

RFC 3875 - The Common Gateway Interface (CGI) Version 1.1 日本語訳

原文URL : <https://datatracker.ietf.org/doc/html/rfc3875>

タイトル : **RFC 3875 - Common Gatewayインタフェース (CGI) バージョン1.1**

翻訳編集 : 自動生成 + 一部修正, ST: **Informational**

Network Working Group	D. Robinson
Request for Comments: 3875	K. Coar
Category: Informational	The Apache Software Foundation
	October 2004

The Common Gateway Interface (CGI) Version 1.1

Common Gatewayインタフェース (CGI) バージョン1.1

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2004).

IESG Note

This document is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this document for any purpose, and in particular notes that it has not had IETF review for such things as security, congestion control or inappropriate interaction with deployed protocols. The RFC Editor has chosen to publish this document at its discretion. Readers of this document should exercise caution in evaluating its value for implementation and deployment.

Abstract

The Common Gateway Interface (CGI) is a simple interface for running external programs, software or gateways under an information server in a platform-independent manner. Currently, the supported information servers are HTTP servers.

The interface has been in use by the World-Wide Web (WWW) since 1993. This specification defines the 'current practice' parameters of the 'CGI/1.1' interface developed and documented at the U.S. National Centre for Supercomputing Applications. This document also defines the use of the CGI/1.1 interface on UNIX(R) and other, similar systems.

Table of Contents

1.	Introduction.	4
1.1.	Purpose	4
1.2.	Requirements	4
1.3.	Specifications	4
1.4.	Terminology	5
2.	Notational Conventions and Generic Grammar.	5
2.1.	Augmented BNF	5
2.2.	Basic Rules	6
2.3.	URL Encoding	7
3.	Invoking the Script	8
3.1.	Server Responsibilities	8
3.2.	Script Selection	9
3.3.	The Script-URI	9
3.4.	Execution	10

本文書の位置付け

このメモはインターネットコミュニティに関する情報を提供します。インターネット規格はあらゆる種類の標準を指定していません。このメモの分布は無制限です。

著作権表示

著作権 (C) インターネット社会 (2004)。

IESGノート

この文書は、インターネット規格の任意のレベルの候補者ではありません。IETFは、この文書の適応度についての知識を否定しており、特にセキュリティ、輻輳制御、または展開されたプロトコルとの不適切な対話などのようなものについてのIETFレビューを受けていないことを注意しています。RFCエディタは、この文書をその裁量で公開することを選択しました。この文書の読者は、実装と展開のためにその価値を評価する際に注意を払うべきです。

概要

Common Gatewayインタフェース (CGI) は、プラットフォームに依存しない方法で、情報サーバーの下の外部プログラム、ソフトウェア、またはゲートウェイを実行するための簡単なインターフェイスです。現在、サポートされている情報サーバーはHTTPサーバーです。

このインターフェイスは、1993年以来ワールドワイドWeb (WWW) によって使用されています。この仕様は、スーパーコンピュータリングアプリケーションの米国国立センターに開発および文書化された「CGI/1.1」インターフェイスの「現在の練習」パラメータを定義しています。この文書は、UNIX (R) およびその他のシステムでCGI/1.1インターフェイスの使用も定義しています。

目次

4.	The CGI Request	10
4.1.	Request Meta-Variables	10
4.1.1.	AUTH_TYPE.	11
4.1.2.	CONTENT_LENGTH	12
4.1.3.	CONTENT_TYPE	12
4.1.4.	GATEWAY_INTERFACE.	13
4.1.5.	PATH_INFO.	13
4.1.6.	PATH_TRANSLATED.	14
4.1.7.	QUERY_STRING	15
4.1.8.	REMOTE_ADDR.	15
4.1.9.	REMOTE_HOST.	16
4.1.10.	REMOTE_IDENT	16
4.1.11.	REMOTE_USER.	16
4.1.12.	REQUEST_METHOD	17
4.1.13.	SCRIPT_NAME.	17
4.1.14.	SERVER_NAME.	17
4.1.15.	SERVER_PORT.	18
4.1.16.	SERVER_PROTOCOL.	18
4.1.17.	SERVER_SOFTWARE.	19
4.1.18.	Protocol-Specific Meta-Variables	19
4.2.	Request Message-Body	20
4.3.	Request Methods	20
4.3.1.	GET.	20
4.3.2.	POST	21
4.3.3.	HEAD	21
4.3.4.	Protocol-Specific Methods.	21
4.4.	The Script Command Line.	21

5.	NPH Scripts	22
5.1.	Identification	22
5.2.	NPH Response	22

6.	CGI Response.	23
6.1.	Response Handling.	23
6.2.	Response Types	23
6.2.1.	Document Response.	23
6.2.2.	Local Redirect Response.	24
6.2.3.	Client Redirect Response	24
6.2.4.	Client Redirect Response with Document	24
6.3.	Response Header Fields	25
6.3.1.	Content-Type	25
6.3.2.	Location	26
6.3.3.	Status	26
6.3.4.	Protocol-Specific Header Fields.	27
6.3.5.	Extension Header Fields.	27
6.4.	Response Message-Body.	28

7.	System Specifications	28
7.1.	AmigaDOS	28
7.2.	UNIX	28
7.3.	EBCDIC/POSIX	29

8.	Implementation.	29
8.1.	Recommendations for Servers.	29
8.2.	Recommendations for Scripts.	30

9.	Security Considerations	30
9.1.	Safe Methods	30
9.2.	Header Fields Containing Sensitive Information	31
9.3.	Data Privacy	31
9.4.	Information Security Model	31
9.5.	Script Interference with the Server.	31
9.6.	Data Length and Buffering Considerations	32
9.7.	Stateless Processing	32
9.8.	Relative Paths	33
9.9.	Non-parsed Header Output	33

10.	Acknowledgements.	33
---------------------	---------------------------	----

11.	References.	33
11.1.	Normative References.	33
11.2.	Informative References.	34

12.	Authors' Addresses.	35
---------------------	-----------------------------	----

13.	Full Copyright Statement.	36
---------------------	-----------------------------------	----

1. Introduction

1.1. Purpose

The Common Gateway Interface (CGI) [22] allows an HTTP [1], [4] server and a CGI script to share responsibility for responding to client requests. The client request comprises a Uniform Resource Identifier (URI) [11], a request method and various ancillary information about the request provided by the transport protocol.

The CGI defines the abstract parameters, known as meta-variables, which describe a client's request. Together with a concrete programmer interface this specifies a platform-independent interface between the script and the HTTP server.

The server is responsible for managing connection, data transfer, transport and network issues related to the client request, whereas the CGI script handles the application issues, such as data access and document processing.

1.2. Requirements

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY' and 'OPTIONAL' in this document are to be interpreted as described in BCP 14, RFC 2119 [3].

An implementation is not compliant if it fails to satisfy one or more of the 'must' requirements for the protocols it implements. An implementation that satisfies all of the 'must' and all of the 'should' requirements for its features is said to be 'unconditionally compliant'; one that satisfies all of the 'must' requirements but not all of the 'should' requirements for its features is said to be 'conditionally compliant'.

1.3. Specifications

Not all of the functions and features of the CGI are defined in the main part of this specification. The following phrases are used to describe the features that are not specified:

'system-defined' The feature may differ between systems, but must be the same for different implementations using the same system. A system will usually identify a class of operating systems. Some systems are defined in section 7 of this document. New systems may be defined by new specifications without revision of this document.

'implementation-defined' The behaviour of the feature may vary from implementation to implementation; a particular implementation must document its behaviour.

1.4. Terminology

This specification uses many terms defined in the HTTP/1.1 specification [4]; however, the following terms are used here in a sense which may not accord with their definitions in that document, or with their common meaning.

'meta-variable' A named parameter which carries information from the server to the script. It is not necessarily a variable in the operating system's environment, although that is the most common implementation.

1. はじめに

1.1. 目的

Common Gateway Interface (CGI) [22]は、HTTP [1]、[4]サーバーとCGIスクリプトがクライアント要求に応答する責任を共有できます。クライアント要求は、統一されたリソース識別子 (URI) [11]、要求方法およびトランスポートプロトコルによって提供される要求に関する様々な付随情報を含む。

CGIは、クライアントの要求を記述するメタ変数として知られる抽象パラメータを定義します。具体的なプログラムのインタフェースと一緒に、これはスクリプトとHTTPサーバー間のプラットフォームに依存しないインターフェースを指定します。

サーバーは、クライアント要求に関連する接続、データ転送、トランスポート、ネットワークの問題を管理する責任がありますが、CGIスクリプトはデータアクセスや文書処理などのアプリケーションの問題を処理します。

1.2. 要件

「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「MAY」および「OPTIONAL」はBCP 14、RFC 2119 [3]に記載されているように解釈される。

実装が実行されたプロトコルの要件の1つ以上を満たすことができない場合は、実装は準拠していません。その特徴の「必須」のすべての要件をすべて満たす実装は、「無条件に準拠」と言われています。その特徴に対する「必須」要件のすべてを満たすが、そのすべての要件のすべてを満たすものではないものは、「条件付き準拠」と言われています。

1.3. 仕様

CGIのすべての機能と機能がこの仕様の主要部分で定義されていません。以下のフレーズは、指定されていない機能を説明するために使用されます。

「システム定義」機能はシステム間で異なる場合がありますが、同じシステムを使用した異なる実装については同じでなければなりません。システムは通常、クラスのオペレーティングシステムを識別します。一部のシステムはこの文書のセクション7で定義されています。この文書の改訂なしに新しい仕様で新しいシステムを定義することができます。

「実装定義」機能の動作は実装と実装に異なる場合があります。特定の実装はその動作を文書化する必要があります。

1.4. 用語

この仕様では、HTTP/1.1仕様で定義されている多くの用語を使用しています[4];ただし、以下の用語は、その文書内のそれらの定義とは関係なく、またはそれらの共通の意味ではない意味で、ここで使用されています。

'meta-variable'サーバーからスクリプトに情報を伝送する名前付きパラメータ。それは必ずしもオペレーティングシステムの環境では変数ではありませんが、最も一般的な実装です。

'script' The software that is invoked by the server according to this interface. It need not be a standalone program, but could be a dynamically-loaded or shared library, or even a subroutine in the server. It might be a set of statements interpreted at run-time, as the term 'script' is frequently understood, but that is not a requirement and within the context of this specification the term has the broader definition stated.

'server' The application program that invokes the script in order to service requests from the client.

2. Notational Conventions and Generic Grammar

2.1. Augmented BNF

All of the mechanisms specified in this document are described in both prose and an augmented Backus-Naur Form (BNF) similar to that used by RFC 822 [13]. Unless stated otherwise, the elements are case-sensitive. This augmented BNF contains the following constructs:

name = definition The name of a rule and its definition are separated by the equals character ('='). Whitespace is only significant in that continuation lines of a definition are indented.

"literal" Double quotation marks (") surround literal text, except for a literal quotation mark, which is surrounded by angle-brackets ('<' and '>').

rule1 | rule2 Alternative rules are separated by a vertical bar ('|').

(rule1 rule2 rule3) Elements enclosed in parentheses are treated as a single element.

rule A rule preceded by an asterisk ('') may have zero or more occurrences. The full form is 'n*m rule' indicating at least n and at most m occurrences of the rule. n and m are optional decimal values with default values of 0 and infinity respectively.

[rule] An element enclosed in square brackets ('[' and ']') is optional, and is equivalent to '*1 rule'.

N rule A rule preceded by a decimal number represents exactly N occurrences of the rule. It is equivalent to 'N*N rule'.

2.2. Basic Rules

This specification uses a BNF-like grammar defined in terms of characters. Unlike many specifications which define the bytes allowed by a protocol, here each literal in the grammar corresponds to the character it represents. How these characters are represented in terms of bits and bytes within a system are either system-defined or specified in the particular context. The single exception is the rule 'OCTET', defined below.

The following rules are used throughout this specification to describe basic parsing constructs.

'script' このインターフェイスに従ってサーバーによって呼び出されるソフトウェア。スタンドアロンのプログラムである必要はありませんが、動的にロードされたライブラリまたは共有ライブラリ、またはサーバー内のサブルーチンである可能性があります。「スクリプト」という用語が頻繁に理解されるので、実行時に解釈される一連のステートメントかもしれませんが、その要件ではなく、この仕様の文脈内では、この用語はより広い定義が記載されています。

'Server' クライアントからの要求をサービスするためにスクリプトを呼び出すアプリケーションプログラム。

2. 表記規則と一般文法

2.1. Augmented BNF

この文書で指定されたすべてのメカニズムは、RFC 822で使用されているものと同様の散文と増強されたバックス-ナウア記法（BNF）の両方で説明されています[13]。特に記載されていない限り、要素は大文字と小文字が区別されます。このAugmented BNFには、次の構成要素が含まれています。

name = 定義ルールとその定義の名前は、等文字（'='）で区切られています。空白は、定義の継続行がインデントされているという点でのみ重要です。

リテラル引用符を除いて、リテラルのテキストを囲みます（'<'、'>'）。

ルール1

（rule1 rule2 rule3） 括弧内に囲まれた要素は単一の要素として扱われます。

ルール アスタリスク（''）が先行するルールは、0以上の出現箇所を持つことができます。フルフォームは、ルールの少なくともnと最大のmの出現を示す「n * mのルール」です。nとmはそれぞれ0と無限大のデフォルト値を持つオプションの10進値です。

[ルール] 角括弧（'['と']'）で囲まれた要素はオプションであり、'* 1ルール'に相当します。

nルール 10進数が前のルールは、ルールの正確にN個の発生を表します。それは'n*nルール'と同じです。

2.2. 基本的なルール

この仕様では、文字の観点から定義されているBNFのような文法を使用します。プロトコルによって許可されているバイトを定義する多くの仕様とは異なり、ここで文法内の各文字通りはそれが表す文字に対応します。これらの文字がシステム内のビットとバイトの点でどのように表されるかは、システム定義または特定のコンテキストで指定されています。単一の例外は、以下に定義されているルール 'Octet'です。

基本的な解析構成要素を説明するには、この仕様を通して以下の規則が使用されます。

alpha	=	lowalpha hialpha
lowalpha	=	"a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
hialpha	=	"A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X"

"Y" "Z"

digit	= "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
alphanum	= alpha digit
OCTET	= <any 8-bit byte>
CHAR	= alpha digit separator "!" "#" "\$" "%" "&" "'" "*" "+" "-" "." "`" "^" "_" "{" " " "}" "~" CTL
CTL	= <any control character>
SP	= <space character>
HT	= <horizontal tab character>
NL	= <newline>
LWSP	= SP HT NL
separator	= "(" ")" "<" ">" "@" "," ";" ":" "\" "<" "/" "[" "]" "?" "=" "{" "}" SP HT
token	= 1*<any CHAR except CTLs or separators>
quoted-string	= "<" *qdtxt ">"
qdtxt	= <any CHAR except "<" and CTLs but including LWSP>
TEXT	= <any printable character>

Note that newline (NL) need not be a single control character, but can be a sequence of control characters. A system MAY define TEXT to be a larger set of characters than <any CHAR excluding CTLs but including LWSP>.

2.3. URL Encoding

Some variables and constructs used here are described as being 'URL-encoded'. This encoding is described in section 2 of RFC 2396 [2]. In a URL-encoded string an escape sequence consists of a percent character ("%") followed by two hexadecimal digits, where the two hexadecimal digits form an octet. An escape sequence represents the graphic character that has the octet as its code within the US-ASCII [9] coded character set, if it exists. Currently there is no provision within the URI syntax to identify which character set non-ASCII codes represent, so CGI handles this issue on an ad-hoc basis.

Note that some unsafe (reserved) characters may have different semantics when encoded. The definition of which characters are unsafe depends on the context; see section 2 of RFC 2396 [2], updated by RFC 2732 [7], for an authoritative treatment. These reserved characters are generally used to provide syntactic structure to the character string, for example as field separators. In all cases, the string is first processed with regard to any reserved characters present, and then the resulting data can be URL-decoded by replacing "%" escape sequences by their character values.

To encode a character string, all reserved and forbidden characters are replaced by the corresponding "%" escape sequences. The string can then be used in assembling a URI. The reserved characters will vary from context to context, but will always be drawn from this set:

reserved	= ";" "/" "?" ":" "@" "&" "=" "+" "\$" "," "[" "]"
----------	--

The last two characters were added by RFC 2732 [7]. In any particular context, a sub-set of these characters will be reserved; the other characters from this set MUST NOT be encoded when a string is URL-encoded in that context. Other basic rules used to describe URI syntax are:

hex	= digit "A" "B" "C" "D" "E" "F" "a" "b" "c" "d" "e" "f"
-----	---

改行（NL）は単一の制御文字である必要はありませんが、一連の制御文字です。システムは、CTLを除くがLWSP>を含む任意の文字よりも大きな文字のセットとなるようにテキストを定義することができます。

2.3. URLエンコーディング

ここで使用されるいくつかの変数と構成は、「URLエンコード」として説明されています。この符号化はRFC 2396 [2]の第2章で説明されています。URLエンコードされた文字列では、エスケープシーケンスはパーセント文字（"%") とそれに続く2つの16進数字で構成され、ここで2つの16進数はオクテットを形成します。エスケープシーケンスは、存在する場合は、us-ascii [9] コード化文字セット内のコードとしてオクテットを持つグラフィック文字を表します。現在、どの文字セットの非ASCIIコードを表すかを識別するためにURI構文内にプロビジョニングされているため、CGIはこの問題をアドホックに基づいて処理します。

符号化されたときに、一部の安全でない（予約済み）文字は異なる意味を持つことがあります。どの文字が安全でないかの定義はコンテキストによって異なります。権威ある治療のために、RFC 2732 [7]によって更新されたRFC 2396 [2]のセクション2を参照のこと。これらの予約された文字は、一般に、フィールド区切り文字として文字列に構文構造を提供するために使用されます。すべての場合において、文字列は最初に存在する予約文字に関して処理され、結果として得られるデータは "%" エスケープシーケンスを文字値で置き換えることによってURLデコードされます。

文字列をエンコードするために、すべての予約された文字と禁止文字は対応する "%" エスケープシーケンスに置き換えられます。その後、文字列をURIの組み立てに使用できます。予約された文字はコンテキストごとに異なりますが、常にこのセットから描画されます。

最後の2文字はRFC 2732 [7]によって追加されました。特定の文脈では、これらの文字のサブセットは予約されます。このセットからの他の文字は、文字列がそのコンテキスト内でURLエンコードされているときにエンコードされてはなりません。URI構文を説明するために使用される他の基本的な規則は次のとおりです。

Orig


```
escaped    = "%" hex hex
unreserved = alpha | digit | mark
mark       = "-" | "_" | "." | "!" | "~" | "*" | "'" | "(" | ")"
```

3. Invoking the Script

3.1. Server Responsibilities

The server acts as an application gateway. It receives the request from the client, selects a CGI script to handle the request, converts the client request to a CGI request, executes the script and converts the CGI response into a response for the client. When processing the client request, it is responsible for implementing any protocol or transport level authentication and security. The server MAY also function in a 'non-transparent' manner, modifying the request or response in order to provide some additional service, such as media type transformation or protocol reduction.

The server MUST perform translations and protocol conversions on the client request data required by this specification. Furthermore, the server retains its responsibility to the client to conform to the relevant network protocol even if the CGI script fails to conform to this specification.

If the server is applying authentication to the request, then it MUST NOT execute the script unless the request passes all defined access controls.

3.2. Script Selection

The server determines which CGI is script to be executed based on a generic-form URI supplied by the client. This URI includes a hierarchical path with components separated by "/". For any particular request, the server will identify all or a leading part of this path with an individual script, thus placing the script at a particular point in the path hierarchy. The remainder of the path, if any, is a resource or sub-resource identifier to be interpreted by the script.

Information about this split of the path is available to the script in the meta-variables, described below. Support for non-hierarchical URI schemes is outside the scope of this specification.

3.3. The Script-URI

The mapping from client request URI to choice of script is defined by the particular server implementation and its configuration. The server may allow the script to be identified with a set of several different URI path hierarchies, and therefore is permitted to replace the URI by other members of this set during processing and generation of the meta-variables. The server

1. MAY preserve the URI in the particular client request; or
2. it MAY select a canonical URI from the set of possible values for each script; or
3. it can implement any other selection of URI from the set.

From the meta-variables thus generated, a URI, the 'Script-URI', can be constructed. This MUST have the property that if the client had accessed this URI instead, then the script would

3. スクリプトを呼び出す

3.1. サーバーの責務

サーバーはアプリケーションゲートウェイとして機能します。クライアントから要求を受け取り、要求を処理するためのCGIスクリプトを選択し、クライアント要求をCGI要求に変換し、スクリプトを実行し、CGI応答をクライアントの応答に変換します。クライアント要求を処理するときは、プロトコルまたはトランスポートレベルの認証とセキュリティを実装する責任があります。サーバは、メディアタイプの変換やプロトコルの削減などのいくつかの追加のサービスを提供するために、要求または応答を変更することもできます。

サーバーは、この仕様書に必要なクライアント要求データに対して翻訳とプロトコル変換を実行する必要があります。さらに、サーバは、CGIスクリプトがこの仕様に適合しなくても、クライアントに対する責任を依頼して関連するネットワークプロトコルに準拠しています。

サーバーが要求に認証を適用している場合は、要求が定義されたすべてのアクセスコントロールに渡されない限り、スクリプトを実行しないでください。

3.2. スクリプト選択

サーバーは、クライアントから提供されたGeneric形式のURIに基づいて実行されるCGIが実行されるスクリプトであるかを決定します。このURIには、"/"で区切られたコンポーネントを持つ階層パスが含まれています。特定の要求に対して、サーバーは個々のスクリプトを使用してこのパスの全部または先頭の部分を識別し、パス階層内の特定の点にスクリプトを配置します。パスの残りの部分があれば、スクリプトによって解釈されるリソースまたはサブリソース識別子です。

このパスのこの分割に関する情報は、後述するメタ変数のスクリプトで利用可能です。非階層型URI方式のサポートは、この仕様の範囲外です。

3.3. Script-URI

クライアント要求URIからスクリプトの選択へのマッピングは、特定のサーバーの実装とその構成によって定義されます。サーバは、スクリプトがいくつかの異なるURIパス階層のセットで識別されることを可能にし得るので、メタ変数の処理および生成中にこのセットの他のメンバによってURIを置き換えることが許可される。サーバー

1. 特定のクライアント要求でURIを保存することがあります。または
2. それは各スクリプトに対して可能な値のセットから標準的なURIを選択するかもしれません。または
3. それはセットから他のURIの選択を実行することができます。

このようにして生成されたメタ変数から、URI、「スクリプト - URI」を構築することができる。これには、クライアントが代わりにこのURIにアクセスした場合、スクリプトはscript_name、

have been executed with the same values for the SCRIPT_NAME, PATH_INFO and QUERY_STRING meta-variables. The Script-URI has the structure of a generic URI as defined in section 3 of RFC 2396 [2], with the exception that object parameters and fragment identifiers are not permitted. The various components of the Script-URI are defined by some of the meta-variables (see below);

```
script-URI = <scheme> "://" <server-name> ":" <server-port>
            <script-path> <extra-path> "?" <query-string>
```

where <scheme> is found from SERVER_PROTOCOL, <server-name>, <server-port> and <query-string> are the values of the respective meta-variables. The SCRIPT_NAME and PATH_INFO values, URL-encoded with ";", "=" and "?" reserved, give <script-path> and <extra-path>.

See section 4.1.5 for more information about the PATH_INFO meta-variable.

The scheme and the protocol are not identical as the scheme identifies the access method in addition to the application protocol. For example, a resource accessed using Transport Layer Security (TLS) [14] would have a request URI with a scheme of https when using the HTTP protocol [19]. CGI/1.1 provides no generic means for the script to reconstruct this, and therefore the Script-URI as defined includes the base protocol used. However, a script MAY make use of scheme-specific meta-variables to better deduce the URI scheme.

Note that this definition also allows URIs to be constructed which would invoke the script with any permitted values for the path-info or query-string, by modifying the appropriate components.

3.4. Execution

The script is invoked in a system-defined manner. Unless specified otherwise, the file containing the script will be invoked as an executable program. The server prepares the CGI request as described in section 4; this comprises the request meta-variables (immediately available to the script on execution) and request message data. The request data need not be immediately available to the script; the script can be executed before all this data has been received by the server from the client. The response from the script is returned to the server as described in sections 5 and 6.

In the event of an error condition, the server can interrupt or terminate script execution at any time and without warning. That could occur, for example, in the event of a transport failure between the server and the client; so the script SHOULD be prepared to handle abnormal termination.

4. The CGI Request

Information about a request comes from two different sources; the request meta-variables and any associated message-body.

4.1. Request Meta-Variables

Meta-variables contain data about the request passed from the server to the script, and are accessed by the script in a system-defined manner. Meta-variables are identified by case-insensitive names; there cannot be two different variables

path_info、およびquery_stringメタ変数の同じ値でスクリプトが実行されたことがあります。スクリプト-URIは、オブジェクトパラメータとフラグメント識別子が許可されていないことを除いて、RFC 2396 [2]のセクション3で定義されている一般的なURIの構造を持ちます。スクリプト-URIのさまざまなコンポーネントは、いくつかのメタ変数によって定義されます（下記参照）。

<Scheme>はserver_protocol、<server-name>、<server-port>、および<query-string>から見つかります。それぞれのメタ変数の値です。script_nameとpath_infoの値、";", "=",と "?"で符号化されたURLエンコード予約済み、<script-path>と<entr-path>を指定します。

PATH_INFOメタ変数の詳細については、セクション4.1.5を参照してください。

スキームとプロトコルは、この方式がアプリケーションプロトコルに加えてアクセス方式を識別するものと同じではありません。たとえば、Transport Layer Security(TLS) [14]を使用してアクセスされるリソースは、HTTPプロトコル[19]を使用するときにHTTPSの方式を持つリクエストURIを持ちます。CGI/1.1は、スクリプトがこれを再構築するための一般的な手段を提供しません。したがって、定義されたScript-URIには、使用されている基本プロトコルが含まれます。ただし、スクリプトは、URIスキームをよりよく推定するためにスキーム固有のメタ変数を利用することができます。

この定義では、適切なコンポーネントを変更することで、パス情報または照会文字列の許可された値を使用してスクリプトを起動するURIを構築することもできます。

3.4. 実行

スクリプトはシステム定義の方法で呼び出されます。特に指定しない限り、スクリプトを含むファイルは実行可能プログラムとして呼び出されます。サーバーはセクション4で説明されているようにCGI要求を準備します。これは、要求メタ変数（実行時にスクリプトに入手可能）およびメッセージデータを要求する要求を含む。要求データはスクリプトにすぐに利用可能である必要はありません。このスクリプトは、このデータがクライアントからサーバーによって受信された前に実行できます。スクリプトからの応答はセクション5および6の説明に従ってサーバーに返されます。

エラー状態が発生した場合、サーバーはいつでも警告なしにスクリプトの実行を中断または終了させることができます。たとえば、サーバーとクライアントとの間のトランスポート障害が発生した場合に発生する可能性があります。そのため、スクリプトは異常終了を処理する準備をしてください。

4. CGIリクエスト

要求に関する情報は2つの異なるソースから来ています。要求メタ変数と関連メッセージ本文。

4.1. メタ変数を要求します

メタ変数には、サーバーからスクリプトに渡された要求に関するデータが含まれており、システム定義の方法でスクリプトによってアクセスされます。メタ変数は、大文字と小文字を区別しない名前で識別されます。ケースのみが異なる2つの異なる変

whose names differ in case only. Here they are shown using a canonical representation of capitals plus underscore ("_"). A particular system can define a different representation.

```
meta-variable-name = "AUTH_TYPE" | "CONTENT_LENGTH" |
                     "CONTENT_TYPE" | "GATEWAY_INTERFACE" |
                     "PATH_INFO" | "PATH_TRANSLATED" |
                     "QUERY_STRING" | "REMOTE_ADDR" |
                     "REMOTE_HOST" | "REMOTE_IDENT" |
                     "REMOTE_USER" | "REQUEST_METHOD" |
                     "SCRIPT_NAME" | "SERVER_NAME" |
                     "SERVER_PORT" | "SERVER_PROTOCOL" |
                     "SERVER_SOFTWARE" | scheme |
                     protocol-var-name | extension-var-name
protocol-var-name  = ( protocol | scheme ) "_" var-name
scheme             = alpha *( alpha | digit | "+" | "-" | "." )
var-name           = token
extension-var-name = token
```

Meta-variables with the same name as a scheme, and names beginning with the name of a protocol or scheme (e.g., HTTP_ACCEPT) are also defined. The number and meaning of these variables may change independently of this specification. (See also section 4.1.18.)

The server MAY set additional implementation-defined extension meta-variables, whose names SHOULD be prefixed with "X_".

This specification does not distinguish between zero-length (NULL) values and missing values. For example, a script cannot distinguish between the two requests http://host/script and http://host/script? as in both cases the QUERY_STRING meta-variable would be NULL.

```
meta-variable-value = "" | 1*<TEXT, CHAR or tokens of value>
```

An optional meta-variable may be omitted (left unset) if its value is NULL. Meta-variable values MUST be considered case-sensitive except as noted otherwise. The representation of the characters in the meta-variables is system-defined; the server MUST convert values to that representation.

4.1.1. AUTH_TYPE

The AUTH_TYPE variable identifies any mechanism used by the server to authenticate the user. It contains a case-insensitive value defined by the client protocol or server implementation.

For HTTP, if the client request required authentication for external access, then the server MUST set the value of this variable from the 'auth-scheme' token in the request Authorization header field.

```
AUTH_TYPE      = "" | auth-scheme
auth-scheme     = "Basic" | "Digest" | extension-auth
extension-auth  = token
```

HTTP access authentication schemes are described in RFC 2617 [5].

4.1.2. CONTENT_LENGTH

The CONTENT_LENGTH variable contains the size of the message-body attached to the request, if any, in decimal number of octets. If no data is attached, then NULL (or unset).

数がありません。ここでは、キャピタルとアンダースコア（「_」）の標準的な表現を使用して表示されます。特定のシステムは異なる表現を定義できます。

スキームと同じ名前を持つメタ変数、およびプロトコルまたはスキームの名前（たとえば、HTTP_ACCEPT）の名前で始まる名前も定義されています。これらの変数の数と意味は、この仕様とは無関係に変わる可能性があります。（4.1.18項も参照）

サーバーは、追加の実装定義定義内容メタ変数を設定します。その名前の先頭に "X_" を付ける必要があります。

この仕様は、ゼロ長（NULL）値と欠損値を区別しません。たとえば、スクリプトは、2つの要求 http://host/script と http://host/script? を区別できません。どちらの場合も、QUERY_STRINGメタ変数はNULLになります。

その値がNULLの場合は、オプションのメタ変数を省略することができます（未設定）。メタ変数値は、特に記載されている場合を除き、大文字と小文字を区別する必要があります。メタ変数の文字の表現はシステム定義です。サーバーは値をその表現に変換する必要があります。

4.1.1. AUTH_TYPE

AUTH_TYPE変数は、ユーザを認証するためにサーバによって使用される任意のメカニズムを識別します。クライアントプロトコルまたはサーバー実装によって定義された大文字と小文字を区別しない値が含まれています。

HTTPの場合、クライアント要求が外部アクセスに対して認証を要求する場合、サーバーは[Request Authorization Header]フィールドの[Auth-Scheme]トークンからこの変数の値を設定する必要があります。

HTTPアクセス認証方式はRFC 2617 [5]に記載されています。

4.1.2. CONTENT_LENGTH

CONTENT_LENGTH変数には、10進数のオクテット数で、要求に添付されているメッセージ本文のサイズが含まれます。データが添付されていない場合は、NULL（または設定解除）します。


```
CONTENT_LENGTH = "" | 1*digit
```

The server MUST set this meta-variable if and only if the request is accompanied by a message-body entity. The CONTENT_LENGTH value must reflect the length of the message-body after the server has removed any transfer-codings or content-codings.

4.1.3. CONTENT_TYPE

If the request includes a message-body, the CONTENT_TYPE variable is set to the Internet Media Type [6] of the message-body.

```
CONTENT_TYPE = "" | media-type
media-type    = type "/" subtype *( ";" parameter )
type          = token
subtype       = token
parameter     = attribute "=" value
attribute     = token
value        = token | quoted-string
```

The type, subtype and parameter attribute names are not case-sensitive. Parameter values may be case sensitive. Media types and their use in HTTP are described section 3.7 of the HTTP/1.1 specification [4].

There is no default value for this variable. If and only if it is unset, then the script MAY attempt to determine the media type from the data received. If the type remains unknown, then the script MAY choose to assume a type of application/octet-stream or it may reject the request with an error (as described in section 6.3.3).

Each media-type defines a set of optional and mandatory parameters. This may include a charset parameter with a case-insensitive value defining the coded character set for the message-body. If the charset parameter is omitted, then the default value should be derived according to whichever of the following rules is the first to apply:

1. There MAY be a system-defined default charset for some media-types.
2. The default for media-types of type "text" is ISO-8859-1 [4].
3. Any default defined in the media-type specification.
4. The default is US-ASCII.

The server MUST set this meta-variable if an HTTP Content-Type field is present in the client request header. If the server receives a request with an attached entity but no Content-Type header field, it MAY attempt to determine the correct content type, otherwise it should omit this meta-variable.

4.1.4. GATEWAY_INTERFACE

The GATEWAY_INTERFACE variable MUST be set to the dialect of CGI being used by the server to communicate with the script. Syntax:

```
GATEWAY_INTERFACE = "CGI" "/" 1*digit "." 1*digit
```

Note that the major and minor numbers are treated as separate integers and hence each may be incremented higher than a

要求にメッセージボディエンティティが伴っている場合に限
り、サーバーはこのメタ変数を設定する必要があります。
CONTENT_LENGTH値は、サーバーが転送コードまたはコンテ
ンツコーディングを削除した後にメッセージ本文の長さを反映
している必要があります。

4.1.3. CONTENT_TYPE

要求にメッセージ本体が含まれている場合、CONTENT_TYPE変
数はメッセージ本文のインターネットメディアタイプ[6]に設定
されます。

型、サブタイプ、およびパラメータ属性名は大文字と小文字を
区別しません。パラメータ値は大文字と小文字が区別される場
合があります。メディアタイプとHTTPでの使用は、HTTP/1.1仕
様のセクション3.7で説明されています[4]。

この変数にはデフォルト値はありません。設定されていない場
合に限り、スクリプトは受信したデータからメディアの種類を
判断しようとします。タイプが不明なままである場合、スクリ
プトはapplication/octet-streamの種類を引き受けるか、（6.3.3
項で説明されているように）要求をエラーで拒否することがで
きます。

各メディアタイプは、オプションと必須パラメータのセットを定
義します。これは、メッセージ本文の符号化文字セットを定義す
る大文字と小文字を区別する値を有する文字セットパラメータを
含み得る。CHARSETパラメーターを省略した場合、次の規則の
どちらでも適用の最初のルールのどちらでもデフォルト値を派生
させる必要があります。

1. いくつかのメディアタイプに対してシステム定義のデフォル
ト文字セットがあるかもしれません。
2. 「テキスト」のメディアタイプのデフォルトはISO-8859-1
[4]です。
3. メディアタイプの指定ではデフォルトで定義されています。
4. デフォルトはUS-ASCIIです。

HTTP Content-Typeフィールドがクライアント要求ヘッダーに
存在する場合、サーバーはこのメタ変数を設定する必要があり
ます。サーバーが添付のエンティティを持つ要求を受信した
が、コンテンツタイプのヘッダーフィールドがない場合は、正
しいコンテンツタイプの決定を試みることがあります。そうし
ないと、このメタ変数を省略する必要があります。

4.1.4. GATEWAY_INTERFACE

GATEWAY_INTERFACE変数は、スクリプトと通信するためにサ
ーバーによって使用されているCGIの方言に設定する必要があり
ます。構文：

メジャー番号とマイナー番号は別々の整数として扱われている
ため、それぞれ1桁より高い増分されてもよい。したがって、

single digit. Thus CGI/2.4 is a lower version than CGI/2.13 which in turn is lower than CGI/12.3. Leading zeros MUST be ignored by the script and MUST NOT be generated by the server.

This document defines the 1.1 version of the CGI interface.

4.1.5. PATH_INFO

The PATH_INFO variable specifies a path to be interpreted by the CGI script. It identifies the resource or sub-resource to be returned by the CGI script, and is derived from the portion of the URI path hierarchy following the part that identifies the script itself. Unlike a URI path, the PATH_INFO is not URL-encoded, and cannot contain path-segment parameters. A PATH_INFO of "/" represents a single void path segment.

```
PATH_INFO = "" | ( "/" path )
path      = lsegment *( "/" lsegment )
lsegment  = *lchar
lchar     = <any TEXT or CTL except "/">
```

The value is considered case-sensitive and the server MUST preserve the case of the path as presented in the request URI. The server MAY impose restrictions and limitations on what values it permits for PATH_INFO, and MAY reject the request with an error if it encounters any values considered objectionable. That MAY include any requests that would result in an encoded "/" being decoded into PATH_INFO, as this might represent a loss of information to the script. Similarly, treatment of non US-ASCII characters in the path is system-defined.

URL-encoded, the PATH_INFO string forms the extra-path component of the Script-URI (see section 3.3) which follows the SCRIPT_NAME part of that path.

4.1.6. PATH_TRANSLATED

The PATH_TRANSLATED variable is derived by taking the PATH_INFO value, parsing it as a local URI in its own right, and performing any virtual-to-physical translation appropriate to map it onto the server's document repository structure. The set of characters permitted in the result is system-defined.

```
PATH_TRANSLATED = *<any character>
```

This is the file location that would be accessed by a request for

```
<scheme> "://" <server-name> ":" <server-port> <extra-path>
```

where <scheme> is the scheme for the original client request and <extra-path> is a URL-encoded version of PATH_INFO, with ";", "=", and "?" reserved. For example, a request such as the following:

```
http://somehost.com/cgi-bin/somescrpt/this%2eis%2epath%3binfo
```

would result in a PATH_INFO value of

```
/this.is.the.path;info
```

An internal URI is constructed from the scheme, server location and the URL-encoded PATH_INFO:

```
http://somehost.com/this.is.the.path%3binfo
```

CGI/2.4はCGI/2.13より低いバージョンであり、これはCGI/12.3より低い。先行ゼロはスクリプトによって無視される必要があり、サーバーによって生成されないでください。

このドキュメントは1.1バージョンのCGIインターフェイスを定義します。

4.1.5. PATH_INFO

path_info変数は、CGIスクリプトによって解釈されるパスを指定します。CGIスクリプトによって返されるリソースまたはサブリソースを識別し、スクリプト自体を識別する部分に従ってURIパス階層の一部から派生します。URIパスとは異なり、PATH_INFOはURLエンコードされていないため、パスセグメントパラメータを含めることはできません。"/"のPATH_INFOは単一のボイドパスセグメントを表します。

値は大文字と小文字が区別され、サーバーはリクエストURIに表示されているようにパスのケースを保存する必要があります。サーバーは、PATH_INFOを許可する値に制限と制限を課し、それが不快と見なされる値が発生した場合にエラーで要求を拒否する可能性があります。これは、スクリプトへの情報の損失を表すかもしれないので、符号化された「/」がPATH_INFOにデコードされるような要求を含むかもしれません。同様に、パス内のUS-ASCII文字以外の文字の処理はシステム定義です。

URLエンコードされたPATH_INFO文字列は、そのパスのSCRIPT_NAME部分に続くScript-URI（セクション3.3を参照）のextra-pathコンポーネントを形成します。

4.1.6. PATH_TRANSLATED

PATH_TRANSLATED変数は、PATH_INFO値を取得し、それを自分の権利でローカルURIとして解析し、それをサーバーのドキュメントリポジトリ構造にマッピングするのに適した仮想間の変換を実行することによって導出されます。結果で許可されている文字セットはシステム定義です。

これは要求によってアクセスされるファイルの場所です。

<scheme>は、元のクライアント要求の方式で、<Extra-Path>はPATH_INFOのURLエンコードされたバージョンです。";", "=", "?"予約済み。たとえば、次のような要求について説明します。

PATH_INFO値をもたらすでしょう

内部URIは、スキーム、サーバーの場所、およびURLエンコードされたPATH_INFOから構成されています。

This would then be translated to a location in the server's document repository, perhaps a filesystem path something like this:

```
/usr/local/www/htdocs/this.is.the.path;info
```

The value of `PATH_TRANSLATED` is the result of the translation.

The value is derived in this way irrespective of whether it maps to a valid repository location. The server **MUST** preserve the case of the extra-path segment unless the underlying repository supports case-insensitive names. If the repository is only case-aware, case-preserving, or case-blind with regard to document names, the server is not required to preserve the case of the original segment through the translation.

The translation algorithm the server uses to derive `PATH_TRANSLATED` is implementation-defined; CGI scripts which use this variable may suffer limited portability.

The server **SHOULD** set this meta-variable if the request URI includes a path-info component. If `PATH_INFO` is `NULL`, then the `PATH_TRANSLATED` variable **MUST** be set to `NULL` (or unset).

4.1.7. QUERY_STRING

The `QUERY_STRING` variable contains a URL-encoded search or parameter string; it provides information to the CGI script to affect or refine the document to be returned by the script.

The URL syntax for a search string is described in section 3 of RFC 2396 [2]. The `QUERY_STRING` value is case-sensitive.

```
QUERY_STRING = query-string
query-string = *uric
uric          = reserved | unreserved | escaped
```

When parsing and decoding the query string, the details of the parsing, reserved characters and support for non US-ASCII characters depends on the context. For example, form submission from an HTML document [18] uses application/x-www-form-urlencoded encoding, in which the characters "+", "&" and "=" are reserved, and the ISO 8859-1 encoding may be used for non US-ASCII characters.

The `QUERY_STRING` value provides the query-string part of the Script-URI. (See section 3.3).

The server **MUST** set this variable; if the Script-URI does not include a query component, the `QUERY_STRING` **MUST** be defined as an empty string ("").

4.1.8. REMOTE_ADDR

The `REMOTE_ADDR` variable **MUST** be set to the network address of the client sending the request to the server.

```
REMOTE_ADDR  = hostnumber
hostnumber   = ipv4-address | ipv6-address
ipv4-address = 1*3digit "." 1*3digit "." 1*3digit "." 1*3digit
ipv6-address = hexpart [ ":" ipv4-address ]
hexpart      = hexseq | ( [ hexseq ] "::" [ hexseq ] )
hexseq       = 1*4hex *( ":" 1*4hex )
```

The format of an IPv6 address is described in RFC 3513 [15].

これは、サーバーのドキュメントリポジトリ内の場所に変換され、おそらくこのようなファイルシステムのパスです。

`PATH_TRANSLATED`の値は翻訳の結果です。

この方法では、有効なリポジトリの場所にマッピングされているかどうかにかかわらず、この方法で派生します。基礎となるリポジトリが大文字と小文字を区別しない名前をサポートしていない限り、サーバーはエキストラパスセグメントの事例を保存する必要があります。リポジトリが文書名に関して大文字と小文字を区別している場合、ケースゼロのみである場合、サーバーは翻訳を通じて元のセグメントの事例を保存する必要はありません。

サーバーが`PATH_TRANSLATED`を派生させるために使用する翻訳アルゴリズムは実装定義です。この変数を使用するCGIスクリプトは、限られた移植性を被る可能性があります。

要求URIにパス情報コンポーネントが含まれている場合、サーバーはこのメタ変数を設定する必要があります。`PATH_INFO`が`null`の場合、`PATH_TRANSLATED`変数は`NULL`（または未設定）に設定する必要があります。

4.1.7. QUERY_STRING

`QUERY_STRING`変数には、URLエンコードされた検索またはパラメータ文字列が含まれています。スクリプトによって返される文書に影響を与えるか、CGIスクリプトに情報を提供します。

検索文字列のURL構文は、RFC 2396 [2]のセクション3で説明されています。`QUERY_STRING`値では大文字と小文字が区別されます。

クエリ文字列を解析および復号化するとき、解析、予約文字、およびUS-ASCII以外の文字のサポートの詳細は、コンテキストによって異なります。たとえば、application/x-www-form-urlencodedを使用したHTML文書からのフォーム送信[18]は、文字 "+", "&", "="が予約されており、ISO 8859-1エンコーディングがある場合があります。US-ASCII文字以外の文字に使用されます。

`QUERY_STRING`値は、スクリプト-URIの照会列部分を提供します。（3.3項を参照）。

サーバーはこの変数を設定する必要があります。Script-URIにクエリコンポーネントが含まれていない場合、`QUERY_STRING`は空の文字列（""）として定義する必要があります。

4.1.8. REMOTE_ADDR

`REMOTE_ADDR`変数は、リクエストをサーバーに送信するクライアントのネットワークアドレスに設定する必要があります。

IPv6アドレスのフォーマットはRFC 3513 [15]に記載されていま

4.1.9. REMOTE_HOST

The REMOTE_HOST variable contains the fully qualified domain name of the client sending the request to the server, if available, otherwise NULL. Fully qualified domain names take the form as described in section 3.5 of RFC 1034 [17] and section 2.1 of RFC 1123 [12]. Domain names are not case sensitive.

```
REMOTE_HOST    = "" | hostname | hostnumber
hostname       = *( domainlabel "." ) toplabel [ "." ]
domainlabel    = alphanum [ *alphahypdigit alphanum ]
toplabel       = alpha [ *alphahypdigit alphanum ]
alphahypdigit  = alphanum | "-"
```

The server SHOULD set this variable. If the hostname is not available for performance reasons or otherwise, the server MAY substitute the REMOTE_ADDR value.

4.1.10. REMOTE_IDENT

The REMOTE_IDENT variable MAY be used to provide identity information reported about the connection by an RFC 1413 [20] request to the remote agent, if available. The server may choose not to support this feature, or not to request the data for efficiency reasons, or not to return available identity data.

```
REMOTE_IDENT = *TEXT
```

The data returned may be used for authentication purposes, but the level of trust reposed in it should be minimal.

4.1.11. REMOTE_USER

The REMOTE_USER variable provides a user identification string supplied by client as part of user authentication.

```
REMOTE_USER = *TEXT
```

If the client request required HTTP Authentication [5] (e.g., the AUTH_TYPE meta-variable is set to "Basic" or "Digest"), then the value of the REMOTE_USER meta-variable MUST be set to the user-ID supplied.

4.1.12. REQUEST_METHOD

The REQUEST_METHOD meta-variable MUST be set to the method which should be used by the script to process the request, as described in section 4.3.

```
REQUEST_METHOD = method
method          = "GET" | "POST" | "HEAD" | extension-method
extension-method = "PUT" | "DELETE" | token
```

The method is case sensitive. The HTTP methods are described in section 5.1.1 of the HTTP/1.0 specification [1] and section 5.1.1 of the HTTP/1.1 specification [4].

4.1.13. SCRIPT_NAME

The SCRIPT_NAME variable MUST be set to a URI path (not URL-encoded) which could identify the CGI script (rather than the script's output). The syntax is the same as for PATH_INFO (section 4.1.5)

```
SCRIPT_NAME = "" | ( "/" path )
```

す。

4.1.9. REMOTE_HOST

REMOTE_HOST変数には、使用可能な場合は、サーバーへの要求を送信するクライアントの完全修飾ドメイン名が含まれています。それ以外の場合はNULL。完全修飾ドメイン名は、RFC 1034 [17]およびRFC 1123のセクション2.1のセクション3.5に記載されているような形式を取ります[12]。ドメイン名は大文字と小文字を区別しません。

サーバーはこの変数を設定する必要があります。ホスト名がパフォーマンス上の理由から使用できないか、そうでなければサーバーはREMOTE_ADDR値を置き換えることがあります。

4.1.10. REMOTE_IDENT

REMOTE_IDENT変数は、利用可能であれば、リモートエージェントへのRFC 1413 [20]リクエストによって接続に関するID情報を提供するために使用されてもよい。サーバーはこの機能をサポートしないか、効率的な理由でデータを要求しないか、利用可能なIDデータを返さないようにすることができます。

返されるデータは認証目的に使用されるかもしれませんが、それに払った信頼レベルは最小限であるはずです。

4.1.11. REMOTE_USER

REMOTE_USER変数は、クライアントによってユーザー認証の一部として提供されるユーザー識別文字列を提供します。

クライアント要求がHTTP認証を要求した場合[5]（例えば、AUTH_TYPEメタ変数が "Basic"または "Digest"に設定されています）、REMOTE_USERメタ変数の値は指定されたuser-IDに設定する必要があります。

4.1.12. REQUEST_METHOD

REQUEST_METHODメタ変数は、セクション4.3で説明されているように、要求を処理するためにスクリプトによって使用されるべきメソッドに設定する必要があります。

この方法では大文字と小文字が区別されます。HTTPメソッドは、http/1.1仕様書[4]のhttp/1.0仕様[1]およびセクション5.1.1のセクション5.1.1で説明されています[4]。

4.1.13. script_name.

SCRIPT_NAME変数は、（スクリプトの出力ではなく）CGIスクリプトを識別できるURIパス（URLエンコードされていない）に設定する必要があります。構文はPATH_INFOと同じです（セクション4.1.5）。

The leading `"/` is not part of the path. It is optional if the path is `NULL`; however, the variable `MUST` still be set in that case.

The `SCRIPT_NAME` string forms some leading part of the path component of the `Script-URI` derived in some implementation-defined manner. No `PATH_INFO` segment (see section 4.1.5) is included in the `SCRIPT_NAME` value.

4.1.14. SERVER_NAME

The `SERVER_NAME` variable `MUST` be set to the name of the server host to which the client request is directed. It is a case-insensitive hostname or network address. It forms the host part of the `Script-URI`.

```
SERVER_NAME = server-name
server-name = hostname | ipv4-address | ( "[" ipv6-address "]" )
```

A deployed server can have more than one possible value for this variable, where several HTTP virtual hosts share the same IP address. In that case, the server would use the contents of the request's `Host` header field to select the correct virtual host.

4.1.15. SERVER_PORT

The `SERVER_PORT` variable `MUST` be set to the TCP/IP port number on which this request is received from the client. This value is used in the port part of the `Script-URI`.

```
SERVER_PORT = server-port
server-port = 1*digit
```

Note that this variable `MUST` be set, even if the port is the default port for the scheme and could otherwise be omitted from a URI.

4.1.16. SERVER_PROTOCOL

The `SERVER_PROTOCOL` variable `MUST` be set to the name and version of the application protocol used for this CGI request. This `MAY` differ from the protocol version used by the server in its communication with the client.

```
SERVER_PROTOCOL = HTTP-Version | "INCLUDED" | extension-version
HTTP-Version    = "HTTP" "/" 1*digit "." 1*digit
extension-version = protocol [ "/" 1*digit "." 1*digit ]
protocol        = token
```

Here, `'protocol'` defines the syntax of some of the information passing between the server and the script (the `'protocol-specific'` features). It is not case sensitive and is usually presented in upper case. The protocol is not the same as the scheme part of the script URI, which defines the overall access mechanism used by the client to communicate with the server. For example, a request that reaches the script with a protocol of `"HTTP"` may have used an `"https"` scheme.

A well-known value for `SERVER_PROTOCOL` which the server `MAY` use is `"INCLUDED"`, which signals that the current document is being included as part of a composite document, rather than being the direct target of the client request. The script should treat this as an HTTP/1.0 request.

4.1.17. SERVER_SOFTWARE

The `SERVER_SOFTWARE` meta-variable `MUST` be set to the

先頭の `"/`はパスの一部ではありません。パスが`NULL`の場合はオプションです。ただし、その場合は変数を設定する必要があります。

`SCRIPT_NAME`文字列は、いくつかの実装定義の方法で派生したスクリプト-URIのパスコンポーネントの先頭部分を形成します。`SCRIPT_NAME`値には、`PATH_INFO`セグメント（4.1.5項を参照）が含まれていません。

4.1.14. SERVER_NAME

`SERVER_NAME`変数は、クライアント要求が指示されているサーバーホストの名前に設定する必要があります。それは大文字と小文字を区別しないホスト名またはネットワークアドレスです。`Script-URI`のホスト部分を形成します。

デプロイされたサーバーはこの変数に対して複数の可能な値を持つことができます。ここで、複数のHTTP仮想ホストは同じIPアドレスを共有します。その場合、サーバーはリクエストのホストヘッダーフィールドの内容を使用して正しい仮想ホストを選択します。

4.1.15. サーバポート

`SERVER_PORT`変数は、クライアントからこの要求を受信したTCP/IPポート番号に設定する必要があります。この値は、`Script-URI`のポート部分で使用されます。

4.1.16. SERVER_PROTOCOL

`SERVER_PROTOCOL`変数は、このCGI要求に使用されるアプリケーションプロトコルの名前とバージョンに設定する必要があります。これは、クライアントとの通信でサーバーによって使用されるプロトコルバージョンとは異なる場合があります。

ここで、`'protocol'`は、サーバーとスクリプトの間の渡しの一部のシンタックスを定義します（`'プロトコル固有の機能'`）。大文字と小文字が区別されず、通常大文字で表示されます。プロトコルは`Script-URI`のスキーム部分と同じではありません。これは、クライアントがサーバーと通信するために使用される全体的なアクセスメカニズムを定義します。たとえば、「HTTP」のプロトコルでスクリプトに到達する要求は、「HTTPS」方式を使用している可能性があります。

サーバーが使用できる`SERVER_PROTOCOL`の有名な値は「含まれています」は「Client」であることが、クライアント要求の直接ターゲットではなく、コンポジットドキュメントの一部として含まれていることを示します。スクリプトはこれをHTTP/1.0要求として扱うべきです。

4.1.17. SERVER_SOFTWARE

`SERVER_SOFTWARE`メタ変数は、CGI要求（ゲートウェイを実

name and version of the information server software making the CGI request (and running the gateway). It SHOULD be the same as the server description reported to the client, if any.

```
SERVER_SOFTWARE = 1*( product | comment )
product         = token [ "/" product-version ]
product-version = token
comment         = "(" *( ctext | comment ) ")"
ctext           = <any TEXT excluding "(" and ">
```

4.1.18. Protocol-Specific Meta-Variables

The server SHOULD set meta-variables specific to the protocol and scheme for the request. Interpretation of protocol-specific variables depends on the protocol version in SERVER_PROTOCOL. The server MAY set a meta-variable with the name of the scheme to a non-NULL value if the scheme is not the same as the protocol. The presence of such a variable indicates to a script which scheme is used by the request.

Meta-variables with names beginning with "HTTP_" contain values read from the client request header fields, if the protocol used is HTTP. The HTTP header field name is converted to upper case, has all occurrences of "-" replaced with "_" and has "HTTP_" prepended to give the meta-variable name. The header data can be presented as sent by the client, or can be rewritten in ways which do not change its semantics. If multiple header fields with the same field-name are received then the server MUST rewrite them as a single value having the same semantics. Similarly, a header field that spans multiple lines MUST be merged onto a single line. The server MUST, if necessary, change the representation of the data (for example, the character set) to be appropriate for a CGI meta-variable.

The server is not required to create meta-variables for all the header fields that it receives. In particular, it SHOULD remove any header fields carrying authentication information, such as 'Authorization'; or that are available to the script in other variables, such as 'Content-Length' and 'Content-Type'. The server MAY remove header fields that relate solely to client-side communication issues, such as 'Connection'.

4.2. Request Message-Body

Request data is accessed by the script in a system-defined method; unless defined otherwise, this will be by reading the 'standard input' file descriptor or file handle.

```
Request-Data    = [ request-body ] [ extension-data ]
request-body    = <CONTENT_LENGTH>OCTET
extension-data  = *OCTET
```

A request-body is supplied with the request if the CONTENT_LENGTH is not NULL. The server MUST make at least that many bytes available for the script to read. The server MAY signal an end-of-file condition after CONTENT_LENGTH bytes have been read or it MAY supply extension data. Therefore, the script MUST NOT attempt to read more than CONTENT_LENGTH bytes, even if more data is available. However, it is not obliged to read any of the data.

For non-parsed header (NPH) scripts (section 5), the server SHOULD attempt to ensure that the data supplied to the script is precisely as supplied by the client and is unaltered by the server.

行している）を作成する情報サーバーソフトウェアの名前とバージョンに設定する必要があります。それがあれば、クライアントに報告されたサーバーの説明と同じであるべきです。

4.1.18. プロトコル固有のメタ変数

サーバーは、要求のプロトコルとスキームに固有のメタ変数を設定する必要があります。プロトコル固有の変数の解釈は、SERVER_PROTOCOLのプロトコルバージョンによって異なります。スキームがプロトコルと同じではない場合、サーバは、スキームの名前をNULL値以外の値に設定してメタ変数を設定することができる。そのような変数の存在は、要求によってスキームが使用されるスクリプトを示す。

使用されるプロトコルがHTTPである場合、 "HTTP_"で始まる名前を持つメタ変数は、クライアント要求ヘッダーフィールドから読み取られた値を含みます。HTTPヘッダーフィールド名は大文字に変換され、すべての出現箇所が "_"に置き換えられ、 "http_"がメタ変数名に入れる前に "HTTP_"を追加しています。ヘッダデータはクライアントによって送信されるように提示され得るか、またはその意味を変更しない方法で書き換えることができます。同じフィールド名を持つ複数のヘッダーフィールドが受信された場合、サーバーはそれらを同じセマンティクスを持つ単一の値として書き換える必要があります。同様に、複数の行にスパンするヘッダフィールドを1行にマージする必要があります。サーバーは、必要に応じて、CGIメタ変数に適したデータの表現（文字セットなど）を変更する必要があります。

サーバーは、受信したすべてのヘッダーフィールドに対してメタ変数を作成する必要はありません。特に、「承認」などの認証情報を搬送するヘッダーフィールドを削除する必要があります。あるいは、 'Content-Length'と 'Content-Type'など、他の変数のスクリプトで利用可能です。サーバーは、「接続」などのクライアント側の通信の問題のみに関連するヘッダーフィールドを削除することがあります。

4.2. メッセージを要求します

要求データはシステム定義の方法でスクリプトによってアクセスされます。特に定義されていない限り、これは '標準入力'ファイル記述子またはファイルハンドルを読むことになります。

CONTENT_LENGTHがNULLではない場合、要求ボディに要求が付属しています。サーバーは、少なくともスクリプトが読み取るために利用可能なその多くのバイトを作成する必要があります。CONTENT_LENGTHバイトが読み込まれた後、サーバーはファイルの終わり条件を通知したり、拡張データを供給してもよい。したがって、より多くのデータが利用可能であっても、スクリプトはCONTENT_LENGTHバイトを読み込もうとしてはならない。ただし、データのいずれかを読む義務がありません。

非解析ヘッダ（NPH）スクリプト（セクション5）の場合、サーバーは、スクリプトに提供されているデータがクライアントによって提供されているとおりに正確に、サーバーによって変更されていないことを確認します。

As transfer-codings are not supported on the request-body, the server MUST remove any such codings from the message-body, and recalculate the CONTENT_LENGTH. If this is not possible (for example, because of large buffering requirements), the server SHOULD reject the client request. It MAY also remove content-codings from the message-body.

4.3. Request Methods

The Request Method, as supplied in the REQUEST_METHOD meta-variable, identifies the processing method to be applied by the script in producing a response. The script author can choose to implement the methods most appropriate for the particular application. If the script receives a request with a method it does not support it SHOULD reject it with an error (see section 6.3.3).

4.3.1. GET

The GET method indicates that the script should produce a document based on the meta-variable values. By convention, the GET method is 'safe' and 'idempotent' and SHOULD NOT have the significance of taking an action other than producing a document.

The meaning of the GET method may be modified and refined by protocol-specific meta-variables.

4.3.2. POST

The POST method is used to request the script perform processing and produce a document based on the data in the request message-body, in addition to meta-variable values. A common use is form submission in HTML [18], intended to initiate processing by the script that has a permanent affect, such a change in a database.

The script MUST check the value of the CONTENT_LENGTH variable before reading the attached message-body, and SHOULD check the CONTENT_TYPE value before processing it.

4.3.3. HEAD

The HEAD method requests the script to do sufficient processing to return the response header fields, without providing a response message-body. The script MUST NOT provide a response message-body for a HEAD request. If it does, then the server MUST discard the message-body when reading the response from the script.

4.3.4. Protocol-Specific Methods

The script MAY implement any protocol-specific method, such as HTTP/1.1 PUT and DELETE; it SHOULD check the value of SERVER_PROTOCOL when doing so.

The server MAY decide that some methods are not appropriate or permitted for a script, and may handle the methods itself or return an error to the client.

4.4. The Script Command Line

Some systems support a method for supplying an array of strings to the CGI script. This is only used in the case of an 'indexed' HTTP query, which is identified by a 'GET' or 'HEAD' request with a URI query string that does not contain any

転送コーディングはリクエストボディでサポートされていないので、サーバはメッセージ本文からそのようなコードを削除し、CONTENT_LENGTHを再計算する必要があります。これが不可能な場合（たとえば、バッファリング要件が大きいため）、サーバーはクライアント要求を拒否する必要があります。メッセージ本文からコンテンツコーディングを削除することもできます。

4.3. リクエストメソッド

REQUEST_METHODメタ変数で指定された要求方法は、応答を生成する際にスクリプトによって適用される処理方法を識別します。スクリプト作成者は、特定のアプリケーションに最も適したメソッドを実装することを選択できます。スクリプトがメソッドを使用して要求を受信した場合、それはそれをサポートしていません。エラーで拒否する必要があります（6.3.3項を参照）。

4.3.1. GET

GETメソッドは、スクリプトがメタ変数値に基づいて文書を作成する必要があることを示します。慣例により、GETメソッドは'safe'と'idempotent'です。

GETメソッドの意味は、プロトコル固有のメタ変数によって変更され洗練されてもよい。

4.3.2. POST

POSTメソッドは、メタ変数値に加えて、スクリプトの処理を実行し、要求メッセージ本文のデータに基づいて文書を作成するために使用されます。一般的な使用はHTML [18]でのフォーム送信は、永続的な影響を持つスクリプトによる処理を開始することを目的としています。このようなデータベースの変更。

スクリプトは、接続されているメッセージ本文を読み込む前にCONTENT_LENGTH変数の値をチェックし、処理前にCONTENT_TYPE値を確認する必要があります。

4.3.3. HEAD

HEADメソッドは、応答メッセージ本文を提供することなく、応答ヘッダフィールドを返すのに十分な処理を実行するようにスクリプトに要求する。スクリプトは、ヘッド要求の応答メッセージ本文を提供してはいけません。そうであれば、サーバーはスクリプトからの応答を読むときにメッセージ本文を破棄する必要があります。

4.3.4. プロトコル固有のメソッド

スクリプトは、HTTP/1.1のPUTおよびDELETEなどのプロトコル固有の方法を実装することができます。そうするときはSERVER_PROTOCOLの値を確認する必要があります。

サーバーは、いくつかのメソッドがスクリプトに対して適切でないか許可されていることを決定し、メソッド自体を処理するか、クライアントにエラーを返すことができます。

4.4. スクリプトコマンドライン

一部のシステムは、CGIスクリプトに文字列の配列を供給する方法をサポートしています。これは '索引付けされた' HTTPクエリの場合にのみ使用されます。これは、符号を付けられていない "="文字を含まないURIクエリ文字列を持つ 'get'または 'head'要

unencoded "=" characters. For such a request, the server SHOULD treat the query-string as a search-string and parse it into words, using the rules

```
search-string = search-word *( "+" search-word )
search-word   = 1*schar
schar         = unreserved | escaped | xreserved
xreserved     = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "," |
               "$"
```

After parsing, each search-word is URL-decoded, optionally encoded in a system-defined manner and then added to the command line argument list.

If the server cannot create any part of the argument list, then the server MUST NOT generate any command line information. For example, the number of arguments may be greater than operating system or server limits, or one of the words may not be representable as an argument.

The script SHOULD check to see if the QUERY_STRING value contains an unencoded "=" character, and SHOULD NOT use the command line arguments if it does.

5. NPH Scripts

5.1. Identification

The server MAY support NPH (Non-Parsed Header) scripts; these are scripts to which the server passes all responsibility for response processing.

This specification provides no mechanism for an NPH script to be identified on the basis of its output data alone. By convention, therefore, any particular script can only ever provide output of one type (NPH or CGI) and hence the script itself is described as an 'NPH script'. A server with NPH support MUST provide an implementation-defined mechanism for identifying NPH scripts, perhaps based on the name or location of the script.

5.2. NPH Response

There MUST be a system-defined method for the script to send data back to the server or client; a script MUST always return some data. Unless defined otherwise, this will be the same as for conventional CGI scripts.

Currently, NPH scripts are only defined for HTTP client requests. An (HTTP) NPH script MUST return a complete HTTP response message, currently described in section 6 of the HTTP specifications [1], [4]. The script MUST use the SERVER_PROTOCOL variable to determine the appropriate format for a response. It MUST also take account of any generic or protocol-specific meta-variables in the request as might be mandated by the particular protocol specification.

The server MUST ensure that the script output is sent to the client unmodified. Note that this requires the script to use the correct character set (US-ASCII [9] and ISO 8859-1 [10] for HTTP) in the header fields. The server SHOULD attempt to ensure that the script output is sent directly to the client, with minimal internal and no transport-visible buffering.

Unless the implementation defines otherwise, the script MUST NOT indicate in its response that the client can send further requests over the same connection.

求によって識別されます。そのような要求の場合、サーバーはクエリ文字列を検索文字列として扱い、ルールを使用して単語に解析する必要があります。

解析後、各検索語はURLデコードされ、オプションでシステム定義の方法でエンコードされ、コマンドライン引数リストに追加されます。

サーバーが引数リストの一部を作成できない場合、サーバーはコマンドライン情報を生成してはいけません。たとえば、引数の数はオペレーティングシステムまたはサーバーの制限よりも大きく、または単語の1つが引数として表現できない可能性があります。

スクリプトは、QUERY_STRING値にエンコードされていない"="文字が含まれているかどうかを確認する必要があり、そのコマンドライン引数を使用しないでください。

5. NPHスクリプト

5.1. 識別

サーバーはNPH（解析されていないヘッダー）スクリプトをサポートすることがあります。これらは、サーバーが応答処理のすべての責任を渡すスクリプトです。

この仕様は、その出力データのみに基づいてNPHスクリプトが識別されるメカニズムを提供しません。したがって、慣例により、特定のスクリプトは1つのタイプ（NPHまたはCGI）の出力を提供することしかできず、したがってスクリプト自体は「NPHスクリプト」として説明されています。NPHサポートを持つサーバーは、おそらくスクリプトの名前または場所に基づいて、NPHスクリプトを識別するための実装定義のメカニズムを提供する必要があります。

5.2. NPH応答

スクリプトがサーバーまたはクライアントにデータを送信するためのシステム定義のメソッドが必要です。スクリプトは常にいくつかのデータを返す必要があります。特に定義されていない限り、これは従来のCGIスクリプトと同じです。

現在、NPHスクリプトはHTTPクライアント要求に対してのみ定義されています。（HTTP）NPHスクリプトは、現在、HTTP仕様[1]、[4]のセクション6で説明している完全なHTTP応答メッセージを返す必要があります。スクリプトはSERVER_PROTOCOL変数を使用して応答の適切な形式を決定する必要があります。特定のプロトコル仕様によって義務付けられている可能性があるように、要求内の一般的またはプロトコル固有のメタ変数を考慮に入れる必要があります。

サーバーは、スクリプト出力が修正されていないクライアントに送信されていることを確認する必要があります。これには、スクリプトがヘッダフィールドに正しい文字セット（US-ASCII [9]、[10]）を使用する必要があります。サーバーは、スクリプト出力がクライアントに直接送信され、最小限の内部および転送不可のバッファリングで送信されるようにします。

実装がそうでないことを定義しない限り、スクリプトはクライアントが同じ接続を介して更なる要求を送信できるという応答に示されてはならない。

6. CGI Response

6.1. Response Handling

A script **MUST** always provide a non-empty response, and so there is a system-defined method for it to send this data back to the server. Unless defined otherwise, this will be via the 'standard output' file descriptor.

The script **MUST** check the REQUEST_METHOD variable when processing the request and preparing its response.

The server **MAY** implement a timeout period within which data must be received from the script. If a server implementation defines such a timeout and receives no data from a script within the timeout period, the server **MAY** terminate the script process.

6.2. Response Types

The response comprises a message-header and a message-body, separated by a blank line. The message-header contains one or more header fields. The body may be NULL.

generic-response = 1*header-field NL [response-body]

The script **MUST** return one of either a document response, a local redirect response or a client redirect (with optional document) response. In the response definitions below, the order of header fields in a response is not significant (despite appearing so in the BNF). The header fields are defined in section 6.3.

CGI-Response = document-response | local-redir-response | client-redir-response | client-redirdoc-response

6.2.1. Document Response

The CGI script can return a document to the user in a document response, with an optional error code indicating the success status of the response.

document-response = Content-Type [Status] *other-field NL response-body

The script **MUST** return a Content-Type header field. A Status header field is optional, and status 200 'OK' is assumed if it is omitted. The server **MUST** make any appropriate modifications to the script's output to ensure that the response to the client complies with the response protocol version.

6.2.2. Local Redirect Response

The CGI script can return a URI path and query-string ('local-pathquery') for a local resource in a Location header field. This indicates to the server that it should reprocess the request using the path specified.

local-redir-response = local-Location NL

The script **MUST NOT** return any other header fields or a message-body, and the server **MUST** generate the response that it would have produced in response to a request containing the URL

scheme "://" server-name ":" server-port local-pathquery

6. CGI 応答

6.1. 応答処理

スクリプトは常に空でない応答を提供する必要があります。したがって、このデータをサーバーに送信するためのシステム定義のメソッドがあります。特に定義されていない限り、これは '標準出力' ファイル記述子を介して行われます。

スクリプトは、要求を処理してその応答を準備するときに REQUEST_METHOD 変数をチェックする必要があります。

サーバは、スクリプトからデータを受信する必要があるタイムアウト期間を実装することができる。サーバー実装がそのようなタイムアウトを定義し、タイムアウト期間内のスクリプトからデータを受信しない場合、サーバーはスクリプトプロセスを終了することがあります。

6.2. 応答タイプ

応答は、メッセージヘッダとメッセージ本文を含む、空白行で区切られています。message-header には、1つ以上のヘッダーフィールドが含まれています。本体はNULLかもしれません。

スクリプトは、文書応答、ローカルリダイレクト応答、またはクライアントリダイレクト（オプションの文書）のいずれかを返す必要があります。以下の応答定義では、応答内のヘッダーフィールドの順序は有意ではありません（BNFに表示されるものではありません）。ヘッダーフィールドはセクション6.3で定義されています。

6.2.1. ドキュメントの応答

CGI スクリプトは文書応答内のユーザーにドキュメントを返すことができます。これは、応答の成功状況を示すオプションのエラーコードです。

スクリプトはContent-Typeヘッダーフィールドを返す必要があります。ステータスヘッダフィールドはオプションであり、ステータス200 'OK' が省略されていれば想定されます。サーバーは、クライアントへの応答が応答プロトコルのバージョンに準拠していることを確認するために、スクリプトの出力を適切な変更を加える必要があります。

6.2.2. ローカルリダイレクト応答

CGI スクリプトは、ロケーションヘッダーフィールドのローカルリソースのURIパスとクエリ文字列 ('local-pathquery') を返すことができます。これは、指定されたパスを使用して要求を再処理する必要があることをサーバーに示しています。

スクリプトは他のヘッダーフィールドまたはメッセージ本文を返してはならず、サーバーはURLを含むリクエストに応じて生成されたことがあるという応答を生成する必要があります。

6.2.3. Client Redirect Response

The CGI script can return an absolute URI path in a Location header field, to indicate to the client that it should reprocess the request using the URI specified.

```
client-redir-response = client-Location *extension-field NL
```

The script **MUST** not provide any other header fields, except for server-defined CGI extension fields. For an HTTP client request, the server **MUST** generate a 302 'Found' HTTP response message.

6.2.4. Client Redirect Response with Document

The CGI script can return an absolute URI path in a Location header field together with an attached document, to indicate to the client that it should reprocess the request using the URI specified.

```
client-redirdoc-response = client-Location Status Content-Type *other-field NL response-body
```

The Status header field **MUST** be supplied and **MUST** contain a status value of 302 'Found', or it **MAY** contain an extension-code, that is, another valid status code that means client redirection. The server **MUST** make any appropriate modifications to the script's output to ensure that the response to the client complies with the response protocol version.

6.3. Response Header Fields

The response header fields are either CGI or extension header fields to be interpreted by the server, or protocol-specific header fields to be included in the response returned to the client. At least one CGI field **MUST** be supplied; each CGI field **MUST NOT** appear more than once in the response. The response header fields have the syntax:

```
header-field      = CGI-field | other-field
CGI-field         = Content-Type | Location | Status
other-field       = protocol-field | extension-field
protocol-field    = generic-field
extension-field   = generic-field
generic-field     = field-name ":" [ field-value ] NL
field-name        = token
field-value       = *( field-content | LWSP )
field-content     = *( token | separator | quoted-string )
```

The field-name is not case sensitive. A NULL field value is equivalent to a field not being sent. Note that each header field in a CGI-Response **MUST** be specified on a single line; CGI/1.1 does not support continuation lines. Whitespace is permitted between the ":" and the field-value (but not between the field-name and the ":"), and also between tokens in the field-value.

6.3.1. Content-Type

The Content-Type response field sets the Internet Media Type [6] of the entity body.

```
Content-Type = "Content-Type:" media-type NL
```

If an entity body is returned, the script **MUST** supply a Content-Type field in the response. If it fails to do so, the server **SHOULD NOT** attempt to determine the correct content type.

6.2.3. クライアントリダイレクト応答

CGIスクリプトは、指定されたURIを使用して要求を再処理する必要があるクライアントに絶対URIパスを返すことができます。

サーバー定義のCGI拡張フィールドを除いて、スクリプトは他のヘッダーフィールドを提供してはなりません。HTTPクライアント要求の場合、サーバーは302 '見つかった' HTTP応答メッセージを生成する必要があります。

6.2.4. クライアントはドキュメントで応答をリダイレクトします

CGIスクリプトは、指定されたURIを使用してリクエストを再処理する必要があることをクライアントに示すように、添付文書と一緒に位置ヘッダーフィールドに絶対URIパスを返すことができます。

ステータスヘッダーフィールドは指定されている必要があり、ステータス値302 'Found'が含まれている必要があります。また、拡張コード、つまりクライアントリダイレクトを意味するもう1つの有効なステータスコードを含めることができます。サーバーは、クライアントへの応答が応答プロトコルのバージョンに準拠していることを確認するために、スクリプトの出力を適切な変更を加える必要があります。

6.3. 応答ヘッダーフィールド

応答ヘッダフィールドは、サーバによって解釈されるCGIまたは拡張ヘッダフィールド、またはクライアントに返される応答に含まれるプロトコル固有のヘッダフィールドのいずれかである。少なくとも1つのCGIフィールドを指定する必要があります。各CGIフィールドは、応答に複数回表示されてはいけません。応答ヘッダーフィールドには構文があります。

フィールド名は大文字と小文字を区別しません。NULLフィールド値は、送信されていないフィールドと同じです。CGI応答内の各ヘッダーフィールドは、1行に指定する必要があります。CGI/1.1は継続行をサポートしません。「:」とフィールド値（フィールド名と ":"の間ではなく、フィールド値の間ではありません）、およびフィールド値のトークン間での空白が許可されています。

6.3.1. Content-Type

Content-Type Responseフィールドは、エンティティ本体のインターネットメディアタイプ[6]を設定します。

エンティティ本体が返された場合、スクリプトは応答にcontent-typeフィールドを指定する必要があります。それが失敗した場合、サーバーは正しいコンテンツタイプを決定しようと

The value SHOULD be sent unmodified to the client, except for any charset parameter changes.

Unless it is otherwise system-defined, the default charset assumed by the client for text media-types is ISO-8859-1 if the protocol is HTTP and US-ASCII otherwise. Hence the script SHOULD include a charset parameter. See section 3.4.1 of the HTTP/1.1 specification [4] for a discussion of this issue.

6.3.2. Location

The Location header field is used to specify to the server that the script is returning a reference to a document rather than an actual document (see sections 6.2.3 and 6.2.4). It is either an absolute URI (optionally with a fragment identifier), indicating that the client is to fetch the referenced document, or a local URI path (optionally with a query string), indicating that the server is to fetch the referenced document and return it to the client as the response.

Location	= local-Location client-Location
client-Location	= "Location:" fragment-URI NL
local-Location	= "Location:" local-pathquery NL
fragment-URI	= absoluteURI ["#" fragment]
fragment	= *uric
local-pathquery	= abs-path ["?" query-string]
abs-path	= "/" path-segments
path-segments	= segment *("/" segment)
segment	= *pchar
pchar	= unreserved escaped extra
extra	= ":" "@" "&" "=" "+" "\$" ","

The syntax of an absoluteURI is incorporated into this document from that specified in RFC 2396 [2] and RFC 2732 [7]. A valid absoluteURI always starts with the name of scheme followed by ":"; scheme names start with a letter and continue with alphanumerics, "+", "-" or ".". The local URI path and query must be an absolute path, and not a relative path or NULL, and hence must start with a "/".

Note that any message-body attached to the request (such as for a POST request) may not be available to the resource that is the target of the redirect.

6.3.3. Status

The Status header field contains a 3-digit integer result code that indicates the level of success of the script's attempt to handle the request.

Status	= "Status:" status-code SP reason-phrase NL
status-code	= "200" "302" "400" "501" extension-code
extension-code	= 3digit
reason-phrase	= *TEXT

Status code 200 'OK' indicates success, and is the default value assumed for a document response. Status code 302 'Found' is used with a Location header field and response message-body. Status code 400 'Bad Request' may be used for an unknown request format, such as a missing CONTENT_TYPE. Status code 501 'Not Implemented' may be returned by a script if it receives an unsupported REQUEST_METHOD.

Other valid status codes are listed in section 6.1.1 of the HTTP specifications [1], [4], and also the IANA HTTP Status Code Registry [8] and MAY be used in addition to or instead of the

しないでください。文字セットパラメータの変更を除いて、値はクライアントに変更されずに送信される必要があります。

それ以外の場合を定義しない限り、プロトコルがHTTPおよびUS-ASCIIの場合は、クライアントに想定されているデフォルトの文字セットがISO-8859-1です。したがって、スクリプトにはCharsetパラメータを含める必要があります。この問題の説明については、HTTP/1.1仕様書[4]のセクション3.4.1を参照してください。

6.3.2. Location

Locationヘッダーフィールドは、スクリプトが実際の文書ではなく文書への参照を返しているサーバーに指定するために使用されます（セクション6.2.3と6.2.4を参照）。それは絶対的なURI（任意選択でフラグメント識別子を持つ）であり、クライアントが参照されている文書を取得すること、またはローカルURIパス（オプションでクエリ文字列を含む）が参照され、サーバーが参照された文書を取得して戻ること示すことを示しています。応答としてクライアントに。

AbsoluteURIの構文は、RFC 2396 [2]とRFC 2732 [7]で指定されたものからこの文書に組み込まれています。有効なAxtriputeURIは常にスキームの名前で始まり、続いて ":";スキーム名は文字で始まり、英数字、 "+", "-"または "."。ローカルのURIパスとクエリは絶対パスでなければならない、相対パスまたはNULLではなく、したがって "/"で始める必要があります。

リクエストに添付されているメッセージ本文（POSTリクエストなど）がリダイレクトのターゲットであるリソースには使用できないことがあります。

6.3.3. 状態

[Status Header]フィールドには、スクリプトの要求の処理の試みの成功のレベルを示す3桁の整数結果コードが含まれています。

ステータスコード200 'OK'は成功を示し、文書応答に想定されているデフォルト値です。ステータスコード302 'Found'は、ロケーションヘッダーフィールドと応答メッセージボディで使用されます。ステータスコード400 'Bad Request'は、欠落CONTENT_TYPEなどの未知の要求形式に使用できます。ステータスコード501 'Not Implemented'は、サポートされていない要求_methodを受信した場合、スクリプトによって返される可能性があります。

その他の有効なステータスコードは、HTTP仕様[1]、[4]、およびIANA HTTPステータスコードレジストリ[8]のセクション6.1.1にリストされており、上記のものに加えて、またはその代わり

ones listed above. The script SHOULD check the value of SERVER_PROTOCOL before using HTTP/1.1 status codes. The script MAY reject with error 405 'Method Not Allowed' HTTP/1.1 requests made using a method it does not support.

Note that returning an error status code does not have to mean an error condition with the script itself. For example, a script that is invoked as an error handler by the server should return the code appropriate to the server's error condition.

The reason-phrase is a textual description of the error to be returned to the client for human consumption.

6.3.4. Protocol-Specific Header Fields

The script MAY return any other header fields that relate to the response message defined by the specification for the SERVER_PROTOCOL (HTTP/1.0 [1] or HTTP/1.1 [4]). The server MUST translate the header data from the CGI header syntax to the HTTP header syntax if these differ. For example, the character sequence for newline (such as UNIX's US-ASCII LF) used by CGI scripts may not be the same as that used by HTTP (US-ASCII CR followed by LF).

The script MUST NOT return any header fields that relate to client-side communication issues and could affect the server's ability to send the response to the client. The server MAY remove any such header fields returned by the client. It SHOULD resolve any conflicts between header fields returned by the script and header fields that it would otherwise send itself.

6.3.5. Extension Header Fields

There may be additional implementation-defined CGI header fields, whose field names SHOULD begin with "X-CGI-". The server MAY ignore (and delete) any unrecognised header fields with names beginning "X-CGI-" that are received from the script.

6.4. Response Message-Body

The response message-body is an attached document to be returned to the client by the server. The server MUST read all the data provided by the script, until the script signals the end of the message-body by way of an end-of-file condition. The message-body SHOULD be sent unmodified to the client, except for HEAD requests or any required transfer-codings, content-codings or charset conversions.

response-body = *OCTET

7. System Specifications

7.1. AmigaDOS

Meta-Variables Meta-variables are passed to the script in identically named environment variables. These are accessed by the DOS library routine GetVar(). The flags argument SHOULD be 0. Case is ignored, but upper case is recommended for compatibility with case-sensitive systems.

The current working directory The current working directory for the script is set to the directory containing the script.

に使用できます。スクリプトは、HTTP/1.1ステータスコードを使用する前にSERVER_PROTOCOLの値を確認する必要があります。スクリプトは、エラー405 'Method Not Allowed'で拒否される場合があります。メソッドを使用して作成されたHTTP/1.1の要求はサポートされていません。

エラーステータスコードを返すことは、スクリプト自体とのエラー状態を意味する必要はありません。たとえば、サーバーによってエラーハンドラとして呼び出されるスクリプトは、サーバーのエラー状態に適したコードを返す必要があります。

reason-phraseは、人間の消費のためにクライアントに返されるエラーのテキスト記述です。

6.3.4. プロトコル固有のヘッダーフィールド

このスクリプトは、SERVER_PROTOCOLの指定によって定義された応答メッセージに関連する他のヘッダーフィールド（HTTP/1.0 [1]またはHTTP/1.1 [4]）を返すことができます。これらが異なる場合、サーバーはヘッダーデータをCGIヘッダー構文からHTTPヘッダー構文に変換する必要があります。たとえば、CGIスクリプトで使用される改行の文字列（UNIXのUS-ASCII LFなど）は、HTTP（US-ASCII CRに続くLF）で使用されているものと同じではない可能性があります。

スクリプトは、クライアント側の通信の問題に関連するヘッダーフィールドを返さないで、サーバーのクライアントへの応答を送信する機能に影響を与える可能性があります。サーバーは、クライアントから返されるそのようなヘッダーフィールドを削除することがあります。スクリプトフィールドとヘッダーフィールドから返されるヘッダーフィールドとそれ自身が送信するヘッダーフィールドの間の競合を解決する必要があります。

6.3.5. 拡張ヘッダーフィールド

追加の実装定義定義されたCGIヘッダーフィールドがあります。そのフィールド名は "X-CGI-"で始める必要があります。サーバーは、スクリプトから受信した「X-CGI-」という名前の認識されていないヘッダーフィールドを無視（および削除）することがあります。

6.4. 応答メッセージ - ボディ

応答メッセージ本文は、サーバーによってクライアントに返される添付文書です。サーバーは、スクリプトがメッセージ本体の終わりを介してメッセージ本体の終わりを通知するまで、スクリプトによって提供されるすべてのデータを読み取る必要があります。メッセージ本文は、ヘッド要求または必要な転送コード、コンテンツコーディング、または文字セット変換を除いて、クライアントに変更されずに送信されるべきです。

7. システム仕様

7.1. AmigaDOS

メタ変数メタ変数は、同じ名前の環境変数でスクリプトに渡されます。これらはDOSライブラリーgetVar（）によってアクセスされます。flags引数は0にする必要があります。

現在の作業ディレクトリスクリプトの現在の作業ディレクトリは、スクリプトを含むディレクトリに設定されています。

Character set The US-ASCII character set [9] is used for the definition of meta-variables, header fields and values; the newline (NL) sequence is LF; servers SHOULD also accept CR LF as a newline.

7.2. UNIX

For UNIX compatible operating systems, the following are defined:

Meta-Variables Meta-variables are passed to the script in identically named environment variables. These are accessed by the C library routine `getenv()` or variable `environ`.

The command line This is accessed using the `argc` and `argv` arguments to `main()`. The words have any characters which are 'active' in the Bourne shell escaped with a backslash.

The current working directory The current working directory for the script SHOULD be set to the directory containing the script.

Character set The US-ASCII character set [9], excluding NUL, is used for the definition of meta-variables, header fields and CHAR values; TEXT values use ISO-8859-1. The PATH_TRANSLATED value can contain any 8-bit byte except NUL. The newline (NL) sequence is LF; servers should also accept CR LF as a newline.

7.3. EBCDIC/POSIX

For POSIX compatible operating systems using the EBCDIC character set, the following are defined:

Meta-Variables Meta-variables are passed to the script in identically named environment variables. These are accessed by the C library routine `getenv()`.

The command line This is accessed using the `argc` and `argv` arguments to `main()`. The words have any characters which are 'active' in the Bourne shell escaped with a backslash.

The current working directory The current working directory for the script SHOULD be set to the directory containing the script.

Character set The IBM1047 character set [21], excluding NUL, is used for the definition of meta-variables, header fields, values, TEXT strings and the PATH_TRANSLATED value. The newline (NL) sequence is LF; servers should also accept CR LF as a newline.

media-type charset default The default charset value for text (and other implementation-defined) media types is IBM1047.

8. Implementation

8.1. Recommendations for Servers

Although the server and the CGI script need not be consistent in their handling of URL paths (client URLs and the PATH_INFO data, respectively), server authors may wish to impose consistency. So the server implementation should specify its behaviour for the following cases:

- 1. define any restrictions on allowed path segments, in particular whether non-terminal NULL segments are permitted;

文字セットUS-ASCII文字セット[9]は、メタ変数、ヘッダーフィールド、および値の定義に使用されます。改行（NL）シーケンスはLFです。サーバーはまた、改行としてCR LFを受け入れる必要があります。

7.2. UNIX

UNIX互換オペレーティングシステムの場合、以下は次のとおりです。

メタ変数メタ変数は、同じ名前の環境変数でスクリプトに渡されます。これらはCライブラリー`getEnv（）`または変数環境によってアクセスされます。

これは`main（）`に`argc`と`argv`引数を使用してアクセスされます。言葉には、Bourneシェルで「アクティブ」な文字がバックスラッシュでエスケープされています。

現在の作業ディレクトリスクリプトの現在の作業ディレクトリは、スクリプトを含むディレクトリに設定する必要があります。

文字セットNULを除くUS-ASCII文字セット[9]は、メタ変数、ヘッダーフィールド、およびCHAR値の定義に使用されます。テキスト値ISO-8859-1を使用します。PATH_TRANSLATED値は、NUL以外の8ビットバイトを含めることができます。改行（NL）シーケンスはLFです。サーバーはまた、改行としてCR LFを受け入れる必要があります。

7.3. EBCDIC/POSIX

EBCDIC文字セットを使用したPOSIX互換オペレーティングシステムの場合は、次のものが定義されています。

メタ変数メタ変数は、同じ名前の環境変数でスクリプトに渡されます。これらはCライブラリールーチン`getEnv（）`によってアクセスされます。

これは`main（）`に`argc`と`argv`引数を使用してアクセスされます。言葉には、Bourneシェルで「アクティブ」な文字がバックスラッシュでエスケープされています。

現在の作業ディレクトリスクリプトの現在の作業ディレクトリは、スクリプトを含むディレクトリに設定する必要があります。

文字セットNULを除くIBM1047文字セット[21]は、メタ変数、ヘッダーフィールド、値、テキスト文字列、およびPATH_TRANSLATED値の定義に使用されます。改行（NL）シーケンスはLFです。サーバーはまた、改行としてCR LFを受け入れる必要があります。

メディアタイプの文字セットのデフォルトテキスト（およびその他の実装定義）メディアタイプのデフォルトの文字セット値はIBM1047です。

8. 実装

8.1. サーバーの推奨事項

サーバーとCGIスクリプトは、URLパスの処理（クライアントURLとPATH_INFOデータ）の処理に一貫している必要はありませんが、サーバーの作者は一貫性を課すことを望むかもしれません。そのため、サーバーの実装は次のような場合にその動作を指定する必要があります。

- 1. 許可されたパスセグメントに対する制限、特に非端末NULLセグメントが許可されているかどうかを定義します。

2. define the behaviour for "." or ".." path segments; i.e., whether they are prohibited, treated as ordinary path segments or interpreted in accordance with the relative URL specification [2];
3. define any limits of the implementation, including limits on path or search string lengths, and limits on the volume of header fields the server will parse.

8.2. Recommendations for Scripts

If the script does not intend processing the PATH_INFO data, then it should reject the request with 404 Not Found if PATH_INFO is not NULL.

If the output of a form is being processed, check that CONTENT_TYPE is "application/x-www-form-urlencoded" [18] or "multipart/form-data" [16]. If CONTENT_TYPE is blank, the script can reject the request with a 415 'Unsupported Media Type' error, where supported by the protocol.

When parsing PATH_INFO, PATH_TRANSLATED or SCRIPT_NAME the script should be careful of void path segments ("/") and special path segments (".", and ".."). They should either be removed from the path before use in OS system calls, or the request should be rejected with 404 'Not Found'.

When returning header fields, the script should try to send the CGI header fields as soon as possible, and should send them before any HTTP header fields. This may help reduce the server's memory requirements.

Script authors should be aware that the REMOTE_ADDR and REMOTE_HOST meta-variables (see sections 4.1.8 and 4.1.9) may not identify the ultimate source of the request. They identify the client for the immediate request to the server; that client may be a proxy, gateway, or other intermediary acting on behalf of the actual source client.

9. Security Considerations

9.1. Safe Methods

As discussed in the security considerations of the HTTP specifications [1], [4], the convention has been established that the GET and HEAD methods should be 'safe' and 'idempotent' (repeated requests have the same effect as a single request). See section 9.1 of RFC 2616 [4] for a full discussion.

9.2. Header Fields Containing Sensitive Information

Some HTTP header fields may carry sensitive information which the server should not pass on to the script unless explicitly configured to do so. For example, if the server protects the script by using the Basic authentication scheme, then the client will send an Authorization header field containing a username and password. The server validates this information and so it should not pass on the password via the HTTP_AUTHORIZATION meta-variable without careful consideration. This also applies to the Proxy-Authorization header field and the corresponding HTTP_PROXY_AUTHORIZATION meta-variable.

9.3. Data Privacy

2. ". "の動作を定義します。または ".."パスセグメント。すなわち、それらが禁止されているか、通常の経路セグメントとして扱われるか、または相対的なURL仕様に従って解釈されるかどうか[2]。
3. パスまたは検索文字列の長さの制限など、実装の制限を定義し、サーバーが解析するヘッダーフィールドのボリューム上の制限です。

8.2. スクリプトの推奨事項

スクリプトがPATH_INFOデータの処理をインテンドしない場合、PATH_INFOがNULLではない場合は404で要求を拒否します。

フォームの出力が処理されている場合は、CONTENT_TYPEが "application/x-www-orf-urlencoded"または "multipart/form-data" [16]であることを確認してください。CONTENT_TYPEが空白の場合、スクリプトはプロトコルによってサポートされている415 'サポートされていないメディアタイプ'のエラーで要求を拒否できます。

PATH_INFOを解析するとき、PATH_TRANSLATEDまたはSCRIPT_NAMEスクリプトは、Void Pathセグメント ("/") と特殊なパスセグメント (". "と ".. ") に注意してください。OSシステムコールで使用する前にパスから削除する必要があります。また、リクエストは404 'Not Found'で拒否されるべきです。

ヘッダーフィールドを返すとき、スクリプトはできるだけ早くCGIヘッダーフィールドを送信し、HTTPヘッダーフィールドの前に送信する必要があります。これはサーバーのメモリ要件を減らすのに役立ちます。

スクリプト作成者は、REMOTE_ADDRとREMOTE_HOSTメタ変数（セクション4.1.8および4.1.9を参照）が要求の最終的なソースを識別できないことに注意する必要があります。それらはサーバーへの即時要求のためにクライアントを識別します。そのクライアントは、実際のソースクライアントの代わりに行動するプロキシ、ゲートウェイ、またはその他の仲介者であり得る。

9. セキュリティに関する考慮事項

9.1. 安全なメソッド

HTTP仕様[1]、[4]のセキュリティ上の考慮事項で説明したように、GETメソッドとヘッドメソッドは 'safe' と 'idempotent' であるべきです（繰り返し要求は単一の要求と同じ効果を持っています）。。完全な議論のためのRFC 2616 [4]のセクション9.1を参照してください。

9.2. 機密情報を含むヘッダフィールド

一部のHTTPヘッダーフィールドは、明示的に構成されていない限り、サーバーがスクリプトに渡されないような機密情報を伝えることができます。たとえば、基本認証方式を使用してサーバーがスクリプトを保護すると、クライアントはユーザー名とパスワードを含む認証ヘッダーフィールドを送信します。サーバーはこの情報を検証しているので、慎重に検討することなくhttp_authorizationメタ変数を介してパスワードを渡すべきではありません。これは、プロキシ認証ヘッダフィールドと対応するHTTP_PROXY_AUTHORIZATIONメタ変数にも適用されます。

9.3. データのプライバシー

Confidential data in a request should be placed in a message-body as part of a POST request, and not placed in the URI or message headers. On some systems, the environment used to pass meta-variables to a script may be visible to other scripts or users. In addition, many existing servers, proxies and clients will permanently record the URI where it might be visible to third parties.

9.4. Information Security Model

For a client connection using TLS, the security model applies between the client and the server, and not between the client and the script. It is the server's responsibility to handle the TLS session, and thus it is the server which is authenticated to the client, not the CGI script.

This specification provides no mechanism for the script to authenticate the server which invoked it. There is no enforced integrity on the CGI request and response messages.

9.5. Script Interference with the Server

The most common implementation of CGI invokes the script as a child process using the same user and group as the server process. It should therefore be ensured that the script cannot interfere with the server process, its configuration, documents or log files.

If the script is executed by calling a function linked in to the server software (either at compile-time or run-time) then precautions should be taken to protect the core memory of the server, or to ensure that untrusted code cannot be executed.

9.6. Data Length and Buffering Considerations

This specification places no limits on the length of the message-body presented to the script. The script should not assume that statically allocated buffers of any size are sufficient to contain the entire submission at one time. Use of a fixed length buffer without careful overflow checking may result in an attacker exploiting 'stack-smashing' or 'stack-overflow' vulnerabilities of the operating system. The script may spool large submissions to disk or other buffering media, but a rapid succession of large submissions may result in denial of service conditions. If the CONTENT_LENGTH of a message-body is larger than resource considerations allow, scripts should respond with an error status appropriate for the protocol version; potentially applicable status codes include 503 'Service Unavailable' (HTTP/1.0 and HTTP/1.1), 413 'Request Entity Too Large' (HTTP/1.1), and 414 'Request-URI Too Large' (HTTP/1.1).

Similar considerations apply to the server's handling of the CGI response from the script. There is no limit on the length of the header or message-body returned by the script; the server should not assume that statically allocated buffers of any size are sufficient to contain the entire response.

9.7. Stateless Processing

The stateless nature of the Web makes each script execution and resource retrieval independent of all others even when multiple requests constitute a single conceptual Web transaction. Because of this, a script should not make any

要求内の機密データは、POST要求の一部としてメッセージ本体に配置され、URIまたはメッセージヘッダーに配置されていません。一部のシステムでは、メタ変数をスクリプトに渡すために使用される環境は、他のスクリプトまたはユーザーに表示されます。さらに、多くの既存のサーバー、プロキシ、およびクライアントが、第三者から表示される可能性があるURIを恒久的に記録します。

9.4. 情報セキュリティモデル

TLSを使用したクライアント接続の場合、セキュリティモデルはクライアントとサーバーの間に適用されます。クライアントとスクリプトの間では適用されます。サーバーのTLSセッションを処理するのはサーバーの責任です。したがって、CGIスクリプトではなくクライアントに認証されているサーバーです。

この仕様は、スクリプトが呼び出されたサーバーを認証するためのメカニズムを提供しません。CGI要求および応答メッセージに強制的な整合性はありません。

9.5. サーバーとのスクリプト干渉

CGIの最も一般的な実装は、サーバープロセスとして同じユーザーとグループを使用して子プロセスとしてスクリプトを呼び出します。したがって、スクリプトがサーバープロセス、その構成、文書、またはログファイルを妨害できないことを確認する必要があります。

スクリプトがサーバーソフトウェアにリンクされている関数を呼び出して（コンパイル時または実行時に）関数を呼び出すことによって実行されると、サーバーのコアメモリを保護するため、または信頼されていないコードを実行できないように注意してください。

9.6. データ長とバッファリングの考慮事項

この仕様は、スクリプトに提示されたメッセージ本文の長さに制限はありません。スクリプトは、任意のサイズの静的に割り当てられたバッファが一度に送信全体を含むのに十分であると仮定しないでください。慎重なオーバーフローチェックなしに固定長バッファを使用すると、オペレーティングシステムの「スタックスマッシング」または「スタックオーバーフロー」の脆弱性を悪用する攻撃者が発生する可能性があります。このスクリプトは、大きな提出物をディスクまたは他のバッファリングメディアにスプールすることができますが、急速な承継の大きな提出がサービス条件の拒否になる可能性があります。メッセージボディのcontent_lengthがリソースの考慮事項より大きい場合、スクリプトはプロトコルバージョンに適したエラーステータスで応答する必要があります。潜在的に適用可能なステータスコードには、503 'Service Unavailable'（HTTP / 1.0およびHTTP / 1.1）、413 '要求エンティティが大きすぎる'（HTTP / 1.1）、および414 'Request-URIが大きすぎる'（http / 1.1）を備えています。

サーバーのスクリプトからのCGI応答の処理にも同様の考慮事項が適用されます。スクリプトによって返されるヘッダーまたはメッセージ本文の長さに制限はありません。サーバーは、任意のサイズの静的に割り当てられたバッファが応答全体を含むのに十分であると仮定してはならない。

9.7. ステートレス処理

Webのステートレス性は、複数の要求が単一の概念Webトランザクションを構成している場合でも、各スクリプトの実行とリソースの検索をすべてのものとは無関係にします。このため、スクリプトは、要求を送信するユーザーエージェントのコンテ

assumptions about the context of the user-agent submitting a request. In particular, scripts should examine data obtained from the client and verify that they are valid, both in form and content, before allowing them to be used for sensitive purposes such as input to other applications, commands, or operating system services. These uses include (but are not limited to) system call arguments, database writes, dynamically evaluated source code, and input to billing or other secure processes. It is important that applications be protected from invalid input regardless of whether the invalidity is the result of user error, logic error, or malicious action.

Authors of scripts involved in multi-request transactions should be particularly cautious about validating the state information; undesirable effects may result from the substitution of dangerous values for portions of the submission which might otherwise be presumed safe. Subversion of this type occurs when alterations are made to data from a prior stage of the transaction that were not meant to be controlled by the client (e.g., hidden HTML form elements, cookies, embedded URLs, etc.).

9.8. Relative Paths

The server should be careful of "." path segments in the request URI. These should be removed or resolved in the request URI before it is split into the script-path and extra-path. Alternatively, when the extra-path is used to find the PATH_TRANSLATED, care should be taken to avoid the path resolution from providing translated paths outside an expected path hierarchy.

9.9. Non-parsed Header Output

If a script returns a non-parsed header output, to be interpreted by the client in its native protocol, then the script must address all security considerations relating to that protocol.

10. Acknowledgements

This work is based on the original CGI interface that arose out of discussions on the 'www-talk' mailing list. In particular, Rob McCool, John Franks, Ari Luotonen, George Phillips and Tony Sanders deserve special recognition for their efforts in defining and implementing the early versions of this interface.

This document has also greatly benefited from the comments and suggestions made Chris Adie, Dave Kristol and Mike Meyer; also David Morris, Jeremy Madea, Patrick McManus, Adam Donahue, Ross Patterson and Harald Alvestrand.

11. References

11.1 Normative References

[1] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[2] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI) : Generic Syntax", RFC 2396, August 1998.

[3] Bradner, S., "Key words for use in RFCs to Indicate

キストについての仮定を任意にしないでください。特に、スクリプトはクライアントから取得したデータを調べて、それらが他のアプリケーション、コマンド、またはオペレーティングシステムサービスへの入力などの機密の目的に使用されることを可能にする前に、それらがフォームとコンテンツの両方で有効であることを確認する必要があります。これらの用途には、システムコール引数、データベース書き込み、動的に評価されたソースコード、および請求またはその他の安全なプロセスへの入力が含まれます（ただし、これらに限定されません）。無効性がユーザーエラー、ロジックエラー、または悪意のあるアクションの結果に関係なく、アプリケーションが無効な入力から保護されることが重要です。

マルチリクエストトランザクションに関与するスクリプトの作成者は、状態情報を検証することについて特に注意してください。望ましくない影響は、そうでなければ安全に推定されるかもしれない提出の部分に対する危険な値の代わりに起因する可能性があります。このタイプのサブバージョンは、クライアントによって制御されることを意図していない（例えば、隠されたHTMLフォーム要素、クッキー、埋め込みURLなど）、トランザクションの以前のステージからのデータに対して変更が行われたときに発生します。

9.8. 相対パス

サーバーは、リクエストURI内のパスセグメントに注意してください。これらはスクリプトパスとエキストラパスに分割される前に、リクエストURIで削除または解決されます。あるいは、PATH_TRANSLATEDを見つけるためにエラクタパスを使用すると、パス解決が予想されるパス階層の外側の翻訳されたパスを提供することを回避するように注意する必要があります。

9.9. 非解析ヘッダ出力

スクリプトが解析されていないヘッダー出力をネイティブプロトコルでクライアントによって解釈するように返す場合、スクリプトはそのプロトコルに関するすべてのセキュリティ上の考慮事項に対処する必要があります。

10. 謝辞

この作業は、「WWW-Talk」メーリングリストで議論から発生したオリジナルのCGIインターフェースに基づいています。特に、Rob Mccool、John Franks、Ari Luotonen、George Phillips、Tony Sandersは、このインタフェースの初期のバージョンを定義し実装するための努力のための特別な認識に値する。

この文書はまた、Chris Adie、Dave KristolとMike Meyerに行われたコメントや提案から大きく恩恵を受けています。David Morris、Jeremy Madea、Patrick Mcmanus、Adam Donahue、Ross Patterson、Harald Alvestrand。

11. 参考文献

11.1 引用文献

[1] Berners-Lee、T.、Fielding、R.およびH. Frystyk、"Hypertext Transfer Protocol - HTTP / 1.0"、RFC 1945、1996年5月。

[2] Berners-Lee、T.、Fielding、R.およびL.Masinter、「Uniform Resource Identifiers（URI）：Generic Syntax」、RFC 2396、1998年8月。

[3] Bradner、S。、「RFCで使用するためのキーワード」、BCP

Requirements Levels", BCP 14, RFC 2119, March 1997.

[4] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[5] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[7] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

[8] "HTTP Status Code Registry", <http://www.iana.org/assignments/http-status-codes>, IANA.

[9] "Information Systems -- Coded Character Sets -- 7-bit American Standard Code for Information Interchange (7-Bit ASCII)", ANSI INCITS.4-1986 (R2002).

[10] "Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1", ISO/IEC 8859-1:1998.

11.2. Informative References

[11] Berners-Lee, T., "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, June 1994.

[12] Braden, R., Ed., "Requirements for Internet Hosts -- Application and Support", STD 3, RFC 1123, October 1989.

[13] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[14] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[15] Hinden R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[16] Masinter, L., "Returning Values from Forms: multipart/form-data", RFC 2388, August 1998.

[17] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[18] Raggett, D., Le Hors, A., and I. Jacobs, Eds., "HTML 4.01 Specification", W3C Recommendation December 1999, <http://www.w3.org/TR/html401/>.

[19] Rescola, E. "HTTP Over TLS", RFC 2818, May 2000.

[20] St. Johns, M., "Identification Protocol", RFC 1413, February 1993.

[21] IBM National Language Support Reference Manual Volume 2, SE09-8002-01, March 1990.

[22] "The Common Gateway Interface", <http://hoohoo.ncsa.uiuc.edu/cgi/>, NCSA, University of Illinois.

14、RFC 2119、1997年3月。

[4] R.、Mogul、J.、Frystyk、H.、Masinter、L.、L.、L.、P.、およびT.Berners-Lee、「Hypertext Transfer Protocol - HTTP / 1.1」、RFC 26161999年6月。

[5] Franks、J.、Hallam-Baker、P.、Hostetler、J.、Lawrence、S.、Leact、P.、Luotonen、A.、およびL.Stewart、「HTTP認証：基本およびダイジェストアクセス認証」、RFC 26171999年6月。

[6] Freed、N.およびN.Borenstein、「Multipurpose Internet Mail Extensions（Multipurpose Internet Mail Extensions（MIME）2：メディアタイプ」、RFC 2046、1996年11月。

[7] Hinden、R.、Carpenter、B.、L. Masinter、「URLのリテラルIPv6アドレスのフォーマット」、RFC 2732、1999年12月。

[8] "HTTPステータスコードレジストリ"、<http://www.iana.org/ashignments/http-status-codes>、IANA。

[9] 「情報システム - 符号化文字セット - 情報交換用の7ビットアメリカ標準コード（7ビットASCII）」、ANSI焼き付け4-1986（R2002）。

[10] 「情報技術 - 8ビットのシングルバイトコード化グラフィック文字セット - 第1部：ラテンアルファベットNo.1」、ISO / IEC 8859-1：1998。

11.2. 参考引用

[11] Berners-Lee、T.、「WWWのユニバーサルリソース識別子：ワールドワイドウェブで使用されているネットワーク上のオブジェクトの名前とアドレスの表現の統一構文」、RFC 1630、1994年6月。

[12] Braden、R.、Ed.、「インターネットホストの要件 - アプリケーションとサポート」、STD 3、RFC 1123、1989年10月。

[13] Crocker、D.、「ARPAインターネットテキストメッセージのフォーマットの標準」、STD 11、RFC 822、1982年8月。

[14] Dierks、T.およびC. Allen、「Thels Protocol Version 1.0」、RFC 2246、1999年1月。

[15] 2003年4月、RFC 3513、「インターネットプロトコルバージョン6（IPv6）アドレッシングアーキテクチャ」、RFC 3513。

[16] 1998年8月、RFC 2388、RFC 2388、RFC 2388、RFC 2388から、Masinter、L. Masinter L. "の値を返す。

[17] Mockapetris、P.、「ドメイン名 - コンセプトと施設」、STD 13、RFC 1034、1987年11月。

[18] Raggett、D.、Le Hors、A.、I. Jacobs、EDS、「HTML 4.01仕様」、W3C勧告1999年12月、<http://www.w3.org/tr/html401/>

[19] Rescola、E. "HTTP over TLS"、RFC 2818、2000年5月。

[20] St. Johns、M.、「識別プロトコル」、RFC 1413、1993年2月。

[21] IBM National Language Supportリファレンスマニュアルボリューム2、SE09-8002-01、1990年3月。

[22] "Common Gateway Interface"、<http://hoohoo.ncsa.uiuc.edu/cgi/>、イリノイ大学NCSA。

12. Authors' Addresses

David Robinson The Apache Software Foundation

EMail: drtr@apache.org

Ken A. L. Coar The Apache Software Foundation

EMail: coar@apache.org

13. Full Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78 and at www.rfc-editor.org, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the ISOC's procedures with respect to rights in ISOC Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

12. 著者の住所

David Robinson Apache Software Foundation

Ken A. L. COAR Apache Software Foundation

13. 完全著作権宣言

著作権（C）インターネット社会（2004）。この文書は、BCP 78とwww.rfc-editor.orgに含まれている権利、ライセンス、制限の対象となります。

この文書と本明細書に含まれる情報は、「現状のまま」基準で提供されており、投稿者、または（いずれかの場合）、インターネット社会とインターネットエンジニアリングのタスクフォースがすべての保証を損なう、または本明細書における情報の使用が、特定の目的のためのあらゆる権利または黙示の保証を侵害しないことを含むがこれらに限定されないが、これに限定されない。

知的財産

この文書に記載されているテクノロジーの実装または使用に関連すると主張される可能性がある、またはそのような権利の下でのライセンスの使用に関連すると主張される可能性がある、またはその他の権利の下にある範囲内である可能性がある、またはその他の権利の使用に関連すると主張する可能性がある、IETFは、IETFを取りません。利用可能です。そのような権利を特定するためにそれが独立した努力をしたことを表していません。ISOC文書の権利に関するISOCの手順に関する情報は、BCP 78およびBCP 79にあります。

IETF事務局へのIETF事務局と利用可能なライセンスの保証のコピー、またはこの仕様書の実装者や利用者による一般的なライセンスまたは許可を得るための試みの結果を得ることができます。IETFオンラインIPRリポジトリから<http://www.ietf.org/ipr>。

IETFは、著作権、特許または特許出願、またはこの規格を実装することが要求される可能性がある技術をカバーする可能性のある他の独自の権利を注意を及ぼすように興味のある当事者を勧めます。ietf-ipr@ietf.orgのIETFに情報を宛先に宛ててください。

謝辞

RFCエディタ機能のための資金は、現在インターネット社会によって提供されています。