FIND Working Group
Internet Draft
<draft-ietf-find-cip-mime-01.txt>
21 November 1997
Expires in six months

J. Allen
WebTV Networks, Inc.
Michael Mealling
Network Solutions, Inc.

MIME Object Definitions for the Common Indexing Protocol (CIP)

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

The Common Indexing Protocol (CIP) is used to pass indexing information from server to server in order to facilitate query routing. The protocol is comprised of several MIME objects being passed from server to server. This document describes the definitions of those objects as well as the methods and requirements needed to define a new index type.

1. Introduction

The Common Indexing Protocol (CIP) is used to pass indexes between servers that combine multiple indexes and/or route queries based on those indexes. The overall framework for the protocol is specified in the CIP Framework document [FRAMEWORK]. This document should be read within the context of that document as there are fundamental concepts contained in the framework that are not fully explained here.

Since there are several different ways to index a given database there will be multiple types of indexes to pass. These indexes may have different transport requirements, different ways of specifying parameters, and different referral rules. These different requirements are handled by encapsulating the indexes within MIME wrappers in order to have a standardized way to specify

those different parameters.

Appendix A contains the actual MIME [RFC2046] registration templates to be sent to the IANA for registration [RFC2048] as soon as this document is accepted by the IESG.

This document uses language like SHOULD and SHALL that have special meaning as specified in "Key words for use in RFCs to Indicate Requirement Levels". [RFC2119]

2.0 CIP Transactions

Messages passed by CIP implementations over reliable transport mechanisms fall into three categories, requests, responses and results. All requests result in either a response or a result. A result sent in response to a request must be interpreted as a successful operation.

Requests, responses and results are formatted as MIME [RFC2046] messages. The specific MIME types involved are defined below.

As with all MIME objects, CIP messages may be wrapped in a security multipart package to provide authentication and privacy. The security policy with respect to all messages is implementation defined, when not explicitly discussed below. CIP implementors are strongly urged to allow server administrators maximum configurability to secure their servers against malicious sent anonymous CIP messages. In general, operations which can permanently change the server's state in a harmful way should only take place upon receipt of a properly signed message from a trusted CIP peer or administrator. Implementors should provide appropriate auditing capabilities so that both successful and failed requests can be tracked by the server administrator.

Since these MIME objects can and will be sent over several different protocols, body termination is specified by the transfer protocol. New protocols are encouraged to use SMTP [RFC821] style body termination.

Finally, since MIME objects can specify their own encoding, the line-breaks contained within each body are defined by the encoding. Thus, instead of specifying them as carriage-return and/or linefeed, the identifier <linebreak> is used. Linebreaks in the headers and separating the body from the headers follow existing standards.

2.1 Common syntactic definitions

There are certain common syntactic elements to all of the CIP transactions. These include type, DSI and the Base-URI.

2.1.1 The "application/index" MIME type tree

Due to requirements in RFC2048 concerning objects that have the same type but different syntaxes, CIP objects will use the application/cip tree but include "facets" [RFC2048] which extend it as other types have done with respect to global elements and vendor specific

enhancements. Thus the tree is divided up into the following branches:

application/index.cmd._command_
application/index.response
application/index.obj._type_
application/index.vnd._xxx_

command is a command as specified here. It contains commands and their arguments.

type identifies what type of CIP index object is contained within the body. It is unique among all other reserved types. Reserved types are those previously documented by other CIP index object specifications, according to standard IETF processes.

xxx is an identifier specified by a vendor for use by that vendor in operations specifically to do with indexes.

All of the above identifiers follow the rules in RFC2048 for valid MIME types. In addition commands, responses and types are limited by this document to consist of from 1 to 20 characters from the set [a-zA-Z0-9-]; that is, all upper and lower case letters, all digits, and the ASCII minus character (decimal 45). Though type names may be specified case sensitively, they must be compared and otherwise processed case insensitively.

Appendix A contains the registration template for the application/index tree that will be sent to the IANA upon approval by the IESG.

2.1.2 DSI

A dataset identifier is an identifier chosen from any part of the ISO/CCITT OID space. The DSI uniquely identifies a given dataset among all datasets indexed by CIP.

As currently defined, OID's are an unbounded sequence of unbounded integers. While this creates an infinite numbering space, it presents problems for implementors dealing with machines with finite resources. To ease implementation, this document specifies an ASCII encoding of the OID, and specifies limits which make implementation easier.

For the purposes of interchange in CIP messages, an OID must conform to the following rules:

```
dsi          = integer *( "." integer)
integer      = all-digits / (one-to-nine *all-digits)
one-to-nine  = "1" / "2" / "3" / "4" / "5" / "6" / "7" /
               "8" / "9"
all-digits   = "0" / one-to-nine
```

Under no circumstances shall the total length of the resulting string exceed 255 characters. OID's which cannot, due to their length, conform to these rules must not be used as CIP dataset identifiers.

An implementation must not attempt to parse the individual integers unless it is prepared to handle arbitrary-length integers. Treating the DSI as anything other than an opaque string of US-ASCII characters is not recommended.

Two CIP DSI's are considered to match if both conform to the above rules and every number matches.

2.1.3. Base-URI

CIP index objects carry base-URI's to facilitate referral generation based on the index object. The base-URI parameter carries a whitespace-delimited list of URL's. URL's are defined in RFC-1738. The exact rules are as follows:

```
base-uri    = genericurl *( 1*whitespace genericurl )
whitespace  = "<space>" (decimal 32) /
              "<tab>"   (decimal 9)  /
              "<cr>"    (decimal 13) /
              "<lf>"    (decimal 10)
genericurl = { as specified in RFC-1738, section 5 }
```

2.2 Response format

All requests must be followed by a response code, except in the cases where a return path is unavailable.

The definition for this MIME type is:

```
MIME type name:          application
MIME subtype name:       index.response
Required parameters:     code
Optional parameters:     charset
Security considerations: (See Section 4)
```

The code parameter contains a 3 digit return code that denotes the status of the last command.

The format of the body is such that the first line is interpreted as the comment corresponding to the code. As with most response codes this comment is intended for human consumption and may not exist and must not be depended on by the protocol. Subsequent lines in the body are reserved for each response to define. In the case where the comment is not given the first must be an empty line.

```
body = comment linebreak payload
comment = { any text }
linebreak = { linebreak characters as defined by transport
              encoding }
payload = { any text }
```

The charset parameter has its normal MIME meaning. Below are several examples:

```
[begin MIME]
Content-type: application/cip.response; code=220


CIP Server v1.0 ready!<linebreak>
[end MIME]

[begin MIME]
Content-type: application/cip.response; code=500

MIME formatting problem<linebreak>
[end MIME]

[begin MIME]
Content-type: application/cip.response; code=520

<linebreak>
[end MIME]
```

While the responses described in this document do not utilize the rest of the lines in the body of a response implementors should take care to not disallow it in the future. A good example would be a message specifying that a poll request did no contain required attributes. This message might look like this:

```
[begin MIME]
Content-type: application/cip.response; code=502

Request is missing required CIP attributes
Missing-Attribute: attribute1
Missing-Attribute: attribute2
Missing-Attribute: attribute3
[end MIME]
```

The meaning of the various digits in the response codes is discussed in RFC-821, Appendix E.

See Appendix B for a list of the valid response codes.

2.3 Command format

A CIP command either initiates an index transfer, interrogates the state of the receiver-CIP (or the server's participation in the mesh), or changes the state of the server (or the server's place in the mesh).

CIP commands are sent as a MIME message of type "application/cip.cmd._command_". The definition for this MIME type tree follows:

MIME type name: application
MIME subtype name: cip.cmd._command_
Optional parameters: type, dsi
Security considerations: (See Section 4)

The format of the body is defined by each command. A general attribute/value pair orientation is preserved throughout the following specified commands. Those developing future command should attempt to maintain that orientation but are not required to do so.

In the following sections, the server's response for each possible value for "command" is defined. Note that the parameters listed as optional above are only optional with respect to the generic MIME form. The optional parameters are only optional with respect to MIME parsing. If one or more of the parameters needed to fulfill a command is missing, a response code of 502 is returned.

Extra optional parameters which are unrecognized must be silently ignored.

2.3.1 No-operation

```
Command Name:        application/cip.cmd.noop
Required parameters: (none)
```

A CIP command with the "command" parameter set to "noop" must be acknowledged with response type code 200 (command OK, no response forthcoming).

This command must not require a signed MIME object. Implementations should accept commands which have been validly signed.

Example:

```
[begin MIME]
Content-type: application/cip.cmd.noop

[end MIME]
```

Note the lack of a body but how the <linebreak> pair is still preserved after the Content-type header.

2.3.2 Poll

```
Request Name:        application/cip.cmd.poll
Required parameters: type, dsi
```

The "poll" command is used by a poller to request the transfer of an index object. It requires the following parameters:

```
type:      The index object type requested
dsi:       The dataset which the index should cover
```

If there are no index objects available for a given DSI, or the receiver-CIP does not support a given index object type, the receiver-CIP must respond with response code 200, (successful, no response forthcoming). Otherwise, the response code must be 201 (successful, response is forthcoming).

The security policy for polling commands is wholly implementation defined. Implementations may be configured to accept or reject anonymous poll commands.

Example:

```
[begin MIME]
Content-type: application/cip.cmd.poll; type="simple";
     dsi= "1.3.5.7.9"

Template: contact name address phone<linebreak>
Start-time: Fri May 30 14:25:30 EDT 1997<linebreak>
End-time: Sat May 31 14:25:30 EDT 1997<linebreak>
[end MIME]
```

2.3.3 DataChanged

```
      Request Name:        application/cip.cmd.datachanged
      Required parameters: type, dsi
```

The "datachanged" command is used by a pollee to notify a poller that the data within an index has changed. It requires the following parameters:

```
      type:      The index object type requested
      dsi:       The dataset which the index should cover
```

If there are no index objects available for a given DSI, or the receiver-CIP does not support a given index object type, the receiver-CIP must respond with response code 200, (successful, no response forthcoming). Otherwise, the response code must be 201 (successful, response is forthcoming).

The body of a DataChanged command is formatted as a simple set of attribute value pairs following the rules of RFC822. The actual attributes and values allowed are defined by the index type specification.

The security policy for DataChanged commands is wholly implementation defined. Implementations may be configured to accept or reject anonymous DataChanged commands.

Example:

```
[begin MIME]
Content-type: application/cip.cmd.datachanged;
     type="simple"; dsi= "1.3.5.7.9"<linebreak>

Time-of-latest-change: Fri May 30 14:25:30 EDT 1997<linebreak>
Time-of-message-generation: Fri May 30 14:25:30 EDT 1997<linebreak>
```

```
   Host-Name: cip.rwhois.net<linebreak>
   Host-Port: 4322<linebreak>
   Protocol: RWhois2.0<linebreak>
   [end MIME]
```

2.3.4 Additional Requests

The request specified above are those required to implement a simple mesh. It is expected that other requests will be developed to handle issues of mesh-management and statistics gathering requests. At this point this an area of additional work. Specifically in the area of mesh management as meshes will tend to be organized around the characteristics of their index type.

2.4. Index Object format

In reply to the "poll" command, a server may choose to send one or more index objects. Regardless of the number of index objects returned, the response must take the form of a MIME multipart/mixed message. Each part must itself be a MIME object of type "application/cip.obj._type_". The definition for this type follows:

```
MIME type name:          application
MIME subtype name:       index.obj._type_
Required parameters:     dsi, base-uri
Optional parameters:     none
Security considerations: (See Section 4)
```

As previously described, each index object is of a particular type. This type is specified in the MIME subtype name since some types may have a different syntax.

The required parameters are to be used as follows:

DSI: The DSI is a string which globally uniquely identifies the dataset from which the index was created.

base-URI: One or more URI's will form the base of any referrals created based upon this index object.

3. Index Type Definition Requirements

Because of the need for application domain specific indices, CIP index objects are abstract; they must be defined by a separate specification. The basic protocols for moving index objects are widely applicable, but the specific design of the index, and the structure of the mesh of servers which pass a particular type of index is dependent on the application domain. While companion documents will describe index objects, there is a set of base requirements and questions those documents must address. This is to ensure that the base assumptions that the CIP protocol makes about its indexes are actually expressible within the index.

Since each type is a MIME type all its own, registration of new types follows the standard registration policies specified in RFC2048.

3.1 Type specific requests

Any index type definition must address the type specific bodies of the Poll and DataChanged requests. All parameters included in the body must be specified.

3.2 The index.obj parameters

3.2.1 Type

See the above definitions for allowed values for type.

A new name must be assigned when any changes to the document describing the index object type are not completely backwards compatible.

3.2.2 DSI

Another attribute is the "DSI", or Dataset Identifier, which uniquely identifies the dataset from which the index was created. The index specification should define the policies for how the DSI is generated. This includes the concept of what a data-set means for the given index.

3.2.3. Base-URI

An attribute of the index object which is crucial for generating referrals is the "Base-URI". The URI (or URI's) contained in this attribute form the basis of any referrals generated based on this index block. The URI is also used as input during the index aggregation process to constrain the possible types of aggregation. This use of the Base-URI is used to deal with meshes that support multiple protocols.

Thus, an index specification should define how the Base-URI applies to the underlying index and how it is changed during the aggregation process.

3.3 Aggregation

All index object specifications must address the issue of aggregation. This is the method by which an index server takes two or more indexes and combines them into one index to be passed on. It is not required that a given index-type aggregate. If it does not it must explicitly address the reasons why and what affect that has on scalability.

If a given index does aggregate, the algorithm for that aggregation must be given. It must also address how that algorithm affects mesh organization and scalability.

Index object document authors should remember that any kind of aggregation should be performed without compromising the ability to correctly route queries while avoiding excessive numbers of missed results. The acceptable likelihood of false negatives must be established on a per-application-domain basis, and is controlled by the granularity of the index and the aggregation rules defined for it by the particular specification.

Nothing in these documents specifically disallows aggregation rules that deal with different index object types. This type of heterogeneous mesh is difficult to formulate at best and thus is not covered by these documents. If document authors wish to attempt such a mesh they should be aware that it is considered an ill understood concept that contains many pitfalls for the mesh builder. In other words, "Here be dragons!".

3.4 Referral Generation Semantics

Since the method by which a client navigates the mesh is by referrals, the document must address how a given access protocol generates a referral from the index. Authors should pay particular attention to the case where an index is accessed by different protocols and the interaction between them. For example, an index that supports referrals being generated for both RWhois and LDAP must understand that one uses a Distinguished Name while the other doesn't. The impacts of these differences on the referral should be clear.

3.5 Matching Semantics

In order to generate a referral the decision of whether or not to do so must be handled by the access protocol. The semantics surrounding this decision have a large impact on the efficiency of searches as well as the requirements on aggregation. Thus, index specification authors must be very clear about how a match is determined.

3.6 Security Considerations

As is customary with Internet protocol documentation, a brief review of security implications of the proposed object must be included. This section may need to do little more than echo the considerations expressed in this document's Security Considerations section.

3.7 Optional Coverage

Because indexing algorithms, stop-lists, and data reduction technologies are considered by some index object designers to be proprietary, it is not necessary to discuss the process used to derive indexing information from a body of source material. When proprietary indexing technologies are used in a public mesh, all CIP

servers in the mesh should be able to parse the index object (and perform aggregation operations, if necessary), though not all of them need to be able to create these proprietary indices from source data.

Thus, index object designers may choose to remain silent on the algorithms used for the generation of indices, as long as they adequately document how to participate in a mesh of servers passing these proprietary indices.

Designers should also seriously consider including useful examples of source data, the generated index, and the expected results from example matches. When the aggregation algorithm is complex, it is recommended that a table showing two indices and the resultant aggregate index be included.

4. Security Considerations

Security considerations come into play in at least the following two scenarios. Indexing information can leak undesirable amounts of proprietary information, unless carefully controlled. At a more fundamental level, the CIP protocol itself requires external security services to operate in a safe manner. Both topics are covered below.

4.1 Secure Indexing

CIP is designed to index all kinds of data. Some of this data might be considered valuable, proprietary, or even highly sensitive by the data maintainer. Take, for example, a human resources database. Certain public bits of data, in moderation, can be very helpful for a company to make public. However, the database in its entirety is a very valuable asset, which the company must protect. Much experience has been gained in the directory service community over the years as to how best to walk this fine line between completely revealing the database and making useful pieces of it available.

Another example where security becomes a problem is for a data publisher who would like to participate in a CIP mesh. The data that publisher creates and manages is the prime asset of the company. There is a financial incentive to participate in a CIP mesh, since exporting indices of the data will make it more likely that people will search your database. (Making profit off of the search activity is left as an exercise to the entrepreneur.) Once again, the index must be designed carefully to protect the database while providing a useful synopsis of the data.

One of the basic premises of CIP is that data providers will be willing to provide indices of their data to peer indexing servers. Unless they are carefully constructed, these indices could constitute a threat to the security of the database. Thus, security of the data must be a prime consideration when developing a new index object type. The risk of reverse engineering a database based only on the

index exported from it must be kept to a level consistent with the value of the data and the need for fine-grained indexing.

Since CIP is encoded as MIME objects, MIME security solutions should be used whenever possible. Specifically when dealing with security between index servers.

4.2 Protocol Security

CIP protocol exchanges, taking the form of MIME messages, can be secured using any technology available for securing MIME objects. In particular, use of RFC-1847's Security Multiparts are recommended. A solid application of RFC-1847 using widely available encryption software is PGP/MIME, RFC-2016. Implementors are encouraged to support PGP/MIME, as it is the first viable application of the MIME Security Multiparts architecture. As other technologies become available, they may be incorporated into the CIP mesh.

If an incoming request does not have a valid signature, it must be considered anonymous for the purposes of access control. Servers may choose to allow certain requests from anonymous peers, especially when the request cannot cause permanent damage to the local server. In particular, answering anonymous poll requests encourages index builders to poll a server, making the server's resources better known.

The explicit security policy with respect to incoming requests is outside the scope of this specification. Implementors are free to accept or reject any request based on the security attributes of the incoming message. When a request is rejected due to authentication reasons, a response code from the 530 series must be issued.

Acknowledgments

Thanks to the many helpful members of the FIND working group for discussions leading to this specification.

Specific acknowledgment is given to Jeff Allen formerly of Bunyip Information Systems. His original version of these documents helped enormously in crystallizing the debate and consensus. Most of the actual text in this document was originally authored by Jeff.

Author's Address

Jeff R. Allen
Bunyip Information Systems, Inc.
310 Ste-Catherine West, Suite 300
Montreal, Quebec H2X 2A1
Canada

Phone: +1-514-875-8611
EMail: jeff@bunyip.com

Michael Mealling
Network Solutions, Inc.
505 Huntmar Park Drive
Herndon, VA 22070

Phone: (703) 742-0400
Email: michael.mealling@RWhois.net

References

[FRAMEWORK]
Allen, J. and M. Mealling, "The Architecture of the Common Indexing Protocol (CIP)", RFC XXX, IETF FIND WG, June 9, 1997.

[RFC2046]
Freed, N., and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, Innosoft, First Virtual Holdings, November 1996.

[RFC2048]
Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: MIME Registration Procedures", RFC 2048, Innosoft, MCI, ISI, November 1996.

[RFC2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.

[RFC821]
Postel, J., "SIMPLE MAIL TRANSFER PROTOCOL", RFC 821, ISI, August 1992.

Appendix A: Media Type Registration Templates

The following templates will be sent to the IANA for registration as soon as this document is accepted by the IESG.

Index tree

To: ietf-types@iana.org
Subject: Registration of MIME media type tree application/index

MIME media type name: application

MIME subtype name: index

Required parameters: none

Optional parameters: none

Encoding considerations: none

Security considerations:

Section 4 of this document should be copied here.

Interoperability considerations:

Published specification:
This draft will be referenced here once it has been approved by the IESG.

Applications which use this media type:
This media type is used to contain information about indices and how they inter-operate to form meshes of index servers.

Additional information:
This media type is not a standalone type. It is the top level of a tree similar to the vnd or prs trees specified in Section 2.1 of RFC2048. There are four specified branches to this tree:

application/index.cmd
application/index.response
application/index.obj
application/index.vnd

Each of these branches is a tree in its own right with types registered below them. See those registrations for more information on the types allowed below those branches.

Person & email address to contact for further information:

Intended usage: LIMITED USE

Author/Change controller:

Command tree

To: ietf-types@iana.org
Subject: Registration of MIME media type application/index.cmd

MIME media type name: application

MIME subtype name: index.cmd

Required parameters: none

Optional parameters: none

Encoding considerations: none

Security considerations:

Section 4 of this document should be copied here.

Interoperability considerations:

Implementors should handle unknown commands gracefully.

Published specification:

This internet draft should be referenced once it has been approved by the IESG.

Applications which use this media type:

This media type is the top of a tree of media types that express commands between hosts that exchange indices for the purpose of routing referrals.

Additional information:

This media type is not a standalone type. It is the top of a tree similar to the vnd and prs trees specified in Section 2.1 of RFC2048. Types registered within this tree are limited to being commands as specified in the document(s) referenced in the "Published specifications" section.

Person & email address to contact for further information:

Intended usage: LIMITED USE

Author/Change controller:

Response tree

To: ietf-types@iana.org
Subject: Registration of MIME media type application/index.response

MIME media type name: application

MIME subtype name: cip.response

Required parameters: code

Optional parameters: none

Encoding considerations: none

Security considerations:

Section 4 of this document should be copied here.

Interoperability considerations:

Implementors should handle unknown responses gracefully

Published specification:

This Internet draft should be referenced once it has been approved by the IESG.

Applications which use this media type:

This media type is used to encode responses to CIP commands passed between hosts that exchange indices for the purpose of routing referrals.

Additional information:

This media type _is_ a standalone type. The code parameter contains the specific response code as specified by Appendix B of the specification document.

Person & email address to contact for further information:

Intended usage: LIMITED USE

Author/Change controller:

Index Object tree

To: ietf-types@iana.org
Subject: Registration of MIME media type application/index.obj

MIME media type name: application

MIME subtype name: index.obj

Required parameters: type, dsi, base-uri

Optional parameters: none

Encoding considerations: none

Security considerations:

Section 4 of this document should be copied here.

Interoperability considerations:

Implementors should handle unknown index objects according to rules specified in the published specification.

Published specification:

This Internet draft should be referenced once it has been approved by the IESG.

Applications which use this media type:

This media type is the top of a tree of media types that express indexes that are exchanged between hosts that operate within a referral mesh.

Additional information:

This media type is not a standalone type. It is the top of a tree similar to the vnd and prs trees specified in Section 2.1 of RFC2048. Types registered within this tree are limited to being representations of indexes that contain some summary of the data found in some database and is used to generate referrals as specified in the above specified publication.

Person & email address to contact for further information:

Intended usage: LIMITED USE

Author/Change controller:

Vendor tree

To: ietf-types@iana.org
Subject: Registration of MIME media type application/index.vnd

MIME media type name: application

MIME subtype name: index.vnd

Required parameters: none

Optional parameters: none

Encoding considerations: none

Security considerations:

Section 4 of this document should be copied here.

Interoperability considerations:

Implementors should handle unknown objects gracefully.

Published specification:

This Internet draft should be referenced once it has been approved by the IESG.

Applications which use this media type:

This media type is the top of a tree of media types that express vendor specific extensions to the framework specified in the published specifications.

Additional information:

This media type is not a standalone type. It is the top of a tree similar to the vnd and prs trees specified in Section 2.1 of RFC2048. Types registered within this tree are limited to being vendor specific extensions to the CIP framework as specified in the publications. Any registrations within this tree are still limited to dealing with indexes, meshes and referrals.

Person & email address to contact for further information:

Intended usage: LIMITED USE

Appendix B: Response Codes

The meaning of the various digits in the response codes is discussed in RFC-821, Appendix E.

The following response codes are defined for use by CIPv3 servers. Implementors must use these exact codes; undefined codes should be interpreted by CIP servers as fatal protocol errors. Instead of defining new codes for unforeseen situations, implementors must adapt one of the given codes. The implementation should attach a useful alternative comment to the reused response code.

Code	Suggested description text Sender-CIP action
220	Initial server banner message
300	Requested CIP version accepted Continue with CIP transaction, in the specified version.
222	Connection closing (in response to sender-CIP close) Done with transaction.
200	MIME request received and processed Expect no output, continue session (or close)
201	MIME request received and processed, output follows Read a response, delimited by SMTP-style message delimiter.
400	Temporarily unable to process request Retry at a later time. May be used to indicate that the server does not currently have the resources available to accept an index.
500	Bad MIME message format Retry with correctly formatted MIME request.
501	Unknown or missing request in application/cip-request Retry with correct CIP command.
502	Request is missing required CIP attributes Retry with correct CIP attributes.
520	Aborting connection for some unexpected reason Retry and/or alert local administrator.
530	Request requires valid signature Sign the request, if possible, and retry.

Otherwise, report problem to the administrator.

531 Request has invalid signature
Report problem to the administrator.

532 Cannot check signature
Alert local administrator, who should cooperate with remote administrator to diagnose and resolve the problem. (Probably missing a public key.)

This Internet Draft expires 6 months from November 1997.