

**CS 580 Client-Server Programming
Spring Semester, 2004
Doc 18 Protocols, HTTP & XML-RPC
Contents**

Protocol..... 2
 Well defined..... 3
 Complete..... 4
 Parsable..... 5
 Available 6
 Protocol Types..... 7
 Protocol Design Issues..... 8
 URI..... 9
 HTML..... 10
 HTTP..... 12
 HTTP Message Format..... 13
 Client Request 14
 Full-Request..... 15
 Server Response..... 17
 Request Methods 25
 XML-RPC..... 28

References

Hypertext Transfer Protocol - HTTP/1.0, Berners-Lee, Fielding, Nielson, rfc1945,
<http://www.w3.org/Protocols/rfc1945/rfc1945>

Hypertext Transfer Protocol -- HTTP/1.1, Fielding, Gettys, Mogul, Masinter, Leach, Berners-Lee, rfc2616, <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

Uniform Resource Identifiers (URI): Generic Syntax, Berners-Lee, Fielding, Masinter, rfc2396
<http://www.ietf.org/rfc/rfc2396.txt>

XML-RPC Specification, <http://www.xmlrpc.com/spec>

Reading

HTTP/1.0 rfc1945, <http://ftp.ics.uci.edu/pub/ietf/http/rfc1945.html>

Post Office Protocol RFC 1939, <http://www.ietf.org/rfc/rfc1939.txt>

Copyright ©, All rights reserved. 2004 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

Protocol

Communication between client and server

Good protocols are hard to design

Requirements for a "good protocol":

- Well defined
- Complete
- Parsable
- Extendable
- Available protocol document

Well defined

Every bit of data sent in either direction has to have its place in the protocol description.

Protocol is a Language

Common formal description:

- BNF and Augmented BNF

Format of the description language needs to be part of the protocol document.

Examples are important

Complete

The protocol must cover **all** possible situations.

- Garbage data
- Old client or server (different protocol versions)
- Illegal requests
- Boundary conditions
- Etc.

Parsable

Both clients and servers are computer programs.

A computer program's IQ is generally 0.

Design goals:

- Distinct information packets or messages
 - Allow parsing independent of semantics
- Consistency
 - Allow for code reuse
- Flexibility
 - For example name-value pairs

Available

Different groups may write clients and servers at different times.

Central registry for Internet protocols

Self regulating:

- RFC - Request For Comment
- IETF - Internet Engineering Task Force

Official:

- ISO
- ANSI

Protocol Types

Two basic types

- Synchronous
- Asynchronous

Typical synchronous

- Client sends request to server
- Server responds with a reply

Examples

- HTTP, POP, SMTP, GOPHER, XMODEM

Typical asynchronous

Client and server both send information to each other concurrently.

Examples

- TELNET, RLOGIN, ZMODEM

A hybrid protocol is also possible

Protocol Design Issues

Protocol design is difficult!

Learn from examples

Some issues

- Protocol extensibility and versioning
- Byte order used for sending values
- ASCII vs. Binary protocol
 - Easy of debugging
 - Efficiency
- Synchronous vs. Asynchronous
 - Protocol overhead
 - Roundtrip delays
- State
 - Who is writing, who is reading?
- Timeouts
 - Timeouts vs. Synchronous protocols

URI

URI = Uniform Resource Identifiers

URL = Uniform Resource Locator

- `gopher://gopher.yoyodyne.com/`
- `news:rec.gardening`
- `http://www.yoyodyne.com/pub/foobar.html`
- `http://www.yoyodyne.com/pub/foobar.html?roger`

Common Internet Scheme Syntax

URL schemes that involve the direct use of an IP-based protocol to a specified host on the Internet use a common syntax for the scheme-specific data:

`//<user>:<password>@<host>:<port>/<url-path>`

HTML

Some Buzz Words

WWW

World Wide Web (or Web, for short)

SGML

Standard Generalized Markup Language
this is a standard for describing markup languages

DTD

Document Type Definition
this is a specific markup language, written using SGML

HTML

HyperText Markup Language
HTML is a SGML DTD.

HTML uses markup tags to tell the Web browser how to display the text

XML

Extensible Markup Language

XHTML

XML + HTML 4.0

What is HTML?

HTML is a language for describing structured documents

HTML does not describe page layout

Web browsers use HTML to render & display a document

HTML is content sent by HTTP

```
<HTML>
<HEAD>
<TITLE>Sample HTML Document</TITLE>
</HEAD>

<BODY>
This is a document
</BODY>
</HTML>
```

HTTP

- Stateless (http 1.0)
- Object-oriented protocol

The typing and negotiation of data representation, allows systems to be built independently of the data being transferred

Assigned port 80

Basic Server-Client Interaction (http 1.0)

Client: Open connection

Server: Accept/Reject connection

Client: Send request

Server: Send response to request

Connection closed

HTTP Message Format

HTTP-message = Simple-Request (HTTP/0.9 messages)
| Simple-Response
| Full-Request (HTTP/1.0 messages)
| Full-Response

Full-Request = Request-Line
*(General-Header | Request-Header | Entity-Header)
CRLF
[Entity-Body]

Full-Response = Status-Line
*(General-Header | Request-Header | Entity-Header)
CRLF
[Entity-Body]

HTTP-header = field-name ":" [field-value] CRLF

Entity-Body = *OCTET

Client Request

Request = Simple-Request | Full-Request
Simple-Request = "GET" SP Request-URI CRLF

Simple-Request Example

```
rohan 11-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '^]'.
GET /courses/fall00/cs580/index.html
<HTML>
<HEAD>
  <TITLE>CS 580: Course Web Site</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

<TABLE BORDER=0 WIDTH="100%">

...stuff removed...

</sub>Visitors since 21-Aug-00
</center>
</BODY>
</HTML>
Connection closed by foreign host.
```

Full-Request

Full-Request = Request-Line
 *(General-Header | Request-Header | Entity-Header)
 CRLF
 [Entity-Body]

Request-Line = Method SP URI SP HTTP-Version CRLF

```
rohan 13-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '^]'.
GET /courses/fall00/cs580/index.html HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Tue, 05 Sep 2000 19:31:14 GMT
Server: Apache/1.3.9 (Unix) PHP/3.0.12
Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT
ETag: "14c199-7e8-39b40e3c"
Accept-Ranges: bytes
Content-Length: 2024
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug
```

```
<HTML>
<HEAD>
  <TITLE>CS 580: Course Web Site</TITLE>
</HTML>
... stuff removed here...
Connection closed by foreign host.
```

Note 2 CRLF are needed to end the full request

HTTP 1.1 Example

rohan 14-> **telnet www.eli.sdsu.edu 80**

Trying 130.191.226.80...

Connected to www.eli.sdsu.edu.

Escape character is '^]'.
GET /courses/fall00/cs580/index.html HTTP/1.1

Connection: close

Host: www.eli.sdsu.edu

HTTP/1.1 200 OK

Date: Tue, 05 Sep 2000 22:41:26 GMT

Server: Apache/1.3.9 (Unix) PHP/3.0.12

Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT

ETag: "14c199-7e8-39b40e3c"

Accept-Ranges: bytes

Content-Length: 2024

Connection: close

Content-Type: text/html

X-Pad: avoid browser bug

<HTML>

<HEAD>

 <TITLE>CS 580: Course Web Site</TITLE>

</HEAD>

...stuff removed here...

</BODY>

</HTML>

Connection closed by foreign host.

Server Response

Example Full-response

HTTP/1.0 200 Document follows

MIME-Version: 1.0

Server: CERN/3.0

Date: Thursday, 21-Mar-96 17:00:45 GMT

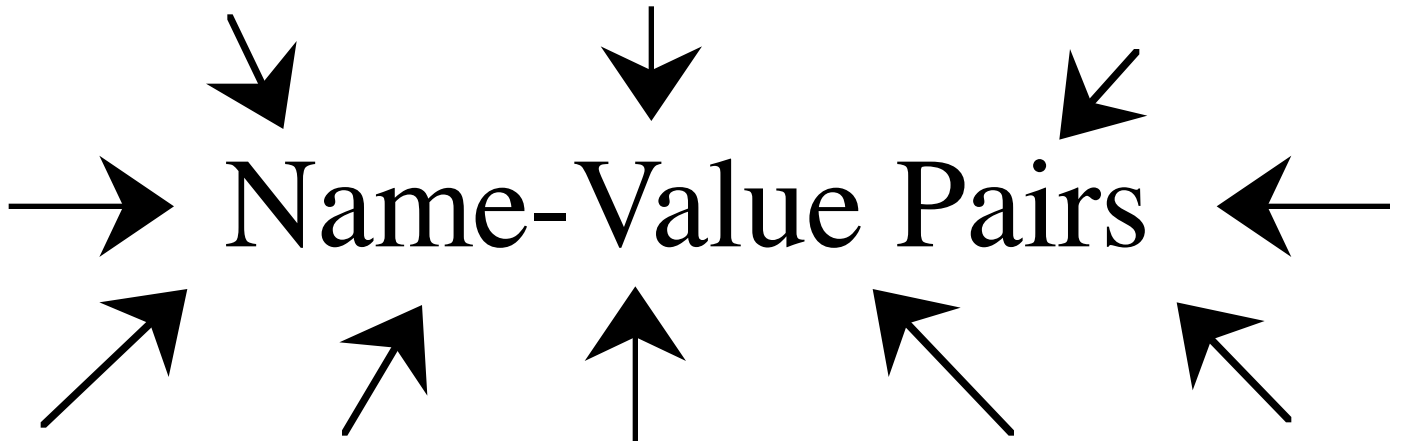
Content-Type: text/html

Content-Length: 2686

Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

field-name	field-value
MIME-Version:	1.0
Server:	CERN/3.0
Date:	Thursday, 21-Mar-96 17:00:45 GMT
Content-Type:	text/html
Content-Length:	2686
Last-Modified:	Tuesday, 27-Feb-96 05:34:12 GMT

What is the big Deal?



What are the data fields in this?

1.0; CERN/3.0; Thursday, 21-Mar-96 17:00:45 GMT;
text/html; 2686; Tuesday, 27-Feb-96 05:34:12 GMT

What are the data fields in this?

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Which is Safer?

Which is Easier to Parse?

Name -Value Pairs are Good

Does Order Matter?

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Server: CERN/3.0
Content-Type: text/html
MIME-Version: 1.0
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT
Date: Thursday, 21-Mar-96 17:00:45 GMT

Extending Protocols

MIME-Version: 1.0

Server: CERN/3.0

Date: Thursday, 21-Mar-96 17:00:45 GMT

Content-Type: text/html

Forwarded: by <http://rohan.sdsu.edu/> for cs.sdsu.edu

Content-Length: 2686

WhitneyInfo: Hi Mom

Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Cookies were added to HTTP by adding a new name-value pair

Clients/servers that were not programmed for cookies ignored the new name-value pair

Name -Value Pairs are Everywhere

Data Files

Which is easier for a program to parse?

Which is safer

```

Allen, Sally      87 92 85 55 74 10
Battista, Joe    92 98 98 55 78 10
Biag, Sam        83 91 78 51 72 8
Chen, Pete       89 92 89 57 79 10
Chen, Roger      74 68 59 61 55 10
    
```

name	course	hwork	exam1	exam2	final	as1
Allen, Sally	87	92	85	55	74	10
Battista, Joe	92	98	98	55	78	10
Biag, Sam	83	91	78	51	72	8
Chen, Pete	89	92	89	57	79	10
Chen, Roger	74	68	59	61	55	10

lastName:Allen, lastName:Sally, course:87, hwork:92, exam1:85, exam2:55, final:74, as1:10

lastName:Battista, lastName:Joe, course:92, hwork:98, exam1:98, exam2:55, final:78, as1:10

lastName:Baig, lastName:Sam, course:83, hwork:91, exam1:78, exam2:51, final:72, as1:8

lastName:Chen, lastName:Pete, course:89, hwork:92, exam1:89, exam2:57, final:79, as1:10

lastName:Chen, lastName:Roger, course:74, hwork:68, exam1:59, exam2:61, final:55, as1:10

Name-Value Pairs and Parameters

Most languages use positional matching for parameters

```
“The cat in the hat came back”.substring( 2, 6);
```

Smalltalk uses name-value pairs

```
‘The cat in the hat came back’ copyFrom: 2 to: 6
```

copyFrom:to: is on method

Keyword (name)	Parameter (value)
copyFrom:	2
to:	6

Name -Value Pairs are Your Friends

Don't program without them

Big Issue: In-line data

If we send binary data or data of unknown format how does receiver know when the data ends?

POP solution

Use termination sequence

Insure that termination sequence does not occur in data

HTTP Solution

Full-Response = Status-Line

*General-Header

*Response-Header

*Entity-Header

CRLF

[Entity-Body]

Send length of data to be sent in header

Request Methods

Method = "GET" | "HEAD" | "PUT" | "POST"
| "DELETE" | "LINK" | "UNLINK"
| extension-method

All HTTP/1.0 servers must support GET and HEAD

Servers should return the Status-Code

"501 Not Implemented"

if the method is unknown.

GET

Retrieves whatever item is identified by the URI.

The URI can refer to a data-producing process, or a script

The produced data which shall be returned as the Entity-Body

HEAD

Identical to GET except that the server must not return any Entity-Body in the response

POST

Request that the origin server accept the item enclosed in the request as a new subordinate of the resource identified by the URI

Allows a uniform function to:

- Annotation of existing documents;
- Posting a message to a bulletin board topic, newsgroup, mailing list, or similar group of articles;
- Providing a block of data (usually a form) to a data-handling process, or a script, which can be run by such a process;
- Extending a document during authorship

These are not always supported

Why?

PUT

The enclosed item in the request is to be stored under the supplied URI

DELETE

Requests that the server delete the resource identified by the given URI

LINK

Establishes one or more Link relationships between the existing resource identified by the URI and other existing resources

UNLINK

UNLINK method removes one or more Link relationships from the existing resource identified by the URI

XML-RPC Requests

Send via HTTP post

User-Agent & Host must be specified

Content-Type is text/xml

Content-Length must be specified and be correct in bytes

Content is in XML and contains single <methodCall>

Example

```
POST /cs580 HTTP/1.1
Host: rugby.sdsu.edu:8008
Content-length: 158
Content-type: text/xml;charset=iso-8859-1
User-Agent: Smalltalk XMLRPC version 0.5 (VisualWorksÆ
NonCommercial, Release 7 of June 14, 2002)
Connection: keep-alive
```

```
<?xml
version="1.0"?><methodCall><methodName>officeDataFor</metho
dName><params>
  <param><value><string>Whitney</string></value></param>
</params></methodCall>
```

XML in Requests

```
<methodCall>
  <methodName>[Character]*</methodName>
  <params>
    <param><value>[Legal type]</value></param>*
  </params>
</methodCall>
```

Characters – string of characters

Legal types

```
<i4> or <int>
<boolean>
<string>
<double>
<dateTime.iso86021>
<base64>
```

```
<struct>
  <member>
    <name>[Character]*</name>
    <value>[Legal type]</value>
  </member>*
</struct>
```

```
<array>
  <data>
    <value>[Legal type]</value>
  </data>*
</array>
```

Response

Unless lower-level error always return 200 OK

Content type is text/xml

Content-length must be present and be correct

Body contains on XML structure <methodResponse>

Example

HTTP/1.0 200 OK

Date: Tue, 8 Oct 2002 23:27:43 -0700

Server: VisualWave TinyHTTP/1.0

Mime-version: 1.0

Content-type: text/xml;

Content-length: 252

```
<?xml
version="1.0"?><methodResponse><params><param><value><arra
y><data>
  <value><string>Whitney</string></value>
  <value><string>P-243</string></value>
  <value><string>594-3535</string></value>
</data></array></value></param></params></methodResponse>
```

Response XML

Response contains one `<methodResponse>` tag

The `<methodResponse>` tag can contain either

- `<params>`
- `<fault>`

`<params>` tag can only contain one `<param>`

methodResponse Structure

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>[Legal type]</value>
    </param>
  </params>
</methodResponse>
```

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>[Legal type]</value>
  </fault>
</methodResponse>
```