1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

**A Design Standards
Document Server**

by

Frederick M. Williams, P.Eng.

B.Eng, Carleton University, Ottawa

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the
requirements for the degree of

Master of Engineering

The Master of Engineering in the Civil and Environmental Engineering Program
is a joint program with the University of Ottawa,
administered by the Ottawa-Carleton Institute for Civil Engineering

Carleton University
Ottawa, Ontario, Canada

December 8, 1993

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-92975-8

Canada

Name FRED WILLIAMS

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

0543 U.M.I.
SUBJECT CODE

SUBJECT TERM

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
General 0289
Ancient 0290
Linguistics 0291
Modern 0401
Literature 0294
General 0295
Classical 0297
Comparative 0298
Medieval 0316
Modern 0591
African 0305
American 0352
Asian 0355
Canadian (English) 0593
Canadian (French) 0311
English 0312
Germanic 0315
Latin American 0313
Middle Eastern 0314
Romance 0314
Slavic and East European

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
General 0321
Biblical Studies 0319
Clergy 0320
History of 0322
Philosophy of 0469
Theology

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical 0310
Business Administration 0272
General 0770
Accounting 0454
Banking 0338
Management 0385
Marketing 0501
Canadian Studies 0503
Economics 0505
General 0508
Agricultural 0509
Commerce-Business 0510
Finance 0511
History 0358
Labor 0366
Theory 0351
Folklore 0578
Geography 0351
Gerontology 0578
History 0578
General

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science 0615
General 0616
International Law and Relations 0617
Public Administration 0814
Recreation 0452
Social Work 0626
Sociology 0627
General 0938
Criminology and Penology 0631
Demography 0628
Ethnic and Racial Studies 0629
Individual and Family Studies 0630
Industrial and Labor Relations 0700
Public and Social Welfare 0344
Social Structure and Development 0709
Theory and Methods 0999
Transportation 0453
Urban and Regional Planning 0453
Women's Studies

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy 0475
Animal Culture and Nutrition 0476
Animal Pathology 0359
Food Science and Technology 0478
Forestry and Wildlife 0479
Plant Culture 0480
Plant Pathology 0817
Plant Physiology 0777
Range Management 0746
Wood Technology

Biology

General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0579
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0433
Radiation 0821
Veterinary Science 0778
Zoology 0472
Biophysics 0786
General 0760
Medical

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
General 0300
Audiology 0992
Chemotherapy 0567
Dentistry 0350
Education 0769
Hospital Management 0758
Human Development 0982
Immunology 0564
Medicine and Surgery 0347
Mental Health 0569
Nursing 0570
Nutrition 0380
Obstetrics and Gynecology 0354
Occupational Health and Therapy 0381
Ophthalmology 0571
Pathology 0419
Pharmacology 0572
Pharmacy 0382
Physical Therapy 0573
Public Health 0574
Radiology 0575
Recreation

Speech Pathology 0460
Taxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry 0485
General 0749
Agricultural 0486
Analytical 0487
Biochemistry 0488
Inorganic 0738
Nuclear 0490
Organic 0491
Pharmaceutical 0494
Physical 0495
Polymer 0754
Radiation 0405
Mathematics 0605
Physics 0986
General 0606
Acoustics 0608
Astronomy and Astrophysics 0748
Atmospheric Science 0607
Atomic 0798
Electronics and Electricity 0759
Elementary Particles and High Energy 0609
Fluid and Plasma 0610
Molecular 0752
Nuclear 0756
Optics 0611
Radiation 0463
Solid State 0346
Statistics 0984
Applied Sciences 0984
Applied Mechanics 0984
Computer Science

Engineering 0537
General 0538
Aerospace 0539
Agricultural 0540
Automotive 0541
Biomedical 0542
Chemical 0543
Civil 0544
Electronics and Electrical 0545
Heat and Thermodynamics 0546
Hydraulic 0547
Industrial 0794
Marine 0548
Materials Science 0549
Mechanical 0551
Metallurgy 0552
Mining 0549
Nuclear 0765
Packaging 0554
Petroleum 0790
Sanitary and Municipal 0428
System Science 0796
Geotechnology 0795
Operations Research 0994
Plastics Technology 0994
Textile Technology

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Department of Civil and Environmental Engineering

The Undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of this thesis

**A Design Standards
Document Server**

submitted by

Frederick M. Williams P.Eng.

in partial fulfillment of the requirements
of the degree of Master of Engineering

A. G. Razaqpur.

Chair, Department of Civil and Environmental Engineering

N. M. Holtz, Thesis Supervisor

M. Baltacioglu, External Examiner

Carleton University

Ottawa, Ontario, Canada

November 1, 1993

Abstract

Design standards are the documents that control and guide the Engineer in any design. Standards are hierarchical in nature and consist of short sections, each of which outline the design requirements for a particular aspect of the overall project.

This thesis presents the design of a Standards' Document Server that will provide client applications with displayable information, formulae and table lookups of design standards. This will allow the client applications to adapt to the latest changes to the standard by being able to access the most current information using a well defined interface to the server and thus to the standard.

The Standards' Document Server will provide the client with: (a) Textual sections of the specification in a displayable form; (b) Mathematical formulae in computable and displayable form; (c) Figures in a displayable form; (d) Conditions of applicability in computable and displayable form; (e) Classifier (keywords which are used to group clauses which pertain to the keyword) searching to locate pertinent clauses; (f) Mathematical symbol searching to locate pertinent clauses and definitions.

Key Words

Client-Server Interface; Standards; Standards Information; Standards Knowledge; Standards Representation; Document Markup; Document Presentation; Display Exchange; Information Interchange; Formulation Exchange; Table Information Exchange.

Acknowledgements

I would like to take this opportunity to express my thanks to my thesis advisor, Neal Holtz, for the time, assistance and advice, which he has given to me so generously. Coming from an MS-DOS background I am particularly grateful to Neal for introducing me to the area of Unix and WorldWideWeb. I thank him sincerely.

I would be remiss if I did not mention the assistance and encouragement which I received from Tim Berners-Lee (timbl@info.cern.ch) and the WorldWideWeb group. Their expertise in the areas of hypertext and network protocols and their patience with my seemingly endless questions was greatly appreciated.

I sincerely thank my family for their understanding and support as I have worked on this thesis. For everything from the endless cups of coffee to the encouragement which they gave me to carry on, I am grateful.

Finally, I would like to make special mention of my mother, Doris Williams, who always instilled the value of education in me. It is to her that I dedicate this thesis.

Contents

Abstract	iii
Acknowledgements	iv
Contents	v
List of Figures	ix
1 Introduction	1-1
1.1 Objectives	1-1
2 Electronic Standards	2-1
2.1 Introduction	2-1
2.2 Representation and Use of Design Specifications	2-1
2.2.1 Decision Tables	2-2
2.2.2 Electronic NBC	2-3
2.2.3 AISC LRFD	2-4
3 Network Computing	3-1
3.1 Network Communications Protocols	3-1
3.1.1 TCP/IP UDP/IP Protocols	3-1
3.2 Client-Server Architecture	3-5
3.3 Client Server Implementations	3-6
3.3.1 File Transfer Protocol (FTP)	3-6
3.3.2 Hypertext Transfer Protocol Version 0.x (HTTP)	3-7
3.3.3 Hypertext Transfer Protocol Version 1.0 (HTTP)	3-8

3.3.4	Remote Terminal Protocol (TELNET)	3-13
3.3.5	Gopher	3-13
4	Document Markup and Presentation	4-1
4.1	Document Markup	4-1
4.1.1	Rich Text Format (RTF)	4-1
4.1.2	Standard Generalized Markup Language (SGML)	4-2
4.1.3	HyperText Markup Language (HTML)	4-3
4.1.4	HyperText Markup Language Plus (HTML+)	4-5
4.1.5	Text EXecutive (TeX)	4-6
4.1.6	Multipurpose Internet Mail Extensions (MIME)	4-8
4.2	Document Editors	4-10
4.2.1	Word and Word for Windows	4-10
4.2.2	WordPerfect and WordPerfect for Windows	4-11
4.2.3	EMACS	4-11
4.3	Document Locators	4-12
4.3.1	Archie	4-12
4.3.2	Wide Area Information Systems (WAIS)	4-13
4.3.3	Uniform Resource Locator (URL)	4-14
4.3.4	Uniform Resource Name (URN)	4-15
4.4	Document Viewers	4-16
4.4.1	Text Based Viewers	4-16
4.4.2	Full Screen Text Based Viewers	4-16
4.4.3	Graphical User Interface Viewers	4-17
5	Design Standards	5-1
5.1	Introduction	5-1
5.2	Structure of Printed Design Standards	5-1
5.2.1	General Structure	5-2
5.2.2	Text	5-3
5.2.3	Tables	5-3

5.2.4	Figures	5-4
5.2.5	Equations	5-5
5.2.6	Definitions	5-5
5.3	Proposed Functionality	5-6
5.3.1	General	5-6
5.3.2	Document Viewers	5-6
5.3.3	Computing Applications	5-7
6	Standards' Server Architecture	6-1
6.1	Introduction	6-1
6.2	Protocol Overview	6-1
6.3	Standards' Document Server Protocol	6-2
6.3.1	Terms Used	6-2
6.3.2	Requests	6-3
6.3.3	Responses	6-9
6.4	Standard's Markup Overview	6-13
6.4.1	Standard's Markup	6-14
7	Client Applications	7-1
7.1	Introduction	7-1
7.2	Reader/Browser	7-1
7.2.1	Hypertext Links	7-2
7.2.2	Table of Contents	7-4
7.2.3	Classifier Searches	7-6
7.2.4	Glossary Search	7-9
7.2.5	Printing	7-10
7.3	Enhanced Reader/Browser	7-10
7.3.1	Equation Solver	7-10
7.4	Design Notes	7-12
7.4.1	Design Notes	7-12
7.5	Design Programs	7-13

8	Implementation Considerations	8-1
8.1	Introduction	8-1
8.2	Server Considerations	8-1
8.2.1	Hypertext Transfer Protocol (HTTP)	8-1
8.2.2	Data Files and Structures	8-2
8.2.3	Responses	8-3
8.2.4	Tolerance to Ill Formed Requests	8-9
8.3	Client Considerations	8-10
8.3.1	Hypertext Markup Language (HTML+)	8-10
8.3.2	Viewers	8-11
8.3.3	Applications	8-12
8.3.4	Tolerance to Ill Formed Responses	8-12
9	Conclusions	9-1
A	Sample HTML+	A-1

List of Figures

3.1	The structure of Internet services	3-2
3.2	Client Server UDP services	3-4
3.3	Example of Client-Server Architecture	3-5
3.4	FTP Client-Server connections	3-7
5.1	Examples of Standard's Table Entries	5-4
7.1	Top Level Table of Contents Display	7-5
7.2	Exploded Table of Contents Display	7-7

1

Introduction

Computers today have become an integral part of the workplace. They provide a efficient method for an engineer to do drafting, designing, modelling, document presentation as well as assisting in the business aspects such as accounting. Engineers also depend upon design standards. With a few exceptions, design standards are not computerized and are used only in their printed form.

Design standards are the documents that control and guide the Engineer in any design. Standards are hierarchical in nature and consist of short sections, each of which outline the design requirements for a particular aspect of the overall project. For any project an engineer must reference, cross reference and re-reference the standards. When a standard is in printed form, this does involve a great deal of "page flipping" to insure that all the applicable sections are reviewed and this can easily result in errors and omissions.

This thesis presents a computerized method of allowing engineers, and engineering applications, to access the information contained within a design standard. It provides for the accessing of standards in methods that are intuitive and presents the information in forms that are familiar.

1.1 Objectives

This thesis presents the design of a Standards' Server Engine that will provide client applications with displayable information, formulae and table lookups of design standards. This will allow the client applications to adapt to the latest changes to the standard by being able to access the most current information using a well defined interface to the server and thus to the standard.

The Standards' Document Server will provide the client with:

- Textual sections of the specification in a displayable form
- Mathematical formulae in computable and displayable form
- Figures in a displayable form
- Conditions of applicability in computable and displayable form
- Classifier (keywords which are used to group clauses which pertain to the keyword) searching to locate pertinent clauses
- Mathematical symbol searching to locate pertinent clauses and definitions

The Standards' Document Server merges three technologies to provide convenient computer based interactive access to standards information. These three technologies are *standards representation*, *network based computing utilizing the client-server model* and *markup and presentation of computerized documents*.

The design is based on the client-server model. The client-server model allows for multiple clients (applications) to access information using a standardized interface without requiring explicit knowledge of the location or original format of the information. By using this model the application author can concentrate on the functionality of the application with the data interface being less of a concern. The model also allows for the updating and modification of the information without the concern of causing problems to dependent applications.

While it is intended that the code be general purpose, most examples and specific implementations will be drawn from the field of structural engineering.

2

Electronic Standards

ABSTRACT *This chapter reviews the major standards representation technologies including the National Building Code and AISC LRFD. The underlying representation methodology utilized is also reviewed.*

See also chapter 3, Network Computing, and chapter 4, Document Markup and Presentation, for reviews of the other technical components.

2.1 Introduction

The representation of standards' information electronically has been ongoing since the mid 1960's. Early work focused on the representation of the quantitative portions of a standard in a manner that could be evaluated by computer without regard to the textual or diagrammatic information contained within it. With the advent of powerful personal computers and graphical user interfaces (GUI) work is now being done in representing standards as "electronic books" using hypertext technology. The "electronic book" however is more concerned with the visual representation of the information than with its useability by an application.

2.2 Representation and Use of Design Specifications

The following sections are a review of major representation technologies in use today for standards information. While this is by no means a complete list of all of the current applications or techniques, these represent examples that are well known and in use today.

2.2.1 Decision Tables

In 1966 S. J. Fenves[16] first put forward the idea of representing standards utilizing decision tables. This work was done in an attempt to circumvent the problem of "hard coding" the rules and provisions of a standard within design software. The decision table approach provides an organizational model which represents the rules and provisions of a standard and also allows for the processing of these. Fenves, along with J. R. Harris[14], R. N. Wright[19] and others, have refined and developed this early work into the decision table standards' representation that is in use today.

Provision and standard have been defined by Harris[14] as:

Provision - a measure that will enable some entity to attain a satisfactory performance with respect to a particular limit state.

Standard - a collection of provisions that are intended to ensure or promote the general objective of the standard.

Each provision within a standard outlines rules which are applied to a single data element implied by the provision. There are two data element types:

Basic Data Element

- the standard contains no rules for evaluation and are supplied by the user of the standard. They do not receive any information but are data suppliers to derived data elements.

Derived Data Element

- the standard provides the rules for evaluation. Information for these elements are from either basic data elements or other derived data elements.

From these data elements, a concise table of logical conditions can be constructed for each given situation, evaluated, and based on the results, appropriate actions can be taken.

This decision table representation of standards has been used in a number of computer based design applications such as Intergraph MicasPlus[25] and CU-STAND[25], developed

at Cornell University. The use of decision tables to represent standards' information, functions well when the client application does not require a great deal of human intervention. The representation does however requires a great deal of manipulation of the original material, by knowledgeable authors, before it can be used.

2.2.2 Electronic NBC

The National Building Code of Canada (NBC) is a hierarchically organized document that lends itself well to computerization utilizing hypertext techniques. The idea of producing a hypertext NBC, known as HyperCode, was first published by S. M. Cornick[10] in 1991. Subsequent development, both by the National Research Council of Canada and private firms, has led to initial versions of "HyperCode" being available.

Cornick categorized the NBC's hierarchy as follows:[10]

Numbering	Category
3	Part
3.5	Section
3.5.1	Subsection
3.5.1.6	Article
3.5.1.6.(1)	Sentence
3.5.1.6.(1).(e)	Clause
3.5.1.6.(1).(e).(i)	Subclause

Within the hypertext document each level forms a node that may be linked with any other node within the document. Definitions, or glossary entries, also form nodes and are linked to the item that they define. The logical linking of a node to others is based on the information contained within the node such as references to other parts of the NBC or definitions. A hypertext viewer can then be utilized to view and "chase the links".

Since accessing the document only by these links can easily result in missing applicable information, queries can be made of the nodes' information and a list of applicable nodes presented to the viewer using a query language such as SQL (Standard Query Language). Use of queries based directly on the information contained with the node does require the user to have intimate knowledge of the terms and phrases used within the NBC. Work is

now underway[31] to classify, using generalized keywords or "classifiers", the information within each node and to allow the user to search the NBC by classifier.

Work is also being done in the area of multimedia and its application to the NBC.

The electronic NBC allows the designer to obtain the required information but only in visual form. Current implementations do not allow for the representation of mathematical formulae or tables in forms that would be useable by other applications. The electronic NBC also requires the user to acquire periodic updates to the information to remain current as with the printed NBC.

2.2.3 AISC LRFD

The American Institute of Steel Construction's (AISC) Load and Resistance Factor Design (LRFD) specification is a document describing the required calculations and checks in American steel design.

A number of software applications such as ELRFD[1] (Electronic Load and Resistance Factor Design) have represented the quantitative provisions of the LRFD in a decision table (see section 2.2.1) representation. This representation allows for the accurate implementation of steel design requirements within design software.

ELRFD, written in the Microsoft Windows environment by Visual Edge Software, provides multiple separate applications for use. The first is a checker/browser. It allows the user to describe the design problem, using menu selections and lists, and to input the required information such as load, section and dimensions. The checker/browser then reviews the decision tables and applies the appropriate ones to the problem to verify the design. The software is designed such that multiple iterations of user input/design verification of the same problem is easily done.

Other applications are available for the modification of decision tables, equations, and design objects which can be used for both customizing an individual installation and the maintenance of the software to insure compliance with the latest LRFD standard.

ELRFD excels at presenting and utilizing the "quantitative provisions" of a standard. It does however only assist an engineer with the "black or white" or quantitative decisions and but does not provide the engineer with the information required to allow an interpretation

of the standards.

3

Network Computing

ABSTRACT *This chapter reviews the basic underlying protocols of Internet networking including TCP/IP and UDP as well as existing Client-Server protocols such as FTP and Gopher. Client-Server protocols for transferring hypertext information such as HTTP is also reviewed.*

See also chapter 2, Electronic Standards, and chapter 4, Document Markup and Presentation, for reviews of the other technical components.

3.1 Network Communications Protocols

The following sections review the underlying communications protocols which are used by the Internet. The number of protocols reviewed is by no means the complete list however, it does represent the most popular methods of network communications.

3.1.1 TCP/IP UDP/IP Protocols

The name TCP/IP[7] results from the two main standards of DARPA (Defence Advanced Projects Research Agency) technology which describes, in detail, the way computers communicate and conventions for interconnecting networks and routing of network traffic. The first standard TCP (Transmission Control Protocol see section 3.1.1) is the protocol that provides reliable, end to end, stream transport while IP (Internet Protocol see section 3.1.1) provides the universal protocol which defines the unit of transfer to be the IP datagram and provides the addressing scheme for hosts and gateways.

The Internet service model uses the TCP/IP protocol to provide it's machine and network independent functions. The figure 3.1 outlines the relationship between applications and the lower level services.

UDP (User Datagram Protocol see section 3.1.1) provides only packet routing unlike

Figure 3.1: The structure of Internet services

the TCP stream delivery. It is used where information exchanged between machines can be delivered within the size of one packet and the loss, or out of order delivery, of the information can be tolerated.

Both TCP and UDP utilize the Internet Protocol (IP) to provide the low level delivery of packets.

Internet Protocol (IP)

IP provides a service that consists of unreliable, best-effort, connectionless packet delivery. Packets may be lost, duplicated or arrive out of order and IP is not able to detect these conditions or report it to the sender or receiver. This is the unreliable portion of the service. The service is considered connectionless because each packet is treated by IP as a single unrelated transaction. Internet discards packets only when all resources or the underlying network fails and therefore the service is called best-effort.

The Internet Protocol defines the use of the Internet datagram as the basic unit of transmission. The datagram is made up of a header, which includes both the source and destination addresses, and the data intended to be transmitted. This is analogous to the physical network which uses the same format for the transmission frame. The difference is that the datagram contains Internet addressing for source and destination whereas the network frame contains network specific addressing.

Transmission Control Protocol (TCP)

TCP is the protocol that defines the format of data and acknowledgements that two computers can use to exchange information to achieve a reliable data transfer. It also defines procedures that can be used to ensure that the data arrives correctly. The specification includes, but is not limited to, the distinguishing of multiple destinations on a singular machine, recovery from errors such as lost or duplicated packets, and how two systems start and complete stream transfer.

TCP makes few assumptions concerning the underlying communications system. It can use dialup telephone lines, local area networks (LANs), fibre optic networks, or low speed long distance networks (WANs). This facility is possible by substituting the appropriate software which supports the required network on the transportation media. It is the ability of supporting multiple transportation media, transparently from applications, that makes the TCP protocol well suited for INTERNET.

The TCP specification defines the protocol and procedures to use for implementing a TCP service. It does not, however, define a detailed application interface. This allows a designer to utilize the native operating system procedures to implement a TCP service. It also allows for the TCP protocol to be implemented on many dissimilar hardware/software platforms.

The interface between applications and TCP is done through the concept of 'protocol ports'. A port (known as a 'socket' in some implementations) is an numeric identifier, through which communications, both send and receive, are done by the application. It was originally intended that a port identifier appear to the application the same way as a file or device identifier would appear to the application. Due to the complexities of network communications, however, this analogy is not transparent to the application.

Protocol port numbers are assigned by agreement or general usage and are published so that application authors can utilize them for particular functionality. For example protocol port 69 is used for TFTP (Trivial File Transfer) while port 5 is used for RJE (Remote Job Entry). Since both source and destination port numbers are 16 bits in length, and by convention must be a positive number, this protocol allows for 32766 possible port numbers.

The use of the protocol port concept allows an application to transmit and receive

Figure 3.2: Client-Server UDP services

information without knowledge of the underlying transmission system and the particularities of the application at the other end. It also allows for multiple sessions to be active at the same time using the same application.

User Datagram Protocol (UDP)

Internet provides UDP to allow the client application to directly address the information sent to a specific application or process on the server and for the server to return information directly to the requesting client application. While IP provides machine to machine addressing, UDP adds to IP the concept of source and destination protocol port addresses. It also provides an optional level of error checking by generating and sending a checksum for each packet of information.

The sending application must know both the Internet address of the intended server as well as the protocol port number of the intended receiver. The concept of port numbers is the same as with TCP.

The protocol allows for multiple send and receive applications to 'multiplex' their sessions over a single communications link. Both client and server implementations of UDP "demultiplex" the incoming information and queue the appropriate packets to their associated protocol port. UDP also collects outstanding packets from the protocol ports and sends them. See Figure 3.2.

Since UDP uses IP directly it is also an unreliable, best-effort, connectionless packet delivery service. Each exchange of information is treated as a single transaction with no

Figure 3.3: Example of Client-Server Architecture

acknowledgement of receipt. Data may be lost or delivered out of order. It is therefore up to both the client and server applications to provide a mechanism to ensure complete data integrity.

3.2 Client-Server Architecture

The term "Client-Server" is used to describe the interaction between applications for the exchange of information in a peer to peer environment. This exchange can occur on the same machine or different machines which are connected by some method. The method of providing a connection could be as simple as a RS232 cable between two machines or using Internet to provide the connection.

The "Client" application is the one that initiates the request for information and the "Server" is the application that responds with the information requested. A simple analogy for client-server model (Figure 3.3) is Directory Assistance provided by the telephone company. When you phone Directory Assistance requesting a phone number you are the "Client" and the "Server" is the operator who responds with the proper phone number.

Peer to peer implies that the applications that are communicating to each other have equal authority to initiate requests. Unlike a poll/response environment where one application is the "Master" and the other is the "Slave" who must wait for the master to prompt before responding; each application within a peer to peer environment may send information or requests without requiring a prompt from the other application.

3.3 Client-Server Implementations

The following sections review a number of current popular Client-Server implementations. The number of protocols reviewed is by no means the complete list however, it does represent a cross section of the styles available.

3.3.1 File Transfer Protocol (FTP)

One of the services provided by the Internet protocol, using TCP/IP, is the file transfer protocol used to transfer complete files from one machine to another. This protocol is designed for use by programs however, it does provide a direct human interface for it's use. The implementation of this human interface is normally known as FTP as well. The protocol provides for remote login, a few simple file commands (such as directory listings, ability to change directories, etc.) and the ability to transfer files. It also allows for the transfer of information between two machines that has been requested by a third machine. The protocol also provides for simple data conversion such as EBCDIC to ASCII during the transfer. The protocol does provide for multiple connections, to different machines or the same machine, during a single 'session' of FTP.

The FTP protocol is different than a normal send/receive protocol in the fact that it requires the establishment of both a data channel and a control channel between the client and the server. The control channel is used for the login information, requests for file transfer, and error reporting during the transfer. The data channel is used only for the exchange of the file data between the two machines. This is done by maintaining two port connections to the client machine, one for data and one for control. See Figure 3.4.

This protocol uses the TELNET (see section 3.3.4) protocol for it's control information while data is exchanged based on a style agreed on between the server and client.

Most implementations of FTP allow the concept of anonymous login. This allows someone, who is not a registered user on a particular system, to use FTP to access and upload files from that particular system. The anonymous login however, generally restricts the information that is viewed or accessed, as defined by the server's system administrator, as well as the FTP commands that can be used.

Figure 3.4: FTP Client-Server connections

3.3.2 Hypertext Transfer Protocol Version 0.x (HTTP)

Version 0.x of the hypertext transfer protocol is currently used by WorldWideWeb as the protocol to transfer hypertext documents. HTTP uses the TELNET (see section 3.3.4) protocol on top of TCP/IP to effect the transfer.

The protocol uses a simple ASCII transfer of information. Documents are requested by issuing a GET followed by the document descriptor and terminated with a CrLf pair. The server returns the document as an ASCII stream. The document is complete when the server closes the connection. Each request and response is treated as a complete transaction.

HTTP allows for the following requests to be made by the client to the server:

GET *DocumentName*

retrieves the requested document. Optionally, if the retrieved document is of an <ISINDEX> type, it can be searched utilizing the construct [*?search-term+search-term ...*] following the *document-name*

SEARCH *Keywords*

searches the index document for keywords specified and returns the results. This request is not currently utilized (see GET above for searching)

SINCE *Date Time*

modifies SEARCH request to limit files inspected to after the specified *Date Time*

BEFORE *Date Time*

modifies SEARCH request to limit files inspected to before the specified *Date Time*

ACCEPT *Format [Penalty]*

informs the server of acceptable document types that can be utilized by the client. The *Penalty* parameter is a numeric measure of the display ability and processing time required by the client for the format specified. The server will, within its capabilities, return the document in a format which involves the least processing penalty to both the client and the server.

PORT

modifies the port number used by the server to reply to requests. If this is not specified the server will respond using the requesting port number.

HEAD *DocumentName*

the server will respond only with the header information for the document(s) requested.

USER *Id*

user name for logging purposes. This may also be used for authentication if **AUTHORITY** is not used.

AUTHORITY *Authentication*

allows for the transmission of an authority string for document access. The contents and interpretation of this string will be particular to the document security system used by the server.

HOST *Name*

name of the client, or calling, host.

The protocol is intended to transfer WorldWideWeb documents. With the use of a keyword tag at the start of each document hypertext **<HTML>**, plain text **<PLAINTEXT>** and indexes **<ISINDEX>** can be transferred and processed.

3.3.3 Hypertext Transfer Protocol Version 1.0 (HTTP)

HTTP is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. It is a generic, stateless, object oriented protocol, which may be used for many similar tasks such as name servers, and distributed object oriented systems, by extending the commands.

or "methods" used. A feature of HTTP is the negotiation of data representation, allowing systems to be built independently of the development of new advanced representations.[3]

The hypertext transfer protocol version 1.0 is an upgrade to the version 0.x of the protocol and is backwards compatible to version 0.x. The general form of a client request is:

Method URI ProtocolVersion CrLf

[HTRQHeader]

[CrLf Data CrLf . CrLf]

The *Method* portion of a client request may be one of the following:

GET	retrieves the object that is identified by the <i>URI</i> . Optionally, if the retrieved object has been identified, by the server, as supporting the method TEXTSEARCH , it can be searched utilizing the construct <i>[?SearchTerm*SearchTerm ...]</i> appended to the <i>URI</i> .
HEAD	retrieves only the HTTP headers for the object identified by the <i>URI</i> .
SHOWMETHOD	returns a description for a given method when applied to the object identified by the <i>URI</i> .
PUT	The data supplied in the body will be used to replace an existing object identified by the <i>URI</i> .
POST	The data supplied in the body will be used to create a new subordinate object which will be linked to the existing object identified by the <i>URI</i> . If successful the new <i>URI</i> created for the object will be returned to the client in the <i>ReasonLine</i> of the returned status line.
REPLY	The data supplied in the body will be used to create a new equal object which will be linked to the existing object identified by the <i>URI</i> . The new <i>URI</i> created for the object will be returned to the client.

The Uniform Resource Identifier *URI* is a string that identifies the object. It may be either a Uniform Resource Locator (URL see section 4.3.3), which represents the address of the object, or a Uniform Resource Name (URN see section 4.3.4) naming the object which has been registered in some name space.

The protocol version, *ProtocolVersion*, is currently set to the string **HTRQ/V1.0** to indicate that the client supports this protocol. If this field is omitted the server will assume that the client supports HTTP version 0.x (see section 3.3.2).

The HTTP request header fields, *HTRQHeader*, allows the client to inform the server of additional information concerning the request. Each line conforms to the RFC822[12] format and is of the form *Item Parameters* CrLf. Request headers must appear immediately after the *Method* line and are delimited from the *Data* by a single blank line. All hypertext request items are optional. The *Item* may be one of the following:

From: The Internet mail address of the requesting user is given in the *Parameters* field. This information may be used by the server for logging purposes or as an insecure method of access control. If the user does not have a valid Internet mail address, information identifying the user should be in the form of an Internet mail address.

Accept: A semi-colon delimited list, in the *Parameters* field, of MIME compatible Content-Type values (see section 4.1.6) which indicate to the server the information formats that can be utilized by the client. A penalty parameter can be associated with each format. It is a numeric measure of the display ability and processing time required by the client for the format specified. The server will, within its capabilities, return the document in a format which involves the least processing penalty to both the client and the server.

Accept-Encoding: A semi-colon delimited list, in the *Parameters* field, of MIME compatible Content-Type values which indicate to the server the types of data encoding that can be supported by the client.

Accept-Language: A semi-colon delimited list in the *Parameters* field, of values which

indicate to the server the preferable language(s) to be used in responses. The format of the language values is yet to be specified.

- User-Agent:** - The *Parameters* field lists the client program name and optionally the version number of the software. If the version number is specified it is delimited from the name with a slash.
- Referer:** - The *Parameters* field contains the URI of the object from which the URI for the current request was derived.
- Authorization:** - The *Parameters* field contains authorization information. The format of the authorization is yet to be specified.
- ChargeTo:** - The *Parameters* field contains account information to which the server can bill access and related charges. The format of the account information is yet to be specified.

The *Data* portion of the request is in a MIME compliant format.

After each client request the server is required to respond, at a minimum, with a status line. The status line is used to communicate to the client a numeric and textual indication of the result of the request. The status line is of the format:

HTTPVersion StatusCode ReasonLineCrLf.

The *HTTPVersion* is a three digit number indicating the version of HTTP being used by the server. For the current version 1.0 this field would be set to 100.

The *StatusCode* is also a three digit number indicating the result of the request. The value of this number, and its required associated client action, is defined in the document "Hypertext Transfer Protocol"[3]. Status codes between 200 and 299 indicate success. Codes between 400 and 599 indicate that an error occurred, and codes within the range 300 to 399 indicate that the object requested has been relocated. Only browsers with link editing capability should automatically relink the reference, all others should consider this an error.

The *ReasonLine* contains human readable text indicating the associated *StatusCode*.

When the *StatusCode* indicates success the lines following the status line are HTTP object headers followed by returned data in a MIME compliant format. The HTTP object headers may be one or more of the following:

Allowed: *Methods*

- A space delimited list of *Methods* which the requesting authorized user may apply to the object requested. The methods may be any of the methods listed in the request section above, or **TEXTSEARCH** if the object is searchable by text, or **SPACEJUMP** if the object can be searched using a coordinate system. While this object header is optional, it must be included if the object can support methods other than **GET** and **HEAD**.

Public: *Methods* - A space delimited list of *Methods* which any requesting user may apply to the object requested (see **Allowed:** for methods). While this object header is optional it must be included if the object is authorized to support public methods other than **GET**.

Content-Length: *Integer*

- Instructs the client to read *Integer* bytes from the server without parsing the information.

Date: *DateInGMT*

- The creation date of the requested object.

Expires: *DateInGMT*

The date on which the object's information is no longer valid.

Last-Modified: *DateInGMT*

- The date on which the object was last modified.

Messageid: *URI* - The unique identifier for the message.

Version-URI: *URI*

- The URI with which the object may be found. Multiple instances of this field will give alternate URIs for the current document.

Live-URI: *URI* - The URI where the most recent version of the object can be found if different than the requesting URI.

Language: *Code* - The code that represents the language in which the object is presented. The format of the language codes is yet to be specified.

Cost: - The cost of retrieving the object is returned. The format of the cost information is yet to be specified.

It is recommended that the HTTP object headers precede the MIME compliant data section, however browsers should allow for HTTP object headers to appear anywhere within the MIME portions of the data section.

3.3.4 Remote Terminal Protocol (TELNET)

TELNET is a keystroke based protocol within the Internet protocol suite that allows for three basic services. The first is the network virtual terminal which provides a remote system interface. The second is a defined method by which both client and server can negotiate transmission options (such as binary or ASCII data transfer). Thirdly TELNET provides for a symmetrical connection between client and server such that either the client or server can be a program instead of a user terminal.

The protocol is simple. The client first requests a connection to the TELNET server using a 'well known' TCP port number. Once the connection is granted the client then transmits the keystrokes from the user terminal to the port number. While this is happening the client is also accepting all the characters sent from the server and displaying them on the user terminal.

The major disadvantage of TELNET is the large amount of overhead required to handle data. Each keystroke is sent from the client, through Internet, to the server who then passes the keystroke to the native operating system. Each character of response travels the same path but in reverse. TELNET however, does provide an easily maintainable access to the server's system without requiring network modifications to the native operating system.

3.3.5 Gopher

Gopher was originally developed to provide a distributed campus information service at the University of Minnesota in 1990. Its unusual name is derived from its intended function of

“go-for”ing information.

Gopher is a shell that allows for transparent client access of information physically located in different locations utilizing other protocols and applications such as TELNET, FTP, ARCHIE, Internet Mail, and WAIS, as well as its own native “Gopher” protocol.

The user interaction with Gopher is by the use of nested menus where, when an item is selected; the appropriate application for that selection is used to obtain either another menu or the information itself. When the Gopher application is started a initial menu is displayed based on the information accessible by the associated Gopher server.

Menus are generated from predefined text files, in menu form, or the output from Archie, WAIS, or a simple Telnet directory listing. Actual data files can be displayed, if a viewer is available, and/or obtained using the FTP or TELNET protocols. At any time mail, using the Internet Mail function, can be sent to the site where the current information screen is obtained from. Links to other Gopher sites can be embedded within text menu files to provide access to other sites.

While Gopher provides easy access to many different types of data, utilizing menu trees, it is strictly designed to handle true tree structures. There is no interlinking of branches within the tree or referencing of branches or associated leaves once a leaf has been accessed. To access another leaf one must first return to the branch. This has the appearance of Hypertext but is not as it is no different than file access on a machine that has a hierarchical file structure.

4

Document Markup and Presentation

ABSTRACT *This chapter reviews document markup representations such as RTF and HTML and document editors which can assist in the document markup. It also covers document locators, which are essential tools and/or techniques within a network environment and document viewers which are client applications that allow a user to display documents, or portions thereof.*

See also chapter 2, Electronic Standards, and 3, Network Computing, for reviews of the other technical components.

4.1 Document Markup

Document markup is a term used to indicate that a document contains indications of how it is to be organized and/or rendered. Markup that only provides information on the document organization is called "*semantic*" markup while markup that only conveys rendering information is called "*formatting*" markup.

There are a large number of "markup" languages available. Some are strictly semantic and others are strictly formatting however the majority of them contain elements of both markup techniques.

There are many more markup languages than are listed here however the ones reviewed here represent the most widely used of them.

4.1.1 Rich Text Format (RTF)

RTF was first publicly documented in the 'March 1983 Microsoft Systems Journal' as a method of encoding formatted text and graphics for easy transfer between applications. It is designed to allow for the exchange of documents, including formatting, between applications on PC DOS platforms and Macintoshes.

An RTF document is made up of plain text which is grouped by control sequences. These sequences are of the form:

`{\ControlName TextGroup}`

For example `{\i italic text}` produces *italic text* when rendered. All control sequences relate to a group of text and RTF delimits the group with the use of `{}`. RTF provides *ControlNames* which the author can use to control the fonts, styles, screen colour, pictures, footnotes, annotations, headers and footers, summary information, fields, bookmarks, as well as the document, section, paragraph, and character formatting properties of the document.

There is no defined extension mechanism for RTF. A particular version of an RTF “reader” (application) will only understand the documented *ControlNames* for that version. The Microsoft definition however, does allow for the identifying of non-standard *control-names*, with the use of `\*`, which are ignored by readers who do not understand them.

While RTF allows for the exchange and representation of text, graphics and tables it does not address mathematical formulations except as represented within a graphic.

4.1.2 Standard Generalized Markup Language (SGML)

SGML is based on an IBM product called GML (Generalized Markup Language) and was originally defined in 1986 and is currently governed by the ISO standard (ISO 8879). The design is based on the idea that an author should not have to worry about the visual presentation of a document, which would be left to the publisher, only it’s organization[6].

A main feature of SMGL is its ability to specify the structure of the document and the logical relationship of its parts. The structure is specified in a DTD or Document Type Definition.

An SGML document is made up of plain text which is bound by control sequences. These sequences are of the form “`<ControlName>`”. Within the SGML language these control names are known as “tags”. For example `<P>` is the control sequence to indicate a new paragraph. Most control sequences relate to a group of text and SGML delimits the group with the use of start and end control sequences. The start sequence is of the form

<ControlName> and is in effect until the end sequence of the form **</ControlName>** is found. The SGML line **<TITLE>This is the Title</TITLE>** defines the title of the document to be "This is the Title". The actual tag or control sequence text is case insensitive therefore **<TITLE>** and **<title>** will be interpreted the same by a SGML browser.

The language definition allows for control names such as **!ENTITY**, **!ELEMENT** and **!ATTLIST** to be used to extend the language. These language extensions can then be used as control names within the document. These extensions are normally defined within a DTD file and not in each document file.

The actual printed or displayed representation of the text marked in SGML is not defined by the SGML language but is up to the SGML browser to decide the representation based on associated output device.

4.1.3 HyperText Markup Language (HTML)

The HyperText Markup Language was created to satisfy the requirements of the World-WideWeb for the processing of structured documents on dis-similar systems. As stated in [5], it was created to:

- Represent existing bodies of information
- Connect information entities with hypertext links
- Scale to a world-wide scope
- Fit into existing and evolving user interface paradigms
- Provide an experimental platform for collaborative hypermedia

HTML is implemented as a particular document definition type in SGML and as such is syntactically the same as SGML. It contains only a reduced subset of the facilities available in SGML. In HTML, for example, a name is restricted to be no more than 40 characters in length whereas in SGML the limit is 255. While this reduction in functionality seems to be restrictive it allows for implementors to create applications using HTML without having to implement the entire suite of SGML functions. The definition of HTML allows for the general group of markup functions:

- Single title for each document
- Support of index type documents which can be searched by the client
- Headers to a minimum of six levels
- Paragraph marking
- Character emphasis such as bold, italic and underline
- Backquoting
- List representation such as bulleted or numbered
- Preformatted text which allows for the inclusion of text that has been formatted before inclusion into the document and will then be displayed in that format
- Support of Universal Resource Locators (URL), a network wide document location scheme
- Hypertext links or anchors

To facilitate the hypertext links between documents the tag "A" (Anchor) has been defined. The anchor tag is used to indicate the relationship between a word or words within the current document to another document. For example, within the HTML document, the string ** Select here for test**, when displayed by a browser, will show only the text "Select here for test" is a distinctive style, selected by the browser's implementor, to indicate that this is a hypertext link. Once the user selects this text the HTML browser will send the string *test/file/test.htm* to the server *xx.yyy.zzz.ca* utilizing the TCP/IP port 80 and the HTTP protocol to obtain the specified document from the server.

Communication of documents between HTML servers and clients does not include the transmission of the DTD, document definition type, which defines HTML. The DTD is well documented and thus browsers are expected to have a local copy of the DTD, if required, to assist in the processing of the HTML document. The HTML markup also does not require the use of **<HEADER>** and **<BODY>** declarations, as does SGML, because the HTML definition requires **<HEADER>** information to be placed before any **<BODY>** markup.

4.1.4 HyperText Markup Language Plus (HTML+)

HTML+ is "a proposed standard for a light weight presentation independent delivery format for browsing and querying information across the Internet"[27]. It is a superset of the original HTML format allowing for:

- Ability of embedding "foreign" data representations
- Enhanced handling of figures (images)
- Definition of tables
- Intrinsic indexing
- Arbitrary grouping of information
- Client-server interface for mouse/cursor or "windowing" interaction

The proposed standard has introduced a number of new tags to provide the new functionality. The tag **<EMBED>** allows for the inclusion of information that is not HTML(+) and would be displayed by browsers which understood how to represent the information. The standard uses the MIME style (see section 4.1.6) to the form of the information. For example **<EMBED TYPE=TEXT/X-TEX>** instructs the browser to process all the information following this tag till the end tag **</EMBED>** to be interpreted as TeX formatted information.

HTML's unique character emphasis tags, such as **** and **<I>**, have been replaced by a single tag **** (EMphasis) which allows for multiple attributes to be assigned to a character string. The HTML+ statement **<EM I>This is an italic line** would be displayed as *This is an italic line*.

The design has defined a number of new attributes for existing HTML markup. The **ROLE** attribute for headers and paragraph tags allows the author to define a segregation of the document into logical role groups. Use of the **INDEX** attribute in emphasis, headers and titles gives the document creators a tool by which they can define a document index, with sub indexes, without creating special document pages by hand. The server can then use these attributes to provide these grouping presentations to the requesting client.

The **TBL** tag allows for the creation of tables within documents. The proposed standard allows for multiple row and column tables with optional titling and bordering.

4.1.5 Text EXecutive (T_EX)

T_EX was developed by Donald E. Knuth in 1983 as a replacement to his first attempt, also known as T_EX (now known as T_EX78), in 1978 to develop a extensive text markup and document formatting language. T_EX is described by the author as the "definitive edition" of the T_EX language[21]. It is a very powerful and extendable language used for the description and typesetting of documents and as such is a very complex language to use and understand. A widely used version of T_EX is L^AT_EX[23], which is a document preparation system built on top of T_EX and provides an additional set of commands to assist in typesetting.

The design of T_EX is similar to the C language in the way that there is a limited number of defined primitives (eg. `\it` (*italics*), `\bf` (**bold face**)) upon which the language is based. This, in conjunction with the ability of T_EX to permit the definition of new or the redefinition of existing control sequences, using such functions as `\def` and `\let`, allows for the rich T_EX function library found in most implementations.

The majority of control sequences within the T_EX language start with a `"\"` character followed by 1 or more ASCII characters representing the name of the function desired (eg. `\bf` is the command for bold face print). Other special symbols exist within the language to delimit commands. The use of the `$` (dollar sign) indicates to T_EX to switch to a 'mathematics mode' which allows for the type setting of mathematical symbols and certain control characters. For example the control sequence `$ \backslash $` allows for the typesetting of a backslash character.

T_EX also supports the concept of grouping which allows the application of a particular command, or commands, to a delimited group of text. With the use of the `{` and `}` to delimit a group and the inclusion of one or more commands within the grouping an author can modify the appearance of the grouped text. these commands will then only apply to the group defined. For example the statement `{\small \bf this is a small bold line}` would result in **this is a small bold line**.

Expression and formatting of mathematical or algebraic formulations is a strong point of T_EX. Information that is bounded, or grouped, by `$` for in-line equations or `$$` for equations that is to be displayed separately from the text. This grouping is interpreted by T_EX using the American Mathematical Society symbols and rules for type setting as opposed

to the standard text type setting rules. For example the equation, from Steel Structures for Buildings (Limit States Design) S16.1 [32], for the exact solution for angle thickness in seated beam shear connections is:

$$t = \frac{-\beta + \sqrt{\beta^2 + 4\alpha\gamma}}{2\alpha}$$

where :

$$\alpha = 0.625\phi^2 F_{ya} F_{yb} w L$$

$$\beta = 2.5\phi w F_{yb} B_r$$

$$\gamma = B_r^2 - \beta(k/2 - 10)$$

F_{ya} = Yield strength of angle

F_{yb} = Yield strength of beam

$$N = 2B_r^2 / \beta - k$$

and is represented in T_EX as:

```


$$t = \frac{-\beta + \sqrt{\beta^2 + 4\alpha\gamma}}{2\alpha}$$

where:

$$\alpha = 0.625 \phi^2 F_{ya} F_{yb} w L$$


$$\beta = 2.5 \phi w F_{yb} B_r$$


$$\gamma = B_r^2 - \beta (k/2 - 10)$$


$$F_{ya} = \text{Yield strength of angle}$$


$$F_{yb} = \text{Yield strength of beam}$$


$$N = 2B_r^2 / \beta - k$$


```

The code example also contains format alignment characters. The `\eqalign` along with the `&` character are used to align the displayed equations along the equal signs.

The production of a document using T_EX is a three step process. The first step is to produce the document text, using any simple text editor, and imbedding within this text all of the T_EX control sequences which will govern the appearance of the final document.

Spacing, section numbering and line length are not important as the TeX processor will attend to these details based on the style of document requested. This text file is then processed by the TeX program and produces what is known as a DVI, DeVice Independent, output file. While the DVI output cannot be viewed directly it can be processed by any number of secondary programs, such as screen viewers and print functions. These have been written to interpret the DVI file and to produce output, on the associated device, as closely as possible to the desired output, based on the abilities of the output device.

TeX enhanced functions, definitions of new control sequences, are normally defined in a style or .STY file. This, in conjunction with the : in definition file, provides the user with the tools and defined automatic functions required to produce a document in the style desired. This document for example is produced with the BOOK.STY standard style file that is modified by the CMTHESIS.STY file. The language allows for the inclusion of one or more .STY files to define the particular document style.

There are some limitations on the abilities of TeX to produce a completely marked up, or typeset, document. There is no provision within the design of TeX to allow for the inclusion of graphics within the document. It does allow for the operator "\special" who's grouped information is not processed by TeX but is passed directly onto the output drivers through the .DVI file. Some implementations have used this function to pass graphical information directly to a custom driver. When this technique is used however, the number and types of output devices that can be used to produce a complete document are severely limited.

4.1.6 Multipurpose Internet Mail Extensions (MIME)

The MIME standard was approved, as defined in RFC1341, in June 1992 to replace the existing Internet mail standard which was established in 1982. While MIME is not a transmission protocol it is also not a true markup language. It does, in fact, provide an envelope which directs the receiver on how to interpret and/or display the contents. Certain aspects of the MIME standard are however, incorporated into "true" markup languages such as HTML+.

The standard, as stated in [20], provides for electronic mail to contain:

- Multiple objects in a single message

- Text having unlimited line length or overall length
- Character sets other than ASCII
- Multi-font messages
- Binary or application specific files
- Images, Audio, Video and multi-media messages

The general form of a MIME directive is:

Content-Type: *Type/Subtype* [*; Parameter*] ...

The *Type* field is used to indicate the general classification of the information to follow. The *Subtype* field is used to indicate a specific instance of the general classification. The optional *Parameter* field(s) is used to provide additional information to the processing application. For example, the MIME directive **Content-Type : Text/x-tex; pages=1-5** instructs the mail receiver to interpret the information as being a plain text file encoded as a TeX document and to display only pages 1 through 5.

MIME currently supports the following content types:

Application	for data that does not fit into the other content types such as pure binary data
Audio	for audio data that requires an audio output device
Image	for image data
Message	for an encapsulated message
Multipart	for mail that contains more than one content type. it requires the definition of the Boundary parameter to separate sections
Text	for purely textual information. This is the default content type
Video	for time based images

Each content type must have a subtype associated with it. Generally the subtype describes the type of encoding used for the data. The standard has only defined a limited number of subtypes. It does, however, allow for the prefix **X-** to be appended to any subtype to indicate that it not currently defined in the standard and will be interpreted only by processes that understand the specific subtype.

The standard is infinitely expandable as it allows for type, as with subtype, to be prefixed with **X-** to indicate an experimental or not currently defined type.

4.2 Document Editors

Mark up languages are generally expressed in ASCII, utilizing printable text for commands. This does allow authors to create documents, utilizing a mark up language, with any ASCII text editor. The resulting document however does not indicate to the author how the final document will be rendered until it is applied to the appropriate mark up processor. While this is a workable situation it requires that the author cycle between writing and displaying to ensure that the document is rendered in the form which he/she has intended.

With the development of sophisticated graphics interfaces, and hardware to run them, a number of WYSIWYG or 'What You See Is What You Get' editors or document entry applications have been created to assist the author to visualize the final result while it is being written. In addition, programmable text editors, such as EMACS, have been enhanced to assist the author in embedding "mark up" transparently, directly into a document but lack the ability to display the finished result directly.

There are a large number of document editors now available and more are becoming available every day. While each of these will have their own features and benefits the ones reviewed here are well known and have been in general use for quite a while.

4.2.1 Word and Word for Windows

Word is a commercial document entry product available from the Microsoft Corporation. It currently runs under a number of operating systems including DOS, MACINTOSH OS, and UNIX. The product is available both as a text based system and a graphical based

system under Windows or X-Windows.

The text based system provides for transparent markup and some indication to the author of the markup utilizing colour and font style. The graphical based system provides the author a direct display of the mark up used.

The default mark up is a proprietary Microsoft format, which is an eight bit encoding scheme. Word however, allows for the exporting of documents to an RTF (see section 4.1.1), plain text, or in formats that are acceptable for other document entry products such as WordPerfect (see section 4.2.2).

4.2.2 WordPerfect and WordPerfect for Windows

WordPerfect is a commercial document entry product available from the WordPerfect Corporation. It currently runs under a number of operating systems including DOS and UNIX. The product is available both as a text based system and a graphical based system under Windows.

The text based system provides for transparent markup and some indication to the author of the markup utilizing colour and font style. The graphical based system provides the author a direct display of the mark up used.

The default mark up is a proprietary WordPerfect format, which is an eight bit encoding scheme. WordPerfect however, allows for the exporting of documents to an RTF (see section 4.1.1), plain text, or in formats that are acceptable for other document entry products such as Word (see section 4.2.1).

It has been announced that the latest version of WordPerfect now supports SGML and with slight modifications to the HTML and HTML+ DDLs can also provide WYSIWYG markup for these mark up languages.

4.2.3 EMACS

The EMACS editor is the de-facto text editor for UNIX based systems. It was originally available only through commercial outlets, however shareware and freeware versions are available for UNIX, DOS and Windows based PC systems as well as MACINTOSH OS systems.

EMACS is a text based editor with no WYSIWYG capabilities. It is however, an extendable and programmable editor. A number of freeware EMACS configuration files are available to assist authors in producing RTF, SGML, HTML, and HTML+ documents.

4.3 Document Locators

Document locators are both tools and addressing schemes which enable applications or users to locate a document or object within the Internet. The following review covers both some locator applications and current and emerging naming standards.

The requirement for a document locator was quickly realized when computers were first joined together by a network. While on small networks it is still feasible to use generic operating system tools (eg. wild-card directory searches) this becomes very unwieldy when the network grows. With the creation and success of Internet and other global spanning networks it is now impossible to locate information unless specialized network-aware tools are used. Naming standards have also become important to describe both the location and access method for documents.

There are a large number of applications which provide document location on the Internet. While this review specifically covers only a few they represent the currently used general methods of locating information.

4.3.1 Archie

The Archie application provides the user the ability to search for full or partial file paths over all of the publicly announced anonymous FTP (see section 3.3.1) accessible information on the Internet.

Once a full or partial file path is entered, as part of the command line, Archie then requests a search of a database, contained on an Archie server, of machine names and complete file paths. The server then returns both the machine name and complete file paths that match the request. The user is then expected to use another application to actually access or obtain the document.

Archie servers are designed to constantly probe the Internet to insure that the database

that they maintain is fairly current. It should be noted that due to the size of the Internet the information contained within an Archie server maybe days or weeks old.

Archie is really a Internet aware LS or DIRectory command. While it can locate files contained in the Internet it does not search or indicate the contents of that file.

4.3.2 Wide Area Information Systems (WAIS)

WAIS is an Internet service that allows for the searching of indexed databases of information. As with Gopher (see section 3.3.5), WAIS finds and accesses information without regard to the information's physical location on the network.

WAIS utilizes the draft ANSI standard Z39.50 which formalizes the method by which one system requests, and receives responses, to search requests from another system.

The user interaction with WAIS is by means of "keyword" searching. Once a WAIS server is selected, usually through a menu selection technique, keywords are entered and the server will then search it's indexes for a match. A list of databases or database entries are then returned to the client with a "score" between 0 and 1000 indicating the success of the search. Items with a score of 1000 indicate an exact match while a score of 0 indicates no match to the keywords requested. Selecting a document from this returned list will access the document for processing by the client.

WAIS supports a large number of information formats from **TEXT** (plain text) to **PS** (postscript) and **GIF** (graphics interchange format). The ability to display and or retrieve the information is a function of the abilities of the WAIS client application being used.

The only access method that WAIS provides is keyword searching. While this can be very powerful, WAIS assumes that every word entered is a keyword. It does not allow for logical operations on the search such as **AND** and **OR** and it also does not support context searches. Context searching is where the order of the keywords is important. An example of this would be the keywords "building standards". WAIS will return with documents for matches to "building", "standards", "building ... standards" and "standards ... building". The scores however, for the last two matches will have the same weight and the user has to inspect each entry to locate the desired documents.

4.3.3 Uniform Resource Locator (URL)

The uniform resource locator is a standardized unique identifier that specifies both the physical address of an object and the retrieval method, utilizing protocols implemented on the Internet. the URL is documented in a proposed RFC "Uniform Resource Locators RFC XXXX" by the Internet Engineering Task Force URL Working Group[4].

The general full form of an URL is:

Scheme: Path[?SearchTerm [+SearchTerm]]

Scheme - the name of the scheme or protocol which will be used to exchange the information. Certain schemes also define the method by which the URL is interpreted. Currently the following schemes are supported:

- HTTP Hypertext Transfer Protocol
- FTP File Transfer Protocol
- AFS Andrew File System (AFS is a registered trademark of Transarc Corporation)
- NEWS Internet News access
- WAIS Wide Area Information Systems access
- PROSPERO
- TELNET Terminal Emulation Protocol
- GOPHER Gopher access

Path - the Internet name or address of the service requested. Depending on the scheme used this may be a physical named address, with or without protocol port number, consisting of a *Host* name with *ObjectPath*, a *CellName*, a *Group* and *Article* or an *Index* and *Document*.

SearchTerm - the search term is used when the client wishes to search the object referenced by the URL. It should be noted that not all schemes support searching in this manner.

The URL RFC also defines a partial URL which can be used once a connection is established with a server. The partial URL is an URL with the *scheme* and server address information removed.

4.3.4 Uniform Resource Name (URN)

The uniform resource name (URN) is an unique identifier which can be used to identify a resource, and is designed to provide persistent naming for network objects. This name will stay the same no matter what the current location(s) of the object is(are)[34].

The URN works in conjunction with a directory service, which provides the client with an URN to URL mapping. This will allow the object to move within the Internet and only require the object's mover to inform the directory service of the new location where the URN identified object can be found.

Unlike the URL, an URN is not intended to be human readable. The general form of the URN is:

URN: *NamingAuthority*: : *OpaqueName* : : :

NamingAuthority - The naming authority consists of two subsections separated by a colon. The first subsection is the name of a protocol or organization which can guarantee the uniqueness and resolvability of the identifier contained in the second subsection. The second subsection, known as an "individual identifier" contains an identifier that, when applied to the naming authority can produce the URLs given the complete URN including the *OpaqueName*. An example of a *NamingAuthority* is **ISBN_Publisher_ID:0_201_12**.

OpaqueName - The opaque name is a string assigned to the object by the *NamingAuthority* which in respect to the complete URN will uniquely identify the object.

The URN naming scheme is currently still a proposal within the Internet.

4.4 Document Viewers

There are a large number of document viewers, or browsers, now available and more becoming available every day. Current development of new browsers is generally towards a graphical interface with support of HTTP, HTML+ and MIME types of information including audio. While each of these will have their own features and benefits the ones reviewed here are well known.

4.4.1 Text Based Viewers

Text based viewers represent the least common denominator type of viewer for WorldWideWeb. They can function on any terminal that has the ability of displaying ASCII text. Hypertext links are represented by an index number, usually bounded by brackets, following the actual link. Users may select a link by typing the number. Additional functionality, such as returning to the Home Page, bookmarks, mailing to a reference, and moving a page forward or backwards is done through commands entered onto the terminal.

These viewers cannot display graphical images, audio or any other type of information that is not ASCII text. They do however provide hints to the user as to the location of these items through areas of the text marked such as **[Image here]**.

Viewers of this type currently in use are WWW, created by the WorldWideWeb group in Cern Switzerland, PerlWWW, a viewer written in the Perl language, and a configuration file that converts EMACS to a WorldWideWeb viewer.

4.4.2 Full Screen Text Based Viewers

Full screen text based viewers are similar to the text based viewers but they require a addressable cursor terminal. Terminals such as the DEC VT series are cursor addressable. Hypertext links are represented by either inverse video text links or completely bracketed links. The user utilizes the cursor keys to both select a link and to scroll the text within the screen. Functionality, such as returning to the Home Page, bookmarks and mailing to a reference is done through commands entered onto the terminal.

These viewers, as with text based viewers, cannot display graphical images, audio or any

other type of information that is not ASCII text. They do however provide hints similar to text based viewers as to the location of these items within the document.

The Lynx browser, from the University of Kansas is typical of this type of viewer as well as NJTT's Browser.

4.4.3 Graphical User Interface Viewers

Graphical user interface (GUI) viewers require a full GUI environment, such as Microsoft Windows Macintosh, NeXTStep and X-Windows, to operate within. They provide the user with full mouse control with drop down menus or buttons to provide all of the functions available within the viewer. Hypertext links are normally marked with colour, underlining, boxing or font style depending on the implementation.

These viewers are capable of displaying graphical images. Time based images and audio are also available when multi-media features are available within the GUI. This is the direction that viewers are taking as GUIs are becoming the standard interface for computer systems.

Currently available GUI viewers are Cello for MS Windows, Mosaic for MS-Windows and X-Windows, Samba for the Macintosh, Erwise for X-motif, and the Browser/Editor for the NeXT systems.

Erwise and Browser/Editor for NeXT represent a new style of viewer that not only accesses and displays the information but also allows the user, when authorized, to modify it utilizing a WYSIWYG style of editor.

5

Design Standards

ABSTRACT *This chapter discusses building codes, such as National Building Code of Canada, and design standards, such as CAN3-S16.1M Steel Structures for Buildings, what they are and how they are used. It also categorizes the different representations of information within a standard.*

5.1 Introduction

Standard is defined as a *“thing serving as basis of comparison; document specifying (inter)nationally agreed properties for manufactured goods etc.”*[28]. Within Canada, engineering design is governed, both by “standards” and “codes”. For example, there is the “Steel Structures for Buildings (Limit States Design)” standard and the “National Building Code of Canada”. While both the terms standard and code are used, they both describe documents which govern the design of an engineering project. For the purpose of this thesis the term “standard” refers to both a standard or a code.

Within this section of the thesis the term “clause” will be used to indicate all the information that is contained between adjacent numbered sections of the original standards document. It may contain text, figures, tables and equations, combinations thereof or no information at all.

5.2 Structure of Printed Design Standards

Design standards are used by Engineers everyday. The National Building Code of Canada, the Ontario Building Code, CAN3-S16.1 which governs design of steel structures, and CSA (Canadian Standards Association) A23.3 which outlines limit state design of concrete structures, are some examples of standards that are in use by Engineers. They are the documents

that control and guide the Engineer in any design. Design Standards are available in a number of forms but the most widely used is the printed form or handbook.

The handbooks contain written text including both the standard and commentary and tables, diagrams, equations and references to other design standards. The following sections describe in detail the types of information that are represented in a handbook or standard.

5.2.1 General Structure

Design standards are made up of sections, or parts, which contain the information pertaining to a particular aspect of the topic covered. For example S16.1, steel design, is broken down into 29 sections ranging from drawings and design requirements to composite beams and connections. Each section is further subdivided into smaller sections or sub-sections which then detail, in a form which may include text, figures, tables and equations, an aspect of the section topic in which it is contained.

The sub-section is usually further subdivided into sub-sub-sections and these may be divided again until a single concept or idea can be covered (clause). Again by example, using S16.1[32]:

13.Member and Connection Resistance

13.11 Bolts in Bearing-Type Connections

13.11.3 Bolts in Tension

Where '13.11.3 Bolts in Tension' presents the definition for T_r . The definition, however, is presented both as a textual definition and as the equation $T_r = 0.75\phi n A_b F_u$ which is defined in terms of ϕ , n , A_b , and F_u which will in turn be defined in other sections or sub-sections of the standard.

The text itself does not provide for a cross reference between an idea presented and the items used to define the idea. The handbook, which includes the standard, uses the commentary to assist in some of these cross reference requirements. The commentary is organized in a similar manner to the standard however, it only covers items which may be open to interpretation as expressed in the standard or items that require an additional explanation to obtain a proper understanding.

To obtain a proper understanding of a particular portion of the standard one generally

requires a great deal of “page flipping” and researching. Each concept is segregated and usually expressed in terms of other concepts contained elsewhere within the commentary and standard. It is often more time consuming to determine whether a item meets the standard than to design the item itself.

5.2.2 Text

Design standards provide the reader with written explanations and clarifications for each concept presented. These, including the embedded symbology, form the textual portion of the standard.

The text portions of a standard is all of the material which is not classified as tables (see section 5.2.3), figures (see section 5.2.4), equations (see section 5.2.5) and definitions (see section 5.2.6).

5.2.3 Tables

Design standards, along with the commentary and appendices, also makes use of tables to convey information to the reader. These tables range from simple two dimensional numeric value tables to multi-dimensional tables containing textual definitions, equations and cross references to other sections.

These tables can be classified into five types of table entries. The criteria used to segregate the table entry types is based on the way the information is presented. These types are:

- **Numeric Entry:** lookup results in a singular numeric value
- **Equation Entry:** lookup results in an equation
- **Reference Entry:** lookup requires a reference to another specification section
- **Textual Entry:** lookup results in a textual explanation
- **Relational Entry:** lookup requires a boolean equation to be resolved

Any particular table contained within a standard can contain one or more of these entry types. All of the entry types are used both as an index as well as an entry within the tables.

Description of Element	Class 1 Plastic Design	Class 2 Compact	Class 3 Noncompact	Class 4 Slender
...	...	...	...	...
Flanges of rectangular hollow structural sections	$\frac{b}{t} \leq \frac{420}{\sqrt{F_y}}$	$\frac{b}{t} \leq \frac{525}{\sqrt{F_y}}$	$\frac{b}{t} \leq \frac{670}{\sqrt{F_y}}$	See Clause 13
...	...	...	...	...

Figure 5.1: Examples of Standard's Table Entries

In fact, most tables represent a mixture of the entry types. For example, from CAN3-S16.1, a line entry from Table 1 'Width-Thickness Ratios: Compression Elements' [32] (see figure 5.1) represents a number of entry types. Both the X and Y axis are Textual Entries which guide the reader to select the proper entry from the table. The entries below the Class 1, Class 2, and Class 3 columns are Equation Entries which, if selected, require the reader to calculate the proper $\frac{b}{t}$ ratio. Finally, the Class 4 column is a Reference Entry which directs the reader to another section of the specification to determine the correctness of his/her design.

Tables can also contain a number of footnotes which have to be referenced to ensure that the usage of the table is correct. These footnotes clarify a particular entry, which may be either a row, column or individual table entry, using any one of the four entry types listed.

5.2.4 Figures

Figures are used within the standard, or commentary, to provide a pictorial explanation of an idea or concept. These diagrams, usually used to present such things as loading configurations, joint details, framing requirements etc. are also used as graphical representations of equations such as X-Y plots.

The standard, or commentary, expects the reader to use the figures to provide additional clarification for ideas that are represented in textual or equation form. There is no basic

information within the standard that is expressed only within a figure.

5.2.5 Equations

Equations are used within the standard and commentary to provide both numerical checks for a design and to compute required values such as load factors, moment resistance, required spacing etc. Equations are expressed in algebraic format and require the reader to reference through the standard to determine both the definition and value required for all the unknowns.

An example, from the National Building Code[26] sentence 4.1.4.2(2), defines the factored load combination.

$$\alpha_D \mathbf{D} + \gamma \psi [\alpha_L \mathbf{L} + \alpha_Q \mathbf{Q} + \alpha_T \mathbf{T}]$$

The factors α_D , α_L , α_Q and α_T are specifically defined in sentence 4.1.4.2(3) whereas ψ is relationally defined in sentence 4.1.4.2(4). The relational definition for ψ is based on the values of $\mathbf{T}$, $\mathbf{Q}$ and $\mathbf{L}$ and require the reader to reference sentence 4.1.2.1(1) for the definitions of these values. The value of γ is determined in sentence 4.1.4.2(5) and is also a relational definition. The definition of $\mathbf{D}$ can also be found in sentence 4.1.2.1(1) however the reader must reference subsection 4.1.5 to determine the proper computation of $\mathbf{D}$.

The whole definition of the factored load combination, as defined in the National Building Code, is further qualified by sentence 4.1.4.2(6) which directs the reader to CSA A23.3 "Code for the Design of Concrete Structures for Buildings", a separate document, for the definition, if the design is in concrete.

5.2.6 Definitions

Each standard contains a section(s), within the document, which provides for a definition of the terms and mathematical symbols used within the standard. The definitions are presented either as a textual definition or a definition using an equation. The definitions also include references to sections of the standard that can provide additional information concerning the term or mathematical symbol being defined.

The sections which may be labelled "Glossary", "List of Symbols", "General Nomenclature" etc. are included here under the general description of definitions.

5.3 Proposed Functionality

5.3.1 General

Programs, known as clients, which utilize a standards server will have access to the information contained in a clause or clauses based on:

1. a list of clause headings that meet a criteria:
 - clauses relevant to certain keywords (classifiers)
 - clauses that are descendants (all clauses below within the hierarchy)
 - clauses that are children (at the level immediately below within the hierarchy)
 - clauses that are siblings (at the same level within the hierarchy)
 - the parent clause
2. a list of classifiers for the current clause and all descendants
3. the glossary entry, or definition, for a given mathematical variable
4. the mathematical formula for a given mathematical variable
5. the glossary entry, or definition, for a given term or phrase
6. portions of a clause called elements
 - the text elements contained within a clause
 - the figure elements contained within a clause
 - the equation elements contained within a clause
 - the table elements contained within a clause

5.3.2 Document Viewers

A client, such as a document viewer, will need to retrieve the textual element of the clause. Depending on the display, or rendering, capabilities of the viewer the figure, equation and table elements associated with the clause may also be retrieved. While it is possible to

obtain only the textual element of a clause, possibly for use with a character based display, the information is not complete unless all the elements are displayed together in context. Viewers, based on a graphical user interface environment, will be able to render all elements of a clause.

The user interface to the standards information would be with a hierarchical list of clause titles. The levels of titles would expand, to the next level, or contract at the request of the user or allow for direct access if the clause identifier is known. Access would also be by classifier(s) which would produce a list of clause titles that meet the classifier(s) selected by the user from a list of valid classifiers produced by the viewer.

Each displayed clause may contain links to other clauses within the current specification or to another. Selecting one of these links will display the referenced, or linked, clause. Selection of a word, phrase or symbol within the displayed clause will display it's definition, if one is available.

An equation evaluation tool should also be available to the user to allow for the solving of displayed equations. The user should only have to provide the data which is unknown to the viewer and server. When the viewer can derive information which can assist the user in selecting an input value, such as a table of yield values for example, it will be displayed and the user allowed to select the appropriate value.

5.3.3 Computing Applications

A computing application is one that requires information from a standard to perform its function but does not have a direct user interface. The interface for these applications will be generally based on classifiers. The classifiers for any particular standard will be published and should not change. For example the string of classifiers **BOLT;TENSION;FAILURE** will always return the clause or clauses pertaining to bolt tension failure. The application can then locate the clause identifiers for this item and then obtain from the standards server only the equations that pertain to bolt tension failure.

If the application does not have the equations which define some of the variables these can also be requested from the server.

As with the viewer applications, computing applications can access the text, figures and

tables that are associated to a clause.

6

Standards' Server Architecture

ABSTRACT *This chapter details the protocols and their extensions required to implement the Standards' Document Server. It also covers the markup requirements for standards to be used with the Standards' Server.*

6.1 Introduction

The Standards' Server architecture consists of three components. The first component is the overall data exchange mechanism. The second is the detailing of the exact commands and extensions to effect the data exchange. The third is the required markup of the actual standards information. Each of these components are discussed in detail within this chapter.

6.2 Protocol Overview

The Standards' Document Server protocol is based on the HTTP V1.0 (HyperText Transfer Protocol (see section 3.3.3)) draft specification. This protocol has been chosen as it is:

- specifically designed for the exchange of information in a hypermedia information system
- designed to be extendable
- closely integrated with MIME and the MIME methodology of encapsulating different representations within a single response
- supported by the WorldWideWeb thus allowing standards' servers to be accessed through the WorldWideWeb
- backwards compatible with HTTP V 0.x protocol which a large number of current Internet browsers support

- the standard on which the next generation of Internet browsers are and likely to be written to

6.3 Standards' Document Server Protocol

The basic architecture of the protocol is that the client sends a single request and server responds. A client request may contain one or more parts and the server response may also have one or more parts. The server is stateless and as such the protocol is designed in a manner which does not require the server to retain any knowledge between transactions.

6.3.1 Terms Used

The following terms are used in the description of the Standards' Document Server protocol.

Text	- A string of ASCII characters which represents the written portion(s) of the standard. The text will be marked up using the HTML+ [27] proposed standard.
Table	- Information which is organized in a tabular form within the standard. Each cell of the table may contain text and/or equation elements.
Figure	- A drawing, chart, image or graph that is contained within the standard. Figures will be represented using industry standard formats such as postscript, GIFF, BMP and TIFF.
Equation	- A mathematical formulation that is contained within the standard. Equations will be represented using T _E X formatting.
Object	- An object, in terms of the standards server, is all the information that is contained between adjacent numbered sections of the original document. The contents of the object may be empty or contain one or more groups of text, tables, figures and equations.
Classifier	A Classifier is a word that identifies the applicability of a standard's section (object) in a design. For example the classifier BEAM would

be assigned to all objects that should be considered in the design of a beam. More than one classifier may be assigned to a particular object.

6.3.2 Requests

The following is a description of the form and functions of the protocol used to access information from the standards' server. While this represents a subset of the entire HTTP V1.0 protocol the server should support the complete HTTP protocol as defined in [3]. Standards' information requests are of the general format:

The following syntax description uses a modified BNF (Backus-Naur Form). The symbol '::=' means 'is defined as'. Symbols set in *italic* type are category names that are further defined elsewhere. Symbols set in **teletype font** are 'terminal' symbols that must appear verbatim. The symbol '|' separates alternatives. Sequences enclosed in '[' and ']' are optional; a '*' suffix means that 0 or more repetitions of the preceding sequence are acceptable.

FullRequest ::= *Method URL ProtocolVersion* CrLf
[*HTRQHeader* CrLf]* CrLf
[*Data*]

Method ::= GET | PUT

URL ::= (*Uniform Resource Locator*)

ProtocolVersion ::= HTTP/V1.0

CrLf ::= (ASCII character 13 (decimal))(ASCII character 10 (decimal))

HTRQHeader ::= *FieldName* : *Value*

FieldName ::= X-EXTRACT | ACCEPT | X-CLASSIFIER

Value ::= *Type / Subtype* [; *Value*]*

Type ::= TEXT | X-TABLE | X-EQN | IMAGE | X-TITLES | X-ALLTITLES
| X-CLASSIFIERS | X-VARIABLES | X-GLOSSARY
| (defined types in Internet RFC1341)

Subtype ::= **X-HTML+** | **X-TEX** | **PLAIN**
| (*defined subtypes in Internet RFC1341*)

Data ::= *MIMEConformingMessage*

Full Request

The protocol requires the full request format, as defined in the HTTP V1.0 RFC, as the **HTRQ** (**HyperText ReQuest**) headers are used to indicate the type(s) of information being requested from the server. If the server receives a simple request of the form *Method URL* CrLf it will assume the **HTRQ** to be **Accept: text/html**.

Methods

The HTTP V1.0 request for comments defines a number of methods that can be applied to the server. Standards' servers should support the complete suite of HTTP methods. The standards' server protocol only utilizes the **GET** and **PUT** methods for accessing or updating standards' information. **GET** is used by the client to access information contained by the server whereas **PUT** is used by authorized users to update the server's information to reflect changes in the standards.

At the time of writing, the original proposed methods, **CHECKIN** and **CHECKOUT**, of ensuring exclusive use of a file for updating have been removed from the HTTP V1.0 request for comment. This, in conjunction with a "yet to be specified" access authorization scheme has severely hindered the development of a "network" file updating scheme for standards' servers utilizing the **PUT** method. It is intended that the Standards' Servers protocol allow authorized authors to update standards across the network once the HTTP V1.0 protocol has been finalized in this area.

Uniform Resource Locator (URL)

The full form of the uniform resource locator used by the standards' server is:

http://HostName or IP-Address: TCP-Port /standard/Name/ Year/SectionNumber

The *HostName* or *IP-Address* is the Internet name or address of the standards' server that is being used.

The *TCP-Port* is 2001 for the experimental version of the server at *HostName* = **www.civeng.carleton.ca**. Registration of a "well known" protocol port number for the standards' document server is intended when a standards' browser has been formalized.

The object identifier is organized in the style of an hierarchical file naming scheme. Depending on the implementation of the server it may, or may not, point directly to a physical file on the server. The root portion of the identifier is **/standard/** which indicates that the rest of the identifier will be interpreted as a standard's object identifier.

The *Name* indicates the actual standard that is to be accessed. For Canadian Standards Association (CSA) documents it will be the CSA identifier with the year omitted. For example, the CSA standard "Steel Structures for Buildings (Limit States Design) CAN3-S16.1-M84" would have the *Name* **CAN3-S16.1-M**. Other documents such as the "National Building Code of Canada" which do not conform to the CSA identifier scheme should be referenced by either a well-known name such as **NBC-CAN** or its ISSN number.

The *Year* is either the 4 digit year in which the standard is published for or the word "latest" to indicate that server should apply the request to the latest version of the standard available.

The *SectionNumber* is the actual section number of the object.

The complete URL should be utilized in all documents created for standards' servers when referencing another standard. The client will normally strip off the protocol identifier (**http:**), *HostName* or *IP-Address* and the *TCP-Port* to form a "partial form URL"[4]. The client then utilizes the stripped off information to establish a connection to the server. The client then uses the "partial form URL" within the request sent to the server. Servers should honour both the complete URL and "partial form URL".

Protocol Version

The protocol version defines the format of the rest of the request. The current version of the hypertext protocol is **HTTP/V1.0**. If the protocol version is not specified, as in a simple request, the server will assume that the client supports **HTTP/V0.9**.

Hypertext Request Headers (HTRQ)

The following hypertext request headers have been defined for the standards' document server. All keywords that begin with **X-** are extensions to the HTTP definition. The requests are case insensitive.

X-EXTRACT: *Request* [*;Request* ...]

- return the objects, as requested, of the standard specified in the URL. Multiple *Requests*, which are also called *Values*, can be made by separating the requests with semi-colons. Valid requests are:

- **TEXT/X-HTML+** returns the textual portion of the object identified in the URL as an HTML+ document. The HTML+ document may contain identified embedded TeX sections. If no textual information is available for the section specified an empty data section is returned.
- **X-EQN/X-TEX** returns the equation portion(s) of the object identified in the URL in TeX format. If no equation information is available for the section specified an empty data section is returned.
- **X-TABLE/X-HTML+** returns the complete table portion(s) of the object identified in the URL as an HTML+ document. The HTML+ document may contain identified embedded TeX sections. If no table information is available for the object specified an empty data section is returned.
- **IMAGE/ImageFormat** returns the figure portion(s) of the object identified in the URL in the specified *ImageFormat*. The initial implementations will utilize the **POSTSCRIPT** format for images. Browsers should first use the **SHOWMETHOD** method to determine the available image formats.
- **X-TITLES/X-HTML+** returns the next level of section numbers and associated titles below the section specified in the URL as

an HTML+ document. The HTML+ document may contain identified embedded TeX sections. If no sections are found it will return the entire section object.

- **X-ALLTITLES/X-HTML+** returns all the levels of section numbers and associated titles below the section specified in the URL as an HTML+ document. The HTML+ document may contain identified embedded TeX sections. If no sections are found an empty data section is returned.
- **X-CLASSIFIERS/PLAIN** returns a list of all the classifiers associated with the section specified and all levels below. This information can be used by the client to provide the user with a valid list of classifiers within the current context.
- **X-GLOSSARY/X-HTML+** returns the standard's textual definition for the term specified in the data portion of the request. The term is case insensitive. The server will only interpret the URL to the *Year* to determine what standard is requested. The HTML+ document may contain identified embedded TeX sections. If no definition exists an empty data section is returned.
- **X-GLOSSARY/X-TEX** returns the standard's equation definition for the term specified in the data portion of the request. The term is expressed in TeX format and is case sensitive. The server will only interpret the URL to the *Year* to determine what standard is requested. If no definition exists an empty data section is returned.

ACCEPT: *Request* [*;Request*]

The method **ACCEPT** is a defined method within the HTTP protocol. The server will accept this method and interpret it as the method as **X-EXTRACT** where applicable.

X-CLASSIFIER: *Classifier* [*;Classifier* ...]

Returns all the section numbers and associated titles, which have been identified with the specified *classifier(s)*, below the section specified in the URL as an HTML+ document. Multiple *classifiers*, which are also called Values, can be specified by separating the classifiers with semi-colons. The HTML+ document may contain identified embedded TeX sections. If no sections are found an empty data section is returned. Reference to classifiers is order dependent. A request of **Beam; Tension** returns objects that are tagged with both classifiers together as well as all objects that are tagged **Tension** which are hierarchically lower, by section, then an object tagged **Beam**. The classifier **Beam; Tension** is not the same as **Tension; Beam** and may not produce the same results. Multiple **X-CLASSIFIER:** requests can be made.

Data

The data portion of a request is used for both specifying additional information required to complete the request (eg. **X-EXTRACT: X-GLOSSARY/X-TEX**) or contains information to update the existing server data utilizing the PUT method.

The data portion of the request is in a MIME compliant format and is separated from the hypertext requests fields by a single empty line. The data portion of the request is terminated by a single line containing a period followed by a CrLf. If the data portion itself requires a single line containing a 'period CrLf' it will be translated to 'period period CrLf' before transmission.

The following is an example of a glossary request utilizing the data portion.

GET http://standard/CAN3-S16.1-M/1984 HTTP/V1.0

Accept: X-Glossary/X-HTML+

MIME-Version: 1.0

Content-Type: Text/Plain

Load combination factor

.

The Content-Type MIME field should reflect the type, or types, of data representation which are contained in the request. If no Content-Type is specified the server will assume **Text/Plain**.

6.3.3 Responses

The following is a description of the form and functions of the protocol used to return requested information to the client from the server. While this represents a subset of the entire HTTP V1.0 protocol the server should support the complete HTTP protocol as defined in [3]. Standards' information responses are of the general format:

The following syntax description uses a modified BNF (Backus-Naur Form). The symbol '::=' means 'is defined as'. Symbols set in *italic* type are category names that are further defined elsewhere. Symbols set in **teletype font** are 'terminal' symbols that must appear verbatim. The symbol '|' separates alternatives. Sequences enclosed in '[' and ']' are optional; a '*' suffix means that 0 or more repetitions of the preceding sequence are acceptable.

Response ::= *HTTPVersion* *StatusCode* *ReasonLine* *CrLf*
[*ResponseHeader* *CrLf*]* *CrLf*
[*Data*]

HTTPVersion ::= 100

StatusCode ::= *Digit* *Digit* *Digit*

Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

ReasonLine ::= *HTML+String*

CrLf ::= (ASCII character 13 (decimal))(ASCII character 10 (decimal))

ResponseHeader ::= *FieldName* : *Value*

FieldName ::= **ALLOW** | **PUBLIC** | **FOR-METHOD**

<i>Value</i>	::= GET PUT GET PUT TEXTSEARCH GET GET TEXTSEARCH <i>Type</i> / <i>Subtype</i>
<i>Type</i>	::= TEXT X-TABLE X-EQN MAGE X-TITLES X-ALLTITLES X-CLASSIFIERS X-VARIABLES X-GLOSSARY <i>(defined types in Internet RFC1341)</i>
<i>Subtype</i>	::= X-HTML+ X-TEX PLAIN <i>(defined subtypes in Internet RFC1341)</i>
<i>Data</i>	::= <i>MIMEConformingMessage</i>

Response

The response line is always sent by the server after any request from the client. It is an HTTP V1.0 standard response line. The fields of the response are separated by a single space.

HTTP Version

The version defines the format of the rest of the response. The current version of the hypertext protocol is HTTP/V1.0 and is expressed as the 3 digit ASCII decimal number 100.

Status Code

The status code is expressed as a 3 digit ASCII decimal number. Status codes are defined in the HTTP document[3]. They are broken down into the following general categories:

2XX Success	Indicates that the request has been successfully acted upon by the server. Requested object information will be returned in the data portion of the response as a MIME message.
3XX Redirection	Indicates to the client that additional action must be taken in order for the server to satisfy the request. This may involve requesting the information in another format or specifying a different URL to access the information. The response header will contain directions for the

client concerning proper access of the information (see Internet draft "Hypertext Transfer Protocol"[3])

4XX Client Error Indicates that the request made by the client is in error. This may be due to an ill formed request, invalid access, or the object requested does not exist. The data portion of the response may contain HTML+ marked up data giving an explanation.

5XX Server Error Indicates that the request cannot be fulfilled due to a server error. This may be due to an internal error or that the server does not support the request. The data portion of the response may contain HTML+ marked up data giving an explanation.

Reason Line

The reason line contains a short human readable text, marked up in HTML+, for the associated status code. This should be ignored by the client if the status is "success", however it should be displayed to the user if the status indicates error along with any additional error messages contained in the data portion of the response.

Response Headers

The response headers will be sent by the server to the client for every successful request. The following are response headers that are used by the standards' server:

ALLOW: GET PUT

PUBLIC: GET - These two response headers are sent in response to a valid request from the client and the returned object is not searchable. The **ALLOW** indicates the methods that can be applied to the requested object by authorized users. The **PUBLIC** indicates methods that can be applied by any client.

ALLOW: GET PUT TEXTSEARCH

PUBLIC: GET TEXTSEARCH

These two response headers are sent in response to a valid request from the client and the returned object is searchable. The **ALLOW** indicates the methods that can be applied to the requested object by authorized users. The **PUBLIC** indicates methods that can be applied by any client.

FOR-METHOD: *valid-request* [*;valid-request*]

This response line is sent for a request with the method **SHOWMETHOD**. It returns a semi-colon separated list of MIME Content-Type style valid requests which can be applied to the object specified in the client request URL.

Data

The data portion of a response, if the request is successful, contains the requested information. If an error has occurred it may contain an additional explanation of the error in HTML+ markup.

The data portion of the response is in a MIME compliant format and is separated from the response headers by a single empty line. The data portion of the response is terminated by a single line containing a period followed by a CrLf. If the data portion itself requires a single line containing a 'period CrLf' it will be translated to 'period period CrLf' before transmission.

The following is an example the response to the glossary request example given in section 6.3.2.

```
100 200 Success
Allow: GET PUT
Public: GET
```

```
MIME-Version: 1.0
Content-Type: X-Glossary/X-Html+
```

```
<?><em b>Load combination factor</em>, <embed type=text/x-tex>$\psi$
</embed> means a factor, given in
<a href="http://www.civeng.carleton.ca:2001/standard/CAN3-S16.1-M/1984/7.2">
Clause 7.2, applied to a specified load for the limit states under
consideration that takes into account the variability of the loads and
```

load patterns and the analysis of their effects.

The Content-Type MIME field will reflect the type or types of data representation which are contained in the response. In this example the **X-Glossary** type is returned to identify to the client that a glossary entry is being returned. This will allow the client, if desired, to render the information in a different style or method from the main text of the document.

6.4 Standard's Markup Overview

The Standard's markup is based on the HTML+ (HyperText Markup Language see section 4.1.4) draft specification. This protocol has been chosen as it is:

- specifically designed for wide area hypertext documents
- designed to be extendable
- designed for rendering on a wide variety of different display types and platforms
- easily decodeable by application that wishes to use the information without displaying it
- closely integrated with MIME and the MIME methodology of encapsulating different information representations within a single response
- transferable over a number of Internet protocols such as FTP, WAIS, Gopher, and HTTP
- supported by the WorldWideWeb thus allowing standards' servers to be accessed through the WorldWideWeb
- backwards compatible with HTML markup which a large number of current Internet browsers support
- the standard on which the next generation of Internet browsers are and will be written to

6.4.1 Standard's Markup

This section covers the markup of a standard's document. The document may utilize any of the documented tags and attributes as defined in HTML+[5]. A number of HTML+ extensions have been defined to facilitate the functionality required for a Standards' Server. These are documented below. A qualifier on the use of full and partial URLs within standards documents has also been included.

Attribute Extensions

The information within a standard is marked up using HTML+ with attribute extensions to facilitate the referencing and identification of information by both the client and the server. These attribute extensions have been identified with the MIME style "X-" prefix.

The following attribute extensions have been defined:

X-SECTION="*ObjectIdentifier*"

- is used within header tags <H?> to identify the object. It is to be a document unique identifier organized in a hierarchical fashion. Within standards this would normally be the same as the section numbers contained with the header information. It is this identifier that is used within the request URL to access the object. It does not form part of the rendered document or is contained in a response to a client request.

X-CLASSIFIERS="*Classifier* [*Classifier* ...]"

- is used within header tags <H?> to indicate the classifiers that apply to the tagged section. It does not form part of the rendered document or is contained in a response to a client request.

X-LOOKUP="*Classifier* [*classifier* ...]"

- is used within an anchor tag <A...> to list classifiers that should be accessed within the anchored document to provide information pertaining directly with the reference.

X-GLOSSARY="*DataRepresentationTermIdentifier*"

- is used within header tags <H?> or page tags <P> to identify the text

and/or equation(s) that will be returned for and **X-GLOSSARY** request. It is to be a document unique identifier. The *data-representation* is a MIME compliant indicator of the representation of the *term-identifier*. It does not form part of the rendered document or is contained in a response to a client request.

Compatibility with HTML+

The HTML+ document defines a number of tags, or attributes, which closely match the functionality required by "specialized" tags utilized for standards processing. While it is hoped that, through discussions with the authors of HTML+, changes can be made to the definition of existing tags and attributes to accommodate standards, or have the incorporation of the new attributes within the HTML+ definition; they are currently separately defined. It was also felt that by creating the new attributes, the defined HTML+ tags and attributes could be used within the document and interpreted properly, if the author wished to do so.

The **X-SECTION** attribute has very similar functionality to the HTML+ tag **GROUP**. The **GROUP** tag, however, can be used to group information into a section, chapter or book type classifications. This is different than **X-SECTION** which, within standards, is intended to identify a unique object (see section 6.3.1) within the document. The **X-SECTION** attribute also performs some of the functionality defined for the **INDEX** attribute, however does not meet all the requirements of the **INDEX** definition.

The **X-CLASSIFIERS** has a role similar to the **ROLE** attribute within HTML+. The **ROLE** attribute however, assumes and requires that a single identifier be used to define it. The **X-CLASSIFIERS**, as defined, allow an **OBJECT** to assume a number of classifiers or roles.

The **X-LOOKUP** has no equivalence within the current HTML+ definition.

It has been requested of the authors of HTML+ that an **ALIAS** tag be defined that would allow authors of documents to associate document specific tags and attributes to an HTML+ defined tag or attribute which can provide some of the functionality required. This would allow non-specific browsers to attempt to interpret the document in the best form possible for the user.

URLs in Anchors

The complete URL should be utilized in all documents created for standards' servers when referencing another standard. Partial URLs may be utilized within a standards document if the reference is to another portion of the same document. Servers should honour both the complete URL and "partial form URL".

7

Client Applications

ABSTRACT *This chapter provides outlines of the expected functionality of client applications designed for use with a standards' document server. It describes both the functionality and the Client/Server exchanges required to obtain that functionality.*

7.1 Introduction

With the demise of text based interfaces and the general acceptance of Graphical User Interfaces (GUIs), such as Windows or X-Windows, to provide the user interface to applications, the following descriptions of applications assume a GUI. The standards' document server applications as described, when implemented using only a textual user interface, will operate correctly, however they will not be able to display equations, figures or functions associated with these, and they will represent tables in a character based format.

The following sections describe planned applications for the Engineer using the standards' document server. When implemented, these applications will provide an Engineer with tools that will allow for rapid and comprehensive access to the portions of specifications that apply to a particular design problem.

7.2 Reader/Browser

The Reader/Browser application provides the user with the ability to scan and review sections of the requested specification. In many ways this application is similar in functionality to existing Internet hypertext browsers such as Mosaic and Cello (see section 4).

Existing Internet hypertext GUI reader/browsers provide the user with:

- **Hypertext links** which allow for the chasing of the document chain by simply selecting the identified associated hypertext link, using the mouse, within the currently

displayed document. Hypertext links within a document are indicated using many different methods such as colour, font style, boxing or underlining.

- **Bookmarks** which allow for the local identification of documents the reader may be interested in referring to again. It also allows for direct access to these “marked” documents without having to follow the original hypertext path.
- **Sessions** which allow the user to invoke independent sessions of the more standard Internet applications such as FTP (see section 3.3.1), Gopher (see section 3.3.5) and Telnet (see section 3.3.4).
- **Configuration** which allows for the assignment of display fonts to hypertext rendering commands.
- **Searching** which allows for searching for text within the currently displayed document or searching of documents tagged as **ISINDEX**.
- **Editing** which allows for the standard GUI inter-application editing tools such as cut, paste, delete, and insert.

The Reader/Browser for the standards’ document server will incorporate all the standard GUI reader/browser functions as well as functions which are unique to standards’ document server. The following sections describe these additional functions as well as documenting the client/server exchange required to implement the functionality.

7.2.1 Hypertext Links

The identification of a hypertext link within the Reader/Browser will be done using font style and colour. Both the font style and colour will be user definable. Other methods such as boxing or underlining were not selected as these methods distract from the overall presentation of the document.

The **<A>** (anchor) tag within an HTML+ document indicates a hypertext link. The Reader/Browser will then render the enclosed text between the **<A>** and **** tags using the defined link font and colour. The following is a example of the exchange between the client and server when a hypertext link is selected.

The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

User Selects from Current Document

```
<a href="http://standard/CAN3-S16.1-M/1983/13.2">See section 13.2</a>
```

Only the text "See section 13.2" will be visible to the user.

Request from Client

```
GET http://standard/CAN3-S16.1-M/1984/13.2 HTTP/1.0
X-Extract: Text/X-HTML+;X-Eqn/X-Tex;X-Table/X-HTML+;Image/Postscript
```

Text, X-Eqn, X-Table, and Image are used to insure complete section information is returned

Response from Server

```
100 200 Success
```

```
Allow: GET PUT
```

```
Public: GET
```

```
MIME-Version: 1.0
```

```
Content-Type: Text/X-HTML+
```

```
<a href="http://standard/CAN3-S16.1-M/1983/13.3">Next</a>
```

```
<a href="http://standard/CAN3-S16.1-M/1983/13.1">Previous</a>
```

```
<h2 x-section="13.2">13.2 Axial Tension</h2>
```

```
<p>The factored tensile resistance <embed type="text/x-tex">$T_r$</embed>,
developed by a member subjected to an axial tension force shall be taken as
```

```
<p>(a) the lesser of
```

```
<p indent>(i)<embed type="x-eqn/x-tex">
```

```
$T_r = \phi A_n F_y \text{ when } A_n / A_g \geq F_y / F_u$</embed>
```

```
<p indent><embed type="x-eqn/x-tex">
```

```
$T_r = \phi \left( F_u \left\{ \frac{A_n}{A_g} \right\} \right) A_n \text{ when } A_n / A_g < F_y / F_u$</embed>
```

```
<p>(b) For pin connections, on net area
```

```
<p indent><embed type="x-eqn/x-tex">$T_r = 0.75 \phi A_n F_y$</embed>
```

The first two lines of the returned HTML+ document contain the server generated Next and Previous anchors. In this example the standards file preprocessor has removed the X-CLASSIFIERS attribute from the "h2" tag.

The Reader/Browser for the standards' document server will provide independent buttons for the "Next" and "Previous" anchors which are returned with every X-Extract request and remove them from the rendered document.

7.2.2 Table of Contents

Standards are highly organized and structured documents (see section 2.2.2 for an example) and the associated table of contents for each standard can provide a quick and easy method of locating relevant and associated sections. The Reader/Browser for the standards' document server will provide a menu item which will generate the high level table of contents for the currently selected standard. Selecting one of the high level entries will expand that entry into the next level of contents. This action is referred to as "exploding" the entry (see figures 7.1 and 7.2). If an entry which has no lower level is accessed, the standards' document server will return the actual entry for the section (see section 7.2.1). The selection of an upper level table of contents entry will collapse the associated lower levels.

The following are two examples of the exchange between the client and server. The first is the exchange when the Table of Contents menu item is selected and the second is when a Table of Contents item is selected.

The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

Request from Client when Table of Contents menu item is selected

GET http://standard/CAN3-S16.1-M/1984 HTTP/V1.0
X-Extract: X-Titles/X-HTML+

Response from Server

100 200 Success
Allow: GET PUT
Public: GET

Table of Contents	
12. Gross and Net Sections	
13. Member and Connection Resistance	
14. Fatigue	
15. Beams and Girders	
16. Open Web Steel Joists	
17. Composite Beams and Columns	
18. General Requirements for Built-Up Members	
19. Stability of Structures and Individual Members	
20. Connections	
21. Bolting Details	
22. Structural Joints Using ASTM A325M, A490M, A325	

Figure 7.1: Top Level Table of Contents Display

NINE-Version: 1.0

Content-Type: X-Titles/X-HTML+

<title>Table of Contents</title>

<body>

<h1 x-section="1.">1. Scope and Application</h1>

<h1 x-section="2.">2. Definition and Symbols</h1>

... additional sections

<h1 x-section="12.">12. Gross and Net Sections</h1>

<h1 x-section="13.">13. Member and Connection Resistance</h1>

<h1 x-section="14.">14. Fatigue</h1>

... additional sections

<h1 x-section="29.">29. Inspection</h1>

</body>

Figure 7.1 shows the user's view of the top level table of contents display.

Request from Client when Table of Contents Entry is Selected

GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0

X-Extract: X-Titles/X-HTML+

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: X-Titles/X-HTML+

<title>Table of Contents</title>

<body>

<h2 x-section="13.1">13.1 General</h2>

<h2 x-section="13.2">13.2 Axial Tension</h2>

<h2 x-section="13.3">13.3 Axial Compression</h2>

<h2 x-section="13.4">13.4 Shear</h2>

<h2 x-section="13.5">13.5 Bending -- Laterally Supported Members</h2>

<h2 x-section="13.6">13.6 Bending -- Laterally Unsupported Members</h2>

<h2 x-section="13.7">13.7 Plastically</h2>

... additional sections

</body>

Figure 7.2 shows the user's view of the revised table of contents display. The server only returns the information requested and as such the client (Reader/Browser) is responsible for maintaining the display (ie inserting/deleting the lower levels of the Table of Contents). The title "Table of Contents" will be returned with all X-Extract requests for X-Titles/X-HTML+.

7.2.3 Classifier Searches

The generalized GUI reader/browser search functionality will be enhanced with a classifier[31] search. When selected it will present a list of currently valid classifiers based on the currently viewed section. When a classifier, or classifiers, have been selected from the list, an exploded table of contents will be returned. This table of contents will contain all sections that have been identified, in the original document, as relating to the classifier(s).

The implementation of this functionality will require two queries to be made to the standards' document server. The first is to request a list of currently valid classifiers and the second is to obtain the selected table of contents. The following is an example of the exchange between the client and server for a classifier search request made by the user.

Table of Contents	
12. Gross and Net Sections	
13. Member and Connection Resistance	
13.1 General	
13.2 Axial Tension	
13.3 Axial Compression	
13.4 Shear	
13.5 Bending -- Laterally Supported Members	
13.6 Bending -- Laterally Unsupported Members	
13.7 Plastically	
14. Fatigue	
15. Beams and Girders	
16. Open Web Steel Joists	

Figure 7.2: Exploded Table of Contents Display

The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

User has requested Classifier Search and the currently displayed document is CAN3-S16.1-M/1984/13. This request is to fill the list of currently valid "classifiers".

GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0
X-Extract: X-Classifiers/Plain

Response from Server

100 200 Success
Allow: GET PUT
Public: GET

MIME-Version: 1.0
Content-Type: X-Classifiers/Plain

member
connection
resistance

```
general
axial-tension
axial-compression
w-shapes
HSS
cold-formed
nonstress-relieved
class-1
class-2
class-3
hot-formed
stress-relieved
singly-symmetric
channels
class-4
.
```

The server responds with a CRLF delimited list of the unique classifiers for section 13. and the levels below. The list may be returned in any order. For this example the list is in order of occurrence.

User Selects axial-compression, HSS and class-1 for the search. The request from the Client is a classifier search using the X-Classifiers method.

```
GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0
X-Classifiers: axial-compression;HSS;class-1
```

Response from Server

```
100 200 Success
Allow: GET PUT
Public: GET

MIME-Version: 1.0
Content-Type: Text/X-HTML+

<title>Table of Contents</title>
<body>
<h3 x-section="13.3.1">13.3.1</h3>
<h3 x-section="13.3.2">13.3.2</h3>
</body>
.
```

The response only contains the heading information which meets the classifiers listed.

Once the Table of Contents is displayed the user may now proceed with the normal Table of Contents usage (see section 7.2.2).

7.2.4 Glossary Search

The generalized GUI reader/browser search functionality will also be enhanced with a glossary search. The user will select the text within the currently displayed document using the standard technique for the GUI, such as word highlighting using the mouse, and then selects the Glossary search menu item (this menu selection could also be connected to the middle or right mouse button). The client then requests from the standards' document server the definition of the "term". If one is found the client will then display the definition in a "pop-up" or separate window depending on the GUI capabilities.

The following is an example of the exchange between the client and server when a glossary term search has been requested.

The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

User Selects Load Combination Factor for a glossary definition.

GET http://standard/CAN3-S16.1-M/1984 HTTP/V1.0
X-Extract: X-Glossary/X-HTML+

MIME-Version: 1.0
Content-Type: Text/Plain

Load Combination Factor
.

Response from Server

100 200 Success
Allow: GET PUT
Public: GET

MIME-Version: 1.0
Content-Type: X-Glossary/X-HTML+

<title>Load Combination Factor</title>
<body>
<p><em b>Load combination factor, <embed type=text/x-text>\$\psi\$
</embed> means a factor, given in

Clause 7.2, applied to a specified load for the limit states under

consideration that takes into account the variability of the loads and load patterns and the analysis of their effects.

</body>

7.2.5 Printing

Even in this age of electronics the printed word is still required. The Reader/Browser for the standards' document server will also have a printing capability. The selection of the menu item **Print** will print the entire currently accessed document section or Table of Contents. If the output device is capable of graphical printing the equations and figures will also be printed otherwise a place-holder (eg [Equation], [Figure]) will be printed in its place.

7.3 Enhanced Reader/Browser

The Enhanced Reader/Browser will contain all of the functionality outlined in Reader/Browser (see section 7.2) as well as the ability, for the user, to solve the equations presented in the standard.

7.3.1 Equation Solver

The Equation Solver will be a separate window invoked by the user through a menu selection within the Reader/Browser. The user will select the equation within the currently displayed document using the standard technique for the GUI, such as highlighting, using the mouse. The equation will then be displayed in the Equation Solver window. The Equation Solver will then attempt to obtain values for all the terms of the requested equation. A list of the terms along with their obtained values will be displayed below the equation. Terms for which no value can be automatically obtained, will be indicated as such.

The user can modify any term's value, using normal GUI techniques, as well as "cutting and pasting" values from the Reader/Browser. This "cutting and pasting" technique allows the user to obtain possible alternative values, for a term, from different sections of the standard or from tables of values contained within the standard.

It is possible that a term, or terms, within the equation can have more than one value such as in a list of yield values for various grades of steel. In this case the Equation Solver

will allow the user to select a value from a list of possible values as defined within the standard.

Terms, for the current equation, that have been previously defined (resolved) within the current session of the Equation Solver will inherit that value. Terms whose definition is another equation will cause the Equation Solver to resolve the defining equation before resolving the currently selected equation.

Most of the outlined functionality listed above, including the actual computations, will be done by the Equation Solver application itself. Obtaining the original equation and the definition of the terms of that equation will be done by exchanges between the client and the standards' document server. The following is an example of the exchange between the client and server when a the exchanges required to obtain the equation and term definitions.

The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

User Selects the following Equation from Section 13.2. It is transferred to the Equation Solver using standard GUI interwindow communications.

<embed type=x-eqn/x-tex>\$T_r = 0.75 \phi A_n F_y\$</embed>

*The Equation Solver uses the **type** to determine the method to decode the equation (in this example **TeX**). The actual equation is $T_r = 0.75 \phi A_n F_y$ and is displayed to the user in this form.*

Client obtains definition of ϕ .

**GET http://standard/CAN3-S16.1-M/1984 HTTP/1.0
X-Extract: X-Glossary/X-TeX**

**MIME-Version: 1.0
Content-Type: x-eqn/x-tex**

\$\phi\$

*the **X-Glossary/X-TeX** request asks only for equation definitions of the term.*

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: x-eqn/x-tex

$\phi = 0.96$

The Equation Solver will repeat this process until all of the terms of the equation have been queried from the standards' document server.

7.4 Design Notes

In this age of electronic design an engineer is still required to create and maintain a set of design notes for any particular project. These notes normally contain the equations and calculations, references to the standards sections that apply, and notes concerning the project. The Design Notes application, in conjunction with the standards' document server, will provide a complete set of design notes for a project on electronic "paper".

7.4.1 Design Notes

The Design Notes application incorporates all the functionality defined in the Reader/Browser (see section 7.2) and the Enhanced Reader/Browser (see section 7.3). Unlike the previous applications, Design Notes will retain all of the "transactions" in a redisplayable form for further reference and will allow the user to save and recall this information at a later date.

When a project is started, the engineer creates a new workspace within Design Notes for that project. As he/she accesses the standard through the standards document server or solves an equation the information is retained. When the workspace is saved it can be recalled at a later date. When recalled, values for terms of equations will assume the last known values for that workspace. As well, all the text, figures and tables that were accessed, or created, previously will be available.

Design Notes will also allow the engineer to annotate, at any point, the information retained. This will allow for comments or design decisions to be recorded within the notes.

The actual design notes will be stored in the server format (see section 6) format and thus can be accessed and linked to the standards document server in the same manner as the original standard information.

The Design Note application does not require any additional functionality from the standards' document server but represents the intended goal for usage, by engineers, of the standards' document server.

7.5 Design Programs

Stand-alone design programs, such as STRUDL and STRESS, can also take advantage of the standards' document server. Programs, such as these, depend currently on "hard-coded" equations to validate the submitted design against the current standard(s). As the standards change so must these programs to insure that the checks are made using the latest equations.

Using the standards' document server, and placing requests to the server for the current valid equations, will insure that these programs will always return to the engineer valid results.

The mechanism by which an application can access equations is the **X-Extract/x-Eqn**. This method, as outlined in section 7.3.1, allows for an application to obtain the latest definition, with respect to the standard requested, of an equation. Resolutions, or values, for the terms of the equations can be obtained through the use of the method **X-Glossary/x-Tex** of accessing the standards' document server (see section 7.2.4).

Since stand-alone applications do not require, and are not are designed to accept, interactive user input, use of the standards' document server to provide equations and equation term values must be designed with this in mind. These applications allow for the "pre-definition" of values or terms before the design is run. If the application obtains, from the standards' document server, a term that cannot be resolved to a singular value, an error will be reported. The user, utilizing the "pre-definition" value functions, can then assign an appropriate value to the term or terms that cannot be defined by the standards document server.

8

Implementation Considerations

ABSTRACT *This chapter provides an implementor of a Standards' Document Server, or a client application, information which will assist in the actual production of a product. It also provides a number of examples that assist in the understanding of particular concepts.*

8.1 Introduction

The actual implementation of a server or client on a particular platform depends greatly on the system resources, techniques and tools available to the software author. This chapter discusses implementation in terms of functionality required and does not intend to constrain the actual implementation of a particular server or client.

8.2 Server Considerations

8.2.1 Hypertext Transfer Protocol (HTTP)

The Standards' Document Server can be implemented utilizing a subsection of the HTTP V1.0 protocol with extensions (see section 6.3.2). However, it is intended that the extensions be incorporated into a fully functional HTTP V1.0 server. This would allow for all the features of HTTP to be utilized by clients.

The server should support both full and partial URLs (see section 4.3.3). When a full URL is used, it should be verified that the reference is to "this" server. If it is not then the error **404 Not Found** is returned. No attempt should be made to forward the request. It is the clients responsibility to insure that the request is forwarded to the proper server. If a partial URL is used the server will assume that it is for "this" server. This support is required to allow for the server to function properly in both direct connect and network

environments.

The URL used for the Standards' Document Server references an object or objects within a standard and not a hierarchical file structure. While a server could be constructed utilizing a hierarchical file structure it should not assume that this is so. URLs, for the standards server, can require interpretation (eg. **latest**) before application.

8.2.2 Data Files and Structures

Appendix A contains sample marked up sections, in HTML+, of a standard. This represents a server's data file. A complete standard, once marked up, will be many megabytes in size. It would be possible to design a server that could process the file, as is, in response to requests however due to the size of the initial data there would be a severe response time penalty involved.

It is recommended that servers be designed to utilize preprocessed standards data files. Standards data is generally static, updated every two or three years, and as such does not require "real time" processing of raw data. While the preprocessing and the resulting data are implementation dependent the following ideas are presented as guide lines as to the types of indexes etc. that should be generated:

- a Section index. This is an index relating section identifiers, as indicated with the **X-SECTION=** attribute, to the location of the actual object within the data. This index should maintain information for the hierarchical nature of section identifiers.
- A Classifier index. This is an index relating a classifier word, as indicated with the **X-CLASSIFIER=** attribute, to the objects that relate to that classifier. It should be noted that the client's reference to classifiers is both unique and hierarchical. A client request of **Tension** must return all the objects that are tagged with that classifier. A request of **Beam; Tension** must return objects that are tagged with both classifiers together as well as all objects that are tagged **Tension** which are hierarchically lower than and object tagged **Beam**. The classifier **Beam; Tension** is not the same as **Tension; Beam** and may not produce the same results.
- A Glossary index. This is an index relating glossary entries, as indicated with the

X-GLOSSARY= attribute, to the location of the actual glossary text with the data.

X-CLASSIFIER and **X-GLOSSARY** information is for use by the server only and does not directly form any part of a response to a client. Ideally it should be removed from the data files generated by the preprocessing.

8.2.3 Responses

Requested Section

The primary function of the server is to return requested sections (objects), or portions thereof, to a client. It is the clients responsibility to request the types of information it requires through the use of the **X-Extract:** request header. The server is not required to know the type of client (eg. browser, application etc.) and is only required to return the requested information.

The server, when returning the textual portion of an object, must include both a *Next* hypertext anchor and a *Previous* hypertext anchor. These must point to the next logical object for *Next* and the previous logical object for *Previous* as they would appear within the printed document. This would allow users to "page" forwards and backwards without having the original author of the standard embed these links in every object. These autogenerated anchors should appear as the first two lines of any returned textual object. If there is no *Next* or *Previous* an empty line "placeholder" must be used. This would then allow the client, if desired, to remove them or to interpret them and present them to the user in another manner such as dedicated buttons within a windowing environment.

The following are some examples of typical client requests and server responses for complete or partial objects of a standard. The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

Client Browser Requests a Complete Object from the Standard

Request from Client

GET http://standard/CAN3-S16.1-M/1984/13.1 HTTP/V1.0

X-Extract: Text/X-HTML+;X-Eqn/X-Tex;X-Table/X-HTML+;Image/Postscript

Text, X-Eqn, X-Table, and Image are used to insure complete section information is returned

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: Text/X-HTML+

<title>CAN3-S16.1-M/1983</title>

<body>

Next

Previous

<h2 x-section="13.1">13.1 General</h2>

<p>To meet the strength requirements of this Standard all factored resistances, as determined in this Clause, shall be greater than or equal to the effect of factored loads determined in accordance with Clause 7.2 and <embed type="text/x-tex"> ϕ </embed> shall be taken to be 0.90 unless otherwise specified.

<p><embed type="x-eqn/x-tex"> $\phi = 0.9$ </embed> unless otherwise noted.</body>

The first two lines of the HTML+ document contain the server generated Next and Previous anchors. In this example the standards file preprocessor has removed the X-CLASSIFIERS attribute from the "h2" tag

Client Application Requests Equations from an Object within the Standard

Request from Client

GET http://standard/CAN3-S16.1-M/1984/13.2 HTTP/V1.0

X-Extract: X-Eqn/X-Tex

X-Eqn is used to request only the equation portions

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: X-Eqn/X-TeX

<title>CAN3-S16.1-M/1983</title>

<body>

$T_r = \phi A_n F_y \text{ when } A_n / A_g \geq F_y / F_u$

$T_r = \phi \left(F_u \left(A_n / A_g \right) \right) A_n \text{ when } A_n / A_g < F_y / F_u$

$T_r = 0.75 \phi A_n F_y$

</body>

For this response the server has removed all the HTML+ envelope around the equations and has returned only the T_EX portions. Only those sections of the requested object with the attribute X-Eqn/X-TeX are returned

Client Application Requests Tables from an Object within the Standard

Request from Client

GET http://standard/CAN3-S16.1-M/1984/13.3.2 HTTP/V1.0

X-Extract: X-Table/X-HTML+

X-Table is used to request only the table portions

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: X-Table/X-HTML+

<title>CAN3-S16.1-M/1983</title>

<body>

<tbl border>

<tt top>Table 2 Values of <embed type=text/x-tex> $\sqrt{F_y / \pi^2 E}$ </embed> and <embed type=text/x-tex> $F_y / \pi^2 E$ </embed>

<th><embed type=text/x-tex> F_y MPa</embed>

<th><embed type=text/x-tex> $\sqrt{F_y / \pi^2 E}$ </embed>

<th><embed type=text/x-tex> $F_y / \pi^2 E$ </embed>

<td>230<td>0.0108<td>0.000 116<tr>

<td>260<td>0.0115<td>0.000 132<tr>

<td>300<td>0.0123<td>0.000 152<tr>

<td>350<td>0.0133<td>0.000 177<tr>

```
<td>380<td>0.0139<td>0.000 192<tr>
<td>400<td>0.0142<td>0.000 203<tr>
<td>480<td>0.0156<td>0.000 243<tr>
<td>700<td>0.0188<td>0.000 355<tr>
</tbl>
</body>
```

For this response the server has replied with only the table portion of the section requested. Since this is not a complete request Next and Previous anchors have not been included.

Hierarchical Object Lists

To provide the client application access to object identifiers and to allow for client implementation of expanding hierarchical indexes and user selection menus the Standards' Document Server protocol allows for two request forms. The first is the **X-TITLES/X-HTML+** which will return the headers and object identifiers for the hierarchical level immediately below the requested object and **X-ALLTITLES/X-HTML+** which returns all the headers and object identifiers below the requested object.

If a request is made with no object identified (eg **/standard/CAN3-S16.1-M/1984**) the server should assume that the client wishes either the first level, if the request is **X-TITLES**, or all levels, if **X-ALLTITLES** is requested. An **X-TITLES** request, if made with an object identifier that has no lower level objects will return the object itself. **X-ALLTITLES** however, will return an empty response for the same situation.

The following are some examples of client requests and server responses for titles and section numbers. The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

Client Application Requests the Next Level of Titles

Request from Client

```
GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0
X-Extract: X-Titles/X-HTML+
```

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: X-Titles/X-HTML+

<title>Table of Contents</title>

<body>

<h2 x-section="13.1">13.1 General</h2>

<h2 x-section="13.2">13.2 Axial Tension</h2>

<h2 x-section="13.3">13.3 Axial Compression</h2>

</body>

The response only contains the heading information for the level immediately below the requested level.

Client Application requests the all levels of titles

Request from Client

GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0

X-Extract: X-AllTitles/X-HTML+

Response from Server

100 200 Success

Allow: GET PUT

Public: GET

MIME-Version: 1.0

Content-Type: X-AllTitles/X-HTML+

<title>Table of Contents</title>

<body>

<h2 x-section="13.1">13.1 General</h2>

<h2 x-section="13.2">13.2 Axial Tension</h2>

<h2 x-section="13.3">13.3 Axial Compression</h2>

<h3 x-section="13.3.1">13.3.1</h3>

<h3 x-section="13.3.2">13.3.2</h3>

<h3 x-section="13.3.3">13.3.3</h3>

</body>

The response only contains the heading information for the level immediately below the requested level.

Access by Classifier(s)

Accessing information by classifier is a powerful tool for the user. It permits the user to select classifiers (keywords) and then have the pertinent standards objects obtained. It is suggested that client applications, such as browsers, first request a list of classifiers from the Standards' Server (see section 8.3.2). This list, which is returned in plain text, should be used to produce a selection list for the user. Once a user has selected one or more classifiers the client can then request the list of headings and object identifiers that pertain to the classifiers selected.

The following are examples of client requests and server responses for classifier lists and titles and section numbers based on classifiers. The information appearing in **teletype font** is the actual text of a request or a response. Text that appears in *italic font* are comments or clarifications. These examples are based on the sample HTML+ included in Appendix A.

Client Application requests list of classifiers for section 13.

Request from Client

```
GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0
X-Extract: X-Classifiers/Plain
```

Response from Server

```
100 200 Success
Allow: GET PUT
Public: GET

MIME-Version: 1.0
Content-Type: X-Classifiers/Plain

member
connection
resistance
general
axial-tension
axial-compression
w-shapes
HSS
cold-formed
nonstress-relieved
```



```
class-1
class-2
class-3
hot-formed
stress-relieved
singly-symmetric
channels
class-4
.
```

The server responds with a CRLF delimited list of the unique classifiers for section 13, and the levels below. The list may be returned in any order. For this example the list is in order of occurrence.

Client Application requests titles by classifiers

Request from Client

```
GET http://standard/CAN3-S16.1-M/1984/13. HTTP/V1.0
X-Classifiers: axial-compression; HSS; class-1
```

Response from Server

```
100 200 Success
Allow: GET PUT
Public: GET

MIME-Version: 1.0
Content-Type: Text/X-HTML+

<title>Table of Contents</title>
<body>
<h3 x-section="13.3.1">13.3.1</h3>
<h3 x-section="13.3.2">13.3.2</h3>
</body>
```

The response only contains the heading information which meets the classifiers listed.

8.2.4 Tolerance to Ill Formed Requests

If the root portion of the URL object identifier is not **standard** the server should not handle the request as a Standards' Document Server reference. If possible this request should be handled directly as an HTTP server.

The server should attempt to always respond with information when possible. If the request is not HTTP V1.0, but an earlier version, it will assume **X-Extract: Text/X-HTML+** as the request header information. If the request is a full URL and not identified as HTTP: the server should assume **Accept: Text/Plain** and return only the textual information of the object requested.

All blocks of information such as table entries, lines, or multiple equations will be terminated with a Crlf character pair. Clients and servers should recognize both the standalone Lf as well as the Crlf character pair to indicate termination of a block of information in keeping with the HTTP V1.0 definition. The detection of a Cr without a following Lf character should be considered an error. As with HTTP the use of the Crlf or Lf as delimitation allows for easy parsing of each block, or line, of information.

8.3 Client Considerations

8.3.1 Hypertext Markup Language (HTML+)

The client application, if it is a viewer (browser), should support the entire HTML+ tag and attribute set. In addition to these, all applications must support of the attributes **X-SECTION** and **X-LOOKUP**. Also the support of **/X-TEX** and initially **/POSTSCRIPT**, for viewers, is also required.

X-SECTION provides the client with the object identifier. This combined with the current URL allows the client to access that particular object from the server. The attribute **X-LOOKUP** is found in conjunction with an anchor tag. The client is to use the values associated with **X-LOOKUP** to perform a **X-CLASSIFIER:** request to the standard specified in the **HREF** of the same anchor.

Initial implementations of client applications must support **/X-TEX** representation. For viewers this will require a T_EX to viewer display format converter. For computing applications it will require a T_EX to executable form converter. Viewer clients must retain the knowledge of the initial format to allow for glossary lookups. The server will understand only the symbol format it originally sent (ie T_EX) and not the rendered format. It is hoped that other mathematical formats will be developed, such as "C" and "Fortran" for comput-

2 OF/DE 2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

ing applications, and that the server will be able to convert from one format to another at the request of the client.

The support of */POSTSCRIPT* is currently required for viewers as initial servers will only supply images in that format. It is hoped that other image formats will be developed and that the server will be able to convert from one format to another at the request of the client.

8.3.2 Viewers

Viewers, or browsers, should be implemented using a windowing environment with full graphics capabilities to provide complete and proper rendering of the standards document. The standards document utilizes embedded mathematical symbols, equations and graphical images (figures) as a normal parts of the document. Without proper rendering of these items information would be lost.

With the use of *X-TITLE/X-HTML+* and *X-ALLTITLES/HTML+* a viewer can provide the user with expanding hierarchical index interface to review or access portions of the document.

Each complete object request will contain a Next and Previous anchor as the first two lines of the response. It is recommended, for GUI viewers, that these two lines be removed by the viewer and the information be used for two integral buttons (Next and Previous) within the display.

List of Classifiers

A viewer should maintain a list of relevant classifiers based on the section of the standards document currently being accessed. A mechanism should be available to allow the user to formulate a request based on selecting one or more classifiers. The mechanism must also allow the user to "order" the selected classifiers. To minimize the amount of "traffic" between the client and the server actual *X-CLASSIFERS* requests should only be made when the user actually selects the list for use.

8.3.3 Applications

Applications are clients who's use of the standards information is not for display or rendering but for calculation, such as design programs. With the use of **X-CLASSIFIER**: an application can ask the server for the object identifiers and titles which match the request. With this information the application can then request from the server the equation portions, using **X-EXTRACT: X-EQN/X-TEX** and the table sections, if required, using **X-EXTRACT: X-TABLE/X-HTML+**. The application can then convert the equations and tables to a form which can be evaluated or interpret them directly.

Initial implementation of standards' servers will utilize **T_EX** for both mathematical symbols and formulae. This will require applications initially to provide either a **T_EX** evaluator, such as MathCad, or a **T_EX** converter. It is hoped that later servers will provide for other mathematical formats such as "C" and "Fortran" for applications, and that the server will be able to convert from one format to another to satisfy the request of the client.

8.3.4 Tolerance to Ill Formed Responses

All clients should ignore any tags or attributes it does not understand. These ignored tags or attributes should not be rendered but absorbed by the client before display.

Mis-matched tag/end tag combinations should also be ignored and not rendered. For example **<h2>Title</h3>** should result in the plain text presentation **Title** without the rendering associated with either **<h2>** or **<h3>**.

All blocks of information such as table entries, lines, or multiple equations will be terminated with a **CrLf** character pair. Clients and servers should recognize both the standalone **Lf** as well as the **CrLf** character pair to indicate termination of a block of information in keeping with the HTTP V1.0 definition. The detection of a **Cr** without a following **Lf** character should be considered an error. As with HTTP the use of the **CrLf** or **Lf** as delimitation allows for easy parsing of each block, or line, of information.

9

Conclusions

The Standards' Server provides a computerized method of allowing engineering applications to access the information contained within a design. The server utilizes queries that are natural and delivers information in a form that can be easily displayed or used for computation. It merges three technologies to provide convenient computer based interactive access to standards information. These three technologies are standards representation, network based computing utilizing the client-server model, and markup and presentation of computerized documents.

The markup and presentation of computerized documents for the standards' server utilizes the HyperText Markup Language (HTML+). This language provides semantic markup of documents and is rich in facilities to provide for hypertext links either within the document or to documents located elsewhere, either locally or across a network.

The standards' server uses the Internet HyperText Transfer Protocol (HTTP V1.0) to provide the network transportation backbone for the client-server model. HTTP allows for the exchange of information while allowing multiple information representations to be contained within a single transaction. It is thus capable of transferring audio and video for multi-media applications, though currently the standards' server does require these features.

Standards are hierarchical in nature and consist of short sections, each of which outline the design requirements for a particular aspect of the overall project. Unlike most documents, standards have discrete elements which make up a section. These elements are text, tables, figures, equations and definitions. A requirement of computerizing the representation of standards is that each of these elements must be obtainable separately as well as together in context with each other. A computerized standard must also be able to present subsets of itself that pertain to a particular design or concept. The extensions defined in this document to HTTP V1.0 and HTML+ allow for these features.

With Standards' Servers and client applications implemented on a network engineers and engineering applications will:

- have access to all the latest standards
- be able to instantly reference other associated documents
- be able to access and resolve elements associated with the standards
- be able to reference only the pertinent parts of standards for a particular problem
- be able to browse the standards

PAGINATION ERROR.

TEXT COMPLETE.

ERREUR DE PAGINATION.

LE TEXTE EST COMPLET.

NATIONAL LIBRARY OF CANADA.

CANADIAN THESES SERVICE.

BIBLIOTHEQUE NATIONALE DU CANADA.

SERVICE DES THESES CANADIENNES.

Bibliography

- [1] M.H. Ackroyd, S.J. Fenves, and W. McGuire. Computerized lrfd specification. June 1988.
- [2] Edward Barrett. *Text, ConText, and HyperText: Writing with and for the Computer*. MIT Press, Massachusetts Institute of Technology, 1988.
- [3] Tim Berners-Lee. Hypertext transfer protocol (working draft). July 1993.
- [4] Tim Berners-Lee. Uniform resource locators rfc xxxx (working draft). July 1993.
- [5] Tim Berners-Lee and Daniel Connolly. Hypertext markup language rfcxxxx (working draft). March 1993.
- [6] Martin Bryan. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Addison Wesley Publishing Company, 1988.
- [7] Douglas E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. Prentice-Hall Inc., 1988.
- [8] Douglas E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture (Second Edition)*. Prentice-Hall Inc., 1991.
- [9] Douglas E. Comer and David L. Stevens. *Internetworking with TCP/IP: Client Server Programming and Applications*. Prentice-Hall Inc., 1991.
- [10] S.M. Cornick. Hypercode: the building code as a hyperdocument. *Engineering with Computers*, 7(1), 1991.

- [11] S.M. Cornick, D.A. Leishman, and J.R. Thomas. Incorporating building regulation into design systems: an object-oriented approach. *ASHRAE Transactions 1990 Paper No. 1682*, 96(2), 1990.
- [12] David H. Crocker. Standard for the format of apra internet text messages rfc822. August 1982.
- [13] Jose R. Cronembold and Kincho H. Law. Automated processing of design standards. *Journal of Computing in Civil Engineering*, 2(3), July 1988.
- [14] J.R. Harris et al. *Technology for the formulation and expression of specifications*. Technical Report, University of Illinois, Urbana, Illinois, 1975. Vol. 1, 2, and 3. Department of Civil Engineering.
- [15] Sunil K. Evt, Sari Khayyal, and Victor E. Sanvido. Representing building product information using hypermedia. *Journal of Computing in Civil Engineering*, 6(1), January 1992.
- [16] S.J. Fenves. Tabular decision logic for structural design. In *Journal of the Structural Division ASCE 92(ST6)*, pages 473-490, December 1966.
- [17] S.J. Fenves, E.H. Gaylord, and S.K. Goel. *Decision Table Formulation of the AISC Specifications*. Technical Report, University of Illinois, Urbana, Illinois, August 1969. Civil Engineering Studies, Structural Research Series No. 347.
- [18] S.J. Fenves and S.K. Goel. *Computer Aided Processing of Structural Design Specifications*. Technical Report, University of Illinois, Urbana, Illinois, November 1969. Civil Engineering Studies, Structural Research Series No. 348.
- [19] S.J. Fenves and R.N. Wright. *The Representation and Use of Design Specifications*. Technical Report, National Bureau of Standards, 1977. Technical Note 940.
- [20] Mark Grand. Mime overview. August 1992.
- [21] Donald E. Knuth. *The T_EXbook*. Addison Wesley Publishing Company, 1984.

- [22] Ed Krol. *The Whole Internet Catalog & User's Guide*. O'Reilly & Associates, Inc., 1992.
- [23] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison Wesley Publishing Company, 1986.
- [24] Daniel C. Lynch and Marshall T. Rose. *Internet System Handbook*. Addison Wesley Publishing Company, 1993.
- [25] W. McGuire. Computers and steel design. June 1992.
- [26] Associate Committee on the National Building Code. *National Building Code of Canada*. "National Research Council of Canada", 1977.
- [27] David Raggett. Html+ (hypertext markup language). July 1993.
- [28] J.B. Sykes. *Concise Oxford Dictionary of current English*. Oxford at the Clarendon Press, 1982.
- [29] Chris Thewalt and David Moskowitz. Automated text generation for building standards. *Journal of Computing in Civil Engineering*, 4(1), January 1990.
- [30] D.J. Vanier. Computerized building regulations. In *Proceedings of the International Conference on Municipal Code Administration, Building Safety and the Computer*, pages 43-62, September 1989.
- [31] D.J. Vanier. A parsimonious classification system to extract project-specific building regulations. December 1992. PHD Thesis in Progress.
- [32] Various. *Handbook of Steel Construction*. Canadian Institute of Steel Construction, 1985.
- [33] Chris Weider. Resource transponders (working draft). March 1993.
- [34] Chris Weider and Peter Deutsch. Uniform resource names (working draft). May 1993.
- [35] Nobuyoshi Yabuki and Kincho H. Law. Hyperdocument model for design standards documentation. *Journal of Computing in Civil Engineering*, 7(2), April 1993.

A

Sample HTML+

The following is a sample of an HTML+ marked up document which can be interpreted and used by a Standards' Server. It is taken from "CAN3-S16.1-M84 Steel Structures for Buildings (Limit State Design)"[32].

Please note that the line numbers to the left of each line do not form part of the data but have been included for ease of reference.

```
1  <!DOCTYPE htmlplus>
2  <h1 x-section="13." x-classifiers="member; connection; resistance">13.
3  Member and Connection Resistance</h1> <p>

4  <h2 x-section="13.1" x-classifiers="general">13.1 General</h2>

5  <p>To meet the strength requirements of this Standard all factored
6  resistances, as determined in this Clause, shall be greater than or equal to
7  the effect of factored loads determined in accordance with <a
8  href="http://standard/CAN3-S16.1-M/1983/7.2"> Clause
9  7.2</a> and <embed type=text/x-tex>${\phi}$</embed> shall be taken to be 0.90
10 unless otherwise specified.

11 <p><embed type=x-eqn/x-tex><x-glossary type="text/x-tex">${\phi}$</x-glossary>
12 = 0.9$</embed> unless otherwise noted.

13 <h2 x-section="13.2" x-classifiers="axial-tension">13.2 Axial Tension</h2>

14 <p>The factored tensile resistance <embed type=text/x-tex>${T_r}$</embed>,
15 developed by a member subjected to an axial tension force shall be taken as
```

16 <p>(a) the lesser of

17 <p indent>(i)<embed type=x-eqn/x-tex>

18 $T_r = \phi A_n F_y$ when $A_n / A_g \geq F_y / F_u$ </embed>

19 <p indent><embed type=x-eqn/x-tex>

20 $T_r = \phi \left(F_u \left\{ \frac{A_n}{A_g} \right\} \right) A_n$ when $A_n / A_g < F_y /$

21 F_u </embed>

22 <p>(b) For pin connections, on net area

23 <p indent><embed type=x-eqn/x-tex> $T_r = 0.75 \phi A_n F_y$ </embed>

24 <h2 x-section="13.3" x-classifiers="axial-compression">13.3 Axial

25 Compression</h2> <p>

26 <h3 x-section="13.3.1" x-classifiers="w-shapes; HSS; cold-formed;

27 nonstress-relieved; class-1; class-2; class 3">13.3.1</h3>

28 <p>The factored axial compressive resistance for W shapes and for hollow

29 structural sections manufactured according to <a

30 href="http://www.civeng.carleton.ca/standard/CAN3-G40.20-M/latest/7.2"

31 x-lookup="cold-formed nonstress-relieved">CSA Standard CAN3-G40.20-M Class C

32 (cold formed nonstress relieved) and conforming to the requirements of <a

33 href="http://standard/CAN3-S16.1-M/1983/11.">Clause 11 for Class 1, 2

34 or 3 sections shall be taken as *:

35 <p>(a)<embed type=x-eqn/x-tex>

36 $\phi \leq \lambda \leq 0.15, C_r = \phi A F_y$ </embed>

37 <p>(b)<embed type=x-eqn/x-tex>

38 $0.15 < \lambda \leq 1.0, C_r = \phi A F_y (1.036 - 0.202 \lambda -$

39 $0.222 \lambda^2)$ </embed>

40 <p>(c)<embed type=x-eqn/x-tex>

41 $1.0 < \lambda \leq 2.0, C_r = \phi A F_y (-0.111 + 0.636 \lambda^{-1} -$
 42 $0.087 \lambda^{-2})$ </embed>

43 <p>(d)<embed type=x-eqn/x-tex>

44 $2.0 < \lambda \leq 3.6, C_r = \phi A F_y (0.009 + 0.887 \lambda^{-2}$
 45 $)$ </embed>

46 <p>(e)<embed type=x-eqn/x-tex>

47 $3.6 < \lambda, C_r = \phi A F_y \lambda^{-2} = \phi A \left[\frac{1,970,000}{(K L / r)^2} \right]$ </embed>

48 <p><embed type=x-eqn/x-tex>

50 $\text{where } \lambda = \left(\frac{K L}{r} \right) \sqrt{\frac{F_y}{\pi^2 E}}$ </embed>

51 <p id="13.2footnote"><em footnote>* These expressions are applicable to the W
 52 shapes normally rolled in Canada and can be assumed to be valid for other
 53 doubly symmetric Class 1, 2 or 3 sections, except for solid round nonstress
 54 relieved cold straighten bars greater than 50 mm diameter (refer to <a
 55 href="http://www.civeng.carleton.ca/standard/CANS37-M/latest">CSA Standard
 56 S37-M). Welded H shapes should have flange edges flame cut. Singly
 57 symmetric, asymmetric, or cruciform sections should be checked as to whether
 58 torsional-flexural buckling is critical
 59 <p>

60 <h3 x-section="13.3.2" x-classifiers="HSS; hot-formed; cold-formed;
 61 stress-relieved; class-1; class-2; class-3">13.3.2</h3>

62 <p>The factored axial compressive resistance for hollow structural sections,
 63 manufactured according to <a href=
 64 "http://www.civeng.carleton.ca/standard/CAN3-G40.20-M/latest"
 65 x-lookup="hot-formed cold-formed stress-relieved">CSA Standard
 66 CAN3-G40.20-M, Class H (hot formed or cold formed stress relieved) and
 67 conforming to the requirements of <a

68 href="http://standard/CAN3-S16.1-M/1983/11.">Clause 11 for Class 1, 2
69 or 3 sections shall be taken as:

70 <p>(a)<embed type=x-eqn/x-tex>

71
$$0 \leq \lambda \leq 0.15, \quad C_r = \phi A F_y$$
</embed>

72 <p>(b)<embed type=x-eqn/x-tex>

73
$$0.15 < \lambda \leq 1.2, \quad C_r = \phi A F_y (0.990 + 0.122 \lambda -$$

74
$$0.367 \lambda^2)$$
</embed>

75 <p>(c)<embed type=x-eqn/x-tex>

76
$$1.2 < \lambda \leq 1.8, \quad C_r = \phi A F_y (0.051 + 0.801 \lambda^{-2})$$

77
$$)$$
</embed>

78 <p>(d)<embed type=x-eqn/x-tex>

79
$$1.8 < \lambda \leq 2.8, \quad C_r = \phi A F_y (0.008 + 0.942 \lambda^{-2})$$

80
$$)$$
</embed>

81 <p>(e)<embed type=x-eqn/x-tex>

82
$$3.6 < \lambda, \quad C_r = \phi A F_y \lambda^{-2} = \phi A \left[\frac{1,970,000}{(K L / r)^2} \right]$$
</embed>

84 <p><embed type=x-eqn/x-tex>

85
$$\text{where } \lambda = \left(\frac{K L}{r} \right) \sqrt{\frac{F_y}{\pi^2 E}}$$
</embed>

86 <p>Values of $\sqrt{\frac{F_y}{\pi^2 E}}$ </embed> and

87 $\frac{F_y}{\pi^2 E}$ </embed> to compute <embed

88 type=text/x-tex> λ </embed> and <embed

89 type=text/x-tex> λ^2 </embed> respectively are give in Table 2.

90 <tbl border>

91 <tr><td><tt top>Table 2 Values of $\sqrt{\frac{F_y}{\pi^2 E}}$ </td></tr>

92 <td> $\frac{F_y}{\pi^2 E}$ </td></tr></tbl>

93 <th><embed type=text/x-tex>\$F_y\$ NPa\$</embed>
 94 <th><embed type=text/x-tex>\$\sqrt{{F_y \over \pi^2 E}}\$</embed>
 95 <th><embed type=text/x-tex>\${F_y \over \pi^2 E}\$</embed>
 96 <td>230<td>0.0108<td>0.000 116<tr>
 97 <td>260<td>0.0115<td>0.000 132<tr>
 98 <td>300<td>0.0123<td>0.000 152<tr>
 99 <td>350<td>0.0133<td>0.000 177<tr>
 100 <td>380<td>0.0139<td>0.000 192<tr>
 101 <td>400<td>0.0142<td>0.000 203<tr>
 102 <td>480<td>0.0150<td>0.000 243<tr>
 103 <td>700<td>0.0168<td>0.000 355<tr>
 104 </tbl>
 105 <p>

 106 <h3 x-section="13.3.3" x-classifiers="HSS; singly-symmetric; channels;
 107 class-4">13.3.3</h3>

 108 <p>The factored compressive resistance, <embed type=text/x-tex>\$C_r\$</embed>
 109 developed by a member subject to an axial compressive force and designated as
 110 a Class 4 section according to <a
 111 href="http://standard/CAN3-S16.1-M/1983/11.">Clause 11 shall be
 112 determined as follows:

 113 <p>(a) for sections less than or equal to 4.5 mm thick and which are not
 114 hollow structural sections, the compressive resistance shall be calculated
 115 in accordance with <a
 116 href="http://www.civeng.carleton.ca/standard/CAN3-S136/latest"> CSA Standard
 117 CAN3-S136;

 118 <p>(b) for hollow structural sections and sections greater than 4.5 mm thick,
 119 which can be shown not to be critical in torsional buckling or not subject to
 120 torsional-flexural buckling, the compressive resistance shall be calculated
 121 in accordance with the requirements of Clause <a href=
 122 "http://standard/CAN3-S16.1-M/1983/13.3.1"> 13.3.1 or <a

123 href="http://standard/CAN3-S16.1-M/1983/13.3.2">13.3.2, as applicable.

124 The area, <embed type=text/x-tex>A</embed>, shall be taken as the effective
125 area determined in accordance with <a

126 href="http://www.civeng.carleton.ca/standard/CAN3-S136/latest> CSA Standard
127 CAN3-S136. The slenderness ratio shall be calculated using gross
128 section properties; and

129 <p>(c) singly symmetric sections, which may be subject to torsional-flexural
130 buckling, and channels with unstiffened flanges shall be designed by rational
131 analysis.

END

1 0-1 1-94

FIN