動作させる場合、HTTPサーバが使用中のTCP通信端点IDと重複しない様にTCP_CEPIDANY(0)指定にて確保することを推奨します。

TCP受付口は１つ使用します。使用するTCP受付口IDは、TCP_REPIDANY(0)にて確保されます。

2.1.3 時刻／カンレンダの起動

HTTPサーバを起動する前に、時刻／カレンダ機能のサービスを開始してください。詳細は「時刻／カレンダAPIのリファレンスマニュアル」を参照ください。

2.1.4 ファイルシステムの準備

HTTPサーバを起動する前に、ファイルシステムのサービスを開始してください。ファイルシステムインタフェースとして、ファイルシステムAPI仕様（コールバック）を提供します。任意のファイルシステムと組み合わせてご使用できます。詳細は使用するファイルシステムのマニュアルを参照してください。

なお、サンプルプログラムとして、簡易ROMファイルシステムを提供します。

2.1.5 HTTP サーバの構築情報

HTTPサーバ起動前に、以下の構築情報を設定してください。詳細は「HTTPサーバの構築マニュアル」を参照してください。

表2-1 HTTPサーバの構築情報

分類	項目	内容
OS 関連	タスク生成情報	サービス開始時に生成される HTTP サーバタスクの生成情報を指定します。
	イベントフラグ生成情報	サービス開始時に生成されるイベントフラグの生成情報を指定します。
	可変長メモリプール生成情報	サービス開始時に生成される可変長メモリプールの生成情報を指定します。
TCP/IP 関連	ウィンドウバッファサイズ	TCP/IP にて使用する送受信用のウィンドウバッファのサイズを指定します。
HTTP サーバ関連	デフォルトコンテンツ名	URL=http://ホスト名/ を指定した場合のデフォルトコンテンツ名を指定します。
	最大コネクション数	一度に接続できる最大コネクション数を指定します。TCP/IP マネージャで指定した TCP 通信端点数を同じ、またはそれ以上の値を設定してください。
	ファイルタイプの登録	拡張子とメディアタイプを登録します。コンテンツの送信時に、content-type ヘッダフィールドに付加されます。
	最大ユーザ CGI の登録数	使用するユーザ CGI の数を設定します。
	システム CGI の On/Off	システム CGI の使用／未使用を設定します。
	ユーザ認証の On/Off	ユーザ認証の使用／未使用を設定します。
	認証用のユーザ登録	ユーザ認証用のユーザ／パスワードを登録を登録します。
	メソッドの On/Off	メソッドの使用／未使用を設定します。
	Realm キーワード設定	ユーザ認証用 Realm キーワードを任意の名称を設定します。
	動的な一部機能の有効設定	Put、Delete メソッド、ユーザ認証を動的に有効／無効設定する機能の使用／未使用を設定します。
	有効設定の初期値	Put、Delete メソッド、ユーザ認証の機能の有効／無効の初期値を設定します。
	持続型接続時間	Keep-Alive 指定による持続型接続でのタイムアウトを指定します。
	http サーバ名称の設定	http サーバ名称を設定します。

2.1.6 HTTP サーバの起動

初期化サービスコールはHTTPサーバの開始や停止する機能を提供します。

表2-2 初期化サービスコール

区分	サービスコール名称	サービスコールの機能
初期化サービスコール	HTTP_init	HTTP サーバを初期化する
	HTTP_start	HTTP サーバのサービスを開始する
	HTTP_stop	HTTP サーバのサービスを終了する

2.2 コンテンツの登録と表示

ネットワーク上のパーソナルコンピュータ、携帯端末、または携帯電話などの上で動作するブラウザソフトで、本HTTPサーバのURLを指定すると、HTTPサーバがファイルシステムAPI（コールバック）を介して、ファイルシステムにアクセスし、任意のコンテンツをブラウザに転送します。

コンテンツの登録は、任意のファイルシステムに従い行なってください。

2.3 ユーザ GGI

CGI機能の利用すると、動的コンテンツの生成ができます。以下にユーザCGI関連サービスコール、ＣＧＩ関連コールバックの一覧を示します。

表2-3 CGI関連サービスコール

区分	サービスコール名称	サービスコールの機能
CGI 関連サービスコール	HTTP_RegisterCGI	CGI を登録します
	HTTP_extractPsStr	コンテンツパラメータの解析
	HTTP_GetFileTypeNoByExt	コンテンツタイプの取得
	HTTP_SendChunkHeader	チャンク形式ヘッダー送信
	HTTP_SendChunkData	チャンク形式データ送信
	HTTP_SendChunkEnd	チャンク形式データ送信終了
	HTTP_ResponsTransfer	通常(一括)コンテンツの送信

表2-4 CGI関連コールバック

区分	コールバック名称	コールバックの機能
CGI 登録	_UserCGI	ユーザ CGI の登録を行なう。
ユーザ CGI 関数	(*usercgi)()	ユーザ CGI コールバック関数。

2.3.1 ユーザ CGI の登録

HTTP_start サービスコールを実行すると、HTTPサーバタスクは、初期化処理にてユーザCGI登録コールバック関数_UserCGI を呼び出します（サンプルソース http_user_cgi.c ）。

_UserCGIでは、HTTP_RegisterCGIサービスコールを使用して、ユーザCGIの登録を行ってください。登録はコンテンツ名、ユーザ認証の有無、CGI関数のアドレスを指定します。構築情報の「最大ユーザCGIの登録数」以上の登録はできません。

サンプルプログラム

```
void _UserCGI(void)
{
        W ercd;

        ercd = HTTP_RegisterCGI( "janken.cgi" , 0, (W (*)())HTTP_CGI_janken );
}
```

2.3.2 ユーザ CGI コールバック関数

ブラウザから登録されたユーザCGIのコンテンツ名を指定すると HTTP_RegisterCGI サービスコールで指定したユーザCGIコールバック関数が呼び出されます。CGI関数では、動的コンテンツ送信用のサービスコールでコンテンツを送信します。

2.3.3 動的コンテンツの送信

動的コンテンツ送信用のサービスコールでは、コンテンツのデータを一度に送る方法（通常のコンテンツ送信）とコンテンツを分割して送る方法（チャンク方式エンコーディング）を提供します。チャンクエン方式コーディングを使用することにより、少量のスタックでの送信が可能となります。

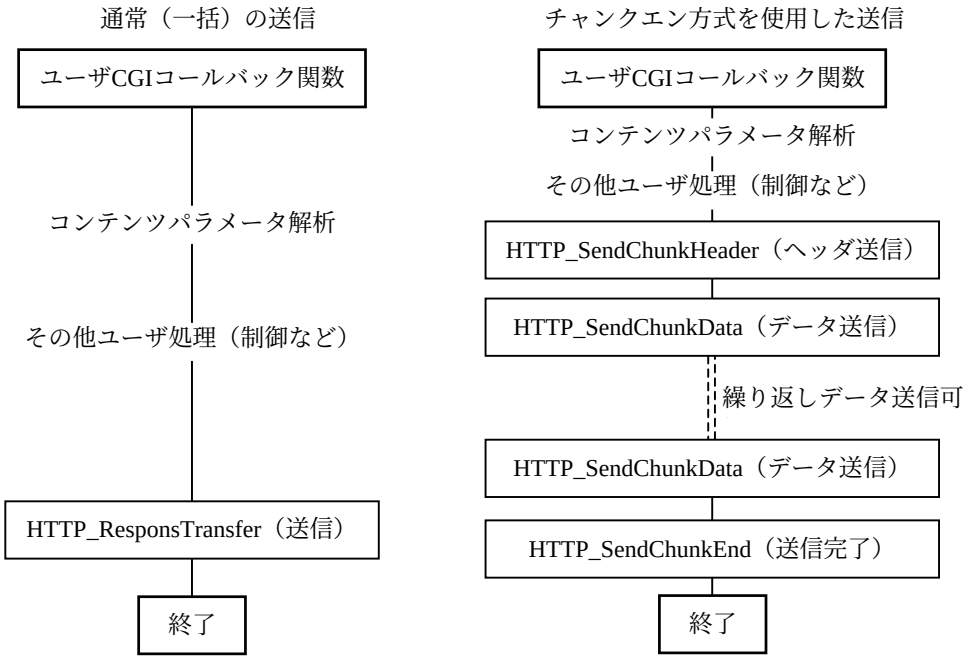

図2-1　動的コンテンツの送信

2.4 システム GGI

以下のシステムCGIが、予め登録されております。構築情報の「システムCGIのOn/Off」を変更することにより、本システムCGIの使用／未使用を選択することができます。

表2-5 システムGGI

項目	URL	内容
フレーム出力	http://ホスト名/admin.cgi	インデックスと各システム CGI をフレームで表示します。
インデックス	http://ホスト名/index.cgi	システム CGI のメニュー画面です。
ユーザ認証パスワード変更	http://ホスト名/passwd.cgi	構築時に登録した認証用ユーザのパスワード変更を行います。
コンテンツ一覧	http://ホスト名/content.cgi	コンテンツの一覧リストを出力します。ファイルシステムに依存する為、サンプルの簡易 ROM ファイルシステムのみ対応。他のファイルシステムでは動作しません。その場合、構築時に本システム CGI を未使用設定してください。
メソッドの有効／無効設定	http://ホスト名/method.cgi	Put、Delete メソッド、ユーザ認証を動的に有効／無効の変更を行います。
時刻／カレンダ設定	http://ホスト名/date.cgi	Put、Delete メソッド、ユーザ認証を動的に有効／無効の変更を行います。
ファイルタイプ一覧	http://ホスト名/filetype.cgi	構築時に登録したファイルタイプの一覧リストを出力します。

2.4.1　フレームとインデックス

URL：http://ホスト名/admin.cgi を実行するとフレーム（admin.cgi ） とインデックス（index.cgi ） が表示されます（図2-2）。使用したいシステムCGIをメニューから選択してください。
HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。

2.4.2　ユーザ認証パスワード変更

インデックスの "PASSWORD" を選択すると、構築時に登録した認証用ユーザのパスワードを変更することができます（図2-2）。
HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。

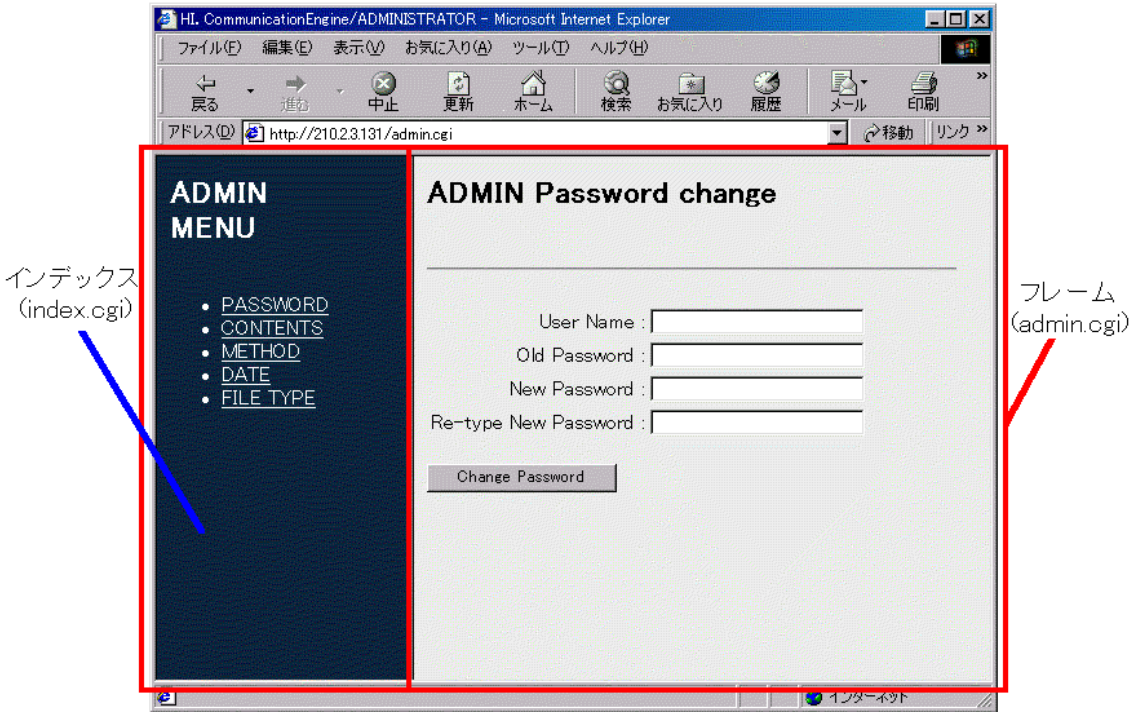

図2-2　ユーザ認証パスワード変更

2.4.3　　コンテンツ一覧

インデックスの "CONTENTS" を選択すると、コンテンツの一覧を表示します（図2-3）。
HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。登録したユーザCGIの一覧は出力しません。
なお、本システムCGIが使用できるのはサンプルの簡易ROMファイルシステム使用時のみです。

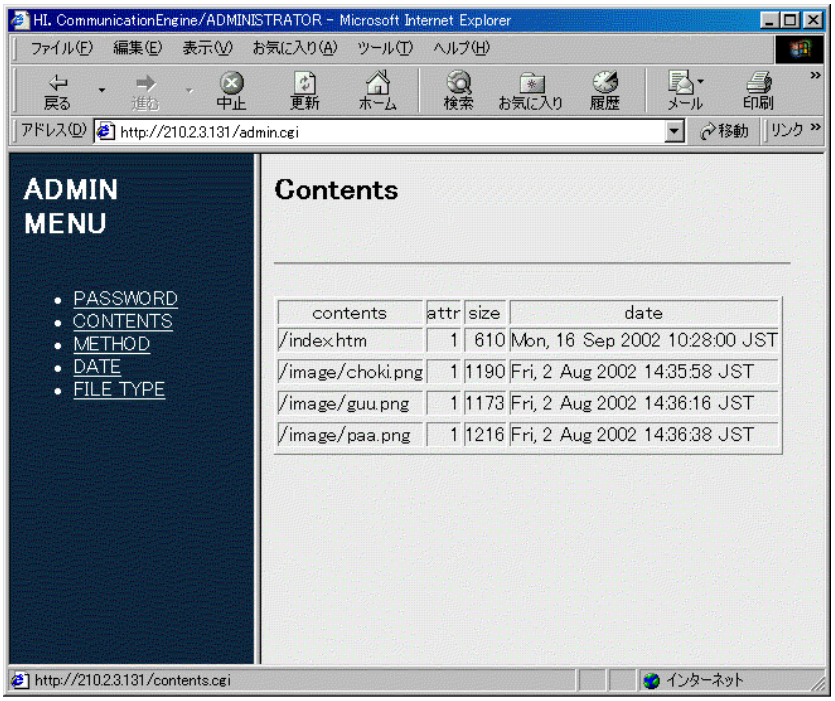

図2-3　コンテンツ一覧

2.4.4　メソッドの有効／無効設定

インデックスの "METHOD" を選択すると、メソッドの有効／無効設定を表示します（図2-4）。
HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。

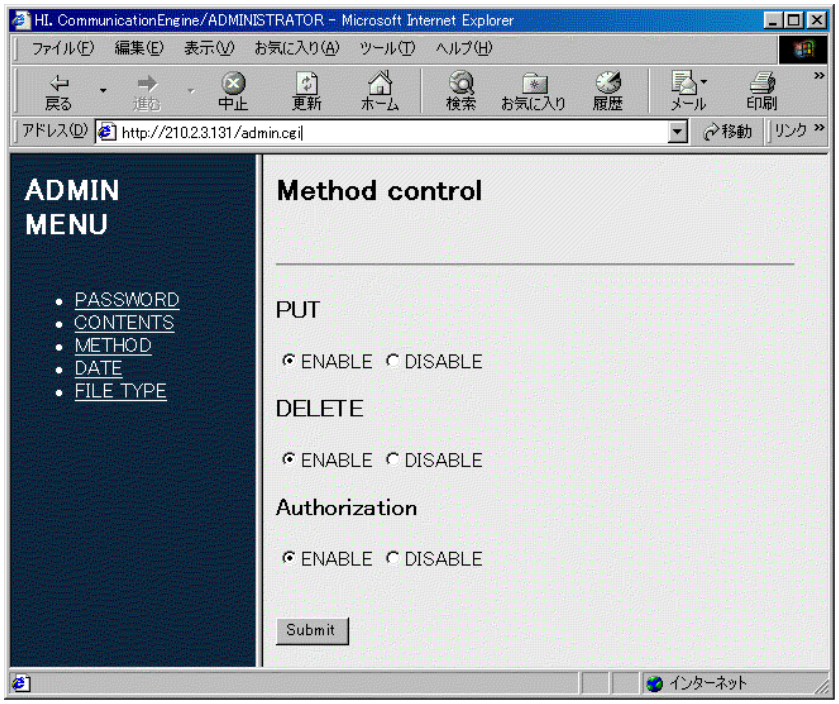

図2-4　メソッドの有効／無効設定

2.4.5　時刻／カレンダ設定

インデックスの "DATE" を選択すると、時刻／カレンダ設定を表示します。日付・時刻の設定をできます（図2-5）。

日付・時刻が正しくないとき、コンテンツが正しく表示されない場合があります。システム起動時に日付・時刻を設定してください。

HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。

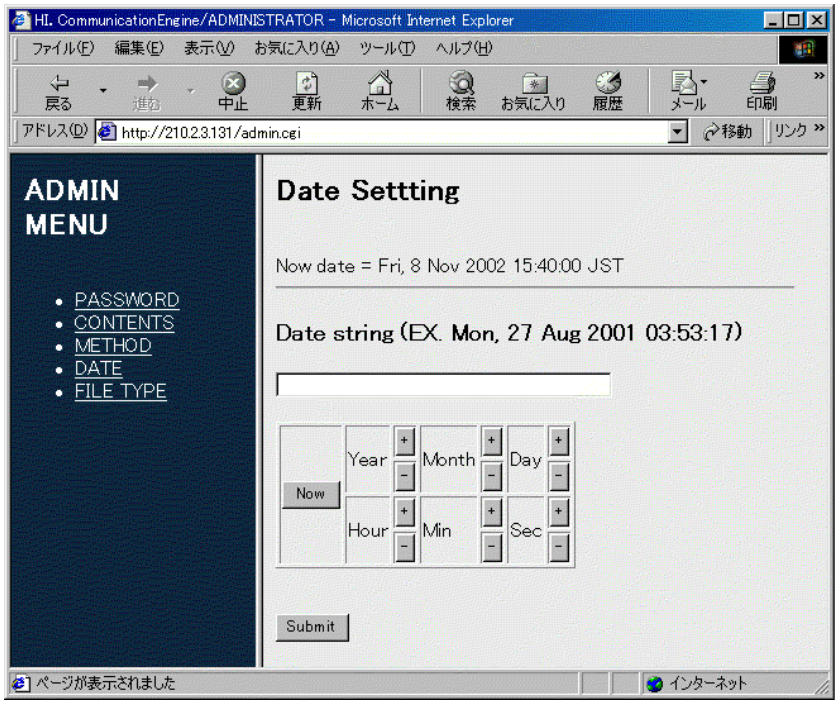

図2-5　時刻／カレンダ設定

2.4.6　　ファイルタイプ一覧

インデックスの "FILE TYPE" を選択すると、構築時に登録したファイルタイプの一覧を表示します（図2-6）。

HTTPサーバの構築情報の設定により使用できない場合があります（詳細は「HTTPサーバの構築マニュアル」を参照してください）。

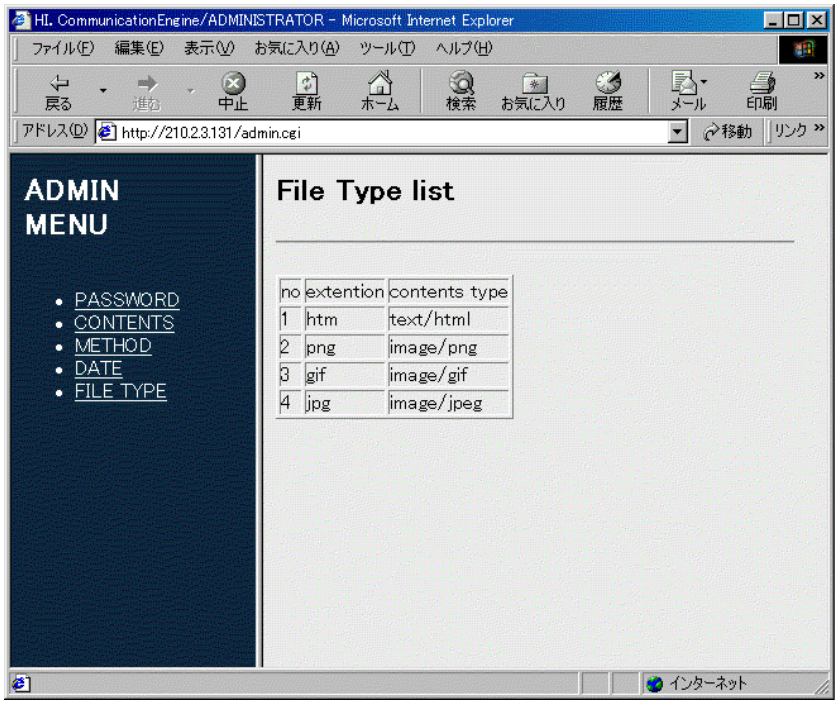

図2-6　ファイルタイプ一覧

2.5 ユーザ認証機能

静的コンテンツはファイルシステムAPIにて、ユーザCGIは登録時に指定することにより、ユーザ認証対象のコンテンツとなります。サポートする認証方法は基本認証（ Base64 ）のみとなります。

ユーザ認証がOKのとき、次からは認証なしでアクセス可能です。アクセスがなくなってから3分後、認証解除されます。そのとき、再度ユーザ認証が必要となります。
認証されたユーザはユーザ名とクライアントのIPアドレスで管理されます。同一のユーザ名で複数のホスト（別IPアドレス）からアクセスした場合、2番目以降に行なったユーザ認証はエラーとなります（"Permission denied"メッセージが送信されます。但し、レスポンスヘッダ上は"HTTP/1.1 200 OK"です）。

コンテンツ毎に特定のユーザを割り当てることはできません。

2.5.1 認証用ユーザの登録

認証ユーザの登録方法には、HTTPサーバ構築情報の「認証用ユーザの登録」による静的登録、以下のサービスコールを使用する動的登録の2通りあります。認証用ユーザのユーザ名、パスワードともにNullを含めた16文字以内です。
登録された認証用ユーザに対して、ユーザ名の変更、パスワードの変更、ユーザの削除が可能です。

表2-6　ユーザ認証用サービスコール

区分	サービスコール名称	サービスコールの機能
ユーザ認証機能	HTTP_authAdd	認証用ユーザを登録します。
	HTTP_authChgUser	認証用ユーザ名を変更します。
	HTTP_authChgPass	認証用ユーザのパスワードを変更します。
	HTTP_authDel	認証用ユーザを削除します。

2.6 ファイルシステム

2.6.1 ファイルシステム API 仕様（コールバック）

本HTTPサーバではファイルシステムに依存しない様に、ファイルシステムAPI仕様を提供します。HTTPサーバでは、本API仕様に基づきファイルシステムにアクセスしております。

ファイルのデータをクライアントに送信する HTTP_fs_send では、HTTP_writeContent サービスコールを使用してデータを送信します。送信処理では少ないスタックサイズで送信できるように、複数回に分割して送信できます。なお、HTTP_fs_getFsize で取得したファイルサイズを正しく送信しなかった場合の動作の保証はされません。

表2-7　ファイルシステムAPI仕様（コールバック）

区分	コールバック名称	コールバックの機能
ファイル API 仕様（コールバック）	HTTP_fs_find	指定したファイルの有無を確認します。
	HTTP_fs_getTM	ファイルの更新時刻を取得します。
	HTTP_fs_getFsisze	ファイルのサイズを取得します。
	HTTP_fs_isitDir	ファイル属性はディレクトリかチェックします。
	HTTP_fs_isitFile	ファイル属性はファイルかチェックします。
	HTTP_fs_checkAuth	ファイルはユーザ認証が必要かチェックします。
	HTTP_fs_send	ファイルのデータをクライアント送信します。

表2-8　ファイル送信用サービスコール

区分	サービスコール名称	サービスコールの機能
ファイル送信用サービスコール	HTTP_writeContent	ファイルのデータをクライアント送信します。

2.6.2 ファイルシステムと PUT／DELETE メソッド

HTTPサーバの機能にPUT／DELETEメソッドがあります。PUTメソッドはコンテンツのアップロード機能、DELETEはコンテンツの削除機能を実現しております。PUT／DELETEメソッドのインプリメントは、コンテンツを管理するファイルシステムに依存します。たとえば、サンプルの簡易ROMファイルシステムは動的な変更ができないため、PUT／DELETEメソッドは使用できません。

PUT／DELETEメソッドのインプリメントについては、当該機能をCソースファイルにて提供してリンクする形式となっております。PUT／DELETEメソッドを使用するさいは、使用するファイルシステムに合わせて、書き換えが必要となります。

表2-9　PUT/DELETEメソッド用コールバック

区分	コールバック名称	コールバックの機能
メソッド機能	HTTP_methPut	PUT メソッドのインプリメント（ファイルシステムに依存するため、コールバックとして提供）
	HTTP_methDelete	DELETE メソッドのインプリメント（ファイルシステムに依存するため、コールバックとして提供）

2.7 レスポンスメッセージ

HTTPサーバの一部のレスポンスメッセージをカスタマイズする機能を提供します。カスタマイズすることによりエラー時に表示されるブラウザの内容を変更することができます。対応するレスポンスメッセージを以下に示します。

表2-10　カスタマイズできるレスポンスメッセージ

ステータスコード	説明
200 OK	通常の"200 OK"では使用されず、認証機能で失敗した場合に表示される"Permission Denied"を表示するケースにのみ使用される。
403 Forbidden	リクエストを理解したが、処理を行なわない。本サーバではディレクトリを指定した場合(CGI 除く)に送信される。
404 Not Found	指定コンテンツがサーバ上にない
405 Method Not Allowd	指定したメソッドが使用できない。
412 Precondition Failed	条件付きヘッダで指定された条件に合わない
413 Request Entity Too Large	リクエストエンティティが大きすぎる。
414 Request-URI Too Large	リクエスト URI が長すぎる。
501 Not Implemented	要求されたメソッドがサポートされていない。
503 Service Unavailable	現時点でサーバはリクエストを処理できません。本サーバでは PUT/DELETE メソッドのサンプルにて使用しています。
505 HTTP Version Not Supported	サーバがリクエストに使われた HTTP バージョンをサポートしていない。

以下にレスポンスメッセージ用コールバック、及び、サービスコールの一覧を示します。

表2-11　レスポンスメッセージ用コールバック

区分	コールバック名称	コールバックの機能
レスポンスメッセージ用コールバック	HTTP_sendmess*XXX*	**レスポンスメッセージを送信します。**

XXX：該当するレスポンスコード

表2-12　レスポンスメッセージ用サービスコール

区分	コールバック名称	コールバックの機能
レスポンスメッセージ用サービスコール	HTTP_sendErrResponse	**レスポンスメッセージに該当する** HTML **メッセージファイルを指定して、レスポンス送信します。**
	HTTP_responsContent	HTML **メッセージファイルがない場合にデフォルトのレスポンスを送信します。**

2.8 C ライブラリ関数

HTTPサーバでは専用のCライブラリ関数を用意しております。

表2-13　Cライブラリ関数

区分	コールバック名称	コールバックの機能
Cライブラリ関数	HIcom_GetStrLenL	**文字列長を計算します。**
	HIcom_GetStrLenW	**文字列長を計算します。**
	HIcom_CpyStrL	**文字列をコピーします。**
	HIcom_CpyStrW	**文字列をコピーします。**
	HIcom_CpyStrLenL	**文字列をコピーします（文字列長指定）。**
	HIcom_CpyStrLenW	**文字列をコピーします（文字列長指定）。**
	HIcom_CatStr	**文字列を結合します。**
	HIcom_CmpStr	**文字列を比較します。**
	HIcom_CmpStrLenL	**文字列を比較します（文字列長指定）。**
	HIcom_CmpStrLenW	**文字列を比較します（文字列長指定）。**
	HIcom_SearchStr	**文字列から指定文字を検索します。**
	HIcom_UpperCh	**アルファベットの小文字を大文字に変換します。**
	HIcom_CmpStrUpper	**アルファベットの大文字に変換後、文字列を比較します。**
	HIcom_CmpStrLenUpperL	**アルファベットの大文字に変換後、文字列を比較します（文字列長指定）。**
	HIcom_CmpStrLenUpperW	**アルファベットの大文字に変換後、文字列を比較します（文字列長指定）。**
	HIcom_CpyMemL	**メモリコピーします。**
	HIcom_CpyMemW	**メモリコピーします。**
	HIcom_SetMemL	**メモリにデータ設定します。**
	HIcom_SetMemW	**メモリにデータ設定します。**
	HIcom_h2b	16 **進文字列の先頭２文字を** 16 **進データ変換します**
	HIcom_h2a	**データを** 16 **進文字列に変換します。**
	HIcom_i2a	**データを** 10 **進文字列に変換します。**
	HIcom_a2h	16 **進文字列をデータに変換します**
	HIcom_a2i	10 **進文字列をデータに変換します**

2.9 Java Package

簡易ROMファイル用のコンテンツ→Cソース変換、および、HTTPサーバの動作確認用にJava Packageを提供します。

本Java Packageを使用するにあたり、Sun Microsystems, Inc.製のJava(TM) 2 SDK, Standard Editionが必要となります。本プログラムは http://java.sun.com/ からダウンロードできます。

提供するJava PackageはJavaソースプログラム形式での提供となります。Java(TM) 2 SDK, Standard Edition にて、コンパイルしてからご使用ください。

```
>  javac *.java                    … コンパイル
>  java クラス名 [引数,...]        … 実行
```

表2-14　Java Package

区分	Java Package の種類	機能
変換ツール	cat.java	簡易 ROM ファイル用のコンテンツ→C ソース変換
動作確認ツール	put.java	PUT メソッド動作確認用プログラム
	delete.java	DELETE メソッド動作確認用プログラム

3. サービスコール

本節では、サービスコール、ＡＰＩ仕様についての詳細な説明を以下の形式で行っています。

No.　サービスコール名　　機能

【発行可能なシステム状態*1】

Ｃ言語インタフェース

マネージャコール呼出し形式*2

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
・	・	・
・	・	・

パケットの構造

リターン値／エラーコード	
リターン値またはニモニック	リターン値またはエラーコードの意味
・	・
・	・
・	・

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サービスコールを発行した場合、システムの正常な動作は保証されません。

3.1 初期化サービスコール

3.1.1 HTTP_init HTTP サーバの初期化

【T／D／L／I】

C言語インタフェース

```
void HTTP_init ( void );
```

パラメータ

無し

リターンパラメータ

無し

解　説

HTTPサーバを初期化します。

HTTPサーバの内部変数を初期化し、サービス開始可能な状態にします。
HTTP_init は、HTTPサーバのサービス開始前に１回だけ実行してください。

3.1.2 HTTP_start HTTP サーバのサービスを開始する

【T】

C言語インタフェース

ER ercd = HTTP_start (UW ipaddr, UH port);

パラメータ		
UW	ipddr	自IPアドレス
UH	port	ポート番号

リターンパラメータ		
ER	ercd	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_OBJ	オブジェクト状態エラー（初期化されていない、または既にサービスが開始している）
E_SYS	システムエラー（OSのリソース確保に失敗）
E_SOCK	TCP通信エラー（TCP通信端点の生成ができない）

解　説

HTTPサーバのサービスを開始します。本サービスコールでは、HTTPサーバタスクの生成、HTTPサーバ内部で使用するOSの可変長メモリプールの生成、TCP通信端点の生成を行ないます。

HTTPサーバタスクと、可変長メモリプールは、HTTPサーバの構築情報（httpd02.cのhttpOsConf)をもとに生成されます。提供時は標準設定の値が設定されています。

```
typedef struct _HTTP_CONFTBL {
        PRI     tskPri;         HTTPサーバタスクのプライオリティ
        ID      tskId;          HTTPサーバタスクのタスクID
        INT     tskSz;          HTTPサーバタスクのスタックサイズ
        ID      flgID;          イベントフラグID
        ID      mplId;          可変長メモリプールID
        INT     mplSz;          可変長メモリプールのサイズ
} HTTP_CONFTBL;

const HTTP_CONFTBL httpOsConf   =   {
        2,                      HTTPサーバタスクのプライオリティ
        0,                      HTTPサーバタスクのタスクID
        0x0600,                 HTTPサーバタスクのスタックサイズ
        0,                      イベントフラグID
        0,                      可変長メモリプールID
        0x1000,                 可変長メモリプールのサイズ
};
```

タスクID、イベントフラグID、メモリプールIDに0を指定した場合、IDは番号は自動的に割り当てられます。IDを指定したい場合は、0以外の値を設定してください。その他、設定の詳細については「2.1.1 OS情報の設定」をご参照ください。**ご使用のOSが HI2000/3 の場合、タスク・可変長メモリプールの生成はOS構築にて行なわれます。そのため、「タスクID」、「可変長メモリプールID」以外の情報は使用されません。**

パラメータのipddrでは自IPアドレスを指定してください。自IPアドレスに0を指定した場合、動作を開始しているIPアドレスを対象にTCP通信端点を生成します。

パラメータのportではポート番号を指定してください。一般的にHTTPサーバのポート番号は80です。

3.1.3 HTTP_stop HTTP サーバのサービスを終了する

【T】

C言語インタフェース

```
void HTTP_stop( void );
```

パラメータ

無し

リターンパラメータ

無し

解　説

HTTPサーバのサービスを停止します。

本サービスコールでは、通信中であっても全てのTCP通信端点、および、HTTPサーバの全リソースを開放後、HTTPサーバタスクを強制終了します。

本マネージャコール発行後、HTTPサーバの全ての機能は停止し、各機能は無効となります。その場合、HTTP_init、HTTP_start 以外のサービスコールは使用できません。

再度HTTPサーバを起動する場合は、HTTP_init、HTTP_start サービスコールを発行してください。

3.2 CGI 関連サービスコール

3.2.1 HTTP_RegisterCGI CGI を登録します

【T】

C言語インタフェース

ER ercd = HTTP_RegisterCGI (B *name, W secure, W (*usercgi)());

パラメータ		
B	*name	CGIコンテンツ名称の文字列を格納した領域のアドレス
W	secure	ユーザ認証フラグ（0:ユーザ認証なし、0以外:ユーザ認証有）
W	(*usercgi)()	コンテンツのリクエスト時に、呼び出されるユーザCGIコールバック関数

リターンパラメータ		
ER	ercd	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（nameがNull、または、nameがNullを含む16文字を超えるとき、または、usercgiがNull、または、usercgiが偶数アドレスでないとき）
E_OBJ	登録済みのCGIコンテンツ名称を指定した
E_NOMEM	メモリ不足（確保済みのCGI登録領域が全て使用済みである）

解　説

ユーザCGIを登録します。HTTPサーバタスクは、初期化処理にてコールバック_UserCGI を呼び出します。_UserCGIでは、HTTP_RegisterCGI サービスコールを使用して、ユーザCGIの登録を行ってください（詳細は、「2.3 ユーザGGI」を参照してください）

パラメータ name ではCGIコンテンツ名称の文字列を指定します。コンテンツ名は、Nullを含む16文字以内で指定してください。

パラメータ secure はユーザ認証フラグです。0以外を指定すると、コンテンツをブラウザから閲覧するときに、ユーザ認証の対象となります。

パラメータ usercgi は、コンテンツのリクエスト時に呼び出されるユーザCGIコールバック関数のアドレスを指定します。ユーザCGIコールバック関数の詳細は、「2.3.2 ユーザCGIコールバック関数」を参照してください。

3.2.2 HTTP_extractPsStr **パラメータの解析**

【T】

C言語インタフェース

```
B *next = HTTP_extractPsStr ( B *par, B **name, B **val );
```

パラメータ		
B	*par	コンテンツパラメータ文字列を格納した領域のアドレス
B	**name	コンテンツパラメータの"Name"を格納する文字列ポインタのアドレス
B	**val	コンテンツパラメータの"Value"を格納する文字列ポインタのアドレス

リターンパラメータ		
B	*next	次のコンテンツパラメータ文字列を格納した領域のアドレス
B	**name	コンテンツパラメータの"Name"を格納した文字列ポインタのアドレス
B	**val	コンテンツパラメータの"Value"を格納した文字列ポインタのアドレス

リターン値／エラーコード	
0	次のコンテンツパラメータ無し
0以外	次のコンテンツパラメータ文字列を格納したアドレス

解　説

ユーザCGIコールバック関数にて、コンテンツパラメータの解析を行ないます。コンテンツパラメータはエンコードされた状態で受信しますが、本サービスコールでは解析と同時にデコードも行います。

ユーザCGIコールバック関数の引数、param1, param2 にはコンテンツパラメータが設定されます（ユーザCGIコールバック関数の詳細については「2.3 ユーザGGI」を参照してください）。

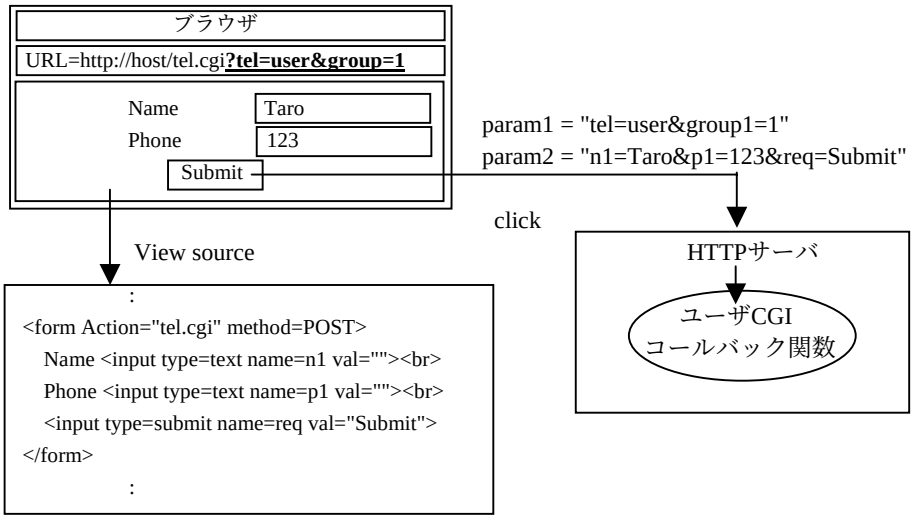

図3-1　コンテンツパラメータ

パラメータ par にはコンテンツパラメータ文字列を格納した領域のアドレスを指定してサービスコールを発行します。リターンパラメータ name, val には、コンテンツパラメータの式（"Name=Value"）のそれぞれ右辺と左辺の文字列を指すポインタが返ります。

また、リターンパラメータ next にはコンテンツパラメータの次の式を指すポインタが設定されます。なお、次の式がない場合はNullポインタ（0）が設定されます。

以下は、コンテンツパラメータ解析のサンプルプログラムです。

サンプルプログラム

```
W _CGI_callback( HTTB_CTB *hcb, W type, B *param1, B param2)
{
        B *par, *next, *name, *val;
        B namae[8], tel[12];
        :
        :
        next = param2;
        while( next ) {
                par = next;
                next = HTTP_extractPsStr( par, &name, &val );

                if( strcmp( name, "n1") = = 0) {
                        strcpy( namae, val);
                } else if( strcmp( name, "p1") = = 0) {
                        strcpy( tel, val);
                }
        }
        :
        :
}
```

3.2.3 HTTP_GetFileTypeNoByExt **コンテンツタイプの取得**

【T】

C言語インタフェース

W fno = HTTP_GetFileTypeNoByExt (B *name);

パラメータ		
B	*name	拡張子の文字列を格納した領域のアドレス

リターンパラメータ		
W	fno	コンテンツタイプ番号

リターン値／エラーコード

無し

解　説

ユーザCGIコールバック関数にて、コンテンツデータ送信サービスコール HTTP_SendChunkHeader、または、HTTP_ResponsTransferを発行するとき、引数にて指定するコンテンツタイプ番号を取得します。

コンテンツタイプ番号はHTTPサーバからブラウザへコンテンツを転送するとき、付加されるヘッダ情報の"Content-Type:"にて指定するメディアタイプを決定するために使用されます。使用する拡張子とメディアタイプは、予め構築情報の「ファイルタイプの登録」にて行う必要があります。

パラメータ name には拡張子（"htm"、"gif"など）の文字列を格納した領域のアドレスを指定します。指定した拡張子が未登録だった場合や、不正だった場合は、コンテンツタイプ番号には0が返ります（0は、構築情報の「ファイルタイプの登録」にて指定するテーブルの0番目のメディアタイプです）。

CGIで、作成する動的コンテンツがHTML形式なら、"htm"と指定します。

3.2.4 HTTP_SendChunkHeader **チャンク形式ヘッダー送信**

【T】

C言語インタフェース

```
ER ercd = HTTP_SendChunkHeader ( HTTP_CTB *hcb, W type );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
W	type	コンテンツタイプ番号

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

リターン値／エラーコード

E_OK	正常終了
E_SOCK	送信エラー（TCP/IPマネージャ tcp_snd_dat エラー）

解　説

ユーザCGIコールバック関数にて、チャンク形式でコンテンツを送信するとき、チャック形式用のレスポンスヘッダの作成と送信を行ないます。本サービスコールはチャンク形式でコンテンツを送信するとき、最初に一回発行する必要があります。

パラメタ *hcb は、ユーザCGIコールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

パラメタ type はコンテンツタイプ番号を指定します。コンテンツタイプ番号は、HTTP_GetFileTypeNoByExt サービスコールを使用して取得します。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

3.2.5 HTTP_SendChunkData **チャンク形式データ送信**

【T】

C言語インタフェース

```
ER ercd = HTTP_SendChunkHeader ( HTTP_CTB *hcb, B *content, W len );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
B	*content	コンテンツデータ文字列を格納した領域のアドレス
W	len	コンテンツデータ文字列の長さ

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

リターン値／エラーコード

E_OK	正常終了
E_SOCK	送信エラー（TCP/IPマネージャ tcp_snd_dat エラー）

解　説

ユーザCGIコールバック関数にて、チャンク形式でコンテンツを送信するとき、コンテンツデータを送信します。

本サービスコールを発行する前には、HTTP_SendChunkHeaderを一回発行する必要があります。
HTTP_SendChunkHeader 発行後から、HTTP_SendChunkEnd を発行するまでの間は、本サービスコールを何回でも発行することができます。コンテンツデータを分割して送信することにより、少ないRAM領域で動的コンテンツの送信が可能となります。

パラメタ *hcb は、ユーザCGIコールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

パラメタ *content はコンテンツデータ文字列を格納した領域のアドレスを指定します。

パラメタ len はコンテンツデータ文字列の長さを指定します。lenに0を指定した場合、何も送信しません。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

3.2.6 HTTP_SendChunkEnd **チャンク形式データ送信終了**

【T】

C言語インタフェース

```
ER ercd = HTTP_SendChunkHeader ( HTTP_CTB *hcb );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

リターン値／エラーコード

E_OK	正常終了
E_SOCK	送信エラー（TCP/IPマネージャ tcp_snd_dat エラー）

解　説

ユーザCGIコールバック関数にて、チャンク形式でコンテンツを送信するとき、コンテンツデータの完了通知を行います。本サービスコールはチャンク形式でコンテンツを送信するとき、最後に一回発行する必要があります。

パラメタ *hcb は、ユーザCGIコールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

3.2.7 HTTP_ResponsTransfer **通常(一括)コンテンツの送信**

【T】

C言語インタフェース

```
ER ercd = HTTP_ResponsTransfer ( HTTP_CTB *hcb, W con, B *content, W type, W len, UW clk, B *etag );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
W	con	コネクション属性
B	*content	コンテンツデータ文字列を格納した領域のアドレス
W	type	コンテンツタイプ番号
W	len	コンテンツデータ文字列の長さ
UW	clk	コンテンツの最終更新時刻
B	*etag	コンテンツのエンティティタグ文字列を格納した領域のアドレス

リターンパラメータ

ER	ercd	リターン値またはエラーコード

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

リターン値／エラーコード

E_OK	正常終了
E_SOCK	送信エラー（TCP/IPマネージャ tcp_snd_dat エラー）

解　説

ユーザCGIコールバック関数にて、コンテンツを通常（一括）送信を行います。

パラメタ *hcb は、ユーザCGIコールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

パラメタ con ではコネクション属性を指定します。コネクション属性には、HTTP_CON_KEEP(0)、または、HTTP_CON_CLOSE(1)を指定することができます。本指定により、"Connection:"ヘッダフィールドで、それぞれ、"Keep-Alive", "close"が設定されます。通常は、HTTP_CON_KEEPを指定してください。

パラメタ *content はコンテンツデータ文字列を格納した領域のアドレスを指定します。

パラメタ type はコンテンツタイプ番号を指定します。コンテンツタイプ番号は、HTTP_GetFileTypeNoByExt サービスコールを使用して取得します。

パラメタ len はコンテンツデータ文字列の長さを指定します。

パラメタ clk、*etag は、CGIでは使用しません。サービスコール発行のさいには、0を指定してください。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

3.3 ユーザ認証機能用サービスコール

3.3.1 HTTP_authAdd 認証用ユーザの登録

【T】

C言語インタフェース

```
ER ercd = HTTP_authAdd ( B *user, B *pass);
```

パラメータ

B	*user	ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス
B	*pass	パスワード（Nullを含む16文字以内）を格納した領域のアドレス

リターンパラメータ

ER	ercd	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（user、passがNull、または、Nullを含めて16文字を超えるとき）
E_OBJ	登録済みのユーザを指定した
E_NOMEM	登録用の領域が不足してます（構築情報にて設定）

解　説

ユーザ認証用のユーザを登録します。

3.3.2 HTTP_authChgUser **認証用ユーザ名の変更**

【T】

C言語インタフェース

ER ercd = HTTP_authChgUser (B *olduser, B *pass, B *newuser);

パラメータ		
B	*olduser	旧ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス
B	*pass	パスワード（Nullを含む16文字以内）を格納した領域のアドレス
B	*newuser	新ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス

リターンパラメータ		
ER	ercd	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（olduser、pass、newuserがNull、または、Nullを含めて16文字を超えるとき）
E_OBJ	旧ユーザは登録されていないか、新ユーザが登録済みです。
E_NG	パスワードが不正です。

解　説

ユーザ認証用のユーザ名を変更します

3.3.3 HTTP_authChgPass **認証用ユーザのパスワード変更**

【T】

C言語インタフェース

ER ercd = HTTP_authChgPass(*user, B *pass, B *new);

パラメータ		
B	*user	ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス
B	*pass	旧パスワード（Nullを含む16文字以内）を格納した領域のアドレス
B	*new	新ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス

リターンパラメータ		
ER	ercd	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（user、pass、newがNull、または、Nullを含めて16文字を超えるとき）
E_OBJ	指定したユーザは登録されていません
E_NG	旧パスワードが不正です。

解　説

ユーザ認証用のユーザ名を変更します

3.3.4 HTTP_authDel 認証用ユーザの削除

【T】

C言語インタフェース

```
ER ercd = HTTP_authDel ( B *user, B *pass);
```

パラメータ

B	*user	ユーザ名（Nullを含む16文字以内）を格納した領域のアドレス
B	*pass	パスワード（Nullを含む16文字以内）を格納した領域のアドレス

リターンパラメータ

ER	ercd	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（user、passがNull、または、Nullを含めて16文字を超えるとき）
E_OBJ	指定したユーザが登録されていません
E_NG	パスワードが不正です。

解　説

ユーザ認証用のユーザを削除します。

全ての認証ユーザを削除した場合、ユーザ認証が必要なコンテンツの閲覧はできなくなるので注意してください。

3.4 ファイルシステム API 用サービスコール

3.4.1 HTTP_writeContent ファイルデータを送信します

【T】

C言語インタフェース

ER ercd = HTTP_writeContent (ID sock, B *buf, W size);

パラメータ		
W	sock	TCP通信端点ID
B	*buf	送信データを格納した領域のアドレス
W	size	送信サイズ

リターンパラメータ		
ER	ercd	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_SOCK	送信エラー（TCP/IPマネージャ tcp_snd_dat エラー）

解 説

ファイルシステムAPI関数 HTTP_fs_send にて、ファイルデータの送信に使用します。ファイルデータは分割して送信することができます。・

パラメタ sock はTCP通信端点IDを指定します。ファイルシステムAPI関数 HTTP_fs_send の引数として取得できるので、そのまま指定してください。

パラメタ *buf は送信データを格納した領域のアドレスを指定します。

パラメタ size は送信データのサイズを指定します。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

なお、ファイルシステムAPI関数 HTTP_fs_send がコールされる前に、HTTPサーバがレスポンスヘッダとして、ファイルサイズを送信済みです。従って、HTTP_fs_sendから、本サービスコールを使用して送信したデータの合計サイズが、ファイルサイズと異なった場合、HTTPサーバの不正動作となりますので、ご注意ください。

3.5 レスポンスメッセージ送信

3.5.1 HTTP_sendErrResponse レスポンス送信(ファイル指定)

【T】

C言語インタフェース

```
ER ercd = HTTP_sendErrResponse ( HTTP_CTB *hcb , B *statuscode, B *htm);
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
B	*statuscode	ステータスコード文字列
B	*htm	HTMLメッセージファイル

リターンパラメータ

ER	ercd	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_NG	指定したHTMLメッセージファイルがない

解　説

HTMLメッセージファイルを指定して、レスポンスメッセージを送信します。本サービスコールでは、ファイルシステムAPIを使用して、指定したHTMLメッセージファイルを送信します。

パラメタ *hcb は、レスポンスメッセージ用コールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

パラメタ * statuscode　はステータスコード文字列(例:”404 Not Found”)を格納した領域のアドレスを指定します。

パラメタ * htm はHTMLメッセージファイル名を格納した領域のアドレスを指定します。

本サービスコールは指定したHTMLメッセージファイルがない場合、エラーリターンします。

3.5.2 HTTP_responsContent **レスポンス送信(汎用)**

【T】

C言語インタフェース

ER ercd = HTTP_sendErrResponse (HTTP_CTB *hcb , B *err, B *errCont, W length, W allow);

パラメータ		
HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
B	*err	ステータスコード文字列
B	*errCont	HTMLコンテンツ文字列
W	length	HTMLコンテンツ文字列の長さ
W	allow	OPTIONメッソッド使用時の”ALLOW”ヘッダ出力フラグ

リターンパラメータ		
ER	ercd	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_NG	送信失敗

解　説

HTMLメッセージファイルを指定して、レスポンスメッセージを送信します。本サービスコールでは、ファイルシステムAPIを使用して、指定したHTMLメッセージファイルを送信します。

パラメタ *hcb は、レスポンスメッセージ用コールバック関数の引数として取得できるので、そのまま指定してください。*hcb では、HTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。

パラメタ * errはステータスコード文字列(例:”404 Not Found”)を格納した領域のアドレスを指定します。

パラメタ * errCont はHTMLコンテンツ文字列を格納した領域のアドレスを指定します。

パラメタ length はHTMLコンテンツ文字列の長さを指定します。

パラメタ allow　はOPTIONメッソッド使用時の”ALLOW”ヘッダ出力フラグを指定します。レスポンスメッセージ送信時は0を指定してください。

本サービスコールは、送信処理（TCP/IPマネージャ tcp_snd_dat）が失敗したときのみ、エラーとなります。

4. コールバック

HTTPサーバでは、以下の（HTTPサーバタスクからユーザ関数を呼び出す）コールバックがあります。コールバック仕様に合わせてご使用ください。

表4-1　コールバック

区分	コールバック名称	コールバックの機能
メソッド機能	HTTP_methPut	PUT メソッドのインプリメント（ファイルシステムに依存するため、コールバックとして提供）
	HTTP_methDelete	DELETE メソッドのインプリメント（ファイルシステムに依存するため、コールバックとして提供）
CGI 登録	_UserCGI	ユーザ CGI の登録を行なう。
ユーザ CGI 関数	(*usercgi)()	ユーザ CGI コールバック関数。
ファイル API 仕様（コールバック）	HTTP_fs_find	指定したファイルの有無を確認します。
	HTTP_fs_getTM	ファイルの更新時刻を取得します。
	HTTP_fs_getFsisze	ファイルのサイズを取得します。
	HTTP_fs_isitDir	ファイル属性はディレクトリかチェックします。
	HTTP_fs_isitFile	ファイル属性はファイルかチェックします。
	HTTP_fs_checkAuth	ファイルはユーザ認証が必要かチェックします。
	HTTP_fs_send	ファイルのデータをクライアント送信します。
レスポンスメッセージ用コールバック	HTTP_sendmess*XXX*	レスポンスメッセージを送信します。

XXX：該当するレスポンスコード

4.1 メソッド機能

4.1.1 HTTP_methPut PUT メソッドのインプリメント

【T】

C言語インタフェース

```
void HTTP_methPut ( HTTP_CTB *hcb, B *meth );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
B	*meth	"PUT"文字列を格納した領域のアドレス

リターンパラメータ

無し

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}  HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

解　説

本コールバックではPUTメソッドを実装します。PUTメソッドはファイルシステムに依存するため、ユーザカスタムによるコールバックとして提供します。

デフォルトのファイルシステムはサンプルの簡易ROMファイルシステムのため、構築情報の「メソッドのOn/Off」にて、PUTメソッドの初期値は未使用となっております。その場合、本コールバックは使用されません。使用する場合には構築情報の「メソッドのOn/Off」にて、PUTメソッドを使用する設定に変更してください。

パラメタ *hcb はHTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。ファイルデータの取得や、レスポンスの送信時に使用されます。

パラメタ *meth は"PUT"文字列を格納した領域のアドレスが設定されております（未使用）。

PUTメソッドの外部仕様は次のようになっております

表4-2　PUTメソッド外部仕様

事象	レスポンス
ファイルの作成に成功した	"http/1.1　204　No Content"
ファイルの作成に失敗した	"http/1.1　503　Service unavialable"

http_m_put.c にてPUTメソッドのインプリメントするためのサンプルを提供します。

ソースプログラム http_m_put.c

```
void HTTP_methPut( HTTP_CTB *htb, B *meth )
{
                :
        /* get now clock */
        HIcom_tapi_getNowClk(&clk);
-------------------------------------------------------------------①
        /*******************************************
         ** INPUT :                                      **
         **      hcb->md.content <-- file name       **
         **      buf              <-- file data       **
         **      size             <-- file size       **
         **      clk              <-- now clock       **
         ** ================================          **
         ** call user file system & create file !! **
         *******************************************
         */

        /* default : 503 service unvailable */
-------------------------------------------------------------------②
        if(0) {
                /* OK Message : 204 No Content */
                HTTP_responsContent(hcb, HTTP_RS_204, 0, 0);
        } else {
                /* NG Message : 503 service unvailable */
                HTTP_sendmess(hcb);
        }
}
```

①時点でファイルの作成に必要な情報、ファイル名、ファイルデータ(*1)、ファイルサイズ、現在の時刻（標準時刻1970年1月1日0時0分0秒からの通産秒数）がそろいますので、ユーザが用意したファイルシステムに従いファイルを作成してください。その結果により、クライアントへのレスポンス（②時点）を決定してください。

(*1) ファイルデータ用のバッファはユーザにて準備する必要があります

4.1.2 HTTP_methDelete DELETE メソッドのインプリメント

【T】

C言語インタフェース

```
void HTTP_methDelete( HTTP_CTB *hcb, B *meth );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
B	*meth	"DELETE"文字列を格納した領域のアドレス

リターンパラメータ

無し

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル、バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

解　説

本コールバックではDELETEメソッドを実装します。DELETEメソッドはファイルシステムに依存するため、ユーザカスタムによるコールバックとして提供します。

デフォルトのファイルシステムはサンプルの簡易ROMファイルシステムのため、構築情報の「メソッドのOn/Off」にて、DELETEメソッドの初期値は未使用となっております。その場合、本コールバックは使用されません。使用する場合には構築情報の「メソッドのOn/Off」にて、DELETEメソッドを使用する設定に変更してください。

パラメタ *hcb はHTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。ファイルデータの取得や、レスポンスの送信時に使用されます。

パラメタ *meth は"DETELE"文字列を格納した領域のアドレスが設定されております（未使用）。

DELETEメソッドの外部仕様は次のようになっております

表4-3 DELETEメソッド外部仕様

事象	レスポンス
ファイルの削除に成功した	"http/1.1 204 No Content"
ファイルの削除に失敗した	"http/1.1 503 Service unavialable"

http_m_delete.c にてDELETEメソッドのインプリメントするためのサンプルを提供します。

ソースプログラム http_m_delete.c

```
void HTTP_methDelete( HTTP_CTB *htb, B *meth )
{
        if(HTTP_anaContent(hcb) != 0) return;
------------------------------------------------------------------ ①
        /*******************************************
         ** INPUT :                              **
         **       hcb->md.content <-- file name   **
         ** ===============================      **
         ** call user file system & delete file !! **
         *******************************************
         */

        /* default : 503 service unvailable */
------------------------------------------------------------------ ②
        if(0) {
                /* OK Message : 204 No Content */
                HTTP_responsContent(hcb, HTTP_RS_204, 0, 0);
        } else {
                /* NG Message : 503 service unvailable */
                HTTP_sendmess(hcb);
        }
}
```

①時点でファイルの作成に必要な情報、ファイル名がわかるので、ユーザが用意したファイルシステムに従いファイルを削除してください。その結果により、クライアントへのレスポンス（②時点）を決定してください。

4.2 CGI 登録

4.2.1 _UserCGI ユーザ CGI の登録を行なう

【T】

C言語インタフェース

```
void _UserCGI( void );
```

パラメータ

無し

リターンパラメータ

無し

解　説

本コールバックではユーザCGIの登録を行います。HTTPサーバタスクの初期化処理で呼び出されます。_UserCGIでは、HTTP_RegisterCGI サービスコールを使用して、ユーザCGIの登録を行ってください（詳細は、「2.3 システムGGI」を参照してください）。

本コールバックの使用例として、サンプルプログラム（http_user_cgi.c）を提供します。

4.3 ユーザ CGI 関数

4.3.1 (*usercgi)() ユーザ CGI コールバック関数

【T】

C言語インタフェース

```
W ercd = (*usercgi)( HTTP_CTB *hcb, W type, B *param1, B *param2 );
```

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ
W	type	メソッドID
B	*param1	コンテンツパラメータ1
B	*param2	コンテンツパラメータ2

リターンパラメータ

E_OK	正常終了
E_OK以外	異常終了

パケットの構造

```
typedef struct _HTTP_HEADR {
        B   content[HTTP_HR_CONT_MAX];      コンテンツ名(HTTP_HR_CONT_MAX=256)
        B   version[HTTP_HR_VER_MAX];       プロトコル､バージョン名(HTTP_HR_VER_MAX =16)
        B   accept[HTTP_HR_ACPT_MAX];       "Accept:"ヘッダ(HTTP_HR_ACPT_MAX=256)
        B   lang[HTTP_HR_LANG_MAX];         "Accept-Language:"ヘッダ(HTTP_HR_LANG_MAX=32)
        B   encoding[HTTP_HR_ENCD_MAX];     "Accept-Encoding:"ヘッダ(HTTP_HR_ENCD_MAX=128)
        B   agent[HTTP_HR_AGNT_MAX];        "User-Agent:"ヘッダ(HTTP_HR_AGNT_MAX=128)
        B   host [HTTP_HR_HOST_MAX];        "Host:"ヘッダ(HTTP_HR_HOST_MAX=128)
        B   auth[HTTP_HR_AUTH_MAX];         "Authorization:"ヘッダ(HTTP_HR_AUTH_MAX =128)
        B   etag[HTTP_HR_ETAG_MAX];         "If-None-Match:"ヘッダ(HTTP_HR_ETAG_MAX=128)
        B   cont_len[HTTP_HR_CLEN_MAX];     "Content-Length:"ヘッダ(HTTP_HR_CLEN_MAX=16)
}   HTTP_HEADR;

typedef struct _HTTP_CTB {
        ID              sd;                 リクエストの応答に使用されるTCP通信端点ID
        UH              port;               リクエストの応答に使用されるポート番号
        UW              modified;           クライアントキャッシュのタイムスタンプの値
                                            （更新チェック用）
        UW              unmodified;         タイムスタンプの値（非更新チェック用）
        H               totalcnt;           ヘッダ情報の総バイト数
        H               rowLen[32];         ヘッダ情報のレコード毎の開始位置
        HTTP_HEADR      md;                 ヘッダ情報
        UW              buffer[HTTP_HEAD_MAX];
                                            ヘッダ取得バッファ(HTTP_HEAD_MAX=2048)
        B               nRow;               ヘッダ情報の文字数
        B               method;             メソッドID
        B               connection;         コネクション属性
        B               send_chunk;         送信用チャンク属性
} HTTP_CTB;
```

解　説

本コールバックはユーザCGIコールバック関数です。登録したユーザCGIコンテンツに対してリクエストが発生すると呼び出されます。ユーザCGIコールバック関数は、動的コンテンツの作成を行います。

パラメタ *hcb はHTTPとリクエスト元のクライアントとのコネクションや、リクエスト情報を管理しております。ファイルデータの取得や、レスポンスの送信時に使用されます。

パラメタ type にはメソッドIDが設定されております。以下にメソッドIDを示します。実際にはGET／POSTメソッドのみが設定され、その他の値は設定されることはありません。

表4-4　メソッドID

メソッド ID	説明
HTTP_NM_UNKNOWUN(0x00)	不正メソッドのリクエスト
HTTP_NM_HEAD(0x01)	HEAD メソッドのリクエスト
HTTP_NM_GET(0x02)	GET メソッドのリクエスト
HTTP_NM_POST(0x04)	POST メソッドのリクエスト
HTTP_NM_PUT(0x08)	PUT メソッドのリクエスト
HTTP_NM_TRACE(0x10)	TRACE メソッドのリクエスト
HTTP_NM_OPTION(0x20)	OPTION メソッドのリクエスト
HTTP_NM_DELETE(0x40)	DELETE メソッドのリクエスト

パラメタ *param1, *param2 はコンテンツパラメータが設定されております。コンテンツパラメータはエンコードされた状態で転送されます。、コンテンツパラメタのデコードと解析は、 HTTP_extractPsStrサービスコールを使用してください。

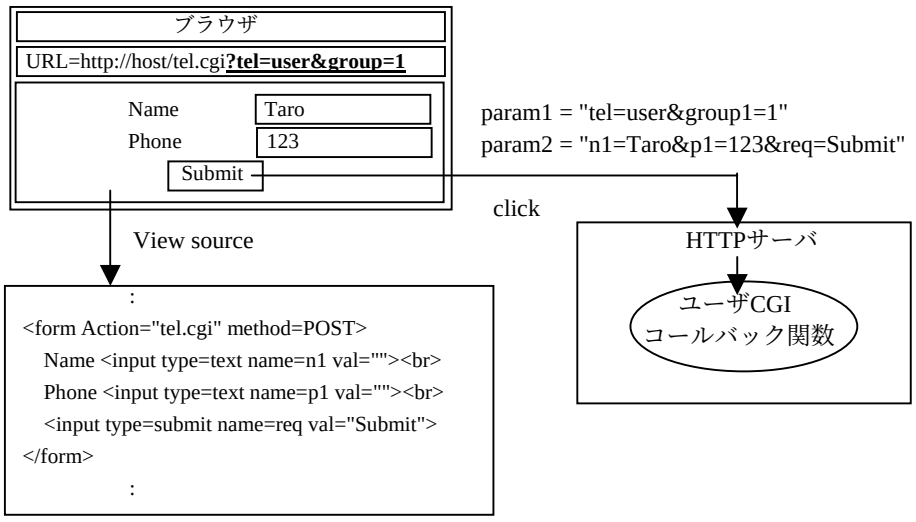

図4-1　コンテンツパラメータ

ユーザCGIコールバック関数では、リクエスト元に対して、必ずコンテンツを送信しなくてはいけません。コンテンツは、一度に送る方法（通常コンテンツの送信）と、分割して送る方法（チャンク形式エンコーディング）があります。詳細は、「2.3 ユーザGGI」を参照してください。

本コールバックの使用例として、サンプルプログラム（http_user_cgi.c）を提供します。ユーザCGIコールバック関数 HTTP_CGI_janken （コンテンツ名"janken.cgi"） は、HTTPサーバとジャンケンをする動的コンテンツのサンプルプログラムです。

4.4 ファイルシステム API 仕様（コールバック）

4.4.1 HTTP_fs_find 指定ファイルの有無チェック

【T】

C言語インタフェース

W ercd = HTTP_fs_find (B *filename);

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
W	ercd	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	
0	指定したファイルは存在しません
0以外	指定したファイルが見つかりました

解　説

パラメータ filename で指定したファイルを探します。

4.4.2 HTTP_fs_getTM **ファイルの更新時刻を取得**

【T】

C言語インタフェース

UW clk = HTTP_fs_getTM (B *filename);

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
UW	clk	ファイルの更新時刻 （標準時刻1970年1月1日0時0分0秒からの通産秒数）

解　説

パラメータ filename で指定したファイルの更新時刻（標準時刻1970年1月1日0時0分0秒からの通産秒数）を取得します。予め、HTTP_fs_find にてファイルがあることを確認してから実行する必要があります。

4.4.3 HTTP_fs_getFsize **ファイルサイズの取得**

【T】

C言語インタフェース

UW size = HTTP_fs_getFsize (B *filename);

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
UW	size	ファイルサイズ

解　説

パラメータ filename で指定したファイルのサイズを取得します。予め、HTTP_fs_find にてファイルがあることを確認してから実行する必要があります。

4.4.4 HTTP_fs_isitDir **属性がディレクトリかをチェック**

【T】

C言語インタフェース

W ercd = HTTP_fs_isitDir (B *filename);

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
W	ercd	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	
0	指定したファイルの属性はディレクトリではありません
0以外	指定したファイルの属性はディレクトリです

解　説

パラメータ filename で指定したファイルの属性がディレクトリかをチェックします。

4.4.5 HTTP_fs_isitFile **属性がファイルかをチェック**

【T】

C言語インタフェース

W ercd = HTTP_fs_isitFile (B *filename);

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
W	ercd	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	
0	指定したファイルの属性はファイルでありません（ディレクトリです）
0以外	指定したファイルの属性はファイルです（ディレクトリでありません）

解　説

パラメータ filename で指定したファイルの属性がファイル(ディレクトリでない)かをチェックします。

4.4.6 HTTP_fs_checkAuth **ユーザ認証の要否チェック**

【T】

C言語インタフェース

```
W ercd = HTTP_fs_checkAuth ( B *filename );
```

パラメータ

B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ

W	ercd	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

0	指定したファイルはユーザ認証が不要です
0以外	指定したファイルはユーザ認証が必要です

解　説

パラメータ filename で指定したファイルについて、ユーザ認証の要否をチェックします。

4.4.7 HTTP_fs_send **ファイルデータの送信**

【T】

C言語インタフェース

```
W ercd = HTTP_fs_send ( B *filename );
```

パラメータ		
B	*filename	ファイル名を格納した領域のアドレス

リターンパラメータ		
W	ercd	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	
0	指定したファイルの送信は成功しました。
0以外	指定したファイルの送信は失敗しました。

解　説

パラメータ filename で指定したファイルを送信します。送信は HTTP_writeContent サービスコールを使用して行います。送信処理では少ないスタックサイズで送信できるように、複数回に分割して送信できます。なお、HTTP_fs_getFsize で取得したファイルサイズを正しく送信しなかった場合の動作の保証はされません。

4.5 レスポンスメッセージ送信（コールバック）

4.5.1 HTTP_sendmess*XXX* レスポンスメッセージ送信

【T】

C言語インタフェース

void HTTP_sendmess*XXX* (HTTP_CTB *hcb);

XXX : 該当するレスポンスコード

パラメータ

HTTP_CTB	*hcb	コネクションコントロールブロックへのポインタ

リターンパラメータ

無し

解　説

レスポンスコードに該当するレスポンスを送信します。

5. C ライブラリ関数

5.1.1 HIcom_GetStrLenL 文字列長を計算します

【T／D／L／I】

C言語インタフェース

```
W len = HIcom_GetStrLenL ( B *buf );
```

パラメータ

B	*buf	文字列

リターンパラメータ

W	len	文字列長

解　説

文字列長を計算します

5.1.2 HIcom_GetStrLenW **文字列長を計算します**

【T／D／L／I】

C言語インタフェース

H len = HIcom_GetStrLenW (B *buf);

パラメータ

B	*buf	文字列

リターンパラメータ

H	len	文字列長

解　説

文字列長を計算します

5.1.3 HIcom_CpyStrL **文字列をコピーします**

【T／D／L／I】

C言語インタフェース

```
W len = HIcom_CpyStrL ( B *dst, B *src );
```

パラメータ

B	*dst	コピー先文字列
B	*src	コピー元文字列

リターンパラメータ

W	len	文字列長

解　説

文字列をコピーします。コピーした文字列長を返します。

5.1.4 HIcom_CpyStrW **文字列をコピーします**

【T／D／L／I】

C言語インタフェース

```
H len = HIcom_CpyStrW ( B *dst, B *src );
```

パラメータ

B	*dst	コピー先文字列
B	*src	コピー元文字列

リターンパラメータ

H	len	文字列長

解　説

文字列をコピーします。コピーした文字列長を返します。

5.1.5 HIcom_CpyStrLenL **文字列をコピーします（文字列長指定）**

【T／D／L／I】

C言語インタフェース

```
W  len = HIcom_CpyStrLenL ( B *dst, B *src, W cnt );
```

パラメータ

B	*dst	コピー先文字列
B	*src	コピー元文字列
W	cnt	コピーする文字列長

リターンパラメータ

W	len	文字列長

解　説

指定した文字列の長さをコピーします。指定した文字列のサイズが指定した文字列の長さより小さい場合、NULLまでをコピーします。コピーした文字列長を返します。

5.1.6 HIcom_CpyStrLenW **文字列をコピーします（文字列長指定）**

【T／D／L／I】

C言語インタフェース

```
H len = HIcom_CpyStrLenW ( B *dst, B *src, H cnt );
```

パラメータ

B	*dst	コピー先文字列
B	*src	コピー元文字列
H	cnt	コピーする文字列長

リターンパラメータ

H	len	文字列長

解　説

指定した文字列の長さをコピーします。指定した文字列のサイズが指定した文字列の長さより小さい場合、NULLまでをコピーします。コピーした文字列長を返します。

5.1.7　HIcom_CatStr　**文字列を結合します**

【T／D／L／I】

C言語インタフェース

```
void HIcom_CatStr ( B *dst, B *src );
```

パラメータ

B	*dst	結合先文字列
B	*src	結合元文字列

リターンパラメータ

無し

解　説

コピー先文字列の終端にコピー先文字列を追加します。

5.1.8 HIcom_CmpStr **文字列を比較します**

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStr ( B *b1, B *b2 );
```

パラメータ

B	*b1	比較文字列1
B	*b2	比較文字列2

リターンパラメータ

W	ercd	リターンコード

リターン値／エラーコード

0	一致
0以外	不一致

解　説

文字列を比較します。

5.1.9 HIcom_CmpStrLenL 文字列を比較します（文字列長指定）

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStrLenL ( B *b1, B *b2, Wcnt );
```

パラメータ

B	*b1	比較文字列1
B	*b2	比較文字列2
W	cnt	文字列長

リターンパラメータ

W	ercd	リターンコード

リターン値／エラーコード

0	一致
0以外	不一致

解　説

指定した文字列の長さを比較をします。指定した文字列のサイズが指定した文字列長さより小さい場合、NULLまでを比較対象とします。

5.1.10 HIcom_CmpStrLenW 文字列を比較します（文字列長指定）

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStrLenW ( B *b1, B *b2, Hcnt );
```

パラメータ

B	*b1	比較文字列1
B	*b2	比較文字列2
H	cnt	文字列長

リターンパラメータ

W	ercd	リターンコード

リターン値／エラーコード

0	一致
0以外	不一致

解　説

指定した文字列の長さを比較をします。指定した文字列のサイズが指定した文字列長さより小さい場合、NULLまでを比較対象とします。

5.1.11 HIcom_SearchStr **文字列から指定文字を検索します**

【T／D／L／I】

C言語インタフェース

```
B *ptr = HIcom_SearchStr ( B *buf, B ch );
```

パラメータ

B	*buf	文字列
B	ch	文字

リターンパラメータ

B	*ptr	リターンコード、または、ポインタ

リターン値／エラーコード

0	指定文字無し
0以外	指定文字の文字列位置を差すポインタ

解　説

文字列から指定文字を検索します。

5.1.12　HIcom_UpperCh　　小文字を大文字に変換します

【T／D／L／I】

C言語インタフェース

```
B ch = HIcom_UpperCh ( B asc );
```

パラメータ

B	asc	文字

リターンパラメータ

B	ch	変換後の文字

リターン値／エラーコード

0	指定文字無し
0以外	指定文字の文字列位置を差すポインタ

解　説

アルファベットの小文字を大文字に変換します。アルファベット小文字以外を指定した場合は変換しません。

5.1.13 HIcom_CmpStrUpper **大文字に変換後、文字列を比較します**

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStrUpper ( B *b1, B *b2 );
```

パラメータ

B	*b1	比較文字列1
B	*b2	比較文字列2

リターンパラメータ

W	ercd	リターンコード

リターン値／エラーコード

0	一致
0以外	不一致

解　説

アルファベットの小文字を大文字に変換して文字列の比較をします。

5.1.14 HIcom_CmpStrLenUpperL **大文字に変換後、文字列を比較します**

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStrUpper ( B *b1, B *b2, W cnt );
```

パラメータ

B	*b1	比較文字列1
B	*b2	比較文字列2
W	cnt	文字列長

リターンパラメータ

W	ercd	リターンコード

リターン値／エラーコード

0	一致
0以外	不一致

解　説

アルファベットの小文字を大文字に変換して指定した文字列の長さを比較をします。指定した文字列のサイズが指定した文字列長さより小さい場合、NULLまでを比較対象とします。

5.1.15 HIcom_CmpStrLenUpperW **大文字に変換後、文字列を比較します**

【T／D／L／I】

C言語インタフェース

```
W ercd = HIcom_CmpStrUpper ( B *b1, B *b2, H cnt );
```

パラメータ		
B	*b1	比較文字列1
B	*b2	比較文字列2
H	cnt	文字列長

リターンパラメータ		
W	ercd	リターンコード

リターン値／エラーコード	
0	一致
0以外	不一致

解　説

アルファベットの小文字を大文字に変換して指定した文字列の長さを比較をします。指定した文字列のサイズが指定した文字列長さより小さい場合、NULLまでを比較対象とします。

5.1.16　HIcom_CpyMemL　　**メモリコピーします**

【T／D／L／I】

C言語インタフェース

```
void HIcom_CpyMemL ( B *dst, B *src, W cnt );
```

パラメータ

B	*dst	コピー先ポインタ
B	*src	コピー元ポインタ
W	cnt	メモリ長

リターンパラメータ

無し

解　説

メモリコピーします。

5.1.17　HIcom_CpyMemW　　**メモリコピーします**

【T／D／L／I】

C言語インタフェース

void HIcom_CpyMemW (B *dst, B *src, H cnt);

パラメータ

B	*dst	コピー先ポインタ
B	*src	コピー元ポインタ
H	cnt	メモリ長

リターンパラメータ

無し

解　説

メモリコピーします。

5.1.18　HIcom_SetMemL　　**メモリにデータ設定します**

【T／D／L／I】

C言語インタフェース

```
W len = HIcom_SetMemL ( B *msgbuf, B c, W cnt );
```

パラメータ

B	*msgbuf	設定先ポインタ
B	c	データ
W	cnt	メモリ長

リターンパラメータ

W	len	メモリ長

解　説

メモリにデータ設定します。

5.1.19　HIcom_SetMemW　　**メモリにデータ設定します**

【T／D／L／I】

C言語インタフェース

H len = HIcom_SetMemL (B *msgbuf, B c, H cnt);

パラメータ		
B	*msgbuf	設定先ポインタ
B	c	データ
H	cnt	メモリ長

リターンパラメータ		
H	len	メモリ長

解　説

メモリにデータ設定します。

5.1.20 HIcom_h2b 16 **進文字列２文字をデータ変換します**

【T／D／L／I】

C言語インタフェース

B ch = HIcom_h2b (B *str);

パラメータ		
B	*str	16進文字列

リターンパラメータ		
B	ch	16進データ

解　説

16進文字列２文字をデータ変換します。大文字アルファベット、及び、数字の16進文字列以外の変換は保証しません。

5.1.21 HIcom_h2a データを 16 進文字列に変換します

【T／D／L／I】

C言語インタフェース

```
void HIcom_h2a ( B *str, UW hex );
```

パラメータ

B	*str	文字列格納ポインタ
UW	hex	データ

リターンパラメータ

B	*str	16進文字列

解　説

データを16進文字列に変換します。

5.1.22 HIcom_i2a　　データを 10 進文字列に変換します

【T／D／L／I】

C言語インタフェース

```
void HIcom_i2a ( B *str, UW dec );
```

パラメータ

B	*str	文字列格納ポインタ
UW	dec	データ

リターンパラメータ

B	*str	10進文字列

解　説

データを10進文字列に変換します。

5.1.23　HIcom_a2h　　16 **進文字列をデータに変換します**

【T／D／L／I】

C言語インタフェース

```
void HIcom_a2h ( B *str, UW *hex );
```

パラメータ

B	*str	16進文字列
UW	*hex	データ格納ポインタ

リターンパラメータ

UW	*hex	データ

解　説

16進文字列をデータに変換します。16進文字列が正しくない場合の変換は保証されません。変換は終端NULLを見つけるまで行います。

5.1.24 HIcom_a2i 10 進文字列をデータに変換します

【T／D／L／I】

C言語インタフェース

```
void HIcom_a2i ( B *str, UW *dec );
```

パラメータ

B	*str	10進文字列
UW	*dec	データ格納ポインタ

リターンパラメータ

UW	*dec	データ

解　説

10進文字列をデータに変換します。10進文字列が正しくない場合の変換は保証されません。変換は終端NULLを見つけるまで行います。

6. Java Package

Java Packageは、HTTPサーバのためのツール郡です。提供形態は、Javaソースファイルとなっております。実行前にコンパイルしてください。

なお、本Java Packageを使用するにあたり、Sun Microsystems, Inc.製のJava(TM) 2 SDK, Standard Editionが必要となります。本プログラムは http://java.sun.com/ からダウンロードできます。

6.1 変換ツール

6.1.1 Java cat 簡易 ROM ファイル用のコンテンツ→C ソース変換

Usage

Java Cat <Source-Dir> <Upload-Dir>

パラメータ	
<Source-Dir>	ホームページのデータを格納したディレクトリ
<Upload-Dir>	変換したCソースファイルを格納するディレクトリ

解　説

ホームページのデータを格納したディレクトリ下のファイルを、簡易ROMファイル用のCソースに変換します。

エラーメッセージ	
Usage: Java Cat <Source-Dir> <Upload-Dir>	文法エラー
(E) Source-Dir is not exist	（E）指定したソースディレクトリが存在しません。
(E) Source-Dir is not directory	（E）指定したソースディレクトリはディレクトリでない。
(I) Upload-Dir is not exist	（I）指定したアップロードディレクトリが存在しません。
(I) Creating Upload-Dir	（I）アップロードディレクトリを生成します。
(E) Can not create Upload-Dir	（E）アップロードディレクトリの生成ができません。
(E) Upload-Dir is not directory	（E）指定したアップロードディレクトリはディレクトリでない。
(E) Can not create http_user_content.c	（E）http_user_content.cの生成ができません。
(W) file name is too long	（W）指定したファイル名が長い

その他、Javaのメッセージ有り

6.2 動作確認ツール

6.2.1 Java put PUT メソッド動作確認用プログラム

Usage

Java Put [Port=<port-Number>] <Hostname> <file-name> <Content >

パラメータ

Port=<port-Number>	ポート番号（省略時は、80）
<Hostname>	ホスト名
<file-name>	ローカルファイル名
<Content>	リモートファイル名

解　説

指定したホスト上のHTTPサーバに対して、PUTメソッドを実行します。

エラーメッセージ

Usage: Java Put [Port=<port-Number>] <hostname> <file-name> <content-name>
<Port-Number> : default 80

	文法エラー
(E) Port-Number is not digit	（E）ポート番号不正。
(E) File is not exist	（E）ファイルが存在しません
(E) <file-name> isdirectory	（E）指定したファイルはディレクトイです。

その他、Java、HTTPサーバのメッセージ有り

6.2.2 Java delete DELETE メソッド動作確認用プログラム

Usage

Java Delete [Port=<port-Number>] <Hostname> <Content>

パラメータ

Port=<port-Number>	ポート番号（省略時は、80）
<Hostname>	ホスト名
<Content>	リモートファイル名

解　説

指定したホスト上のHTTPサーバに対して、DELETEメソッドを実行します。

エラーメッセージ

Usage: Java Delete [Port=<port-Number>] <Hostname> <Content>
<Port-Number> : default 80

	文法エラー
(E) Port-Number is not digit	（E）ポート番号不正。
	その他、Java、HTTPサーバのメッセージ有り

付録A　　サービスコール エラーコード一覧

表A-1　HTTPサーバのサービス コールエラーコード

エラーコード	値	説　明
E_NG	－０ｘ０１	エラーリターン
E_SOCK	－０ｘ０１	TCP通信エラー
E_SYS	－０ｘ０５	システムエラー
E_NOMEM	－０ｘ０Ａ	メモリ不足
E_PAR	－０ｘ２１	パラメータエラー
E_OBJ	－０ｘ３Ｆ	オブジェクト状態不正

付録B　　制限事項

本 HTTP サーバは、HTTP/1.1 仕様に準拠しておりますが、ご使用にあたり以下の制限事項がございますので、ご確認ください。

B.1　　ヘッダフィールドと処理

ヘッダフィールドと本 HTTP サーバの動作を記載します。

（1） リクエストヘッダフィールド

説明にて記載される*hcb は構造体 HTTP_CTB の領域を指すポインタでユーザ CGI コールバック関数の引数です。必要であれば、ユーザ CGI コールバック関数内で参照することができます。

- Accept

  hcb->md.accept にコピー（但し、領域のサイズはNull含む256文字まで）

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Accept-Charset

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Accept-Encoding

  hcb->md.encording にコピー（但し、領域のサイズはNull含む128文字まで）

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Accept-Language

  hcb->md.lang にコピー（但し、領域のサイズはNull含む32文字まで）。

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Authorization

  hcb->md.auth にコピー（但し、領域のサイズはNull含む128文字まで）。

  本HTTPサーバは本リクエストヘッダを処理します（**但し、サポートする認証方法は基本認証：Base64のみ**）

- From

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Host

  hcb->md.host にコピー（但し、領域のサイズはNull含む128文字まで）

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- If-Modified-Since

  hcb->modified に時刻をクロック変換して格納

  本HTTPサーバは本リクエストヘッダを処理します（但し、GET,HEADメソッドのみ）

- If-Match

  本HTTPサーバは本リクエストヘッダフィールドを無視します

- If-None-Match
  hcb->md.etag にコピー（但し、領域のサイズはNull含む128文字まで）

  本HTTPサーバは本リクエストヘッダを処理します（**但し、**GET,HEADメソッドのみ。**弱い比較のみサポート。強い比較（"*"含む）は、未サポート**）

- If-Range
  本HTTPサーバは本リクエストヘッダフィールドを無視します

- If-Unmodified-Since
  hcb->unmodified に時刻をクロック変換して格納。

  本HTTPサーバは本リクエストヘッダを処理します（但し、GET,HEADメソッドのみ）

- Max-Forward
  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Proxy-Authorization
  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Range
  本HTTPサーバは本リクエストヘッダフィールドを無視します

- Referer
  本HTTPサーバは本リクエストヘッダフィールドを無視します

- User-Agent
  hcb->md.agent にコピー（但し、領域のサイズはNull含む128文字まで）。

  本HTTPサーバは本リクエストヘッダフィールドを無視します

（2） エンティティヘッダフィールド

説明にて記載される*hcb は構造体 HTTP_CTB の領域を指すポインタでユーザ CGI コールバック関数の引数です。必要であれば、ユーザ CGI コールバック関数内で参照することができます。

- Allow
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはOPTIONメソッドのレスポンス、または、"HTTP/1.1 405 Method Not Allowed"レスポンスにて使用します。

- Content-Base
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-Encoding
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-Language
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-Length
  hcb->md.cont_len にコピー（但し、領域のサイズはNull含む16文字まで）

  本HTTPサーバはリクエストで本エンティティヘッダフィールドを処理します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用します

- Content-Location
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-MD5
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-Range
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本エンティティヘッダフィールドを使用しません

- Content-Type
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはGET/POST/HEADメソッドのレスポンスにて使用します。

- Etag
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはGET/ HEADメソッドのレスポンスにて使用します。

- Last-Modified
  本HTTPサーバはリクエストで本エンティティヘッダフィールドを無視します

  本HTTPサーバはGET/ HEADメソッドのレスポンスにて使用します。

（3） 汎用ヘッダフィールド

説明にて記載される*hcb は構造体 HTTP_CTB の領域を指すポインタでユーザ CGI コールバック関数の引数です。必要であれば、ユーザ CGI コールバック関数内で参照することができます。

- Connection
  hcb->connection に"Keep-Alive"なら0、それ以外なら1を設定します。

  本HTTPサーバはリクエストで本汎用ヘッダフィールドを処理します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用します

- Date
  本HTTPサーバはリクエストで本汎用ヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用します

- Pragma
  本HTTPサーバはリクエストで本汎用ヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用しません

- Transfer-Encoding

  本HTTPサーバはリクエストで本汎用ヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用します（**但し、**POSTメソッドのみ**。"chunked"のみサポート**）

- Upgrade

  本HTTPサーバはリクエストで本汎用ヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用しません

- Via

  本HTTPサーバはリクエストで本汎用ヘッダフィールドを無視します

  本HTTPサーバはレスポンスで本汎用ヘッダフィールドを使用しません

（4） キャッシュコントロール

- Age

  本HTTPサーバはリクエストで本キャッシュコントロールを無視します

  本HTTPサーバはレスポンスで本キャッシュコントロールを使用しません

- Cache-Control

  本HTTPサーバはリクエストで本キャッシュコントロールを無視します

  本HTTPサーバはレスポンスで本キャッシュコントロールを使用しません

- Expires

  本HTTPサーバはリクエストで本キャッシュコントロールを無視します

  本HTTPサーバはレスポンスで本キャッシュコントロールを使用しません

- Warning

  本HTTPサーバはリクエストで本キャッシュコントロールを無視します

  本HTTPサーバはレスポンスで本キャッシュコントロールを使用しません

（5） レスポンスヘッダフィールド

- Accept-Ranges

  本HTTPサーバは本レスポンスヘッダフィールドを使用します（**但し、**GET,HEAD,OPTIONメソッドのみ。**指定は"none"のみ**）

- Authentication-Info

  本HTTPサーバは本レスポンスヘッダフィールドを使用しません

- Location

  本HTTPサーバは本レスポンスヘッダフィールドを使用しません

- Proxy-Authenticate

  本HTTPサーバは本レスポンスヘッダフィールドを使用しません

- Proxy-Authenticate-info

  本HTTPサーバは本レスポンスヘッダフィールドを使用しません

- Public

  本HTTPサーバは本レスポンスヘッダフィールドを使用します（但し、OPTIONメソッドのみ）

- Retry-After

  本HTTPサーバはレスポンスで本レスポンスヘッダフィールドを使用しません

- Server

  本HTTPサーバは本レスポンスヘッダフィールドを使用します

- Vary

  本HTTPサーバはレスポンスで本レスポンスヘッダフィールドを使用しません

- Warning

  本HTTPサーバはレスポンスで本レスポンスヘッダフィールドを使用しません

- WWW-Authenticate

  本HTTPサーバはレスポンスで本レスポンスヘッダフィールドを使用しません

（5） その他

- 総バイト数が 2047（Null 含む、但し、改行コード 0x0a,0x0d は含まない）を超えるリクエストヘッダを受信した場合、本 HTTP サーバはレスポンスメッセージ"HTTP/1.1 413 Request Entity Too Large"を返し、そのリクエストの処理は行ないません。

- 総バイト数が 255（Null 含む）を超えるコンテンツ名がリクエストされた場合、本 HTTP サーバはレスポンスメッセージ"HTTP/1.1 414 Request-URI Too Large"を返し、そのリクエストの処理は行ないません。なお、コンテンツ名のサイズは、パラメタ部（ http://ホスト名 /xxxx.cgi**?p1=xx&p2=yyyy**"）も含みます。

B.2　コールバック全般

（1） コールバック関数内で無限ループやタスクウエイトに入った場合、HTTP サーバタスク自体が無限ループやウエイトに入ることを意味します。その場合、HTTP サーバのサービスは処理されません。

B.3　ユーザCGI関連

（1） コンテンツの送信をチャンク形式エンコーディング方式で行なうとき、初めに HTTP_SendChukHeader サービスコールを発行せずに HTTP_SendChukData、HTTP_SendChukEnd サービスコールを発行した場合、動作は保証されません。

（2） コンテンツの送信をチャンク形式エンコーディング方式で行なうとき、最後に HTTP_SendChukEnd サービスコールを発行しない場合、動作は保証されません。

（3） コンテンツの送信で HTTP_SendChukHeader、HTTP_SendChukData、HTTP_SendChukEnd サービスコールと、HTTP_ResponseTransfer サービスコールを併用して使用した場合、動作は保証されません。

（4） ユーザ CGI コールバック関数でコンテンツの送信を行なわなかった場合、動作は保証されません。

（5） ユーザ CGI コールバック関数は HTTP サーバタスクとして呼び出されます。ユーザ CGI コールバック関数で時間がかかる処理を実行した場合、HTTP サーバの性能が低下するおそれがあります。

（6） ユーザ CGI コールバック関数から、別なタスクを起動してそのタスクからコンテンツの送信を行なうことはできません（制御等、コンテンツ送信以外の目的で別タスクを起動することはできます）。

（7） HTTP サーバからコールバックされたユーザ CGI コールバック関数以外から、HTTP_SendChukHeader、HTTP_SendChukData、HTTP_SendChukEnd、HTTP_ResponseTransfer サービスコールを発行することはできません。

B.4　ファイルシステムAPI仕様（コールバック）関連

（１） HTTP_fs_send にて、HTTP_fs_Fsize で取得したファイルサイズを正しく送信しなかった場合、動作の保証はされません。

B.5　HTTPサーバ全般

（１） 使用している TCP 通信端点の数に対して、リクエスト要求が多い環境で使用した場合、コネクションが確立できるまでリクエストが保留される、またはコネクションの確立に失敗する場合があります[*1]。以下の対策を行なってください。

- ご使用形態に応じて TCP 通信端点の数を調整（増やす）してください（推奨）。
- HTTP サーバの構築情報「持続型接続時間」にて、Keep-Alive する時間を短く設定してください。設定方法の詳細は HTTP サーバの構築マニュアルを参照してください。

[*1] HTTP/1.1 の持続型接続機能により、コンテンツ送信完了後も接続は保持されます。ブラウザ側が接続を切断しないとき、接続が残る場合があります。但し、Keep-Alive 時間（デフォルト値 15 秒）が経過した後に、接続を強制切断（RST を送信）します。

（２） 正しい日付・時刻（正しい OS 時間の動作）でない状態での HTTP サーバの動作は保証されません。ハードウェア機構（RTC： Real Time clock など）による時間管理、または NTP（ : Network Time Protocol ）等を利用することをお勧めします。

（３） 日本語のコンテンツ名（例： URI="http://210.2.3.131/あいうえお.htm" 等）はサポートしておりません。

（４） 本 HTTP サーバは SSL（Secure Sockets Layer）対応サーバではありません。

（５） HTTPサーバは使用するTCP通信端点IDをTCP_CEPIDANY(0)指定で確保します。同一システム上で通信を行なうプログラムを動作させる場合、HTTPサーバが使用中のTCP通信端点IDと重複しない様にTCP_CEPIDANY(0)指定にて確保することを推奨します。

（６） TCP受付口は１つ使用します。使用するTCP受付口IDは、TCP_REPIDANY(0)にて確保されます。

B.6　動作確認済みのブラウザ

本 HTTP サーバは、以下のブラウザにて動作確認を行っております。

- Internet Explorer 5.00（Microsoft）
- Netscape 6.01（Netscape Communications Corporation）

HI.CommunicationEngine
HTTPサーバ リファレンスマニュアル
CM7000HTD02J-7

発行年月　　2008年　9月　　第7版
発　　行　　株式会社 ルネサス北日本セミコンダクタ
編　　集　　株式会社 ルネサス北日本セミコンダクタ