VALIDATION AND VERIFICATION OF E-MAIL ADDRESSES

FINAL YEAR PROJECT

PRESENTED AS PART OF THE REQUIREMENT FOR AWARD WITHIN THE UNDERGRADUATE MODULAR SCHEME AT

GLOUCESTERSHIRE UNIVERSITY

BY

GLENN TURNER

Course:	Computing with Business Computer Systems
Module code:	CO303
Student number:	s9701050
Student e-mail address:	s9701050@glos.ac.uk
Project website:	http://final.glennturner.co.uk/
Supervised by:	Jon Wise
Start/finish dates:	October 2001 – 8th May 2002

Declaration

DECLARATION:-
This dissertation is the product of my own work. I agree that it may be made available for reference and photocopying at the discretion of the University.

Glenn Turner

Date:

Many thanks to Gyroscope.com in providing test data for this project

Abstract

E-mail addresses are crucial to the Internet community providing a quick, easy, cost effective means of contacting people to provide information or services. Along with the World Wide Web (WWW), e-mail is one of the key building blocks of the Internet, and arguably one of the reasons why the use of the Internet has grown so rapidly.

E-mailing provides the ability to market products or services to existing or potential customers at a low cost. This means a list of e-mail addresses is often regarded as a highly prized asset by website administrators. The amount of e-mail addresses in a company's database can even increase the company's value as it indicates the amount of potential customers it has. As a result many companies place a high degree of importance to the reliability (correctness) of gathered e-mail addresses.

The value of an e-mail address creates a need to check the 'correctness' of e-mail addresses entering into the company's databases. Periodic checking of e-mail addresses in a database would allow removal of addresses that are no longer in use. These unused addresses increase as people change their e-mail address, often by changing their internet service provider.

This document investigates a number of methods to find the 'correctness' of an e-mail address, discusses how successful the methods found are likely to be and how feasible it would be to use them for commercial purposes.

Contents

List of Figures

Introduction

The beginnings of e-mail

In late 1971 a computer engineer called Ray Tomlinson used the ARPANET to send the first e-mail between two Digital PDP-10 computer systems (Campbell, 1998). The e-mail was sent using a modified version of his own program named SNDMSG. SNDMSG had originally been designed to allow users to leave messages for each other on a single computer. It was subsequently modified to make use of a protocol which allowed files to be copied between machines. When SNDMSG was combined with this protocol it allowed the transmission of the first e-mail. The @ symbol was chosen to differentiate between messages for the local machine and outbound messages. Tomlinson said: "I used the @ sign to indicate that the user was 'at' some other host rather than being local." (compulit.uta.edu, nd)

It was never envisaged that e-mail would become so popular or used as widely as it is today. At the time Tomlinson sent the first e-mail only 15 computers were connected to the network and most people knew everyone using the system. E-mail is now in use worldwide by approximately 400 million people (software-aus.com.au, 2001) .

The problems of incorrect email addresses

If a telephone or fax number is missing a digit the person will realise this when they try to use it. Sending a test e-mail can often take days before it is returned with 'recipient unknown' or a similar error. For personal use this is just a inconvenience, but for commercial use, the accuracy of customer email addresses is an important factor. Should 1% of all customers incorrectly type their email address, this could potentially lead to a large loss of business.

E-mail addresses become redundant as customers change their ISP. Surveys have shown that 40% of people using e-mail change there address at least once a year (Marcus, 2001). Less than a third of those that change their e-mail address notify retailers or Internet services that rely on there e-mail address. The result is an increasing number of redundant e-mail addresses in company databases.

In the case of e-mail newsletters the target audience is reduced, and mail server performance suffers for every incorrect address. This is because the server will make multiable attempts to send the e-mail over a period of time.

As well as losing customers, incorrect e-mail addresses can also cost companies time and money trying to find the correct address, and may lead to even customer dissatisfaction.

For example; an order is placed online with a request for next day delivery, but with an incorrect e-mail address. The company discovers the item is out of stock and sends an e-mail to the customer notifying them of the problem (using the incorrect address).

The next day the customer phones the company trying to find out what has happened to the order.

Email validation & verification : a possible solution

In the past I have worked with e-mail systems such as sendmail and MS Exchange, and have been required to diagnose problems with remote mail servers. One of the techniques used to determine faults is to manually connect to the mail server using a number of mail protocols. As a result I have gained knowledge of the protocols and how the servers react. I also manage mailing lists for e-mail newsletters which can quickly fill with redundant addresses over a period of time. These need to be removed as the e-mails are returned but this is a time-consuming and tedious job. This lead me to wonder if it is possible to verify an address before sending the e-mail. I began to speculate on the type of application that would be required to handle such verification. I also realised that the resulting application could also be employed to verify e-mail addresses at the point of entry into a computer system. At the time of commencing this project I had been unable to find a company that was able to provide such a service.

Project Aim

To establish if it is feasible to produce a software program to validate and verify e-mail addresses. If this is the case a working program that uses the methodologies discovered will be implemented. The program will then be tested to check how effective it is likely to be in a commercial environment.

The difference between validation and verification is explained on page 14

Project Approach

The project will be spilt into two halves; validation and verification. The two types of checking require two different approaches. Validation of an e-mail address will check the syntax and requires detailed knowledge of the e-mail address structure. Verification requires the e-mail address to be checked against another source. I can see a use of validation in the verification program, for this reason I feel the validation section should be carried out first. Both validation and verification sections will have there own conclusions with an overall conclusion at the end of the project. Where ever possible e-mail standards will be consulted with particular attention paid to standards that are in common use.

Project Objectives

- Website showing project progress
- Abstract
- Introduction to subject
- Project Approach
- Validation verses Verification
- Define and document e-mail address structure
- Evaluate methods for validating e-mail
- Choose methods
- Implement validating software
- Obtain test data and test software
- Evaluate effectiveness of the validation
- Research DNS, MX RECORD, SMTP systems
- Methodology to verify e-mail addresses
- Produce software to verify e-mail addresses
- Obtain test data and test software
- Evaluate effectiveness of the verification
- Project Evaluation

Project Deliverables

- A full working prototype system for finding e-mail addresses within documents, that can validate and verify those addresses
- The project document
- Conclusion of the project (within the project document)
- Publish documents on project website (http://final.glennturner.co.uk)

Proposed project timetable

Pre October	2001	General research
4th October	2001	First Project meeting
Late October	2001	Project chosen & website up and running.
January	2002	Validation completed
February	2002	Verification theory documented
April	2002	Verification application working
8th May	2002	Project to be finished and documented

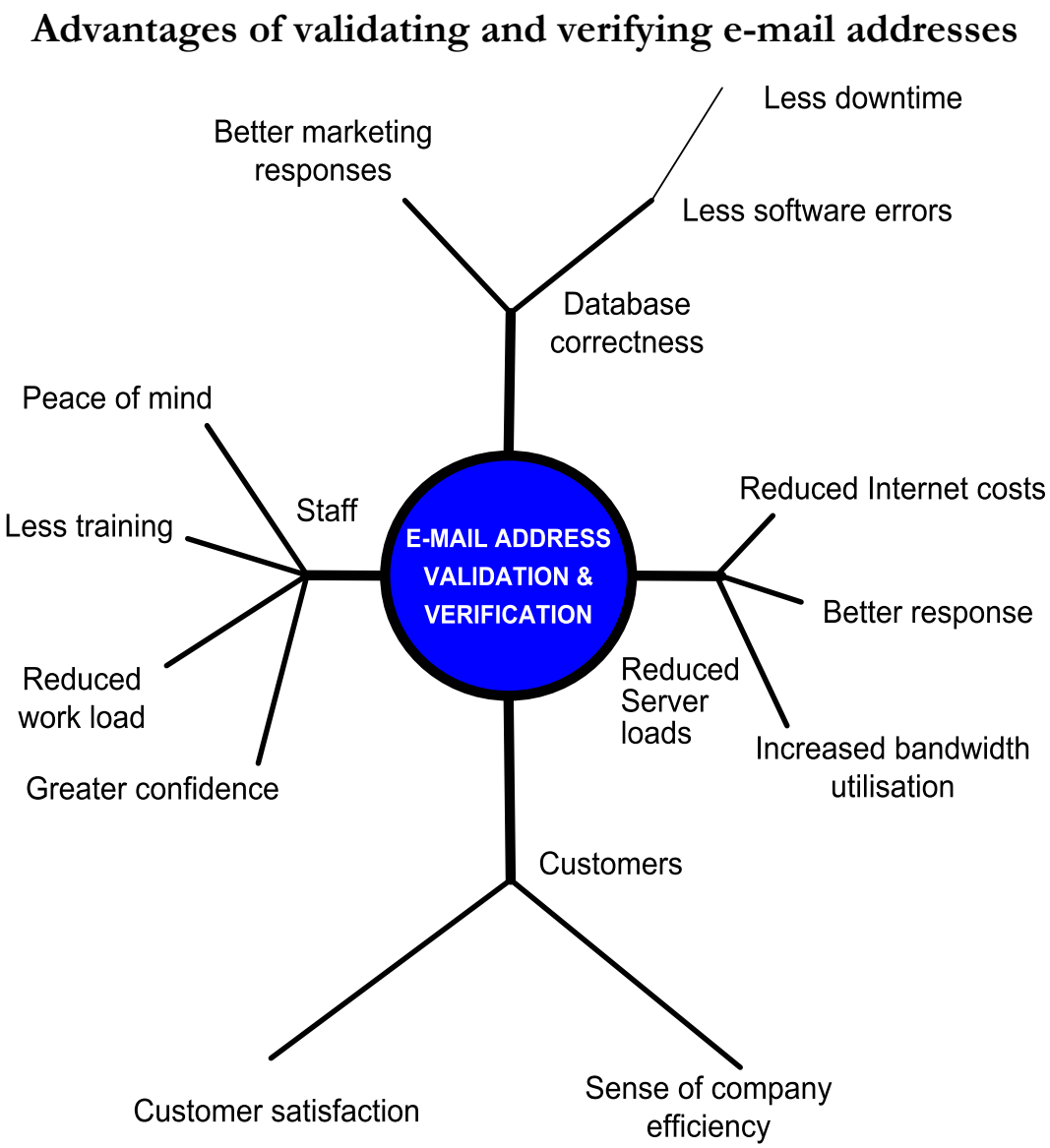

Figure 1: Advantages of validation and verification

Figure 1 shows a mind map on the advantages of validating and verifying e-mail addresses. It gives a diagrammatic overview of what could be achieved if the project is successful.

What is validation?

In computing terms data validation is testing to see if data complies with a set of defined characteristics. For example if a computer program takes the date of birth from a user it can validate the date of birth. This may be making sure it is numeric and not a letter or symbol. For example; If a user was born in 1960 and but types 196o by mistake (the letter o rather than a zero) then validation can pick this error up. But if the user typed 1860 instead of 1960 validation alone will not pick this error up. The validation process does not require extra user input to determine that the data is incorrect.

What is verification?

Data verification is to confirm that the data is correct or is likely to be correct. In terms of data input verification may include asking the operator/user to re-enter the data or confirm that it is correct. Techniques such as range checking on the data can be used ensure that the data is likely to be correct. Using the same scenario as the validation example, the user enters his/her date of birth into to the program. Range checks can be used to establish if the age is between a predefined range (e.g. 0 to 150). The user may also be asked to re-enter the data of birth in which case the computer compares the two, or the entered data is displayed on the screen and the user is asked to confirm if it is correct.

The most visible form of verification is re-entry of data or confirmation by the user but verification can take place without user intervention. For example when a file is copied between two storage drives verification is used to check consistency between the source and destination. The source data needn't be checked because file error checking can be used on the destination.

Quotes on validation and verification

"Data validation is the process of getting the computer to check to see if the data is valid (sensible in the context in which it is being used). "

(Bradley, 1995:423)

Figure 2: Validation definition

"Data validation, important though it is, can't detect all errors…Verification is a means of checking to see if the data being entered is likely to be correct"

(Bradley, 1995:423)

Figure 3: Verification definition

Why not just verify?

In the case of validating and verifying e-mails addresses verifying can be used on its own to test for correctness. However, because verifying tends to be more time consuming than validating it maybe quicker and more efficient (in terms of processing power) to validate first, before verify and having delays on anomalous addresses.

Validation

Approach to validating e-mail addresses

In order to validate e-mail addresses the structure of the e-mail address needs to be understood and have an explicit definition for common and possible usage. The Internet Engineering Task Force is the international body with the task of establishing open standards for the Internet. It does this through RFCs (Request For Comments) documents that are published on the Internet itself. Some of the RFCs are standards, with others used as advice and recommendations (Bradner, 1996). Using the latest RFC standards the approach is to break e-mail addresses down into manageable parts such as the local part and domain. If needed these will be further broken down. Once all these smaller parts are defined and understood they can then be constructed to form validation for whole e-mail addresses. Definition of the parts and overall validating will be finalised using regular expression conventions.

See Figure 33: Regular Expression Syntax

RFC2822

There are now thousands of RFCs on many topics regarding the Internet and surrounding technologies. RFCs can also be superseded by newer specifications. It is therefore important to select the appropriate RFC before using it to extract the e-mail address structure. There are a number of web search facilities specifically for searching RFCs. I used the search engine at http://www.rfc-editor.org/cgi-bin/rfcsearch.pl to search for any RFCs relating to e-mail systems. This gave a large number of RFCs in the

search results. RFCs are well referenced and I very quickly came across references to RFC822 (the basis of modern e-mail addressing) and RFC2822 which supersedes RFC822.

RFC2822 (Resnick, 2001) will be used to define the structure of the e-mail addresses, as it is the current standard which *"specifies a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages."*

Structure of an e-mail address

An e-mail address can be defined using the sections below. For common western languages (e.g. English/American) use, only the following symbols would be used:

@ Compulsory to delimit the user section from the rest of the e-mail address.

. When used on the right hand side of the "@" they delimit domains and their sub-domains .

_ Underscores are valid within user section (but not at beginnings or ends).

- Hyphens again valid in both domain and user sections (not at beginnings or ends).

a-z Valid in either upper or lower case.

0-9 Valid (but not valid at the start of some domains – depends on TLD).

( ) Brackets can be used as comments.

! Bangs can be used as a command to an e-mail server to forward the e-mail to another server. Some servers will ignore this character others will reject the e-mail. The e-mail can also be rejected if the 'other' e-mail server does not exist or is not accessible.

[ ] When used to the right of the "@" symbol brackets can be used to replace the entire domain with an IP address in the format [xxx.xxx.xxx.xxx] e.g. [192.168.0.1]

 # $ % & ' * + - = ? ^ _ ` { | } ~

These symbols can be used to the left of the "@" without any underlining functions or uses (current standards). They are simply treated as 'normal' characters.

Figure 4: Common structure of an e-mail address

Figure 5: Examples of working e-mail addresses

sales@store.gyroscope.com
sales@gyroscope.com
firstname.lastname@gyroscope.com

Figure 6: Examples of less common (but usable) addresses

“:-)” @gyroscope.com
!#$%&'*+-/=?^_`{|}~@gyroscope.com

Figure 7: Examples of addresses that are now becoming obsolete

sales@[213.171.193.18]
sales(comment)@gyroscope.com
sales(test)!gyroscope@gyroscope.com

```
Received: From pcow004o.blueyonder.co.uk [195.188.53.119] by
mailserver02.fasthosts.co.uk
   (Matrix SMTP Mail Server v(1.3)) ID=CDBAA517-829E-40F2-A592-
235046670B34 ; Mon, 29 Apr 2002 15:04:59 +0000
Received: from mail pickup service by blueyonder.co.uk with
Microsoft SMTPSVC;
       Mon, 29 Apr 2002 15:08:53 +0100
Content-Class: urn:content-classes:message
From: <glennturner@blueyonder.co.uk>
To: <":-)"@glennturner.co.uk>
Subject: test
Date: Mon, 29 Apr 2002 15:08:53 +0100
Message-ID: <5ee201c1ef87$64a51b50$7735bcc3@blueyonder.net>
MIME-Version: 1.0
Content-Type: text/plain;
      charset="us-ascii"
Content-Transfer-Encoding: 7bit
X-Mailer: Microsoft CDO for Windows 2000
Thread-Index: AcHvh2SlV6EOCltjEdaQvQCQJ9GOQA==
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2014.211
X-RCPT-TO: <":-)"@glennturner.co.uk>
```

Figure 8: Raw Message sent successfully to ":-)"@glennturner.co.uk

To prove these e-mail addresses work test messages were sent to them. Figure 8 Shows the raw message sent to ":-)"@glennturner.co.uk

User

The user section of an e-mail address is also know as the local part and is only used towards the end of the e-mail delivery process. Traditionally, the user section of the address would define a single user on a computer system. All users of the same domain would normally be physically local to the computer system. However, with the use of POP boxes and the ISP service model this is no longer the case. Users can 'dial in' from anywhere in the world to collect their e-mail. E-mails can also be forwarded or alias another e-mail address. So ***sales@gyroscope.com*** may be forwarded to ***glenn@gyroscopes.co.uk*** . This now makes identifying someone's geographic location from their e-mail address almost impossible.

Hostname (Often known as sub-domains)

The hostname is prepended onto the domain. Historically each sub-domain would have been another computer system (often a single machine). Today it is more likely to be created/used on the same machine as its 'parent domain' for a different website or service. Note: the one of the most used sub-domains is www which is a sub-domain in its own right.

Domain

The domain is a unique name containing sections that are separated with a single dot. Each section can have a maximum of 63 characters, with a the whole domain having a maximum length of 255 characters. The domain resolves to an IP address. Using the example "www.gyroscope.com" "com" is the TLD (Top Level Domain) with "gyroscope" being a sub domain of "com". "www" is a sub domain of "gyroscope.com". It has recently become common to call the first sub domain of the TLD "the domain" with any sub domains known as "sub domains".

TLD (Top Level Domain)

TLD's are the highest level of domain and the most significant. All countries have their own top level domain name (244 in total), plus there are other non-geographic domains such as com and org. Most TLDs are controlled by national organisations but a few are controlled by a multiple number of international bodies in cases such as com , net and org domains.

See Figure 32 for a full list of Top level domains

Case sensitivity of e-mail addresses

The growth of e-mail systems were popularised by UNIX based operating systems (Marshall, nd). E-mail systems inherited case sensitivity like that used on UNIX file systems. This sensitivity only applied to the user part of the address because everything after is governed by the DNS system which is not case sensitive. The case sensitivity on modern mail systems can be set independently of the operating system. So UNIX mail systems maybe case insensitive. It is in effect down to each server that is receiving and sending e-mails. Most systems at present are using case insensitivity. The latest SMTP RFC (RFC 2821) discourages case sensitive systems due to interoperability (Klensin, 2001).

Length of an e-mail address

In RFC1123 (section 6.1.3.5) its says:

> *"The DNS defines domain name syntax very generally – string of labels each containing up to 63 8-bit octets, separated by dots, and with a maximum total of 255 octets."*
>
> (RFC1123, 1989:79)

Interpreting the RFC 2822 pseudo code specification

The RFC2822 specification defines the structure of an e-mail address predominantly for SMTP but the structures it defines are used by other mail systems. Most e-mails sent over the Internet are at some point sent via SMTP for part of the delivery. Therefore the SMTP messaging standards are the most applicable standards to be used to define e-mail address validation The specification in RFC2822 has been written in a modular manner that allows easy understanding of elements but has to be 'expanded' in order for it to be converted to a programming language. The syntactic notation used in RFC2822 is written in Augmented Backus-Naur Form (ABNF) notation which is specified in RFC2234 (Crocker, 1997) . The following steps show how I got to the validation specification.

1. From 3.4.1. "Addr-spec specification" . This is the highest form of 'description' of an e-mail in the specification. It defines that a e-mail address has a "local-part", at the beginning, an "@" in the centre and the "domain" at the end.

```
addr-spec  = local-part "@" domain
```

2. RFC2822 has a specification for the "local-part" which is:

```
local-part = dot-atom / quoted-string / obs-local-part
```

Using this I made substitution to expand the definition. Hence the result:

```
(dot-atom / quoted-string / obs-local-part ) "@" domain
```

3. Using the "domain" definition:

```
domain = dot-atom / domain-literal / obs-domain
```

I made a similar substitution on the domain section. The result:

```
(dot-atom / quoted-string / obs-local-part ) "@" (dot-atom / domain-literal / obs-domain)
```

4. "obs-domain" expands to:

```
([CFWS] 1*atext [CFWS] *("." [CFWS] 1*atext [CFWS]))
```

which is obsolete so this can be removed (denoted by the "obs-"). Hence the definition:

```
(dot-atom / quoted-string / obs-local-part ) "@" (dot-atom / domain-literal)
```

5. "obs-local-part" is also obsolete so this can be removed. Hence:

```
(dot-atom / quoted-string) "@" (dot-atom / domain-literal)
```

6. "dot-atom" is substituted/expanded

```
(([CFWS] dot-atom-text [CFWS]) / quoted-string) "@" (([CFWS] dot-atom-text [CFWS]) / domain-literal)
```

7. "quoted-string" is substituted/expanded

```
(([CFWS] dot-atom-text [CFWS]) / ( [CFWS] DQUOTE *([FWS] qcontent) [FWS] DQUOTE [CFWS])) "@" (([CFWS] dot-atom-text [CFWS]) / domain-literal)
```

8. "domain-literal" is substituted/expanded

```
(([CFWS] dot-atom-text [CFWS]) / ( [CFWS] DQUOTE *([FWS] qcontent) [FWS] DQUOTE [CFWS])) "@" (([CFWS] dot-atom-text [CFWS]) / [CFWS] "[" *([FWS] dcontent) [FWS] "]" [CFWS] )
```

9. White spaces are removed and any unnecessary brackets.

```
(dot-atom-text / (DQUOTE *(qcontent) DQUOTE)) "@" dot-atom-text /
"[" *(dcontent) "]")
```

10. dot-atom-text is substituted

```
((1*atext *("." 1*atext)) / (DQUOTE *(qcontent) DQUOTE)) "@" (
1*atext *("." 1*atext) ) / "[" *(dcontent) "]")
```

11. qcontent is substituted

```
((1*atext *("." 1*atext)) / (DQUOTE *(qtext / quoted-pair) DQUOTE))
"@" ( 1*atext *("." 1*atext) ) / "[" *(dcontent) "]")
```

12. dcontent is substituted

```
((1*atext *("." 1*atext)) / (DQUOTE *(qtext / quoted-pair) DQUOTE))
"@" ( 1*atext *("." 1*atext) ) / "[" *(dtext / quoted-pair) "]")
```

13. quoted-pair = ("\" text) / obs-qp
obs-qp is now obsolete so quoted-pair is substituted with ("\" text)

```
((1*atext *("." 1*atext)) / (DQUOTE *(qtext / ("\" text)) DQUOTE))
"@" ( 1*atext *("." 1*atext) ) / "[" *(dtext / ("\" text)) "]")
```

14. Changes to the domain-literal addressing

See "Limitations and scope of the validation" for the reasons why.

[0-255.0-255.0-255.0-255]

Hence:

```
((1*atext *("." 1*atext)) / (DQUOTE *(qtext / ("\" text)) DQUOTE))
"@" ( 1*atext *("." 1*atext) ) / ("[" 0-255 "." 0-255 "." 0-255 "."
0-255 "]"))
```

15. The following are keys for above definition

```
DQUOT        =   %d34

Atext        =   ALPHA / DIGIT / ; Any character except controls,
                 "!" / "#" /     ;  SP, and specials.
                 "$" / "%" /     ;  Used for atoms
                 "&" / "'" /
                 "*" / "+" /
                 "-" / "/" /
                 "=" / "?" /
                 "^" / "_" /
                 "`" / "{" /
                 "|" / "}" /
                 "~"

qtext        =   NO-WS-CTL /     ; Non white space controls
                 %d33 /          ; The rest of the US-ASCII
                 %d35-91 /       ;  characters not including "\"
                 %d93-126        ;  or the quote character

dtext        =   NO-WS-CTL /     ; Non white space controls
                 %d33-90 /       ; The rest of the US-ASCII
                 %d94-126        ;  characters not including "[",
                                 ;  "]", or "\"

NO-WS-CTL    =   %d1-8 /         ; US-ASCII control characters
                 %d11 /          ;  that do not include the
                 %d12 /          ;  carriage return, line feed,
                 %d14-31 /       ;  and white space characters
                 %d127

text         =   %d1-9 /         ; Characters excluding CR and LF
                 %d11 /
                 %d12 /
                 %d14-127 /
                 obs-text
```

Regular Expressions

The ABNF format used in the RFC is descriptive but needs to be converted into something that a computer can understand. An interpreter could be written for ABNF but this would be a significant amount of work and beyond the scope of this project. The ABNF to validate an e-mail address could be converted directly to code but the amount of code would be considerable. Instead I have chosen to implement the ABNF into regular expressions which can be executed in an application. Regular expressions have been designed to manipulate and validate string data. One of the main reasons why

I have chosen to use regular expressions is that it is portable between many languages and is not dependent on any one operating system (Friedl, 1998).

"At a low level, a regular expression describes a chunk of text. You might use it to verify a user's input, or perhaps to sift through large amounts of data. On a higher level, regular expressions allow you to master your data. Control it. Put it to work for you. To master regular expressions is to master your data."

(Friedl, 1999:XV)

Regular expressions were invented by Professor Stephen Kleene in the mid 1950s to manipulate "regular sets". UNIX operating systems use regular expressions widely to manipulate and validate string data (Howe, 1997) and have become popular for web page scripting.

Converting ABNF (Augmented Backus-Naur Form) notation into Regular Expressions

Using the RFC2822 ABNF I have define the parts of the e-mail address structure and combined the whole structure. In this next section the ABNF is ported to regular expressions it needs to be converted in to something a computer can understand.

ABNF

```
[" 0-255 "." 0-255 "." 0-255 "." 0-255 "]"
```

Regular expression

```
IPNo="(([1-2][0-4][\d])|([1-2][5][0-5])|([1-9][0-8])|([\d]))"
RegIPaddr = "\[" & IPNo & "(\." & IPNo & "){3}\]"
```

ABNF

```
((1*atext *("." 1*atext))
```

Regular expression

```
([\w!#$%&*+-/=?^_`{|}~'])+([\.]([\w!#$%&*+-/=?^_`{|}~'])+)*
```

ABNF

```
(DQUOTE *(qtext / ("\" text)
```

Regular expression

```
([\x22]([\w]|([\\\][\011\014\127\012\001\002\003\004\005\006\007\008
\009]))*[\x22])
```

The regular expression that handles the domain i.e.

```
([\w!#$%&*+-/=?^_`{|}~'])+([\.]([\w!#$%&*+-/=?^_`{|}~'])+)*
```

does not take into consideration of the domain name system (Mockapetris, 1983). The expression can be improved by adding checking for that the domain structure is valid. It is known that all

"Note that while upper and lower case letters are allowed in domain names no significance is attached to the case. That is, two names with the same spelling but different case are to be treated as if identical.

The labels must follow the rules for ARPANET host names. They must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. There are also some restrictions on the length. Labels must be 63 characters or less."

(RFC883, 1983:56)

The updated regular expression

```
((([a-zA-Z0-9\-]{1,62})+[\.])+[a-zA-Z0-9\-]*)
```

The result is as follows (regular expression):

```
IPNo  ="(([1-2][0-4][\d])|([1-2][5][0-5])|([1-9][0-8])|([\d]))"
RegIPaddr = "\[" & IPNo & "(\." & IPNo & "){3}\]"


RegLocal =
"([\w!#$%&*+-/=?^_`{|}~'])+([\.]([\w!#$%&*+-/=?^_`{|}~'])+)*"

RegQstring =
"([\""]([\w]|([\\\][\011\014-
\127\012\001\002\003\004\005\006\007\008\009]))*[\""])"

RegDomain = "((([\w][\w\d\-]{1,62})+[\.])+[\w\_\-]*)"

emailRegExp =
"((" & RegLocal & "|" & RegQstring & ")\@
(" & RegDomain & "|" & RegIPaddr & "))"
```

Implementation of the program

With the regular expression 'containing' most of the validation work a language needed to be chosen to implement the application. Only a small amount of code needs to be created so I did not feel the validation application needed to use the same language as the verification application. I wanted a language that could quickly implement the regular expression and could be used on the project website. I chose ASPs (Active Server Pages) because they allow the validation program to be used on the project web site.

Limitations and scope of the validation

Although using regular expressions has increased the portability of the program to allow validation in ASPs (VBscript), Perl, Shell scripts, PHP, Visual Basic, Python, JSP, Javascript, Java and even Delphi (with the aid of a 3rd party module). Many of the above languages do not support recursive calls in regular expressions which is needed in order to meet the full RFC standards (RFC 2822). Recursive calls can be included into the regular expressions but at the time of writing support could only be found in Python and newer versions of Perl and Shell scripts.

The following standards are missing from the validation as implemented:

- **Comments in e-mail addresses**

  RFC 2822 defines how comments can be added to the local-part of the e-mail address. A comment starts with "(" and ends with ")" and allows a number of characters to be placed into the comment. An unlimited number of comments are allowed and comments can be placed within comments providing the corresponding correct number of open and close brackets are used. This type of recursive checking cannot be carried out solely in all versions of regular expressions. A decision was made to leave this because of that reason.

- **RFC 2822 4.4. Obsolete Addressing**

  Section 4.4 of RFC 2822 deals with obsolete addressing. It is worth noting that there are a couple of rules that are now obsolete that apply to a single e-mail address. In the past addresses were allowed to have a "route portion" before being enclosed in "<" and ">". This was used to route e-mail via a number of servers. This is now ignored. Carriage returns with spaces were allowed between the periods in both the local-part and domain. I chose not to implement these as support for these features are being phased out.

- **Some domain-literal addressing missing "[something]"**

  The domain-literal was finally expanded to

  ```
  "[" *(dtext / ("\" text)) "]"
  ```

  which allows a wide range of characters to be placed in the brackets that the mail server needs to interpret. Looking at the "4.1.3 Address Literals" section in RFC 2821 the contents within the brackets have been left open either for local interpretation (e.g. a CNAME in the DNS) or as an IP address or for a future

use of the IPv6 addresses. RFC822 (Crocker, 1982) gives a better explanation and goes on to strongly discourage use of the domain literals. Using domain-literals effectively undermines the DNS by simply not using the DNS system.

"Note:
THE USE OF DOMAIN-LITERALS IS STRONGLY DISCOURAGED. It is permitted only as a means of bypassing temporary system limitations, such as name tables which are not complete."

(RFC822, 1982:30)

Because of this I have decided to remove most of the flexibility from domain-literal specification by only allowing an IPv4 address format

- **IPv6**

  The current domain-literal specification would allow the use of the IPv6 address format. I chose to only implement IPv4 because the IPv6 roll out is still in it early stages.

  The website at http://www.ipv6.org/ maintains a list of servers connected to the Internet (all be it possibly not a complete list) (ipv6.org, 2001)

Optional Top Level Domain Checking

The current regular expression should work very well but I have thought of another technique to reduce the number of incorrect e-mail addresses. All the TLDs are known so a check can be created to validate the TLD against a known list. Figure 9 shows the complied list of TLDs in regular expression format. I have included this check as an option in the final validation application. I've done this because more TLDs are currently being added to the domain name system. When the new TLDs become available they will need to be added to the expression, which may cause administrative problems.

It should also be noted that TLD checking maybe conceived as validation or verification. I personally consider it as a borderline case.

Figure 9 was complied from the following reputable sources

The World Wide Alliance of Top Level Domain-names list of county domains
http://www.wwtld.org/member_list/countrycodesort0917.php

A RFC on "Domain Name System Structure and Delegation"
http://www.isi.edu/in-notes/rfc1591.txt

ICANNs webpage on "SEVEN NEW TLD PROPOSALS SELECTED FOR INTRODUCTION" http://www.icann.org/tlds/

```
(ac|ad|ae|aero|af|ag|ai|al|am|an|ao|aq|ar|arpa|as|at|au|aw|az|ba|bb|
bd|be|bf|bg|bh|bi|biz|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|c
g|ch|ci|ck|cl|cm|cn|co|com|cr|cu|cv|cx|cy|cz|de|dj|dk|dm|do|dz|ec|ed
u|ee|eg|eh|er|es|et|fi|fj|fk|fm|fo|fr|fx|ga|gb|gd|ge|gf|gh|gi|gl|gm|
gov|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|il|in|info|in
t|io|iq|ir|is|it|jm|jo|jp|ke|kg|kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|
li|lk|lr|ls|lt|lu|lv|ly|ma|mc|md|mg|mh|mil|mk|ml|mm|mn|mo|mp|mq|mr|m
s|mt|mu|museum|mv|mw|mx|my|mz|na|name|nc|ne|net|nf|ng|ni|nl|no|np|nr
|nu|nz|om|org|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|pro|pt|pw|py|qa|re|ro|ru
|rw|sa|sb|sc|sd|se|sg|sh|si|sj|sk|sl|sm|sn|so|sr|st|sv|sy|sz|tc|td|t
f|tg|th|tj|tk|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|um|us|uy|uz|va|vc|
ve|vg|vi|vn|vu|wf|ws|ye|yt|yu|za|zm|zw)
```

Figure 9: TLD checking in regular expression format

Validation Testing

The e-mail address	Description of test	Expect Result
user@test.com	Simple address	Valid
Firstname.surname@test.com	Address with dot somewhere before @	Valid
sales@test.co.uk	Address with two sub domains	Valid
sales@thisis.test.co.uk	Address with three sub domains	Valid
-@test.co.uk	Address with hyphen in user	Valid
user@te-st.co.uk	Address with hyphen in domain	Valid
Us_er@test.co.uk	Address with underscore in user	Valid
sales..@test.co.uk	Address with two dots in user	Valid
“test”@test.co.uk	Address with user in quotes	Valid
test@[1.1.1.1]	Address with IP address	Valid
“:-)”@glennturner.co.uk	Address with a quoted smiley	Valid
!#$%&'*+-/=?^_`{|}~@glennturner.co.uk	Address with a number of symbols for user	Valid
s9701050@[193.61.84.34]	Address with IP address after @	Valid

Figure 10: A selection of valid test data used

The e-mail address	Description of test	Expect Result
@test.com	Without anything before @	Invalid
This.user@	Without anything after @	Invalid
.@test.com	Address with just dot before @	Invalid
Name.@test.com	Address with letters, then dot, then @	Invalid
sales@com	Address with no sub domain	Invalid
User@te_st.co.uk	Address with underscore in domain	Invalid
test@test[1.1.1.1]	Address with confused IP address	Invalid
s9701050@[256.61.84.34]	Address with confused IP address	Invalid

Figure 11: A selection of invalid test data used

Validation Synopsis

I would classify the validation program as a success. It does everything I have set out to do. The RFC was more complicated than I had expected because there are many features an e-mail address can have which are rarely used, such as quotations. A few parts of the RFC have been missed such as the ability to handle Comments in e-mail addresses. Whenever such parts have been missing they have been documented so they can be included at a later date, if required.

The program has succeeded in successfully recognising valid and invalid test data that was used to check the syntax in the application. One surprising result was the ability of the validation application to pick out an e-mail address from surrounding text. For example, if an e-mail was placed in a sentence the validation application can highlight it.

Implementing such validation when an e-mail address enters into a computer system will reduce the amount of incorrect addresses in the system. This clearly is beneficial. The ability to pick out address has been used before but until now I have been unaware of its use. MS Word, Outlook and Excel (2002 versions) all use automatic hyper-linking. When an e-mail address is typed into a document it is automatically recognised and turned into a hyperlink.

Verification

The methodology to verify e-mail addresses

The most effective way an e-mail address can be verified is to check it against the server that administrates the e-mail address (the mail server). The first task is to establish a way to find the mail server from the e-mail address. This should be quite easy to do through the DNS system as the mail servers themselves do this on a regular basis to contact one another. Once the address of the mail server is found, contact can be made with the server.

The easiest publicly accessible way to communicate with the mail server is via SMTP. To find out if this protocol can be used to verify an e-mail address, the protocol needs to be checked to see if it is capable of carrying out this function. If it is capable of doing this, the relevant information must be extracted from the standards to form the basis of the verification. The theory can then be tested by creating a prototype application although I may create simple applications throughout the development to test certain aspects so to avoid complications with others parts of the software.

Outline methodology

- Chose application platform
- Find a way to obtain the mail server from the e-mail address
- Check that the SMTP standards allow verification
- Extract information from standards to allow creation of a prototype application
- Build verification application
- Chose test data
- Run tests
- Discuss results

Platform Decision

The platform which I will use to develop and test the application needs to be known prior to designing the software, so the limitations can be worked around and be able to utilize its features. The following criteria are those that have been chosen because of importance.

- **Portability + Availability**

  For my own personnel use I intend to leave my options open to allow a move to Linux platform at a future date. It would be useful to choose a language that will work across Windows and Linux systems. If I chose to release the final application as freeware then this would also maximise the amount of people that could use the software by releasing the software on both platforms at once.

- **Productivity**

  While creating the application it will be beneficial to resolve problems with the design of the application and protocols rather than struggling with programming language. In order for the time scales to be met a platform must be chosen that allows good productivity when creating an application that will interact with the Internet. A language with good Internet components is therefore a high priority.

- **Performance**

  The final verification application could verify one e-mail address at a time, however I do foresee a need use to check entire lists of addresses or even a database. In this case speed will be an issue so an application that can handle more than one address being verified at a time will reduce the wait dramatically.

Shortlist of Systems with advantages and disadvantages

- **Visual Basic**

  I've had I a lot of previous experience with Visual Basic. I feel confident with the language and it has some great Internet components that would simplify creating the application(s). From experience the applications produced are slower than its rivals from Borland because Visual Basic applications are semi-complied. This often requires a number of DLLs to be carried around with them. Visual Basic applications are non-portable, only running on resent Windows operating systems.

- **Borland C++ builder**

  C++ builder has now become portable between Windows and Linux systems. An application can be created on either system and converted to the other by simply copying across the source files and re-compiling. The registry is one of the few areas that can cause problems when converting. C++ builder is fast and has many components to aid development of Internet applications. Unfortunately I don't have any experience of C++ builder despite using a few variations of C++. I'm always happy about learning a new language but because of the time restrictions and the fact of the Delphi/Kylix option this came a very close runner up. Regular expressions are not included in the libraries.

- **ASPs (Active Server Pages)**

  After just spending a year working with ASPs (VB script) I gave some consideration to using them for the verification application. They are good for working with databases, web pages and sending e-mails, but lack many of the Internet components compared to Visual Basic, Delphi/Kylix or C++ builder. In particular they can't connect to a customised socket, fetch other web pages or make a telnet connection without obtaining 3rd party add-ins or building your own components. Spawning other threads is also quite complex and unreliable.

  It would be easy for people to use over the web but few would be able to get it running locally unless they had there own web server running. Basic regular expressions are supported. Screen ouput is provided through web pages e.g. HTML, XML.

- **Perl**

  Very portable across a wide range of systems including Windows and Linux.

  There are many examples scripts available for this language with a bias to the Internet. It is also a fast interpreted language and has very good support for Internet applications, which include being able to connect using sockets, telnet and sending e-mails. Cross-platform versions tend not to have GUI interfaces. Regular expressions are supported.

- **Borland Delphi/Kylix**

  Pascal was the first language that I leant and is one of my favorite languages. Delphi implements an object oriented version of Pascal, with Kylix simply being a Linux version of Delphi. Delphi and C++ builder share the same complier which it is quite unique in being able to compile two different languages. The Delphi IDE is nearly identical to C++ builders and it shares the ability to be fast and have great components to aid development of Internet applications. Unfortunately neither Delphi/Kylix nor C++ builder have regular expression built in.

- **Java**

  Like Perl Java is very portable, however in the latest versions of Windows and IE support has been removed. The java engine now needs to be downloaded and installed before java can be used on a windows system. Like Perl it has good socket support and general Internet components. A GUI development environment would probably need to be chosen to aid the development speed. Regular expressions are supported.

ASPs and Perl were ruled out because most people are unlikely to be able to run them locally. Visual Basic was discarded because of a lack of portability leaving Java, C++ builder and Delphi/Kylix. C++ builder and Delphi/Kylix are identical apart from the language and because I prefer Pascal over C++ I chose Delphi. Leaving Java or Delphi which was a tough choice but I finally chose Delphi because of the built in GUI developer. A search on the web found a number of 3rd party regular expression components for Delphi/Kylix. Without these it would have been a great disadvantage.

Finding the mail server

The first task is to find the mail server from the e-mail address. Anything after the "@" is the domain. A DNS query can be created to find the IP address of the mail server(s) for that particular domain. The mail server maybe the same server as the DNS server or it may be in a different country with no particular relation to the DNS server apart from the domain itself.

Using the web tools at http://www.demon.net/external/ any domain can be queried and the results viewed on a web page. The figure below shows the MX record results (mail exchangers) for the domain "hotmail.com" (the mail servers are on the right of each line). There maybe more than one machine accepting e-mail for any given e-mail address. But they always form a 'system' and an e-mail could go to any one of them. In the case of "hotmail.com" e-mail accounts there are 14 mail servers.

```
hotmail.com mail is handled (pri=5) by mx07.hotmail.com
hotmail.com mail is handled (pri=5) by mx08.hotmail.com
hotmail.com mail is handled (pri=5) by mx09.hotmail.com
hotmail.com mail is handled (pri=5) by mx10.hotmail.com
hotmail.com mail is handled (pri=5) by mx11.hotmail.com
hotmail.com mail is handled (pri=5) by mx12.hotmail.com
hotmail.com mail is handled (pri=5) by mx13.hotmail.com
hotmail.com mail is handled (pri=5) by mx14.hotmail.com
hotmail.com mail is handled (pri=5) by mx15.hotmail.com
hotmail.com mail is handled (pri=5) by mx01.hotmail.com
hotmail.com mail is handled (pri=5) by mx02.hotmail.com
hotmail.com mail is handled (pri=5) by mx04.hotmail.com
hotmail.com mail is handled (pri=5) by mx05.hotmail.com
hotmail.com mail is handled (pri=5) by mx06.hotmail.com
```

Figure 12: DNS MX results for "hotmail.com"

Notice that the list is in a sequence but does start at any particular number. The hotmail DNS in fact changes the server that is at the top for every query. It makes sense that the hotmail DNS query results are rotated to ensure the load is evenly balanced. Normally

another mail system wanting to commutate with the hotmail mail servers would pick a mail server with the highest priority rating. If two or more servers with the same priority are listed the first server is used. In Figure 12 the "mx07.hotmail.com" server would be used.

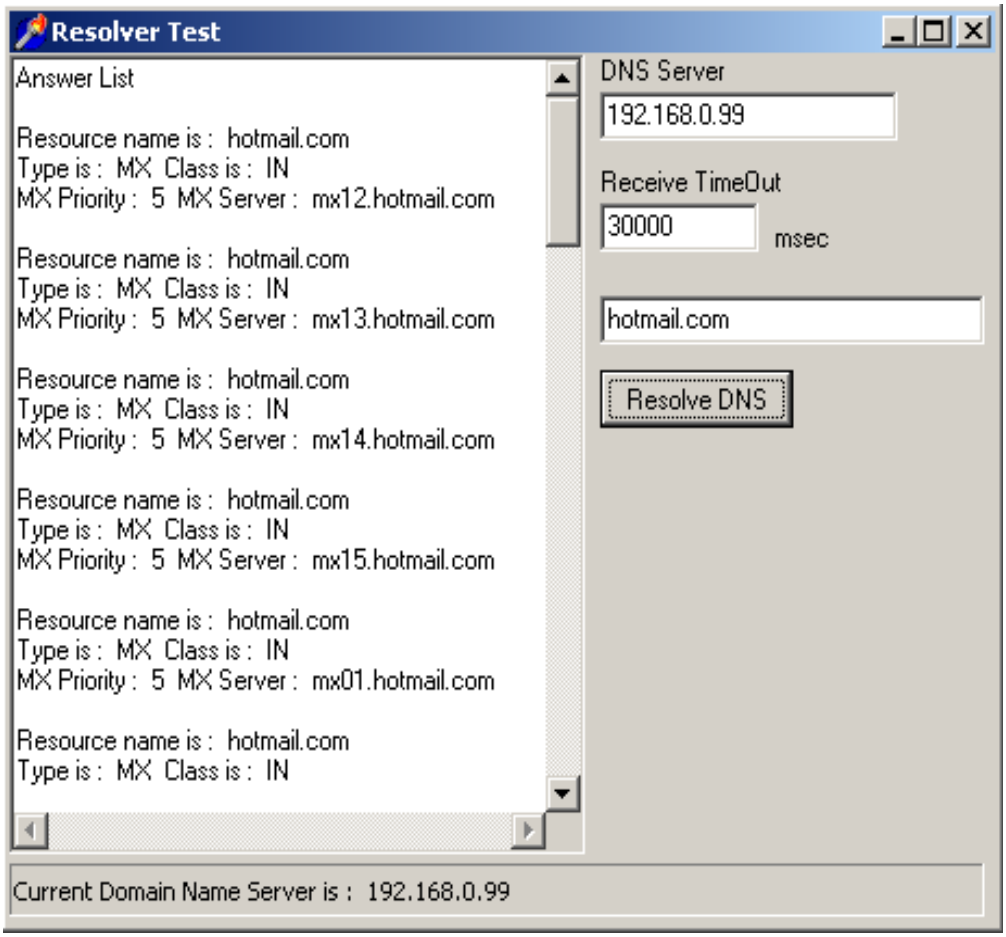

Figure 13: Application to test lookup of MX records

A test program to resolve the MX records

Using Borland Delphi 6 a simple test application was written to gain familiarity of the Borland DNS components and make sure that the MX resolver part of the final verification application could be integrated smoothly. Using the Borland "TIdDNSResolver" Internet component finding the MX records was simple, only requiring minimal programming. Figure 13 shows the application running with the results (left side), chosen domain, maximum timeout and the chosen DNS server to perform the query. In this case the local DNS server was "192.168.0.99".

The application was then improved by creating a single object that fetched the MX servers and reported this back as a string. This removed any unnecessary output and allowed the object to be easily used for any future applications.

Regular Expressions in Delphi

The usefulness of regular expressions particularly the ability to find e-mail addresses in text without much programming effort was made apparent with the validation application. Delphi does not support regular expressions without the use of 3rd party tools. So before continuing much further I wanted to find a solution to this problem. Andrey Sorokin of Russia has developed a free component for Delphi that provides regular expressions (Sorokin, 2001). The code itself has been ported from Henry Spencer's C source into Object Pascal. The speed of the regular expressions are particularly impressive. As with the Borland DNS components a small test application was created to familiarise myself.

Introduction to SMTP communication

SMTP communication can be used across the Intranet, or in a LAN or WAN environment utilizing a TCP/IP based protocol. During an SMTP communication process two systems talk to one another, the sender is known as the 'SMTP client' and receiver known as the 'SMTP server'. Delivery of messages can be made to a number of addresses but communication can only occur between two systems for each connection. SMTP servers allow multiple simultaneous connections to increase the availability to accept and send mail. Multiple messages from a single server destined for, or via, a particular server would normally be sent through a single connection in sequence. Mail can be passed to its final destination via a relay or gateway which separates networks or domains. Most mail servers can also act as mail forwarders but normally this is disabled when not required.

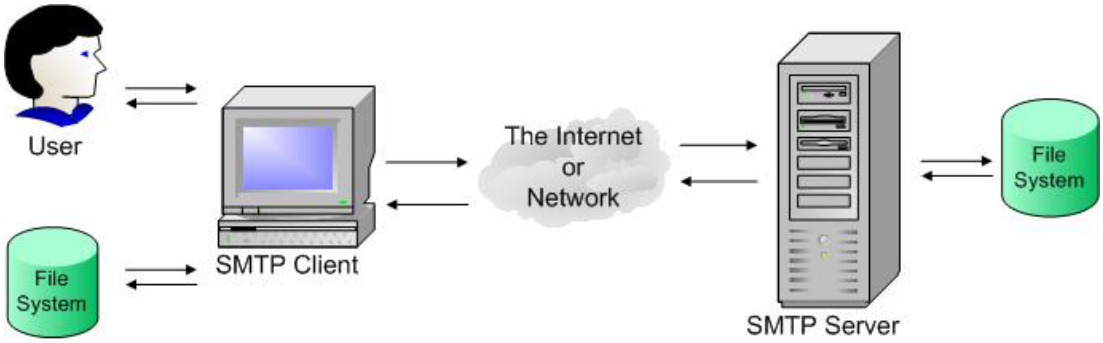

Figure 14: SMTP common usage

The common e-mail client

Assuming the most common set-up of Internet access connection is via either ISDN, modem, xDSL or Cable modem. A dynamic IP address is given for use while connected to the Internet from a 'pool' of addresses. This means that the IP address will change after a relatively short period of time or between 'dial-ins'. If an e-mail is sent with the reply address resolving to a dynamic IP address there is a very high chance that the IP

address will be different when the recipient replies to the e-mail. As a result it is impractical to receive or send e-mail directly from an Internet account without having a static IP address or 3rd party e-mail service.

There is one exception to this problem

The Dynamic DNS service allows you to alias a dynamic IP address to a static hostname, allowing your computer to be more easily accessed from various locations on the Internet.

It is envisaged that the verification application can communicate with the SMTP server via a telnet connection, which could be easily achieved using a component provided by Delphi. However the commands of the SMTP/telnet connection and the sequence they should be in are yet to be discovered.

The SMTP standard

RFC 2821 is the current standard for SMTP. In order to verify an address we only need a subset of the commands used for sending an e-mail. Early on in the reading of the standard it became clear that there are only 8 commands that are compulsory for any SMTP server to recognise. It seemed appropriate to get a brief understanding of these compulsory commands to make a decision on which of these commands would need to be used by the verification application. The table below shows the minimum commands that must be supported to allow SMTP to work. They are required by all SMTP servers.

Command	RFC2821 Section	Description
"EHLO" or "HELO"	4.1.1.1	These two commands are used to identify the client to the SMTP server and must be the first command used. "EHLO" supersedes "HELO" but both are still supported. Both commands are case insensitive. In both cases the command is given then a space followed by the domain of the client. Finally a carriage return and line feed is given. e.g. "Helo test.com "
"MAIL"	4.1.1.2	Used to start a mail transaction. Although the verification application will not be sending an e-mail this command must be used in order to be able to give the "RCPT", "VRFY" or "EXPN" commands. This command notifies the SMTP server where the e-mail is coming from (what the from address of the e-mail will be). An example command is "MAIL FROM:<user@test.com>"
"RCPT"	4.1.1.3	Short for recipient. This defines where and ultimately who the e-mail would be going to. However in the case of the verification application the response from this command can be used to verify an address. An example of this command would be: "RCPT TO:<user@email2test.com>"
"DATA"	4.1.1.4	This command sends the body, title and generally all information about the e-mail. The command parameters are quite complex. The verification application does not need to send this command because it will never send an e-mail.

"RSET"	4.1.1.5	This command aborts the current message, resetting the SMTP communication back to the point after the "HELO" or "EHLO" command. This command maybe useful if wanting to attempt a number of checks whether they are different e-mail addresses or different checks on the same address
"NOOP"	4.1.1.9	It stands for no operation. It is a test command that requests the server to return an OK response.
"QUIT"	4.1.1.10	The quit command can be issued at any time. Once a response is given the server will close the connection.
"VRFY"	4.1.1.6	As RFC2821 says "This command asks the receiver to confirm that the argument identifies a user or mailbox".

Figure 15: Minimum Implementation of SMTP

After reading though the RFC 2821 standard it became clear that the "NOOP" command would not be required in the verification application because this was simply a test command to check that the server is responding. Likewise the "DATA" command should not be used because this is only needed to send an e-mail. A key feature of SMTP is the fact that after any command is given a response is given back from the SMTP server to indicate the server status of how successful the operation was. It is the response from the server that will be used to verify an e-mail address. A complete e-mail will not be sent, instead the application will terminate the session to the server, following the verification step.

211 System status, or system help reply
214 Help message
(Information on how to use the receiver or the meaning of a
particular non-standard command; this reply is useful only
to the human user)
220 <domain> Service ready
221 <domain> Service closing transmission channel
250 Requested mail action okay, completed
251 User not local; will forward to <forward-path>(See section 3.4)
252 Cannot VRFY user, but will accept message and attempt delivery
354 Start mail input; end with <CRLF>.<CRLF>
421 <domain> Service not available, closing transmission channel
(This may be a reply to any command if the service knows it must shut down)
450 Requested mail action not taken: mailbox unavailable (e.g., mailbox busy)
451 Requested action aborted: local error in processing
452 Requested action not taken: insufficient system storage
500 Syntax error, command unrecognized
(This may include errors such as command line too long)
501 Syntax error in parameters or arguments
502 Command not implemented (see section 4.2.4)
503 Bad sequence of commands
504 Command parameter not implemented
550 Requested action not taken: mailbox unavailable
(e.g., mailbox not found, no access, or command rejected for policy reasons)
551 User not local; please try <forward-path> (See section 3.4)
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed
(e.g., mailbox syntax incorrect)
554 Transaction failed
(Or, in the case of a connection-opening response, "No SMTP service here")

(RFC2821, 2001:45)

Figure 16: SMTP Reply Codes in Numeric Order – RFC2821

Response Codes

SMTP responses are given as a three digit number followed by a space and a textual explanation. The text explanation varies from server to server but the three digit number is standardised. The three digit number is standardised for all responses to all the commands. The first digit shows the overall result. One is reserved for extended SMTP. Two means a success, three means the action is pending, four means there was a temporary error and five there was a permanent error. The second and third digits of the number classify and define the error. A list of common response codes are in Figure 16.

Sequence of commands

Section 4.1.4 of RFC2821 defines how the order of commands should be given.

The first command should be either "HELO" or "EHLO" followed by the clients host name. This maybe checked by the server against the IP address that the client is using, so it is important that the application can customise this property. It does recommend that "VRFY" can be used without any other commands but this is only a recommendation and may not be supported by all servers. Once the 'session' has started by using "HELO" or "EHLO" then any of the following commands can be used any number of times : "NOOP", "HELP", "EXPN", "VRFY", and "RSET". The sequence to use the "DATA" command can be ignored because this won't ever be sent by the verification program. If the "MAIL" command is given then it should be followed by the commands "RCPT" and "DATA". The standard does allow the sequence to be broken by using a "RSET". Finally "QUIT" must be the last command, and is used to exit from the session.

Verification Possibilities

After looking at all the commands, how they should be sequenced and their possible responses it is now possible to formulate a set of commands to verify an e-mail address. Two possible methods have been identified. These methods may need refining during testing.

Method 1

The first method uses the "VRFY" command. The client opens communication using either "HELO" or "EHLO" command. The "VRFY" command is then sent with the e-mail address needing to be verified. It's the response from this command that will determine whether the address does or does not, or whether the server refuses to verify the addresses (probably for security reasons). A "RSET" command is then given to assure that no e-mail is being sent and then the "QUIT" command is given to exit from the SMTP communication.

```
Command  1: "HELO myserver" or "EHLO myserver"
Response 1: 250 Requested mail action okay, completed
Command  2: "VRFY test@example.com"
Response 2: 250 Requested mail action okay, completed
Command  3: "RSET"
Response 3: 250 Reset state
Command  4: "QUIT"
Response 4: 221 Service closing transmission channel
```

Method 2

The second method is similar to the first except that it uses the "RCPT" command to verify the address rather than "VRFY". Before that command can be given the "MAIL" command must be issued, and a 'from' e-mail address.

```
Command  1: "HELO myserver" or "EHLO myserver"
Response 1: 250 ok
Command  2: "MAIL FROM: me@mysever.com"
Response 2: 250 ok
Command  3: "RCPT TO: test@test.com"
Response 3: 250 ok
Command  4: "RSET"
Response 4: 250 Reset state
Command  5: "QUIT"
Response 5: 221 Service closing transmission channel
```

It is also possible that a combination of the two methods may work better by verifying the address both ways and then combining the two results to form an overall verification result.

EXPN

"*Server implementations SHOULD support both VRFY and EXPN. For security reasons, implementations MAY provide local installations a way to disable either or both of these commands through configuration options or the equivalent.*"

(RFC2821 2001:22)

"This command asks the receiver to confirm that the argument Identifies a mailing list, and if so, to return the membership of that list. If the command is successful, a reply is returned containing information as described in section 3.5. This reply will have multiple lines except in the trivial case of a one-member list."

(RFC2821, 2001 : 35)

It is possible that the EXPN command can be used in a similar way to VRFY.
The EXPN command can only verify an e-mail address if it is used as an alias to mailing list on the mail server. Most e-mail addresses are not used in this way so it is anticipated that verification using EXPN will not be successful.

Manually testing via a telnet connection

Using telnet under Windows 2000 a manual connection can be made with a mail server. In the example below an attempt to verify a hotmail account resulted in the hotmail mail server giving a 550 code response. This provided successful validation of the first e-mail address (as the example address did not exist). Further tests showed that the server also responded as expected when given an address that existed. The return result for an existing address was 250.

```
[Contacting mx01.hotmail.com [64.4.55.71]...]
[Connected]
220-HotMail (NO UCE) ESMTP server ready at Tue, 26 Feb 2002
05:10:16 -0800
220 ESMTP spoken here
HELO glennturner.co.uk
250 Requested mail action okay, completed
MAIL FROM:<validemail@glennturner.co.uk>
250 Requested mail action okay, completed
RCPT TO:<email.test@hotmail.com>
550 Requested action not taken: mailbox unavailable
RSET
250 Reset state
QUIT
221 Service closing transmission channel
[Connection closed]
```

Figure 17: An example telnet connection

A simple SMTP test application

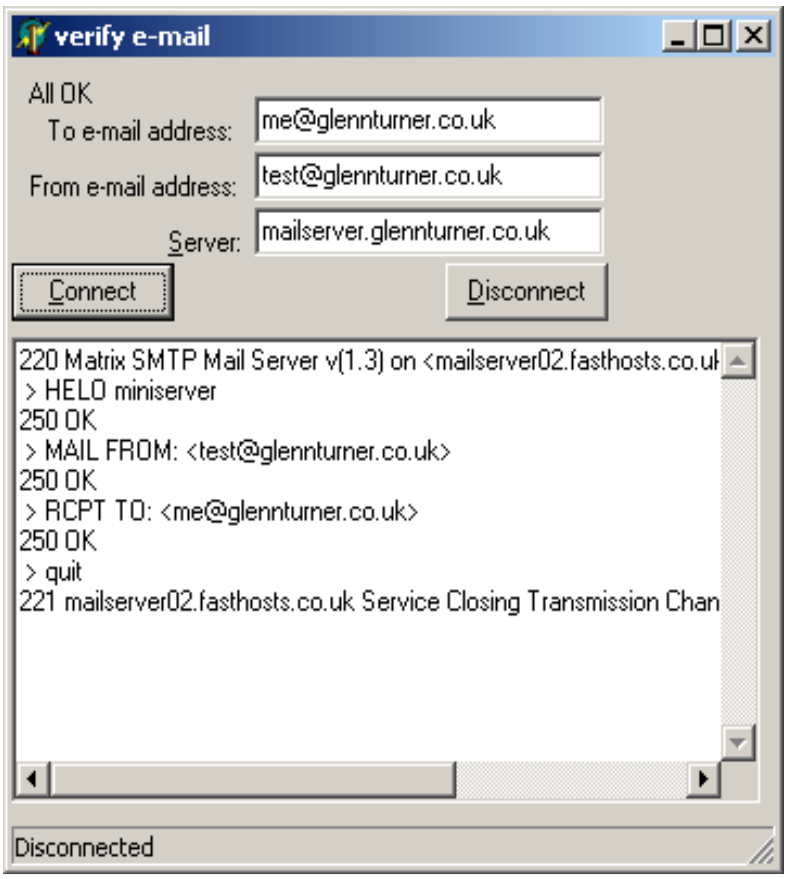

Figure 18: A SMTP test application

A simple application was written to familiarize myself with the telnet components in Delphi. This application did not use the MX record resolver object, so the user has to type in server address, along with the 'from address', and the address to verify. The program was entirely successful in allowing me to understand the telnet component. Once the application was completed the next stage of the development was started.

Ending the session

Creating the SMTP test application has shown that closing the connection can sometimes take a while because the mail server fails to respond promptly. There is an option to close the socket without notifying the mail server. This would be quicker, but I felt it would go against the etiquette of computer communication, or even against SMTP standards. I sought advice on the matter, and found RFC 2821 held the answer even giving recommended timeouts. E-mail address verification is relatively simple when compared to actually sending an e-mail, so I chose to set the timeout for any command to 30 seconds. If tests show this is a problem then this can easily be increased.

Issues with verifying

New provisions in mail servers are constantly being added to stop spamming. In particular Sendmail 8.9 released an "Anti-Spam Configuration Control" to aid mail administrators (sendmail.org, nd). This tool allows easy control over many aspects of the mail system, enabling the server to reject incoming mail based on information including the source of the e-mail. Worried that configuring the server to stop spam may also stop verification of e-mail addresses I again consulted RFC2821.

"A server MUST NOT return a 250 code in response to a VRFY or EXPN command unless it has actually verified the address. In particular, a server MUST NOT return 250 if all it has done is to verify that the syntax given is valid. In that case, 502 (Command not implemented) or 500 (Syntax error, command unrecognized) SHOULD be returned. As stated elsewhere, implementation (in the sense of actually validating addresses and returning information) of VRFY and EXPN are strongly recommended. Hence, implementations that return 500 or 502 for VRFY are not in full compliance with this specification."

(RFC2821, 2001:22)

From the above statement it is very clear that the server has to verify the address to fully comply with the standard. However, there are still return values that can be given if the mail administer does not wish to allow e-mail verification. See the two statements below.

"There may be circumstances where an address appears to be valid but cannot reasonably be verified in real time, particularly when a server is acting as a mail exchanger for another server or domain. "Apparent validity" in this case would normally involve at least syntax checking and might involve verification that any domains specified were ones to which the host expected to be able to relay mail. In these situations, reply code 252 SHOULD be returned. These cases parallel the discussion of RCPT verification discussed in section 2.1. Similarly, the discussion in section 3.4 applies to the use of reply codes 251 and 551 with VRFY (and EXPN) to indicate addresses that are recognized but that would be forwarded or bounced were mail received for them. Implementations generally SHOULD be more aggressive about address verification in the case of VRFY than in the case of RCPT, even if it takes a little longer to do so."

(RFC2821, 2001:23)

"As discussed in section 3.5, individual sites may want to disable either or both of VRFY or EXPN for security reasons. As a corollary to the above, implementations that permit this MUST NOT appear to have verified addresses that are not, in fact, verified. If a site disables these commands for security reasons, the SMTP server MUST return a 252 response, rather than a code that could be confused with successful or unsuccessful verification."

(RFC2821, 2001: 65)

There therefore could be mail servers that will not allow verification and will give a 252 or 251 error code. But no servers, provided they abide by the mail standards will falsely respond.

Reverse lookups

During manual testing using telnet it was noticed that one mail server was refusing to accept the "MAIL FROM" command. I realised that the server could be doing a reverse lookup on the domain of the from e-mail address. By changing domain to the actual domain for the Internet connection, ("pc-62-30-95-242-st.blueyonder.co.uk" at the time of writing) the mail server started accepting the "MAIL FROM" command. It is therefore important when verifying to use the actual domain in the from address.

The final verification application

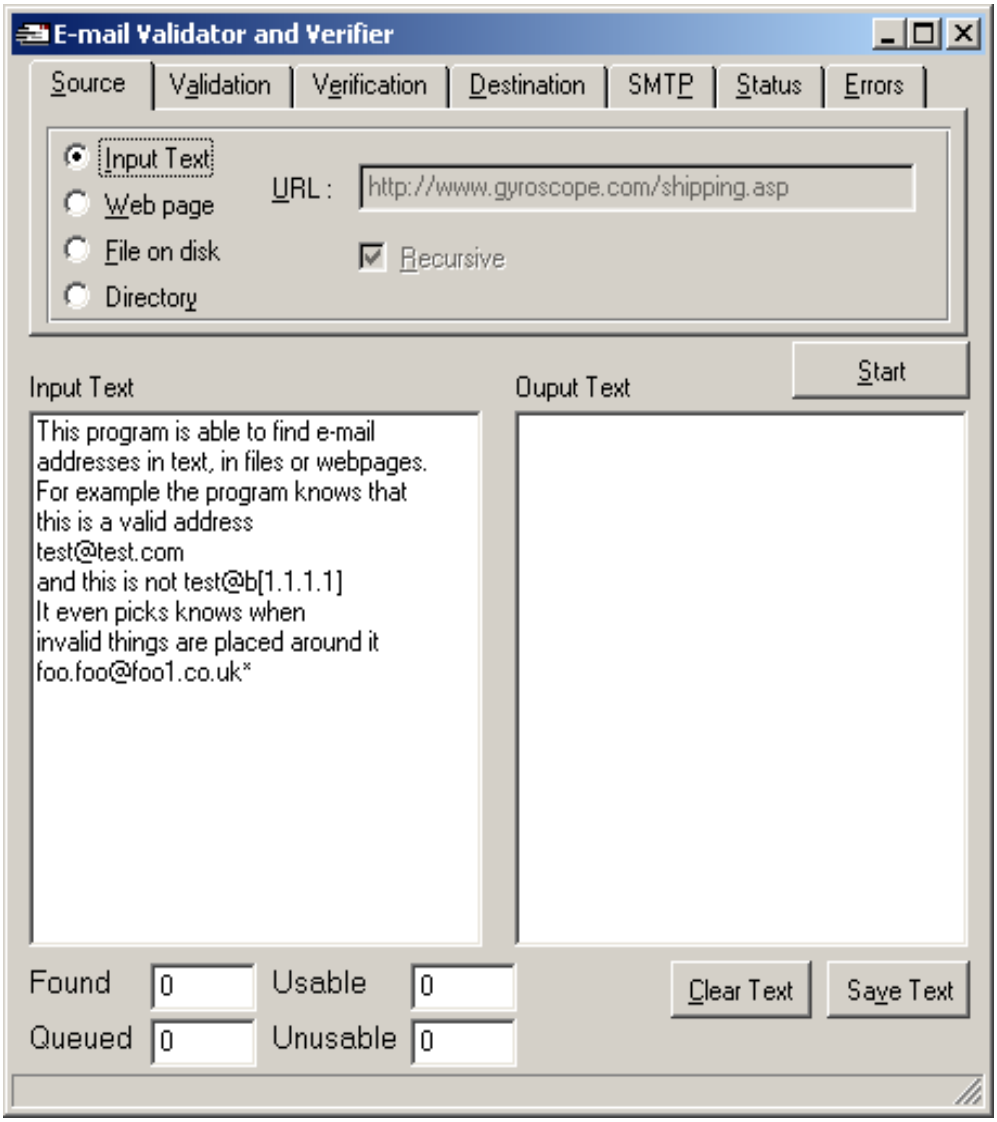

Figure 19: The verification application running

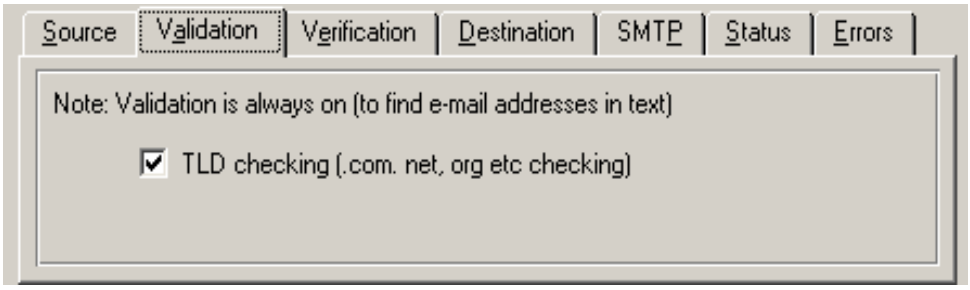

Figure 20: The validation tab

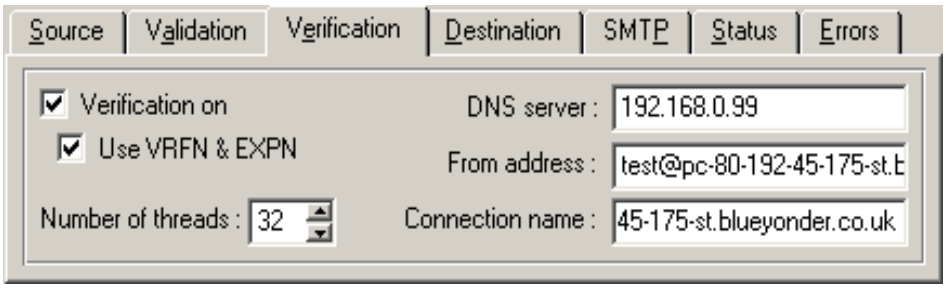

Figure 21: The verification tab

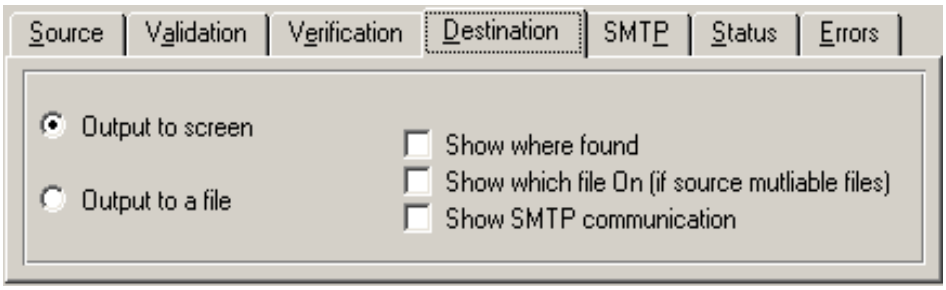

Figure 22: The destination tab

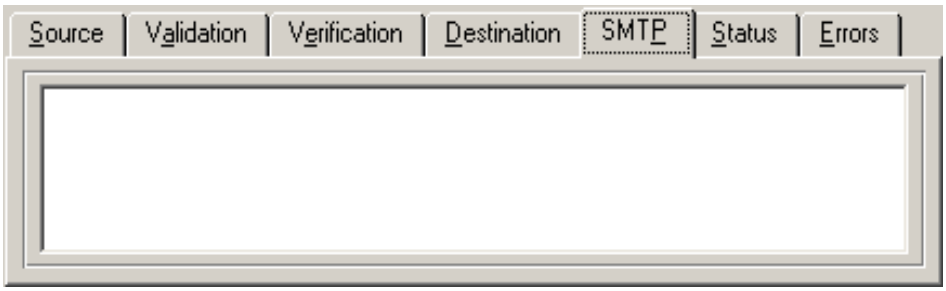

Figure 23: The SMTP tab

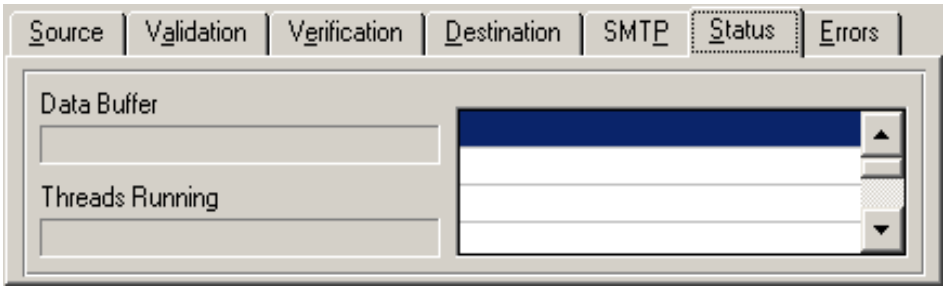

Figure 24: The status tab

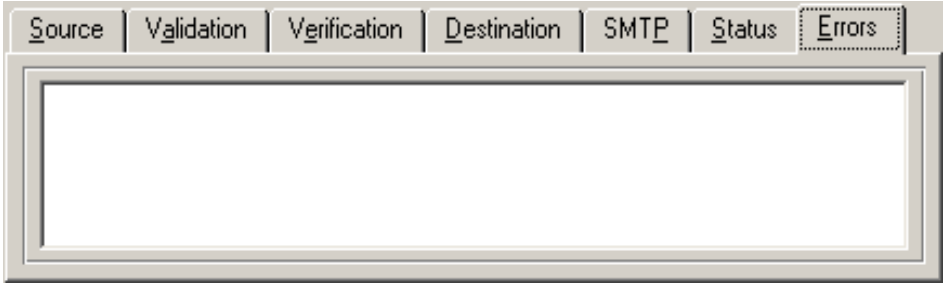

Figure 25: The Errors tab

Key Features of the program

- Automatically detects e-mail addresses in text
- Validates/verifies e-mail addresses from multiple sources (Text box, Web page, File, Directory)
- Optional Top Level Domain (TLD) checking
- Optional Verification of e-mail addresses
- Optional using of VRFY when verifying
- Multi-Threaded (choose between 1-99 threads to match the speed of system)
- Outputs to either the screen or file
- DNS server, connection name and from address can all be customised
- Program shows results in real-time
- SMTP, Status and errors tabs are used for debugging

Problems producing the prototype

The interface of the program was created first (without any underlying code) to get an idea how I wanted the data and options to be presented. Validation checking was first added, this required the extra regular expression module. Then parts of the previous verification test programs were added, including the MX resolver. Then each extra feature was added. The program worked well but was very slow, during the verification stage the interface 'locked up'. This was because while the program was waiting for a reply from a mail server the application was in an endless loop.

The program found an e-mail address to verify, then tried to verify it, and then went back to find another e-mail address. This was terribly inefficient.

A thread was added with the sole purpose just to verify. A data queue was put between the thread searching for e-mails and the thread verifying. This was done to enable the

searching thread to keep running. Unfortunately the queue could grow very quickly, while the verification thread had only processed a few addresses. Since the queue was in RAM I didn't want use all the memory, which could happen if the program was asked to verify a large number of addresses. I also wanted to add more verification threads so more e-mail addresses could be verified simultaneously.

Figure 26 shows the system devised to solve the problem. The system is based around a queue that can store a maximum of 100 e-mail addresses. I single thread is used to find e-mail addresses from the users chosen source. These are placed into the queue when they are found. If the queue is full the thread 'pauses' itself. When a thread is paused it requires no processing power. If the thread can't find any addresses a flag is raised and the thread stops. While the queue is not empty a number of verification threads will be running. These run independently, each taking a single address off the queue, verifying it and storing the results to the user's chosen destination e.g. file, screen. Any point where a thread may change any common data, such as the queue or interface is known as a critical section. In Delphi the "synchronize" method can be used to only allow one thread at a time to access a 'resource'. Without the "synchronize" method a number of threads could be writing at the same time, corrupting the data.

Apart from those mentioned there were no major problems developing the prototype.

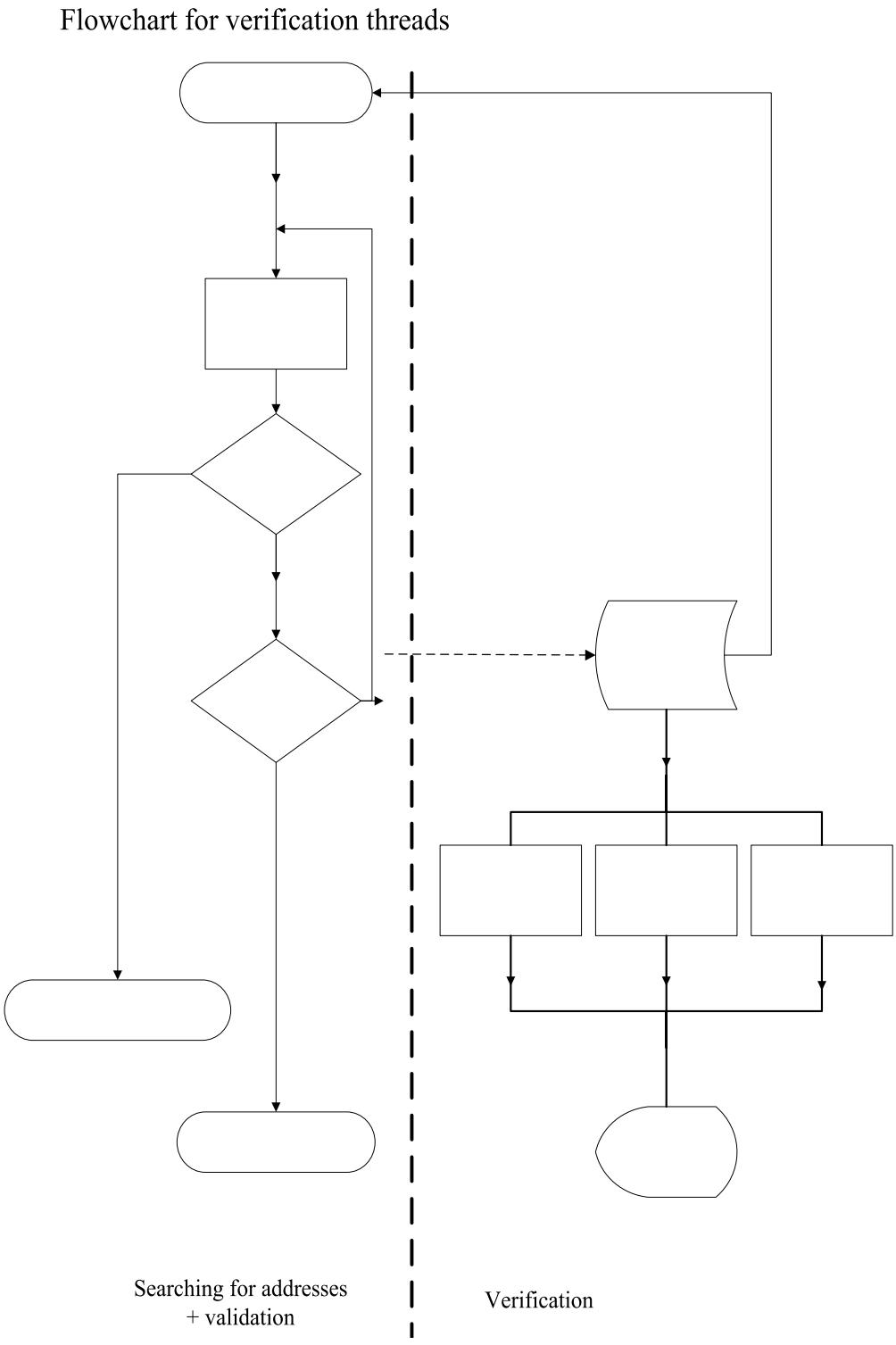

Figure 26: Flowchart for verification threads

Testing

The Test Data

Test data has been kindly donated by Gyroscope.com. The test data consists of 250 e-mail addresses that were collected as part of an opt-in newsletter over a course of an 18 month period. Gyroscope.com is an international site with the majority of visitors living in English speaking countries. The list of e-mail address reflects this with a wide range of TLDs. Addresses range from hotmail accounts to ones with .nz for a top level domain. The data protection 1998 (United Kingdom, 1998) prohibits the publication of these addresses without consent from the people or companies these e-mail addresses refer to. Verification to remove redundant e-mail addresses is considered as maintenance and is not prohibited. The addresses have not been filtered in anyway prior to testing and are not in any particular order.

Verification Testing

The tests check what percentage of e-mail addresses the application can verify using the VRFY command. It is anticipated that if there are any problems with the verification they will show up in these tests. Once the tests have taken place the results will be spilt into three groups. Those that have been verified as e-mail addresses (working), those that have been verified as non-working and those that can't be verified (if any). The addresses that have been marked as working and those that can't be verified will be included in Aprils Gyroscope.com newsletter. Provided no newsletter e-mails are 'bounced' we have proved the verification application has correctly verified those addresses. Only the addresses that are marked by the application to be non-working

need to be checked. A second newsletter will be sent out to those addresses. If the verification application has correctly verified them, all of them will bounce.

RCPT Testing

All 250 e-mail addresses will be subjected to a second type of verification using the RCPT command rather than the VRFY command. No e-mails will be sent to confirm the results. This is because the RCPT command was less successful in manual tests (telnet) at correctly verifying an e-mail address. The RCPT results will just be used to compare results with the VRFY command.

Speed/Thread Testing

A final set of tests will be conducted to test how effective threads are to the application. The tests will use the same test data as before but use a different number of threads for each test. The tests will be timed to see how much quicker the application is with threads and find an optimum number of threads.

The Test System

The test system is a dual processor PC with Celeron 400 processors. The PC has half a gigabyte of RAM and is running Windows 2000 Advanced Server. Internet connection is via a Telewest cable modem with 512kilobits download and 128Kilobits upload. The Internet connection is accessed through a Linux Smoothwall machine (IBM 350-P166) that acts as a DNS server, Proxy and firewall.

Results

Starting with the thread speed tests, Figure 27 shows the results in the form of a bar chart. Using a single thread to verify all 250 addresses took 1182 seconds (almost 20 minutes). Using 8 threads to verify took only 165 seconds which is about a 7^{th} of the time compared to using one thread. As more threads were used so the total time to verify the addresses reduced. Using 32 threads is the approximate optimum number of threads for the test system. More than 32 threads and the total verification time starts to increase. This is due a number of possible factors; the processors are spending more time switching between threads than verifying; the maximum amount of bandwidth for the Internet connection has been reached; or processing power is being taken up by the real-time data on the interface. Looking at the Windows task manager did not prove conclusive, but indicated which most of the processing power was being used refresh the interface.

Two bar charts show the verification results from the mail servers. Figure 29 shows the frequency of each response while using only the RCPT command to verify. Comparing the RCPT and VRFY commands, the RCPT command was the least successful. Figure 29 shows 6% of e-mail addresses could not be found by the mail server. In 4.4% of cases the mail server responded with error code 550 which represents that the mailbox is unavailable (incorrect e-mail address). 0.4% of responses gave a temporary error. Normally the addresses with temporary errors would put into their own list to be checked again at a later date. Despite not being as successful as the VRFY command, the RCPT command still recognised that 10.4% of the addresses were not working.

Figure 28 shows the verification results from using the VRFY command. The most noticeable column in Figure 28 is the verifiable column showing that 84.4% of the 250

e-mail addresses are verifiable, leaving just 14.8% that can't be verified and 0.8% that had a temporary error. 48% of addresses were verified as correct using VRFY compared to 89.20% using RCPT.

The two sets of newsletters sent out, which confirmed results of the verification application showed that the application was working well. All the e-mails expected to bounce did (up to two days later) and those that should have been received did not bounce. Only a few e-mails were sent back saying the person was out of the office.

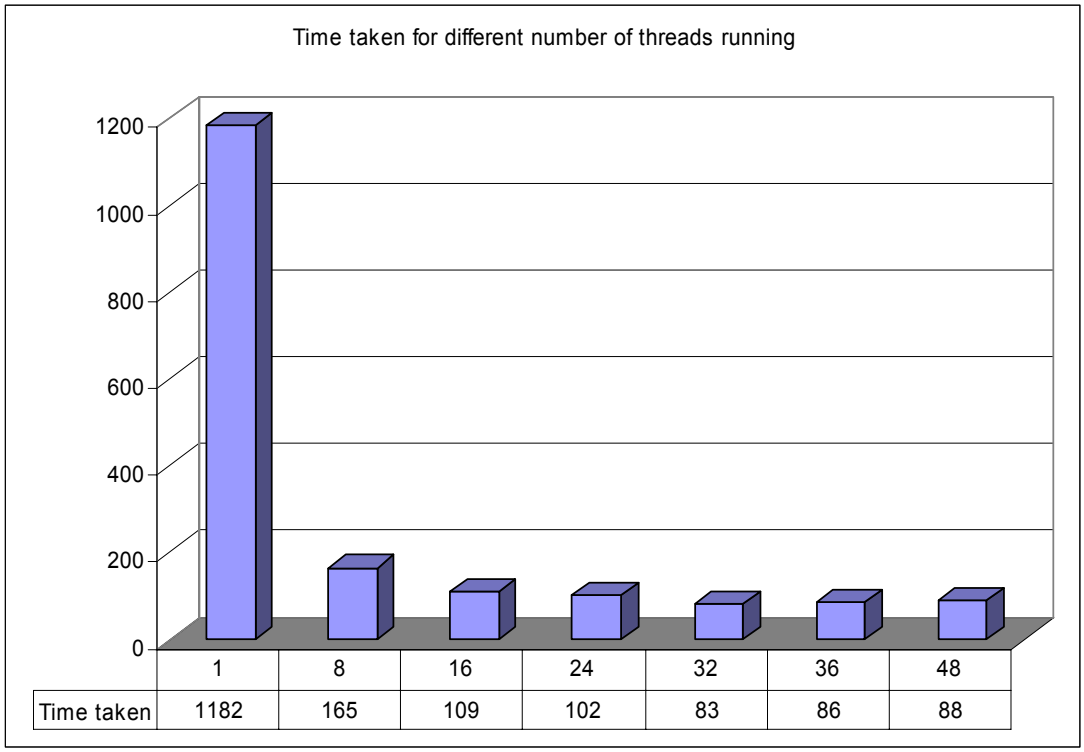

Figure 27: The time taken when different number of threads are running

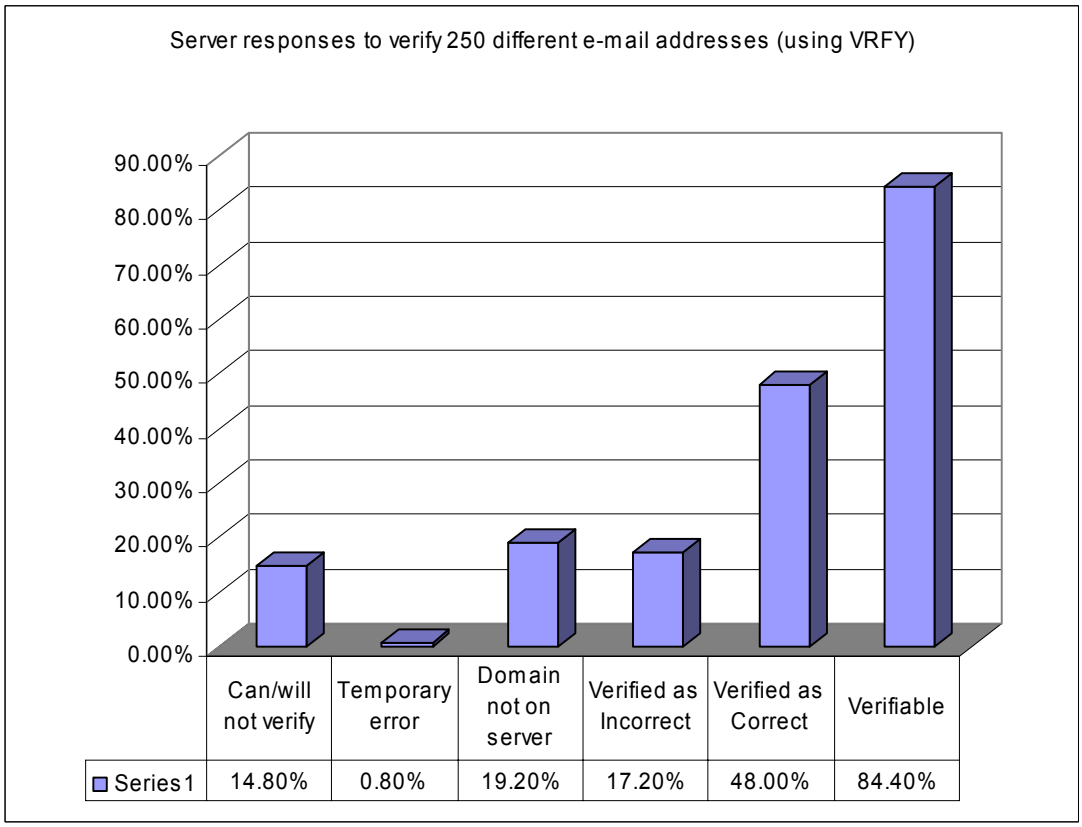

Figure 28: Server responses to verify 250 different e-mail addresses

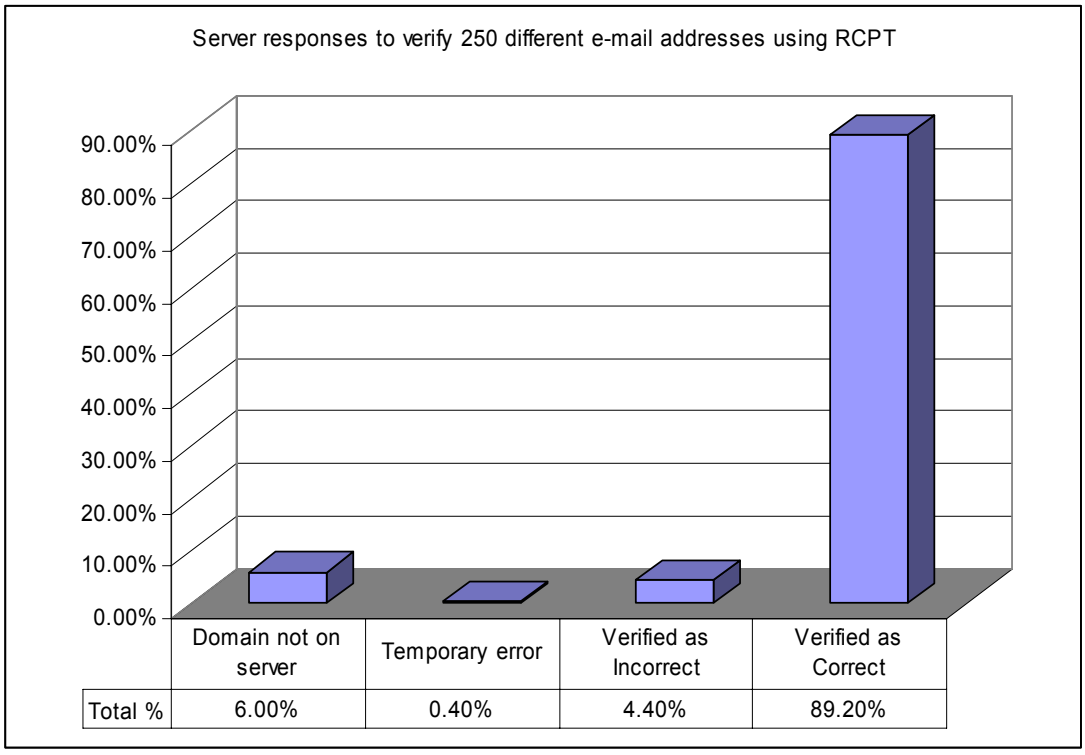

Figure 29: Server responses to verify 250 different e-mail addresses using RCPT

Conclusion

Product

In conclusion I would regard the project as a success. The final application was able to successfully validate e-mail addresses and verify them in the majority of cases.

When I started this project I was not aware of the complexity of the e-mail address structure. Despite the complexity, validation is performed each time an e-mail is sent. It is feasible to add validation to general software, but without 3rd-party tailored components to assist, the work evolved is not proportional with the results received. An unexpected consequence of validation was the ability to detect the position of an e-mail address in surrounding text. This has many applications including detecting addresses in files, documents and web pages.

Using the verification program, e-mail addresses are split into three groups. Those that are working; those that are not working and those that can't be verified. 84.4% of e-mail addresses from the test data were verified. Leaving just 15.6% of addresses where the 'correctness' could not be established. The ability to test such a high percentage of addresses held in company databases is a significant tool for administrators to protect against 'dirty data' issues. It is questionable whether such verification should be used at the point of data entry into systems, due to the application's inability to say absolutely if the address is correct or not. In certain circumstances, email addresses that cannot be verified may be assumed to be correct, however this would largely depend upon the administrator's requirements for gathering accurate information.

A possible alternative is to use the verification in an advisory roll, so when a user types an address into a system, a warning notification can be given to determine whether the email address has been entered correctly.

The only negative aspect to the project is regarding the future of email address verification. It is unknown if, in the future that the number of servers rejecting the verification process will increase, in an attempt to reduce spamming.

During the project a number of programs were found that can perform various forms verification on e-mail addresses. Some just check that the e-mail address's domain exists while the HexGadgets software does similar checking to the prototype application. Figure 30 gives a list of the programs trading name and a URL.

Trading Name	**URL**
Mail Utilities	http://www.mailutilities.com/amv/
ASPmx	http://www.internext.co.za/stefan/aspmx/
HexGadgets	http://www.hexillion.com/
Clean address	http://www.runnertechnologies.com/

Figure 30: A list of companies selling e-mail address verification products

Process

The Project Aim on page 10 started "To establish if it is feasible to produce a software program to validate and verify e-mail addresses". From this, a target was set to create a prototype program to establish if this was the case. I would have preferred to have produced UML (Unified modelling Language) diagrams for the prototype. Unfortunately time constraints did not allow for any formal designs. It was more important to create a working prototype than use certain techniques and fail to get the program functioning because of a lack of time.

Given time I would have liked to have produced an Internet version for the verification program with the interface being a web page. As Delphi can create IIS (Internet Information Server) components this would have been relatively easy task.

The verification test data should be a good representation of average e-mail addresses. If the verification program was used in a commercial environment the results should be similar to the tests. I would have liked to have had test data with invalid addresses to test against the validation program. Unfortunately a source of such data could not be found. This type of data would have allowed the project to consider how effective validation is compared to verification.

Figure 31 shows what I call an "e-mail address correctness hierarchy". It shows the methods that can be taken to increase the correctness of an e-mail address. At the bottom there is no checking, and at the top there is theoretical absolute correctness. As you go further up the column the e-mail address correctness increases, 'dirty data' reduces and the complexity of the checking increases. The verification in the prototype produced is the second highest process, which currently can only be bettered by theoretical absolute correctness.

Learning

Introducing threads into the application has been a major achievement for me during the project. At the start of the project I had never created a multi-threaded program. I now have an understanding of the limitations and benefits of threads. I'm sure this will be of use for future applications. In this project threads proved to be very useful to increase the speed of the application.

I've used regular expressions in the past for very simple programs and scripts but never in such scale like I have in this project. It surprised me how concise and efficient regular expressions are and I would be more likely to use them in future programs. I've also been made more aware of the complexity of mail systems, there protocols, standards and limitations.

During the project the limitations of the Harvard referencing system at times have caused problems with references to web based resources that had no authors or creation dates. This problem was discovered when I found a reference to Sendmail spam checking on Sendmail's own site without an author or creation date. Clarification was given by the University regarding such cases which resolved the matter.

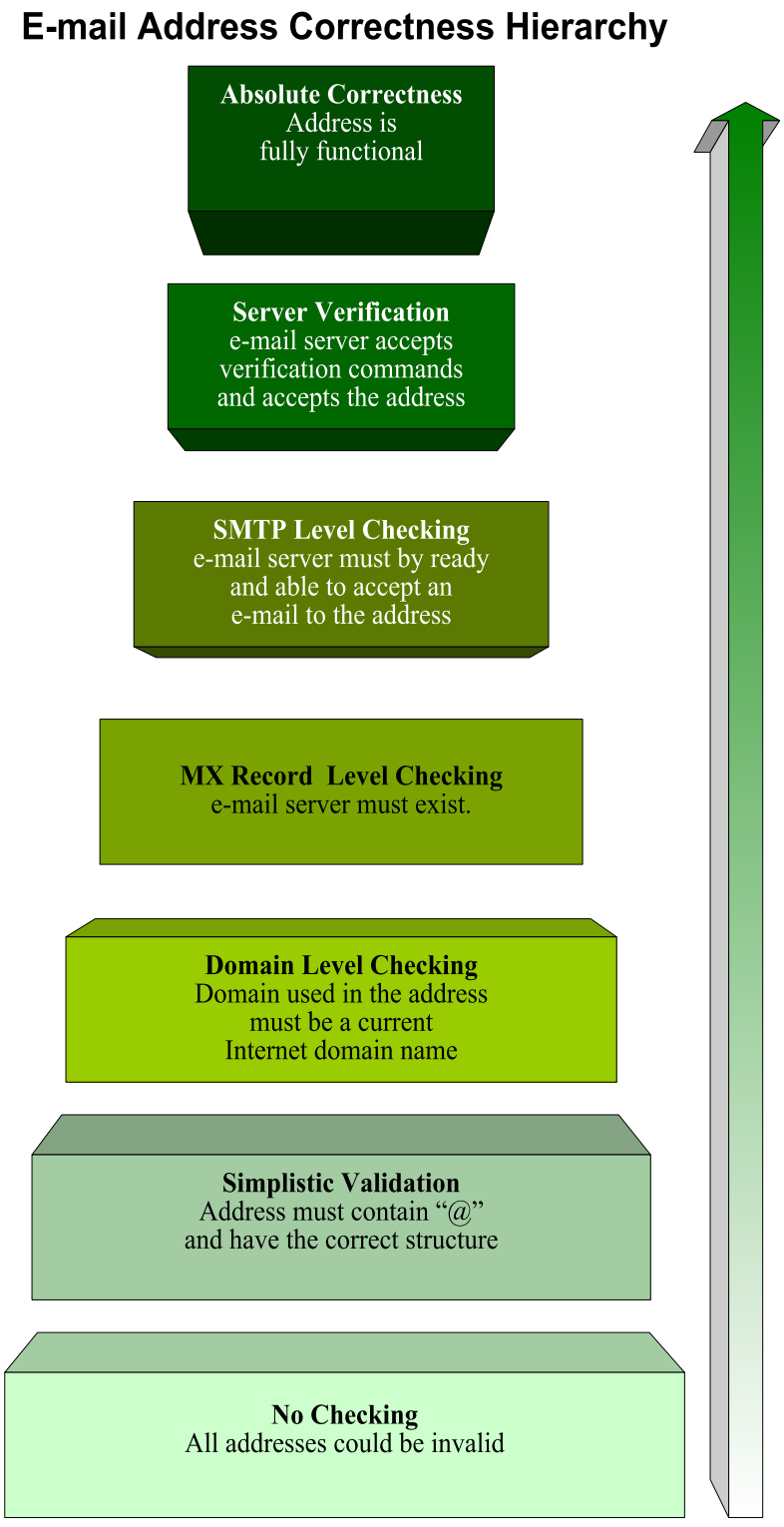

Figure 31: E-mail address correctness hierarchy

Bibliography

Validation

Hexillion Technologies (nd) Hexillion [Online] [Cited 9 Oct 2001]
Available from URL: http://www.hexillion.com/

Internic (2002) The Domain Name System: A Non-Technical Explanation – Why Universal Resolvabilty Is Important [Online] [Cited 1 Nov 2001]
Available from URL: http://www.internic.net/faqs/authoritative-dns.html

INT Media Group (nd) ASP 101 - Active Server Pages Resource Index - DNS Functions [Online] [Cited 9 Oct 2001] Available from URL:
http://asp101.aspin.com/home/components/internet/dns?cob=asp101

Moten, L. (nd) Catching Bogus Addresses - active server pages - ASP [Online] [Cited 7 Aug 2001] Available from URL: http://www.planet-source-code.com/xq/ASP/txtCodeId.6803/lngWId.4/qx/vb/scripts/ShowCode.htm

ServerObjects Inc., (nd) Products - AspMX™ 1.5 [Online] [Cited 9 Oct 2001]
Available from URL: http://www.serverobjects.com/products.htm#aspmx

Shiran, Y. (nd) Validating email Address [Online] [Cited 10 March 2000]
Available from URL: http://www.webreference.com/js/tips/000310.html

Verification

Finer, J. (nd) The Forgotten Art of Email Address Validation [Online] [Cited 20 Jan 2002] Available from URL:
http://www.4guysfromrolla.com/webtech/093000-1.shtml

Thus PLC, (nd) Demon Internet: Internet Query Tools [Online] [Cited 10 Jan 2002]
Available from URL: http://www.demon.net/external/

SMTP Security

Wirtschafts University, (nd) VRFY, EXPN, and Security [Online] [Cited 20 Jan 2002]
Available from URL: http://www.wu-wien.ac.at:8082/rfc/rfc2821.hyx/7.3_bVRFY,_bEXPN,_band_bSecurity

History

Brodbelt, M. (nd) A Brief History of Mail [Online] [Cited 20 Jan 2002] Available from URL: http://www.coruscant.demon.co.uk/mike/sendmail/history.html

Computer Literacy (nd) E-mail History [Online] [Cited 20 Jan 2002] Available from URL: http://compulit.uta.edu/mailhist.html

Nua Internet Surveys & ComputerScope Ltd. (2001) How Many Online? [Online] [Cited 20 Jan 2002] Available from URL: http://www.nua.ie/surveys/how_many_online/index.html

e-mail Systems

Albitz, P. and Liu, C. (1998) DNS and BIND. 3rd ed. Sebastopol: O'Reilly & Associates

Christiansen, T. and Torkington, N. (1999) Sendmail. Sebastopol: O'Reilly & Associates

Costales, B. and Allman, E. (1997) Sendmail. 2nd ed. Sebastopol: O'Reilly & Associates

GBdirect, (nd) An Overview of Internet Email [Online] [Cited 20 Jan 2002] Available from URL: http://ebusiness.gbdirect.co.uk/howtos/mail-system.html

Maor et al, SMTP Simple Mail Transfer Protocol [Online] [Cited 20 Jan 2002] Available from URL: http://raddist.rad.com/networks/1998/smtp/smtp.htm

Programming and languages

Friedl, J.E.F. (1998) Mastering Regular Expressions. 7th printing. Sebastopol: O'Reilly & Associates

Netscape Communications Corp., (2000) RegExp [Online] [Cited 20 Jan 2002] Available from URL: http://developer.netscape.com/docs/manuals/js/core/jsref15/regexp.html

Siever, E. and Spainhour, S. and Patwardhan, N. (1999) PERL in a nutshell. Sebastopol: O'Reilly & Associates

Sorokin, A. V. (2001) Delphi freeware components: regular expressions [Online] [Cited 21 April 2002] Available from URL: http://anso.virtualave.net

Weissinger, A.K. (1999) ASP in a nutshell. Sebastopol: O'Reilly & Associates

Internet Standards

Braden, R. (1989) Requirements for Internet Hosts -- Application and Support [Online] [Cited 8 Dec 2001] Available from URL: http://www.rfc.net/rfc1123.html

Deering, S. & Hinden, R. (1998) Internet Protocol Version 6 (IPv6) Specification [Online] [Cited 8 Dec 2001] Available from URL: http://www.rfc.net/rfc2460.html

Gilligan, R. & Thomson, S. & Bound, J. & Stevens, w. (1997) Basic Socket Interface Extensions for IPv6 [Online] [Cited 21 Apr 2002] Available from URL: http://www.rfc.net/rfc2133.html

Internet society, (nd) RFC index search engine [Online] [Cited 25 Apr 2002] Available from URL: http://www.rfc-editor.org/cgi-bin/rfcsearch.pl

Klensin, J. et al (2001) Internet Message Format RFC2821 [Online] [Cited 23 Feb 2002] Available from URL: http://www.rfc.net/rfc2821.html

Mockapetris, P. (1983) DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION [Online] [Cited 18 Mar 2002] Available from URL: http://www.rfc.net/rfc883.html

Postel, J. B. (1982) Simple Mail Transfer Protocol RFC821 [Online] [Cited 10 Feb 2002] Available from URL: http://www.rfc.net/rfc821.html

Postel, J. B. (1994) Domain Name System Structure and Delegation [Online] [Cited 18 Mar 2002] Available from URL: http://www.rfc.net/rfc1591.txt

Resnick, P. (2001) Internet Message Format RFC2822 [Online] [Cited 10 Feb 2002] Available from URL: http://www.rfc.net/rfc2822.html

General

Bradley, R. (1995) Understanding computer science for advanced level, 3rd ed. Cheltenham: Stanley Thornes Publishers ltd

Harris, R. & Rickman, R. (1997) Workshop Studies – Workbook, 8th ed. Cheltenham: Cheltenham & Gloucester College of Higher Education

Illingworth, V. et al (1996) Dictionary of computing, 4th ed. New York: Oxford University Press.

SmoothWall Ltd, (2002) Smoothwall [Online] [Cited 25 Apr 2002] Available from URL: http://www.smoothwall.org

World Wide Alliance of Top Level Domain-names, (nd) ccTLD Contact List [Online] [Cited 25 Apr 2002] Available from URL: http://www.wwtld.org/member_list/countrycodesort0917.php

References

Campbell, T. (1998) The first e-mail message – Who sent it and what it said, [Online] [Cited 8 Dec 2001] Available from URL: http://www.pretext.com/mar98/features/story2.htm

Computer Literacy, (nd) E-mail history, [Online] [Cited 20 Jan 2002] Available from URL: http://compulit.uta.edu/mailhist.html

The Radicati Group Inc., (2001) "Messaging Software: Market and product Analysis. 2001-2005" [Online] [Cited 24 Apr 2002] Available from URL: http://www.software-aus.com.au/pdf/01MessagingBrochure.pdf

Marcus, B. (2001) E-mail Changes Thwarting Web Marketers [Online][Cited 24 Apr 2002] Available from URL: http://www.digitrends.net/mna/index_14181.html

Bradner, S. (1996) The Internet Standards Process -- Revision 3 [Online][Cited 25 Apr 2002] Available from URL: http://www.ietf.org/rfc/rfc2026.txt

Resnick, P.(2001) Internet Message Format RFC2822 [Online] [Cited 10 Feb 2002] Available from URL: http://www.rfc.net/rfc2822.html

Marshall, D. (nd) History of the Internet, [Online] [Cited 5 May 2002] Available from URL: http://www.netvalley.com/archives/mirrors/davemarsh-timeline-1.htm

Klensin, J. et al, (2001) Simple Mail Transfer Protocol [Online] [Cited 4 Nov 2001] Available from URL: http://www.imc.org/rfc2821

Crocker, D. (1997) Augmented BNF for Syntax Specifications [Online] [Cited 23 Feb 2002] Available from URL: http://www.imc.org/rfc2234

Friedl, J.E.F. (1998) Mastering Regular Expressions. 7th printing. Sebastopol: O'Reilly & Associates

Howe, D. (1997) regular expression from FOLDOC [Online] [Cited 26 Apr 2002] Available from URL: http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=regular+expressions

Mockapetris, P. (1983) DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION [Online] [Cited 18 Mar 2002] Available from URL: http://www.rfc.net/rfc883.html

Crocker, D. (1982) Standard for the Format of ARPA Internet Text Messages [Online] [Cited 23 Feb 2002] Available from URL: http://www.imc.org/rfc822

(2001) IPv6: The Next Generation Internet!, [Online] [Cited 21 Feb 2002] Available from URL: http://www.ipv6.org/

Sorokin, A. V. (2001) Delphi freeware components: regular expressions [Online] [Cited 21 April 2002] Available from URL: http://anso.virtualave.net

(nd) Anti-Spam Configuration Control, [Online] [Cited 1 May 2002] Available from URL: http://www.sendmail.org/m4/anti-spam.html

United kingdom, Data Protection Act 1998, London: HMSO

Appendix

The current Top Level Domains (TLDs) (24/2/2002)

.ac	.bo	.dk	.gq	.ki	.mp	.pf	.so	.vc
.ad	.br	.dm	.gr	.km	.mq	.pg	.sr	.ve
.ae	.bs	.do	.gs	.kn	.mr	.ph	.st	.vg
.aero	.bt	.dz	.gt	.kp	.ms	.pk	.sv	.vi
.af	.bv	.ec	.gu	.kr	.mt	.pl	.sy	.vn
.ag	.bw	.edu	.gw	.kw	.mu	.pm	.sz	.vu
.ai	.by	.ee	.gy	.ky	.museum	.pn	.tc	.wf
.al	.bz	.eg	.hk	.kz	.mv	.pr	.td	.ws
.am	.ca	.eh	.hm	.la	.mw	.pro	.tf	.ye
.an	.cc	.er	.hn	.lb	.mx	.pt	.tg	.yt
.ao	.cd	.es	.hr	.lc	.my	.pw	.th	.yu
.aq	.cf	.et	.ht	.li	.mz	.py	.tj	.za
.ar	.cg	.fi	.hu	.lk	.na	.qa	.tk	.zm
.as	.ch	.fj	.id	.lr	.name	.re	.tm	.zw
.at	.ci	.fk	.ie	.ls	.nc	.ro	.tn	
.au	.ck	.fm	.il	.lt	.ne	.ru	.to	
.aw	.cl	.fo	.in	.lu	.net	.rw	.tp	
.az	.cm	.fr	.info	.lv	.nf	.sa	.tr	
.ba	.cn	.fx	.int	.ly	.ng	.sb	.tt	
.bb	.co	.ga	.io	.ma	.ni	.sc	.tv	
.bd	.com	.gb	.iq	.mc	.nl	.sd	.tw	
.be	.com	.gd	.ir	.md	.no	.se	.tz	
.bf	.cr	.ge	.is	.mg	.np	.sg	.ua	
.bg	.cu	.gf	.it	.mh	.nr	.sh	.ug	
.bh	.cv	.gh	.jm	.mil	.nu	.si	.uk	
.bi	.cx	.gi	.jo	.mk	.nz	.sj	.um	
.biz	.cy	.gl	.jp	.ml	.om	.sk	.us	
.bj	.cz	.gm	.ke	.mm	.org	.sl	.uy	
.bm	.de	.gn	.kg	.mn	.pa	.sm	.uz	
.bn	.dj	.gp	.kh	.mo	.pe	.sn	.va	

Figure 32: The current Top Level Domains

Extracts from RFC2822

2.2.2. Structured Header Field Bodies

```
WSP             =       ASCII value 32 (Space) and  ASCII value 9
(horizontal tab)
```

3.2.1. Primitive Tokens

```
CRLF            =       ASCII value 11 and ASCII value 13

NO-WS-CTL       =       %d1-8 /         ; US-ASCII control
characters
                        %d11 /          ;  that do not include the
                        %d12 /          ;  carriage return, line
feed,
                        %d14-31 /       ;  and white space
characters
                        %d127

text            =       %d1-9 /         ; Characters excluding CR
and LF
                        %d11 /
                        %d12 /
                        %d14-127 /
                        Obs-text
```

3.2.2. Quoted characters

```
quoted-pair     =       ("\" text) / obs-qp
```

3.2.3. Folding white space and comments

```
FWS             =       ([*WSP CRLF] 1*WSP)

ctext           =       NO-WS-CTL /     ; Non white space controls

                        %d33-39 /       ; The rest of the US-ASCII
                        %d42-91 /       ;  characters not including
"(",
                        %d93-126        ;  ")", or "\"

ccontent        =       ctext / quoted-pair / comment

comment         =       "(" *([FWS] ccontent) [FWS] ")"

CFWS            =       *([FWS] comment) (([FWS] comment) / FWS)
```

3.2.4. Atom

```
atext            =       ALPHA / DIGIT / ; Any character except
controls,
                         "!" / "#" /     ;  SP, and specials.
                         "$" / "%" /     ;  Used for atoms
                         "&" / "'" /
                         "*" / "+" /
                         "-" / "/" /
                         "=" / "?" /
                         "^" / "_" /
                         "`" / "{" /
                         "|" / "}" /
                         "~"

atom             =       [CFWS] 1*atext [CFWS]

dot-atom         =       [CFWS] dot-atom-text [CFWS]

dot-atom-text    =       1*atext *("." 1*atext)
```

3.2.5. Quoted strings

```
DQUOTE           =       ASCII value 34

qtext            =       NO-WS-CTL /     ; Non white space controls

                         %d33 /          ; The rest of the US-ASCII
                         %d35-91 /       ;  characters not including
"\"
                         %d93-126        ;  or the quote character

qcontent         =       qtext / quoted-pair

quoted-string    =       [CFWS] DQUOTE *([FWS] qcontent) [FWS]
DQUOTE [CFWS]
```

3.2.6. Miscellaneous tokens

```
word             =       atom / quoted-string
```

3.4.1. Addr-spec specification

```
addr-spec        =       local-part "@" domain

local-part       =       dot-atom / quoted-string / obs-local-part

domain           =       dot-atom / domain-literal / obs-domain

domain-literal   =       [CFWS] "[" *([FWS] dcontent) [FWS] "]"
[CFWS]
```

```
dcontent        =       dtext / quoted-pair

dtext           =       NO-WS-CTL /     ; Non white space controls

                        %d33-90 /       ; The rest of the US-ASCII
                        %d94-126        ;  characters not including
"[",
                                        ;  "]", or "\"
```

4.1. Miscellaneous obsolete tokens

```
obs-qp          =       "\" (%d0-127)

obs-text        =       *LF *CR *(obs-char *LF *CR)

obs-char        =       %d0-9 / %d11 /          ; %d0-127 except CR
and
                        %d12 / %d14-127         ;  LF
```

4.4. Obsolete Addressing

```
obs-local-part  =       word *("." word)

obs-domain      =       atom *("." atom)
```

ASP to validate an e-mail address

```
<html>
<title>VALIDATION AND VERIFICATION OF E-MAIL ADDRESSES</title>
<body>
<center>
<h2>VALIDATION AND VERIFICATION<br> OF E-MAIL ADDRESSES</h2>
PRESENTED AS PART OF THE <br>
REQUIREMENT FOR AWARD WITHIN<br>
THE UNDERGRADUATE MODULAR SCHEME <br>
AT: <a href="http://www.glos.ac.uk/">GLOUCESTERSHIRE
UNIVERSITY</a><br>
BY: <a href="http://www.glennturner.co.uk/">GLENN TURNER</a>
</div>

<%
 '*********************************************************************
 '               Finds e-mail addresses in a string
 '               ==================================
 ' Written by : Glenn Turner
 ' Date       : 14/11/2001
 ' Updated    : 15/11/2001 - now recognises IP addresses
 '
 ' Yet to:
 '   find correct tlds
 '   Does not take in account total length (any size)
 '
 '*********************************************************************

Function emailRegExp(byval ValidateTLD)
  Dim RegIPNo
  Dim RegDomain
  Dim RegLocal
  Dim RegIPaddr
  Dim RegQstring
  Dim ValidTLDs
  Dim RegDomainTLD

  ValidTLDs = _
  "(aero|biz|com|edu|museum|org|name|net|pro|mil|info|int|"&_
  "ac|ad|ae|af|ag|ai|al|am|an|ao|aq|ar|arapa|as|at|au|aw|az|ba|" &_
  "bb|bd|be|bf|bg|bh|bi|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|" &_
  "cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|cr|cu|cv|cx|cy|cz|de|dj|" &_
  "dk|dm|do|dz|ec|ee|eg|eh|er|es|et|fi|fj|fk|fm|fo|fr|fx|ga|" &_
  "gb|gd|ge|gf|gh|gi|gl|gov|gm|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|"
&_
  "hr|ht|hu|id|ie|il|in|io|iq|ir|is|it|jm|jo|jp|ke|kg|" &_
  "kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|" &_
  "ma|mc|md|mg|mh|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|" &_
  "mv|mw|mx|my|mz|na|nc|ne|nf|ng|ni|nl|no|np|nr|nu|nz|" &_
  "om|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|pt|pw|py|qa|re|ro|ru|"&_
  "rw|sa|sb|sc|sd|se|sg|sh|si|sj|sk|sl|sm|sn|so|sr|st|sv|sy|sz|" &_
  "tc|td|tf|tg|th|tj|tk|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|um|" &_
  "us|uy|uz|va|vc|ve|vg|vi|vn|vu|wf|ws|ye|yt|yu|za|zm|zw)"


'*****************************************************************
  ' Finds a number between 0-255
  '     e.g.    100-249       120-5/220-225     10-99      0-9
```

```
'*********************************************************************
  RegIPNo="(([1-2][0-4][\d])|([1-2][5][0-5])|([1-9][0-8])|([\d]))"


'*********************************************************************
  ' Finds a IP address in the format [X.X.X.X]

'*********************************************************************
  RegIPaddr = "\[" & RegIPNo & "(\." & RegIPNo & "){3}\]"


'*********************************************************************
  'Finds a domain e.g. server1.test.foo.com (TLD  2 to 6
characters.)

'*********************************************************************
  RegDomain = "((([a-zA-Z0-9\-]{1,62})+[\.])+[a-zA-Z0-9\-]*)"

  ' Do the same this time validating the TLD.

  RegDomainTLD = "((([a-zA-Z0-9\-]{1,62})[\.])+" & ValidTLDs & ")"


'*********************************************************************
  'Local part, Note: Two dots cannot be together

'*********************************************************************

  RegLocal = _
    "[\w\*\$\?\#\%\&\^\+\-\=\_\`\{\|\}\~\'\!\/]+" & _
    "([\.][\w\*\$\?\#\%\&\^\+\-\=\_\`\{\|\}\~\'\!\/]+)*"
  RegQstring = _
    "([\x22]([\w]|([\\\]" & _
    "[\011\014-\127\012\001\002\003\004\005\006\007\008\009]))*" & _
    "[\x22])"

  if ValidateTLD then
    emailRegExp = "((" & RegLocal & "|" & RegQstring & ")\@"&_
                  "(" & RegDomainTLD & "|" & RegIPaddr & "))"
  else
    emailRegExp = "((" & RegLocal & "|" & RegQstring & ")\@"&_
                  "(" & RegDomain & "|" & RegIPaddr & "))"
  end if
end function


  Dim String2Search
  Dim RegExp
  Dim foundmatched
  Dim foundmatch
  Dim email2validate
  Set RegExp = New RegExp

  string2search = request("email2validate")

  email2validate = request("email2validate")
  email2validate = """:-)""@test.co.uk"
  if trim(email2validate) = "" then
    email2validate = _
    "This is a test string test@test.com to find" & vbcrlf & _
```

```
    "e-mail foo.foo@foo1.co.uk* addresses within" & vbcrlf & _
    "a document. test@[62.63.1.255] test@b[1.1.1.1] x[1.1.1.256]"
  end if

if len(string2search) > 0 then
  'Response.write mailRegExp & "<br><br>"

  With RegExp
    if request("validateTLD") = "on" then
      .Pattern = emailRegExp(true)
    else
      .Pattern = emailRegExp(false)
    end if
    .IgnoreCase = False
    .Global = True
  End With

  Set foundmatch = RegExp.Execute(String2Search)


  %><br><br>
  <table border="1">
  <tr>
  <td bgcolor="#FF0000"><b><font
color="FFFFFF">Results</font></b></td>
  </tr><tr><td><%
  If foundmatch.Count > 0 Then
    string2search = _
    RegExp.replace(String2Search, "<font color='#CC3333'>$1</font>")
  end if

  Response.write "<PRE>" & String2Search &
"</PRE></td></tr></table>"

  If foundmatch.Count = 0 Then
    Response.Write vbcrlf & _
      "<B>The data given was found NOT to" & _
      " be a valid e-mail address</B><br>" & _
      "If this is not the case please " & _
      "e-mail me at "&_
      "<a
href='final@glennturner.co.uk'>final@glennturner.co.uk</a>" & _
      " so I can update the program."
  End If

  Set RegExp = nothing

end if
%>
<center>
<h2>e-mail validation program</h2>
Enter one or more e-mail addresses in the area below<br>
(May be mixed in with other text)
<form ACTION="<%=Request.ServerVariables("SCRIPT_NAME")%>"
METHOD="POST" id=form1 name=form1>
  <textarea rows=5 cols=60 name="email2validate"
id="email2validate"><%=email2validate%></textarea><br>
  <input type="checkbox" <%if request("validateTLD") = "on"
then%>CHECKED<%end if%> name="validateTLD" id="validateTLD">
  <input type="submit" value="Validate" name="submit" id="submit">
</form>
```

```
</center>

<table align="center">
<tr><td width="600">
<b>Abstract</b>

<div align="justify">
E-mail addresses are crucial to the Internet community providing a
quick,
easy, cost effective means of contacting people to provide
information
or services. Along with the World Wide Web (WWW) e-mail is one of
key
building blocks of the Internet and arguably one of the reasons why
the
use of the Internet has grown so rapidly.
<br>
<br>
E-mailing provides the ability to market products or services to
existing
or potential customers at a low cost. This means a list of e-mail
addresses are often regarded as a highly prized asset by website
administrators. The amount of e-mail addresses in a company's
database
can even increase the company's value as it indicates the amount of
potential customers it has. As a result many companies place a high
degree of importance to the reliability (correctness) of gathered e-
mail
addresses.
<br>
<br>
The value of an e-mail address creates a need to check the
'correctness'
of e-mail addresses entering into the company's databases. Periodic
checking of e-mail addresses in a database would reduce the amount
of
addresses that are no longer in use. These unused addresses increase
as
people change their e-mail address, often by changing their internet
service provider.
<br>
<br>
This document investigates a number of methods to find the
'correctness'
of an e-mail address, discusses how successful the methods found are
likely to be and how feasible it would be to use them for commercial
purposes.
<br>
<br>
</div>
</td></tr></table>

<A HREF="Dissertation.pdf">Download the full report (Soon to be
finished)</A>
</body>
</html>
```

e-mail test messages

Test message to !#$%&'*+-/=?^_`{|}~@glennturner.co.uk

```
Received: From pcow004o.blueyonder.co.uk [195.188.53.119] by
mailserver04.fasthosts.co.uk
   (Matrix SMTP Mail Server v(1.3)) ID=0CB55809-2246-4991-9833-
7BD996B7C921 ; Mon, 29 Apr 2002 15:46:32 +0000
Received: from mail pickup service by blueyonder.co.uk with
Microsoft SMTPSVC;
       Mon, 29 Apr 2002 15:46:53 +0100
Content-Class: urn:content-classes:message
From: <glennturner@blueyonder.co.uk>
To: <!#$%&'*+-/=?^_`{|}~@glennturner.co.uk>
Subject: test
Date: Mon, 29 Apr 2002 15:46:52 +0100
Message-ID: <61ae01c1ef8c$b3825e90$7735bcc3@blueyonder.net>
MIME-Version: 1.0
Content-Type: text/plain;
      charset="us-ascii"
Content-Transfer-Encoding: 7bit
X-Mailer: Microsoft CDO for Windows 2000
Thread-Index: AcHvjLOAyl+ndlt+EdaQvQCQJ9GOQA==
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2014.211
X-RCPT-TO: <!#$%&'*+-/=?^_`{|}~@glennturner.co.uk>
```

ASCII code table

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	DEL

Regular Expression Syntax

Core JavaScript Reference 1.5, [Online], Available from URL: *http://developer.netscape.com/docs/manuals/js/core/jsref15/regexp.html#1207831*, [2001 Nov. 15]

Character	Meaning
\	For characters that are usually treated literally, indicates that the next character is special and not to be interpreted literally. For example, `/b/` matches the character 'b'. By placing a backslash in front of b, that is by using `/\b/`, the character becomes special to mean match a word boundary. -or- For characters that are usually treated specially, indicates that the next character is not special and should be interpreted literally. For example, `*` is a special character that means 0 or more occurrences of the preceding character should be matched; for example, `/a*/` means match 0 or more a's. To match `*` literally, precede the it with a backslash; for example, `/a\*/` matches 'a*'.
^	Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character. For example, `/^A/` does not match the 'A' in "an A", but does match the first 'A' in "An A."
$	Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character. For example, `/t$/` does not match the 't' in "eater", but does match it in "eat".
*	Matches the preceding item 0 or more times. For example, `/bo*/` matches 'boooo' in "A ghost booooed" and 'b' in "A bird warbled", but nothing in "A goat grunted".
+	Matches the preceding item 1 or more times. Equivalent to `{1,}`. For example, `/a+/` matches the 'a' in "candy" and all the a's in "caaaaaaandy".
?	Matches the preceding item 0 or 1 time. For example, `/e?le?/` matches the 'el' in "angel" and the 'le' in "angle." If used immediately after any of the quantifiers `*`, `+`, `?`, or `{}`, makes the quantifier non-greedy (matching the minimum number of times), as opposed to the default, which is greedy (matching the maximum number of times). Also used in lookahead assertions, described under (?=), (?!), and (?:) in this table.
.	(The decimal point) matches any single character except the newline character.

	For example, `/.n/` matches 'an' and 'on' in "nay, an apple is on the tree", but not 'nay'.
`(x)`	Matches 'x' and remembers the match. These are called capturing parentheses. For example, `/(foo)/` matches and remembers 'foo' in "foo bar." The matched substring can be recalled from the resulting array's elements `[1]`, ..., `[n]` or from the predefined `RegExp` object's properties `$1`, ..., `$9`.
`(?:x)`	Matches 'x' but does not remember the match. These are called non-capturing parentheses. The matched substring can not be recalled from the resulting array's elements `[1]`, ..., `[n]` or from the predefined `RegExp` object's properties `$1`, ..., `$9`.
`x(?=y)`	Matches 'x' only if 'x' is followed by 'y'. For example, `/Jack(?=Sprat)/` matches 'Jack' only if it is followed by 'Sprat'. `/Jack(?=Sprat|Frost)/`matches 'Jack' only if it is followed by 'Sprat' or 'Frost'. However, neither 'Sprat' nor 'Frost' is part of the match results.
`x(?!y)`	Matches 'x' only if 'x' is not followed by 'y'. For example, `/\d+(?!\.)/` matches a number only if it is not followed by a decimal point. `/\d+(?!\.)/.exec("3.141")` matches 141 but not 3.141.
`x|y`	Matches either 'x' or 'y'. For example, `/green|red/` matches 'green' in "green apple" and 'red' in "red apple."
`{n}`	Where `n` is a positive integer. Matches exactly `n` occurrences of the preceding item. For example, `/a{2}/` doesn't match the 'a' in "candy," but it matches all of the a's in "caandy," and the first two a's in "caaandy."
`{n,}`	Where `n` is a positive integer. Matches at least `n` occurrences of the preceding item. For example, `/a{2,}` doesn't match the 'a' in "candy", but matches all of the a's in "caandy" and in "caaaaaaandy."
`{n,m}`	Where `n` and `m` are positive integers. Matches at least `n` and at most `m` occurrences of the preceding item. For example, `/a{1,3}/` matches nothing in "cndy", the 'a' in "candy," the first two a's in "caandy," and the first three a's in "caaaaaaandy". Notice that when matching "caaaaaaandy", the match is "aaa", even though the original string had more a's in it.
`[xyz]`	A character set. Matches any one of the enclosed characters. You can specify a range of characters by using a hyphen. For example, `[abcd]` is the same as `[a-c]`. They match the 'b' in "brisket" and the 'c' in "ache".
`[^xyz]`	A negated or complemented character set. That is, it matches anything that is not enclosed in the brackets. You can specify a range of characters by using a hyphen. For example, `[^abc]` is the same as `[^a-c]`. They initially match 'r' in "brisket" and 'h' in "chop."
`[\b]`	Matches a backspace. (Not to be confused with `\b`.)
`\b`	Matches a word boundary, such as a space. (Not to be confused with `[\b]`.) For example, `/\bn\w/` matches the 'no' in "noonday";`/\wy\b/` matches the 'ly' in "possibly yesterday."
`\B`	Matches a non-word boundary. For example, `/\w\Bn/` matches 'on' in "noonday", and `/y\B\w/` matches 'ye' in

	"possibly yesterday."
\c*X*	Where *X* is a letter from A - Z. Matches a control character in a string. For example, `/\cM/` matches control-M in a string.
\d	Matches a digit character. Equivalent to `[0-9]`. For example, `/\d/` or `/[0-9]/` matches '2' in "B2 is the suite number."
\D	Matches any non-digit character. Equivalent to `[^0-9]`. For example, `/\D/` or `/[^0-9]/` matches 'B' in "B2 is the suite number."
\f	Matches a form-feed.
\n	Matches a linefeed.
\r	Matches a carriage return.
\s	Matches a single white space character, including space, tab, form feed, line feed. Equivalent to `[ \f\n\r\t\u00A0\u2028\u2029]`. For example, `/\s\w*/` matches ' bar' in "foo bar."
\S	Matches a single character other than white space. Equivalent to `[^ \f\n\r\t\u00A0\u2028\u2029]`. For example, `/\S/\w*` matches 'foo' in "foo bar."
\t	Matches a tab.
\v	Matches a vertical tab.
\w	Matches any alphanumeric character including the underscore. Equivalent to `[A-Za-z0-9_]`. For example, `/\w/` matches 'a' in "apple," '5' in "$5.28," and '3' in "3D."
\W	Matches any non-word character. Equivalent to `[^A-Za-z0-9_]`. For example, `/\W/` or `/[^$A-Za-z0-9_]/` matches '%' in "50%."
n	Where *n* is a positive integer. A back reference to the last substring matching the *n* parenthetical in the regular expression (counting left parentheses). For example, `/apple(,)\sorange\1/` matches 'apple, orange', in "apple, orange, cherry, peach." A more complete example follows this table.
\0	Matches a NUL character. Do not follow this with another digit.
\xhh	Matches the character with the code hh (two hexadecimal digits)

\uhhhh	Matches the character with code hhhh (four hexadecimal digits).

Figure 33: Regular Expression Syntax

Source code for verification application

Verify.pas

```
unit verify;
{$DEFINE UsingWindows}

interface
{
*********************************************************************

  Finds e-mail addresses and verifies them (if chosen)

  Written by : Glenn Turner
  Purpose    : As part of a final year degree project at
               Gloucestershire University
  Start Date : 21/01/2002
  Updated    : N/A

  Thanks to:

  Andrey V. Sorokin <anso@mail.ru> http://anso.virtualave.net/
    for porting regular expressions to delphi

  Henry Spencer (University of Toronto)
    for developing the original regular expression C sources in 1986

  Albert A. Mavrin<amavr@yahoo.com>
    for creating the DirDialog component for Delphi 32 (May 1999)
    http://www.geocities.com/amavr

  Jon Wise at Gloucestershire University
    for general advice on creation of this program

*********************************************************************
}
uses
  SysUtils, Types, Classes, QGraphics, QControls, QForms, QDialogs,
  QStdCtrls, QTypes, QExtCtrls, IdBaseComponent, IdComponent,
  IdTCPConnection, IdTCPClient, IdHTTP, QComCtrls, verifyaddr,
  StrUtils, QGrids, IniFiles, SyncObjs,
  RegExpr, stringqueue, dirdialog; // <- My modules

type
  TVerifythread = class(TThread)
    private
       // Internal variables
       Verify1      : TVerifier;// Object that can verify addresses.
       Iamrunning   : boolean;  // If this thread running.
       ThreadNo     : Integer;  // Unique ID for this object/thread.
       FinderRunning: boolean;  // Is this finder thread running?

       // Input Info
       DNSserver    : string;   // The DNS server to use
       ThisSerName  : string;   // Call ourself this when 'SMTPing'
       Temail       : string;   // The e-mail verifying
       Femail       : string;   // From e-mail used 'SMTPing'.
       VRFY         : boolean;  // Whether we want to use VRFY
       EmailInfo    : string;   // Where it was found (positions)
       EmailSource  : string;   // Source of e-mail i.e. file name
```

```
      // Output Info
      Resultdata     : string; // Error code (if any)
      RawResult      : string; // Raw result from SMTP.
      SMTP           : string;

      procedure SynLastOfGroup();
      procedure SynPrepare();
      procedure SynOutput();
      Procedure SynStatus();
    protected
      procedure Execute(); override;
      procedure OnTerminate;
    public
      procedure AssignNumber(Threadnumber : integer);
    end; { of class declaration }

//*****************************************************************
// End of TVerifythread declaration
//*****************************************************************

  TFinderthread = class(TThread)
    private
      stringInput    : Tstrings;    // Input  (String)

      TypeOfSearch   : integer;     // webpage, input, file etc
      TLDcheckOn     : boolean;     // Options
      VerificationOn : boolean;
      Recursive      : boolean;     // Used with directory option

      EmailsFound    : integer;     // Summary results
      Errors         : string;

      emailfound     : string;      // Working data
      wherefound     : string;
      SearchingNow   : boolean;
      Qsize          : Integer;     // Need know queue size so
                                    // we don't overflow it

      Saving2disk    : string;      // When saving to a file
      InputSource    : string;      // Stores source infomation.
      debuginfo      : string;      // For Debugging

      procedure Msgbox();
      procedure SynSearchingNow();
      Procedure SynQSize();
      procedure SynStats();
      procedure SynErrors();
      procedure AddEmail2Q();
      procedure SynInitialize();
      procedure FindInInput();
      procedure FindinDirectory();
      procedure Wait4Queue();

      procedure CheckString(regularexpression : string;
                            var Astring       : string;
                            source            : string;
                            usingFile         : string);
    protected
      procedure Execute; override;
```

```
    end; { of class declaration }

//*****************************************************************
// End of TFinderthread declaration
//*****************************************************************

  TVerifyForm = class(TForm)
    TabControl          : TTabControl;  // On the main form
    ClearButton         : TButton;
    SaveFileButton      : TButton;
    StartButton         : TButton;
    IdHTTP              : TIdHTTP;
    SaveDialog1         : TSaveDialog;
    OpenDialog1         : TOpenDialog;
    Timer1              : TTimer;
    Lblinput            : TLabel;
    InputMemo           : TMemo;
    Lbloutput           : TLabel;
    sbrstatus           : TStatusBar;
    OutputMemo          : TMemo;
    FoundLabel          : TLabel;
    QueuedLabel         : TLabel;
    UsableLabel         : TLabel;
    UnusableLabel       : TLabel;
    NoFoundEdit         : TEdit;
    NoQueuedEdit        : TEdit;
    UsableEdit          : TEdit;
    UnusableEdit        : TEdit;

    SourceGroup         : TGroupBox;    // On source tab
    URLedit             : TEdit;
    RecurSearchCheck    : TCheckBox;
    webpageRadio        : TRadioButton;
    InputTextRadio      : TRadioButton;
    FileRadio           : TRadioButton;
    DirectoryRadio      : TRadioButton;

    ValidationGroup     : TGroupBox;    // On validation tab
    TLDCheck            : TCheckBox;
    ValidationNoteLabel : TLabel;

    VerificationGroup   : TGroupBox;    // On verification tab
    VRFYCheck           : TCheckBox;
    VerificationCheck   : TCheckBox;
    ConNameLabel        : TLabel;
    ConNameEdit         : TEdit;
    DNSLabel            : TLabel;
    DNSserverEdit       : TEdit;
    FromemailEdit       : TEdit;
    FromLabel           : TLabel;
    ThreadsSpinEdit     : TSpinEdit;
    NoThreadsLabel      : TLabel;

    DestinationGroup    : TGroupBox;    // On Destination tab
    Output2ScreenRadio  : TRadioButton;
    Output2FileRadio    : TRadioButton;
    ShowwherefoundCheck : TCheckBox;
    TagFileOnCheck      : TCheckBox;
    ShowSMTPCheckBox    : TCheckBox;

    SMTPGroup           : TGroupBox;    // On SMPT tab
```

```
    TextBrowser         : TTextBrowser;

    StatusGroup         : TGroupBox;    // On Status tab
    ThreadProgressBar   : TProgressBar;
    QueueProgressBar    : TProgressBar;
    ThreadsOnLabel      : TLabel;
    BufferLabel         : TLabel;
    StringGrid1         : TStringGrid;
                                        // On Error tab
    ErrorGroup          : TGroupBox;
    TextBrowser1        : TTextBrowser;

    procedure TabControlChange(Sender: TObject);
    procedure StartButtonClick(Sender: TObject);
    procedure ClearButtonClick(Sender: TObject);
    procedure SaveFileButtonClick(Sender: TObject);
    procedure InputTextRadioClick(Sender: TObject);
    procedure webpageRadioClick(Sender: TObject);
    procedure FileRadioClick(Sender: TObject);
    procedure DirectoryRadioClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure VerificationCheckClick(Sender: TObject);
    procedure Output2FileRadioClick(Sender: TObject);
    procedure FormClose(Sender: TObject;
                        var Action: TCloseAction);

  private
    { Private declarations }
    iFHWrite          : textfile; // File handle to write
                                  // results to  disk.
    saving2disk       : string;
    Caption           : string;

    // Finding / validating
    FindThread        : TFinderthread;
    FindingEmails     : boolean;  // searching for e-mail now?

    // Verifing
    Verifiers         : array[0..64] of TVerifythread;
    VerifierOn        : array[0..64] of boolean;      // Running.
    VerifierID        : array[0..64] of cardinal;     // ThreadId.
    InputQ            : Tstringqueue;
    InputSourceQ      : Tstringqueue;
    InputInfoQ        : Tstringqueue;

    procedure StartVerifyThreads();
    procedure StopAllThreads();
    procedure SaveSettings();
    procedure LoadSettings();
    procedure DumpResults(emailaddress : string;
                          correctness  : string;
                          SMTP         : string;
                          Source       : string;
                          Comments     : string);
  public
    { Public declarations }
  end;

//***************************************************************
```

```
Const
  ValidTLDs =
  '(aero|biz|com|edu|museum|org|name|net|pro|mil|info|int|'      +
  'ac|ad|ae|af|ag|ai|al|am|an|ao|aq|ar|as|at|au|aw|az|ba|'       +
  'bb|bd|be|bf|bg|bh|bi|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|'    +
  'cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|cr|cu|cv|cx|cy|cz|de|dj|'    +
  'dk|dm|do|dz|ec|ee|eg|eh|er|es|et|fi|fj|fk|fm|fo|fr|fx|ga|'    +
  'gb|gd|ge|gf|gh|gi|gl|gm|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|' +
  'hr|ht|hu|id|ie|il|in|io|iq|ir|is|it|jm|jo|jp|ke|kg|'          +
  'kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|' +
  'ma|mc|md|mg|mh|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|'             +
  'mv|mw|mx|my|mz|na|nc|ne|nf|ng|ni|nl|no|np|nr|nu|nz|'          +
  'om|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|pt|pw|py|qa|re|ro|ru|'       +
  'rw|sa|sb|sc|sd|se|sg|sh|si|sj|sk|sl|sm|sn|so|sr|st|sv|sy|sz|' +
  'tc|td|tf|tg|th|tj|tk|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|um|' +
  'us|uy|uz|va|vc|ve|vg|vi|vn|vu|wf|ws|ye|yt|yu|za|zm|zw)';

  //*************************************************************
  // Finds a number between 0-255
  //   e.g.    100-249       120-5/220-225     10-99      0-9
  //*************************************************************
  RegIPNo='(([1-2][0-4][\d])|([1-2][5][0-5])|([1-9][0-8])|[\d])';

  //*************************************************************
  // Finds a IP address in the format [X.X.X.X]
  //*************************************************************
  RegIPaddr = '\[' + RegIPNo + '(\.' + RegIPNo + '){3}\]';

  //*************************************************************
  //Finds domain e.g. server1.test.foo.com (TLD  2 to 6 characters.)
  //*************************************************************
  RegDomain = '(([\w]+[\.])+[\w\d\-]{2,6})';

  // Do the same this time validating the TLD.

  RegDomainTLD = '((([\w][\w\d\-]{1,62})[\.])+' + ValidTLDs + ')';

  //*************************************************************
  // Local part, Note: Two dots cannot be together
  //*************************************************************

  RegLocal =
    '[\w\*\$\?\#\%\&\^\+\-\=\_\`\{\|\}\~\''\!\/]+' +
      '([\.][\w\*\$\?\#\%\&\^\+\-\=\_\`\{\|\}\~\''\!\/]+)*';


  RegExp =
    '(' + RegLocal + ')\@(' + RegDomainTLD + '|' + RegIPaddr + ')';

  RegExp2 =
    '((' + RegLocal + ')\@(' + RegDomain + '|' + RegIPaddr + '))';

  FinderBufferSize = 100;
  BufferWaittime   = 1000; // Milliseconds
var
  VerifyForm : TVerifyForm;

implementation

{$R *.xfm}
```

```
//***************************************************************

procedure TFinderthread.CheckString(regularexpression : string;
                                    var Astring       : string;
                                    source            : string;
                                    usingFile         : string);
var
  res          : boolean;
  first        : boolean;
  finished     : boolean;
  strstart     : string;
  strend       : string;
  RegExpEngine : TRegExpr;
begin
  RegExpEngine := TRegExpr.Create;

  { regular expression precompilation
    ( When you assign Expression property,
      TRegExpr automatically compiles the r.e.).
    Note:
      if there are errors in regular expression
      TRegExpr will raise exception.
  }

  finished := false;
  first    := true;
  while not finished do //Check until no matches left
  begin
    try
      RegExpEngine.Expression := regularexpression;
      try
        if not first then
          res := RegExpEngine.ExecNext // search for next match
        else
        begin
          // search from first position
          res := RegExpEngine.Exec(Astring);
          //r.compile;
          first := false;
        end;
        if res then
        begin // found
          if RegExpEngine.MatchPos[0] > 0 then
          begin
            str(RegExpEngine.matchpos[0], strstart );
            str(RegExpEngine.matchLen[0], strend   );
            emailfound := RegExpEngine.Match[0];

            // Inc counter (e-mail addresses found)
            inc(emailsFound);

            wherefound := source + ' ' + strstart + ' - ' + strend;

            Synchronize(AddEmail2Q); // Adds e-mail to queue

            if VerificationOn then
              Synchronize(VerifyForm.StartverifyThreads);

            Synchronize(SynStats);
          end
```

```
          else
            finished:= true; // not found
        end
        else
          finished:= true;

        except on E:Exception do begin
          // during regular expression compilation or execution
          Errors := Errors + 'Error: "' + E.Message + '"';
          Synchronize(msgbox);
        end;
      end; //try

      except on E:Exception do
      begin // during regular expression compilation or execution
        Errors := 'Error compilation exception.';
        if E is ERegExpr then
          Errors := 'Error compilation exception (RegExpr).';

        Synchronize(msgbox);
        Synchronize(SynErrors);
        // continue exception processing
        raise Exception.Create (E.Message);
      end; //try
    end; //try
  end; //while

  RegExpEngine.Free;
end;

//***************************************************************

procedure TFinderthread.SynInitialize();
begin
   Recursive               := VerifyForm.RecurSearchCheck.checked;
   Saving2disk             := VerifyForm.saving2disk;
   TLDcheckOn              := VerifyForm.TLDcheck.checked;
   Stringinput             := VerifyForm.InputMemo.lines;
   VerificationOn          := VerifyForm.VerificationCheck.checked;
   emailsFound             := 0;
   emailFound              := '';
   Errors                  := '';
   VerifyForm.inputQ       := Tstringqueue.create();
   VerifyForm.InputInfoQ   := Tstringqueue.create();
   VerifyForm.InputSourceQ := Tstringqueue.create();
   debuginfo               := '';
   If VerifyForm.InputTextRadio.checked then TypeOfSearch := 1;
   If VerifyForm.webpageRadio.checked   then TypeOfSearch := 2;
   If VerifyForm.FileRadio.checked      then TypeOfSearch := 3;
   If VerifyForm.DirectoryRadio.checked then TypeOfSearch := 4;
end;

//***************************************************************

procedure TFinderthread.SynStats();
begin
  VerifyForm.NoFoundEdit.text  := cstr(EmailsFound);
  VerifyForm.NoQueuedEdit.text := cstr(VerifyForm.inputQ.size);
end;

//***************************************************************
```

```
procedure TFinderthread.SynQsize();
begin
  Qsize := VerifyForm.inputQ.size;
end;

//***************************************************************

procedure TFinderthread.SynErrors();
begin
  VerifyForm.TextBrowser.text := VerifyForm.TextBrowser.Text+errors;
end;

//***************************************************************

procedure TFinderthread.Msgbox();
begin
  showmessage('VERIFY ERROR:' + debuginfo);
end;

//***************************************************************

procedure TFinderthread.AddEmail2Q();
begin
  if VerificationOn then  // if verifing add to verify queue
  begin
    VerifyForm.InputQ.Add(emailfound);
    VerifyForm.InputSourceQ.Add(InputSource);
    VerifyForm.InputInfoQ.Add(wherefound);
  end
  else
    VerifyForm.DumpResults(emailfound,'','',Saving2disk,wherefound);
end;

//***************************************************************

procedure TFinderthread.SynSearchingNow();
begin
  VerifyForm.FindingEmails := SearchingNow;
  VerifyForm.QueueProgressBar.position :=
    strtoint(VerifyForm.NoQueuedEdit.text);
end;

//***************************************************************

procedure TFinderthread.FindInInput();
var
  strtemp : string;
  i       : integer;
  lineno  : string;
begin
  i := -1;
  SearchingNow := true;
  Synchronize(SynSearchingNow);

  if Stringinput.count > 0 then        // If not empty
    //for i := 0 to Stringinput.Count do // For every line
    while (i <= Stringinput.Count) AND (NOT Terminated) do
    begin
      inc(i);
      if VerificationOn then           // No need to wait for queue
```

```
        Wait4Queue;       // if verifing
      str(i,lineno);
      strtemp := Stringinput[i];
      if TLDcheckon then
        CheckString(regexp, strtemp,
                    'Line : ' + lineno + ' characters ', '')
      else
        CheckString(regexp2, strtemp,
                    'Line : ' + lineno + ' characters ', '');
    end;

  SearchingNow := false;
  Synchronize(SynSearchingNow);
end;

//***************************************************************

procedure TFinderthread.FindinDirectory();
var
  DirDialog   : TDirDialog;
  afiles      : Tstrings;
  path        : string;
  i           : integer;
  FileCaption : string;
  Fstr        : string;
  FileNoOn    : integer;
  sourcefile  : Tstrings;

begin
  afiles := TStringList.Create;

  try
    // Ask for a directory to search through.
    DirDialog := TDirDialog.Create(application);
    DirDialog.Execute;
    path := DirDialog.DirName;
    DirDialog.Free;

    //caption := 'Searching directory... ' + path;
    //VerifyForm.TimerON; // Show searching

    if path <> '' then
    begin
      getfiles(path,'*.*', afiles,
                        Recursive,
                        false,
                        VerifyForm.Caption);
      i := 0;
      while (i <= (afiles.count - 1) ) AND (NOT Terminated) do
      //for i := 1 to afiles.count - 1 do
      begin
        inc(i);

        // Write which file on (results file).
        FileCaption :=  '[ ' + afiles.Strings[i] + ' ]';

        try
          File2String(afiles.Strings[i], Fstr); // Get File
          str2plaintxt(Fstr);                   // binary convert
          sourcefile := TStringList.Create;     // Spilt into rows
          sourcefile.text := fstr;
```

```
          fstr := '';

          // Process line by line
          FileNoOn := 0;
          //for FileNoOn := 1 to sourcefile.count - 1 do
          while (FileNoOn <= (sourcefile.count - 1)) AND
                (NOT Terminated) do
          begin
            inc(FileNoOn);
            if VerificationOn then  // Don't wait for the queue
              Wait4Queue;           // if verifing
            fstr := sourcefile[FileNoOn];
            CheckString(Regexp, fstr, afiles.Strings[i],
                        FileCaption);
          end;
          sourcefile.free;

        except on E:Exception do
          VerifyForm.outputmemo.lines.Add(
            'Error: "' + E.Message + '"');
        end; {try}

      end;
    end;

  finally
    afiles.Free;
  end;

end;

//***************************************************************

procedure TFinderthread.Wait4Queue();
var
  FinishEvent     : Tevent;
begin
  Synchronize(SynQsize); // Get queue size before going into loop

  // Create an Event (Trigger)
  if Qsize > FinderBufferSize then
  begin
    FinishEvent := TEvent.Create(Nil, True, False, 'Finderwait');

    while (Qsize > FinderBufferSize) do
    begin
      // Clear any calls to the event.
      FinishEvent.ResetEvent;

      // Wait (stop processing !!! until event is triggered)
      // Call in "SMTPconDisconnect"
      if FinishEvent.WaitFor(BufferWaittime) <> wrSignaled then
      begin
        // Do nothing. CPU will be idle.
        // CPU Usage will be near 0% (NOT 100%. ike you may think)
        // Its an interrupt not a loop!
      end;
      Synchronize(SynQsize); // Get queue size again.
    end;
    FinishEvent.free;
  end;
```

```
end;

//***************************************************************

procedure TFinderthread.execute();
begin
  Synchronize(SynInitialize); // Get settings from user

  if ((TypeOfSearch = 1) or (TypeOfSearch = 2) or (TypeOfSearch =
3)) then
    FindInInput();
  if TypeOfSearch = 4 then
    FindInDirectory();
end;

//***************************************************************
// End of TFinderthread Thread
//***************************************************************

//***************************************************************
// Start of TVerifythread Thread
//***************************************************************

procedure TVerifythread.Execute();
begin
  Iamrunning := true;
  Synchronize(SynStatus);

  try

    // Initialize variables + synchronize data
    Synchronize(Synprepare);

    // Do while queue not emtpy and not finished!

    While ( (FinderRunning = true) or (Temail <> '') ) AND
          ( NOT Terminated ) do
    begin
      if (Temail <> '') then
      begin
        Verify1                 := TVerifier.Create();
        Verify1.DNSserver       := DNSserver;
        Verify1.ThisServersName := ThisSerName;

        Rawresult := Verify1.verify(Temail, Femail, VRFY);
        if Rawresult = '' then
          resultdata := ' OK '
        else
          resultdata := ' BAD [' + cstr(rawresult) + ']';

        // For debugging
        SMTP := SMTP + verify1.Details;

        // Initialize variables + synchronize data
        try
          Verify1.free;
        finally
          Synchronize(SynOutput);
        end;
      end;
```

```
      Synchronize(Synprepare);  // Must do last
    end;

  finally
    Iamrunning := false;
    Synchronize(SynStatus);
  end;
end;

//***************************************************************

procedure TVerifythread.Onterminate();
begin
  Verify1.free; // Clean Up

  Iamrunning := false;
  Synchronize(SynStatus);
  Synchronize(SynLastOfGroup);
end;

//***************************************************************

procedure TVerifythread.SynPrepare();
begin
  Resultdata      := '';
  SMTP            := '';
  DNSserver       := VerifyForm.DNSserverEdit.text;
  ThisSerName     := VerifyForm.ConNameEdit.text;
  EmailSource     := VerifyForm.inputSourceQ.Remove;
  EmailInfo       := VerifyForm.inputInfoQ.Remove;
  Temail          := VerifyForm.inputQ.Remove;
  Femail          := VerifyForm.FromemailEdit.text;
  VRFY            := VerifyForm.VRFYCheck.checked;
  FinderRunning   := VerifyForm.FindingEmails;

  VerifyForm.StringGrid1.Cells[0, Threadno] :=
    '[' + cstr(threadno) + ']' +
    Temail +
    ' (' + TimeToStr(now) + ')';
end;

//***************************************************************

procedure TVerifythread.AssignNumber(Threadnumber : integer);
begin
  Threadno := Threadnumber;
end;

//***************************************************************

procedure TVerifythread.SynStatus();
begin
  VerifyForm.VerifierID[Threadno] := threadid;
  VerifyForm.VerifierOn[Threadno] := Iamrunning;
end;

//***************************************************************

procedure TVerifythread.SynOutput();
begin
  VerifyForm.DumpResults(Temail, resultdata, SMTP,
```

```
                          EmailSource, EmailInfo);

  VerifyForm.TextBrowser.text  := SMTP; // For 'live action'
  if rawresult = '' then
    VerifyForm.UsableEdit.text :=
      inttostr(strtoint(VerifyForm.UsableEdit.text)+ 1)
  else
    VerifyForm.UnUsableEdit.text :=
      inttostr(strtoint(VerifyForm.UnUsableEdit.text)+ 1);

  VerifyForm.NoQueuedEdit.text := cstr(VerifyForm.inputQ.size);
end;

//***************************************************************

procedure TVerifythread.SynLastOfGroup();
var
  Threadsrunning : integer;
  I              : integer;
  Maxsize        : integer; // No of possible threads running.
begin
  // Checks to see if this last verify thread to terminate

  Threadsrunning := 0;
  // Array starts at 0 so - 1
  Maxsize := VerifyForm.ThreadsSpinEdit.value - 1;

  for I := 0 to maxsize do
  begin
    if VerifyForm.VerifierOn[i] = true then
      inc(Threadsrunning);
  end;

  // This was the last thread to run.
  // Clear up. e.g. Close any open files etc.
  if threadsrunning = 0 then
  begin
    // ******* If writting results to disk *******
    if VerifyForm.Saving2disk <> '' then
      CloseFile(VerifyForm.iFHWrite);
    // ****************************************
  end;
end;

//***************************************************************
// End of TVerifythread Thread
//***************************************************************

procedure TVerifyForm.DumpResults(emailaddress : string;
                                  correctness  : string;
                                  SMTP         : string;
                                  Source       : string;
                                  Comments     : string);
var
  Finaloutput : string;
begin
  // Save or display results, either validated or verified
  Finaloutput := emailaddress + ', ' + correctness;

  if ShowwherefoundCheck.checked then
    Finaloutput := Finaloutput + ', ' + Comments;
```

```
  if TagFileOnCheck.checked then
    Finaloutput := Finaloutput + ', ' + Source;

  if ShowSMTPCheckBox.checked then
    Finaloutput := Finaloutput + ', ' + SMTP;

  if Saving2disk <> '' then
    Writeln(iFHWrite, Finaloutput)
  else
    //OutputMemo.lines.add(Finaloutput);
    // Could use this but the line below is much quicker
    OutputMemo.Append(Finaloutput);
end;

//***************************************************************

procedure TVerifyForm.StartVerifyThreads();
var
  I       : integer;
  Maxsize : integer;
begin
  // Gentleman... Start your threads...
  Maxsize := ThreadsSpinEdit.value - 1; // Array starts at 0 so - 1

  for I := 0 to maxsize do
  begin
    if VerifierOn[i] = false then
    begin
      application.ProcessMessages;
      verifierOn[i] := false;
      verifierID[i] := 0;
      verifiers[i]  := TVerifythread.create(true);
      verifiers[i].FreeOnTerminate := true;
      verifiers[i].AssignNumber(i);
      verifiers[i].resume;
    end;
  end
end;

//***************************************************************

procedure TVerifyForm.StopAllThreads();
var
  i : integer; // Loop control variable.
begin
  // Gentleman... Start stop your threads...

  // Stop the finder thread
  if FindThread <> nil then
    FindThread.Terminate;

  // Stop verify threads
  for I := 0 to high(VerifierOn) do
  begin
    if VerifierOn[i] = true then // if its running stop it
      verifiers[i].Terminate;
  end;
end;

//***************************************************************
```

```
procedure TVerifyForm.TabControlChange(Sender: TObject);
begin
  Sourcegroup.top           := 28;
  ValidationGroup.top       := 28;
  verificationGroup.top     := 28;
  DestinationGroup.top      := 28;
  SMTPGroup.top             := 28;
  StatusGroup.top           := 28;
  ErrorGroup.Top            := 28;

  Sourcegroup.visible       := false;
  ValidationGroup.visible   := false;
  verificationGroup.visible := false;
  DestinationGroup.visible  := false;
  SMTPGroup.visible         := false;
  StatusGroup.visible       := false;
  ErrorGroup.visible        := false;

  Sourcegroup.BringToFront;
  ValidationGroup.BringToFront;
  verificationGroup.BringToFront;
  DestinationGroup.BringToFront;
  SMTPGroup.BringToFront;
  StatusGroup.BringToFront;
  ErrorGroup.BringToFront;

  case Tabcontrol.tabindex of
    0: Sourcegroup.visible       := true;
    1: ValidationGroup.visible   := true;
    2: VerificationGroup.visible := true;
    3: DestinationGroup.visible  := true;
    4: SMTPGroup.visible         := true;
    5: StatusGroup.visible       := true;
    6: ErrorGroup.visible        := true;
  else
    Sourcegroup.visible := true; // Just in case.
  end;
end;

//***************************************************************

procedure TVerifyForm.StartButtonClick(Sender: TObject);
var
  Fstr : string;
begin
  if StartButton.caption = '&Start' then
  begin
    StartButton.caption := '&Stop';
    savesettings;

    ThreadProgressBar.max := ThreadsSpinEdit.value;
    ThreadProgressBar.position := 0;

    // ******* If writting results to disk *******
    if VerifyForm.Saving2disk <> '' then
    begin
      AssignFile(iFHWrite, Saving2disk);
      if fileexists(Saving2disk) then
        append(iFHWrite)
      else
```

```
        rewrite(iFHWrite);
    end;
    // ****************************************

    if FileRadio.checked then
      if OpenDialog1.execute then
      begin
        //InputMemo.Lines.LoadFromFile(OpenDialog1.FileName);
        File2String(OpenDialog1.FileName, Fstr); // Get File
        //str2plaintxt(Fstr);                    // Binary convert
        InputMemo.Lines.Clear;

        //InputMemo.lines.Text := fstr;
        InputMemo.append(Fstr);
      end;

    if webpageRadio.checked then
    begin
      InputMemo.lines.clear;
      InputMemo.append(idHTTP.Get(URLedit.Text));
    end;

    FindThread := TFinderthread.create(true);
    Findthread.FreeOnTerminate := true;
    FindThread.resume;

    // Turn the timer on. No need to turn it off (it does it itself)
    Timer1.enabled := true;
  end
  else
  begin
    StartButton.enabled := false;
    StartButton.caption := '&Start';
    StopallThreads;
    StartButton.enabled := true;
  end;
end;

//*********************************************************************

procedure TVerifyForm.ClearButtonClick(Sender: TObject);
begin
  OutputMemo.lines.clear;
  usableedit.text      := '0';
  UnusableEdit.text    := '0';
  NoFoundEdit.text     := '0'; // Set number e-mails found to zero.
end;

//*********************************************************************

procedure TVerifyForm.SaveFileButtonClick(Sender: TObject);
begin
  if SaveDialog1.Execute then
    OutputMemo.lines.SaveToFile(SaveDialog1.FileName);
end;

//*********************************************************************

procedure TVerifyForm.InputTextRadioClick(Sender: TObject);
begin
  URLedit.enabled          := false;
```

```
  RecurSearchCheck.enabled := false;
  TagFileOnCheck.checked   := false; // Can't show which file on
end;

//***************************************************************

procedure TVerifyForm.webpageRadioClick(Sender: TObject);
begin
  RecurSearchCheck.enabled := false;
  URLedit.enabled          := true;
  TagFileOnCheck.checked   := false; // Can't show which file on
end;

//***************************************************************

procedure TVerifyForm.FileRadioClick(Sender: TObject);
begin
  URLedit.enabled          := false;
  RecurSearchCheck.enabled := false;
  TagFileOnCheck.checked   := false; // Can't show which file on
end;

//***************************************************************

procedure TVerifyForm.DirectoryRadioClick(Sender: TObject);
begin
  URLedit.enabled          := false;
  RecurSearchCheck.enabled := true;
  TagFileOnCheck.checked   := true; // Can show which file on
end;

//***************************************************************

procedure TVerifyForm.Timer1Timer(Sender: TObject);
var
  i              : integer;
  Threadsrunning : integer;
begin
  if length(caption) > 80 then
    VerifyForm.sbrStatus.simpletext := '...' + rightstr(caption, 80)
  else
    VerifyForm.sbrStatus.simpletext := caption;

  Threadsrunning := 0;
  for i := 0 to (ThreadsSpinEdit.value - 1) do
  begin
    if VerifierOn[i] then inc(Threadsrunning);
  end;
  ThreadProgressBar.caption :=
    cstr(Threadsrunning) + ' out of ' + cstr(ThreadsSpinEdit.value);

  ThreadProgressBar.position := Threadsrunning;
  if (Threadsrunning = 0) then
  begin
    QueueProgressBar.position := 0;
    timer1.enabled := false;
  end
  else
  begin
    // don't run this if no threads running
    QueueProgressBar.position := VerifyForm.inputQ.size;
```

```
    Application.ProcessMessages;
    Application.ProcessMessages;
  end;

end;

//***************************************************************

procedure TVerifyForm.FormCreate(Sender: TObject);
begin
  Timer1.Enabled := false;
  FindingEmails  := false; // Currently not finding e-mails
  ThreadProgressBar.max := ThreadsSpinEdit.value;
  ThreadProgressBar.position := 0;
  ThreadProgressBar.caption := '';

  loadsettings;
end;

//***************************************************************

procedure TVerifyForm.FormResize(Sender: TObject);
const
  textstart     = 155;
  memobottomgap = 225;
  CentreGap     = 15;
var
  width     : integer;
  Farleft   : integer;
begin
  width               := (VerifyForm.width - 35) div 2;
  Farleft             := (VerifyForm.width div 2) + 10;

  TabControl.width    := (width * 2) + CentreGap;
  InputMemo.width     := width;
  InputMemo.top       := textstart;
  InputMemo.height    := VerifyForm.height - memobottomgap;

  OutputMemo.left     := InputMemo.left + width + CentreGap;
  OutputMemo.width    := width;
  OutputMemo.top      := textstart;
  OutputMemo.height   := VerifyForm.height - memobottomgap;

  SaveFileButton.top  :=
    OutputMemo.top + OutputMemo.height + CentreGap;

  SaveFileButton.Left :=
    (TabControl.width - SaveFileButton.width) + 7;

  ClearButton.left    :=
    (SaveFileButton.Left - ClearButton.width) - 10;

  ClearButton.top     := SaveFileButton.top;
  StartButton.left    := SaveFileButton.Left - 18;
  FoundLabel.top      := VerifyForm.height - 65;
  QueuedLabel.top     := VerifyForm.height - 45;
  UsableLabel.Top     := VerifyForm.height - 65;
  UnUsableLabel.top   := VerifyForm.height - 45;
  NoFoundEdit.top     := VerifyForm.height - 65;
  NoQueuedEdit.top    := VerifyForm.height - 45;
```

```
  UsableEdit.top      := VerifyForm.height - 65;
  UnusableEdit.Top    := VerifyForm.height - 45;
  Lbloutput.left      := Farleft;
end;

//***************************************************************

procedure TVerifyForm.VerificationCheckClick(Sender: TObject);
begin
  if VerificationCheck.checked then
    begin
      VRFYCheck.enabled       := true;
      DNSserverEdit.enabled   := true;
      FromemailEdit.enabled   := true;
      ConNameEdit.enabled     := true;
      ThreadsSpinEdit.enabled := true;
    end
  else
    begin
      VRFYCheck.enabled       := false;
      DNSserverEdit.enabled   := false;
      FromemailEdit.enabled   := false;
      ConNameEdit.enabled     := false;
      ThreadsSpinEdit.enabled := false;
    end;
end;

//***************************************************************

procedure TVerifyForm.SaveSettings();

var
  {$IFDEF UsingWindows}
    Settings : TMemIniFile;
  {$ENDIF}
begin
  {$IFDEF UsingWindows}
    Settings := TMemIniFile.create('settings.ini');

    // ***** Source tab *****
    Settings.WriteString(
      'general','URLedit',URLedit.text);
    Settings.WriteBool(
      'general','RecursiveSearchCheck', RecurSearchCheck.checked);
    Settings.WriteBool(
      'general','InputTextRadio', InputTextRadio.checked);
    Settings.WriteBool(
      'general','WebpageRadio', webpageRadio.checked);
    Settings.WriteBool(
      'general','FileRadio', FileRadio.checked);
    Settings.WriteBool(
      'general','DirectoryRadio', DirectoryRadio.checked);

    // ************** Validation tab **************
    Settings.WriteBool(
      'general','TLDCheck',TLDCheck.checked);

    // ************** Verification tab **************
    Settings.WriteBool(
      'general','VerificationCheck',VerificationCheck.checked);
    Settings.WriteBool(
```

```
      'general','VRFYCheck',VRFYCheck.checked);
    Settings.WriteInteger(
      'general','Threads', ThreadsSpinEdit.Value);

    Settings.WriteString(
      'general','DNSserverEdit',DNSserverEdit.text);
    Settings.WriteString(
      'general','FromemailEdit',FromemailEdit.text);
    Settings.WriteString(
      'general','ConNameEdit',ConNameEdit.text);

    Settings.UpdateFile;
    Settings.free
  {$ENDIF}
end;

//***************************************************************

procedure TVerifyForm.LoadSettings();

var
  {$IFDEF UsingWindows}
    Settings : TMemIniFile;
  {$ENDIF}
begin
  {$IFDEF UsingWindows}
    Settings := TMemIniFile.create('settings.ini');

    // *************** Source tab ***************
    URLedit.text := Settings.ReadString(
       'general','URLedit','http://www.gyroscope.com/shipping.asp');

    RecurSearchCheck.checked := Settings.ReadBool(
       'general','RecursiveSearchCheck',false);

    InputTextRadio.checked := Settings.ReadBool(
       'general','InputTextRadio',true);

    webpageRadio.checked := Settings.ReadBool(
       'general','webpageRadio',false);

    FileRadio.checked := Settings.ReadBool(
       'general','FileRadio',false);

    DirectoryRadio.checked := Settings.ReadBool(
       'general','DirectoryRadio',false);

    // *************** Validation tab ***************
    TLDCheck.checked := Settings.Readbool(
      'general','TLDCheck',true);

    // *************** Verification tab ***************
    VerificationCheck.checked    := Settings.Readbool(
      'general','VerificationCheck',true);
    VRFYCheck.checked            := Settings.Readbool(
      'general','VRFYCheck',true);
    ThreadsSpinEdit.Value        := Settings.ReadInteger(
      'general','Threads',2);
    DNSserverEdit.text           := Settings.ReadString(
      'general','DNSserverEdit','192.168.0.99');
    FromemailEdit.text           := Settings.ReadString(
```

```
      'general','FromemailEdit','s9701050@glos.ac.uk');
    ConNameEdit.text             := Settings.ReadString(
      'general','ConNameEdit','testminiserver');

    //webpageRadio.checked := true;
    Settings.free
  {$ENDIF}
end;

//***************************************************************

procedure TVerifyForm.FormClose(Sender: TObject;
                                var Action: TCloseAction);
begin
  StopAllThreads;
  Savesettings;
end;

//***************************************************************

procedure TVerifyForm.Output2FileRadioClick(Sender: TObject);
begin
  if SaveDialog1.execute then
  begin
    saving2disk := SaveDialog1.FileName;
  end;
end;

//***************************************************************

end.
```

MXloolup.pas

```
unit MXlookup;

interface

uses
 IdDNSResolver, SysUtils,  messages, Classes, IdBaseComponent,
 IdComponent, IdUDPBase, IdUDPClient,  IdTCPConnection, IdTCPClient,
 IdWhois, QForms;

type
  TMXlookerup = class
  private
    { Private declarations }
    DnsResource: TIdDNSResourceItem; // Used to extract details
    DNSResolver: TIdDNSResolver;
    results: string;

  public
    { Public declarations }
    constructor create();
    destructor  Destroy();     Override;
    function    FullResults(): string;
    function    GetMXrecord(DomainV : string;
                            DNS     : string;
                            timeout: integer): string;

  end;

//*****************************************************************

implementation

constructor TMXlookerup.create();
begin
  DNSResolver := TIdDNSResolver.Create(Application);
end;

//*****************************************************************

destructor TMXlookerup.Destroy();
begin
  //DnsResource.free;  //No need to free this one.
  DNSResolver.free;
  inherited;
end;

//*****************************************************************

function TMXlookerup.GetMXrecord(DomainV : string;
                                 DNS     : string;
                                 timeout : integer):string;
var
  aQueryType: integer;
begin { ConnectBtnClick }
  aQueryType := 15; // MX records
  try
    results := '';
    DnsResolver.Host           := DNS;
    DnsResolver.ReceiveTimeout := TimeOut;
```

```
    DnsResolver.ClearVars;
    with DnsREsolver.DNSHeader do
    begin
      Qr      := False;  // False is a query; True is a response
      Opcode  := 0;      // 1 is an Iquery return <domainname>
      RD      := True;   // Request Recursive search
      QDCount := 1;      // Just one Question
    end;
    DnsREsolver.DNSQDList.Clear;
    with DnsREsolver.DNSQDList.Add do // And the Question is ?
    begin
      if Length(DomainV) = 0 then
        results := 'Domain Not Given!'
      else
        QName := DomainV;
      QType := aQueryType;
      QClass := cIN;
    end;

    try
       DNSResolver.ResolveDNS;
    except
      On Exception do
      begin
       // Do exception handling
       results := 'DNS error!';
      end;
    end;
    GetMXrecord := FullResults;
  except
  end;
end; { ConnectBtnClick }

//*********************************************************************

function TMXlookerup.FullResults() :string;
var
  Idx: Integer;

begin { DisplayResults }
  with DNSResolver do
  begin
    if DnsAnList.Count > 0 then
    begin
      results := '';
      for Idx := 0 to DnsAnList.Count - 1 do
      begin
        DnsResource := DnsAnList[Idx];
        // Only interested in MX records
        if DnsResource.aType = cMx then
          results := results + DNsResource.Rdata.MX.Exchange + ', '
      end;
    end;
  end;
  FullResults := results;
end; { DisplayResults }

//*********************************************************************

end.
```

Verifyaddr.pas

```
unit verifyaddr;

interface

uses
  windows, messages, spin,  SysUtils,  Classes,
  IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,
  IdTelnet, MXlookup, SyncObjs, DateUtils, QForms, IdWinsock,
  idException;

type
  TClientEvent = procedure of object; // Required for events.

  TVerifier = class
  private
    SMTPcon       : TIdTelnet;   // The SMTP connector
    MXlookup      : TMXlookerup; // and MX record finder.

    FinishEvent   : Tevent;  // For event handling.

    ServerName    : string;  // Name of OUR DNS server 4 this program
    OurServerName: string;  // Name of OUR server (this program).
    TOemail       : string;  // Address we are checking.
    FROMemail     : string;  // From address (can be bogus).
    MXser         : string;  // Mail server to 'talk' to.
    SMTPport      : integer; // Port number. Normally 25.

    SMTPlineOn    : integer; // Line on (commands sent).
    Communicating: boolean; // Started? Communicating? Finished?
    VRFYchecking : boolean; // Using the VRFY+EXPN SMTP commands?
    VRFYresult    : integer; // The result of the VRFY command.
    EXPNresult    : integer; // The result of the EXPN command.
    MaxAttempts   : integer; // Maximum attempts to mail server
    MaxSecsWait   : integer; // Maximum seconds wait from server
    CheckEveryMS : integer; // Check response every X Milliseconds

    // These are for formatting return details
    SendColour       : string; // Holds colour for commands sending
    ResponseColour   : string; // Holds colour for responses
    AppComments      : string; // Holds colour for comments
    Newline          : string; // Replaces CR/LF

    function         AfterAT(email: string): string;
    function         BeforeAT(email: string): string;
    function         smtp2html(line: string; color: string): string;
    procedure        SMTPsend(Command: string);
    procedure        SMTPconOnDataAvailable(Buffer: string);
    procedure        SMTPconConnected(Sender: TObject);
    procedure        SMTPconConnect;
    procedure        SMTPconDisconnect;
  public
    { Public declarations }
    SMTPtxt          : string;
    errorcode        : string;
    constructor      Create();
    function         Verified(): boolean;
    function         Verify(Temail : string;
                            Femail : string;
                            VRFY   : boolean): string;
```

```
  protected
    FOnFinished     : TClientEvent;
    FOnErrorOccured : TClientEvent;
  published
    property SMTPportNo     : integer      read  SMTPport
                                           write SMTPport;
    property Verifiying     : boolean      read  Communicating;
    property MXserver       : string       read  MXser;
    property Details        : string       read  SMTPtxt;
    property ThisServersName: string       read  OurServerName
                                           write OurServerName;
    property DNSserver      : string       read  ServerName
                                           write ServerName;
    property OnFinished     : TClientEvent read  FOnFinished
                                           write FOnFinished;
    property OnErrorOccured : TClientEvent read  FOnErrorOccured
                                           write FOnErrorOccured;
  end;

//*********************************************************************

const
  MaxAttempts    = 3;
  MaxSecsWait    = 10;
  CheckEveryMS   = 100;
  SendColour     = '006600';
  ResponseColour = 'FF0000';
  AppComments    = '666600';

implementation

//*********************************************************************

constructor TVerifier.create();
begin
  // Initialize and set defaults.
  mxlookup                := TMXlookerup.create();
  SMTPcon                 := TIdTelnet.Create(Application);
  SMTPcon.OnDataAvailable := SMTPconOnDataAvailable;
  SMTPcon.OnConnect       := SMTPconConnect;
  SMTPcon.OnConnected     := SMTPconConnected;
  SMTPcon.OnDisconnect    := SMTPconDisconnect;

  // initialize variables (in case not given any)
  ServerName              := '192.168.0.99';
  OurServerName           := 'VerifyServer';
  SMTPport                := 25;
  Newline                 := '<br>';
  VRFYchecking            := true;
  MaxAttempts             := 3;
  MaxSecsWait             := 10;
  CheckEveryMS            := 100;
  SendColour              := '006600';
  ResponseColour          := 'FF0000';
  AppComments             := '666600';
end;

//*********************************************************************

function TVerifier.smtp2html(line: string; color: string): string;
```

```
begin
  Line := stringreplace(line,'<','&lt;', [rfReplaceAll]);
  Line := stringreplace(line,'>','&gt;', [rfReplaceAll]);
  result :=
    '<font color="#' + color + '">'+
    '<code>' + line + '</code></font>' + newline
end;

//***************************************************************

function TVerifier.verify(Temail : string;
                          Femail : string;
                          VRFY   : boolean):string;
var
  charat        : integer;   // Used to store '@' position in e-mail
  attempts      : integer;   // Connection attempts.
  waituntil     : Tdatetime; // A Date/Time to wait for when
communicating.
                             // (Over this and the app will
disconnect)
begin
  // Remember the e-mail addresses
  TOemail       := Temail;
  FROMemail     := Femail;
  VRFYchecking  := VRFY;

  // Initialize variables.
  communicating := true;
  attempts      := 0;
  errorCode     := '';
  SMTPtxt       := '';
  VRFYresult    := 0;
  EXPNresult    := 0;
  if VRFYchecking then
    SMTPlineOn  := 10
  else
    SMTPlineOn  := 0;


  // Go get the name of the MX mail server to 'talk' to.
  MXser         := mxlookup.GetMXrecord(AfterAT(TOemail),
ServerName, 20000);

  mxlookup.Free;  // Finished with. So remove from memory.

  // Use first MX record
  charat        := Pos(',', MXser);
  charat        := charat - 1;
  MXser         := copy(MXser, 0, charat);

  if MXser <> '' then
  begin
    // Initialize telnet/SMTP connection.
    SMTPcon.Host  := MXser;
    SMTPcon.port  := smtpport;

    SMTPtxt       := smtp2html('[Found MX server: ' + MXser +
']',AppComments);

    // Make a number of attempts to 'talk' to the mail server
//communicating
```

```
    while (attempts < MaxAttempts) and (Not smtpcon.Connected) do
    begin

      try
        SMTPcon.Connect; // Start connection.
      except
        on EIdSocketError do
        begin
          inc(attempts);
          SMTPtxt := SMTPtxt + newline + 'Socket connect error,
attempt: ' +
                     inttostr(attempts);
        end;
        else
        begin
          // Do exception handling
          inc(attempts);
          SMTPtxt := SMTPtxt + newline + 'Cannot connect, attempt: '
+
                     inttostr(attempts);
        end;
      end; // try
    end; // while

    if smtpcon.Connected then
    begin
      // Normally the "SMTPconOnDataAvailable" will now be
processing WHILE
      // this is executing! We now need to wait for it to finish
before returning
      // the result. Hence the following code.

      // Create an Event (Trigger)
      FinishEvent := TEvent.Create(Nil, True, False, 'Verifywait');

      // Set the Maximum wait time
      waituntil := IncSecond(now, MaxSecsWait);

      // While still 'talking' to mail server and not over maximum
wait time.
      while (communicating = true) AND (comparetime(waituntil, now)
= 1) do
      begin
        // Clear any calls to the event.
        FinishEvent.ResetEvent;

        // Wait (stop processing !!! until event is triggered)
        // Call in "SMTPconDisconnect"
        if FinishEvent.WaitFor(CheckEveryMS) <> wrSignaled then
          begin
            // Do nothing. CPU will be idle.
            // CPU Usage will be near 0% (NOT 100%. ike you may
think)
            // Its an interrupt not a loop!
          end;
      end;

      FinishEvent.free;

      try
        //SMTPcon.disconnect; // Make sure it disconnects.
```

```
      finally
      end;
    end;

    // Emtpy string if no error.
    if VRFYchecking then
      if errorcode = '' then
        begin
          if (VRFYresult = 250) or (EXPNresult = 250) then
            Result := errorcode
          else
            if (VRFYresult = 250) then
              Result := inttostr(EXPNresult)
            else
              Result := inttostr(VRFYresult);
        end
      else
        Result := errorcode
    else
      Result := errorcode;
  end
  else
    Result := '-1';

  If NOT SMTPcon.Connected then
    SMTPcon.Free;
end;

//*********************************************************************

function TVerifier.Verified(): boolean;
begin
  if ErrorCode = '' then
    result := true
  else
    result := false;
end;

//*********************************************************************

procedure TVerifier.SMTPsend(Command: string);
begin
  try
    if (communicating) and SMTPcon.connected then
      SMTPcon.WriteLn(Command);
  finally
  end;
end;

//*********************************************************************

function TVerifier.AfterAT(email: string): string;
var
  charat : integer;
begin
  charat   := Pos('@', email);
  charat   := charat + 1; // Move past the @ sign
  AfterAT  := copy(TOemail, charat, length(email));
end;

//*********************************************************************
```

```
function TVerifier.BeforeAT(email: string): string;
var
  charat : integer;
begin
  charat   := Pos('@', email);
  charat   := charat - 1; // Move before the @ sign
  BeforeAT := copy(TOemail, 0, charat);
end;

//***************************************************************


procedure TVerifier.SMTPconOnDataAvailable(Buffer: string);
const
  CR = #13;
  LF = #10;
var
  Start, Stop, ReturnCode: Integer;
  Command, returned: string;
begin
  try

  Start := 1;
  Stop  := Pos(CR, Buffer);
  if Stop = 0 then
    Stop := Length(Buffer) + 1;

  // while there is data to process from server.
  while ((Start <= Length(Buffer)) and SMTPcon.connected) do
  begin
    // This is the returned data.
    returned := Copy(Buffer, Start, Stop - Start);

    // Get the response code, < 250 is good.
    val(Copy(Buffer, Start, 3),ReturnCode,ReturnCode);

    // Store the server response for later.
    SMTPtxt := SMTPtxt + smtp2html(returned, responsecolour);

    if SMTPlineOn = 12 then
    begin
      VRFYresult := ReturnCode;
      if ReturnCode = 250 then SMTPlineOn := 17 ;//SMTPlineOn + 1;
    end;
    if SMTPlineOn = 14 then
    begin
      EXPNresult := ReturnCode;
      if ReturnCode = 250 then SMTPlineOn := 17 ;//SMTPlineOn + 1;
    end;

    if (ReturnCode <= 250) or
       (SMTPlineOn = 12) or (SMTPlineOn = 13) or
       (SMTPlineOn = 14) or (SMTPlineOn = 15) then
    begin
      case SMTPlineOn of
        // Basic verification
        0: Command := 'HELO ' + ThisServersName; //HELO
        1: Command := 'MAIL FROM: <' + FROMemail + '>';
        2: Command := 'RCPT TO: <' + TOemail + '>';
        3: Command := 'quit';
```

```
          4: SMTPconDisconnect;

          // For verification when using VRFY/EXPN
          10: Command := 'EHLO ' + ThisServersName; //HELO
          11: Command := 'VRFY ' + TOemail; //+ BeforeAT(TOemail); '
          12: Command := 'RSET' + #13 + #11;//CRLF;
          13: Command := 'EXPN ' + BeforeAT(TOemail);
          14: Command := 'RSET' + #13 + #11;
          15: Command := 'MAIL FROM: <' + FROMemail + '>';
          16: Command := 'RCPT TO: <' + TOemail + '>';
          17: Command := 'quit';
          18: SMTPconDisconnect;
          19: Command := ''{nothing} ;
        else
          Command := '';
          SMTPtxt := SMTPtxt +
            smtp2html('[Error Loop failure!]', appcomments);
        end;

        // If no errors send another command.
        if Command <> '' then
        begin
          // Store request for later.
          SMTPtxt := SMTPtxt + smtp2html(command, sendcolour);

          SMTPsend(Command); // Send command to server.
        end;

        SMTPlineOn := SMTPlineOn + 1; // Move to next command.
      end
      else
      begin
        errorcode := inttostr(ReturnCode);
        // time to quit.
        SMTPtxt   := SMTPtxt + smtp2html('[FAILED!]',appcomments);
      end;

      // This section is to 'handle' SMTP responses.

      Start := Stop + 1;
      if Start > Length(Buffer) then
        Break;
      if Buffer[Start] = LF then
        Start := Start + 1;
      Stop := Start;
      while (Buffer[Stop] <> CR) and (Stop <= Length(Buffer)) do
        Stop := Stop + 1;
    end;

    finally
    end;
end;

//***************************************************************

procedure TVerifier.SMTPconConnected(Sender: TObject);
begin
  SMTPtxt := SMTPtxt + smtp2html('[Connected]',appcomments);
end;

//***************************************************************
```

```
procedure TVerifier.SMTPconConnect;
begin
  SMTPtxt := SMTPtxt + smtp2html('[Connecting]',appcomments);
end;

//****************************************************************

procedure TVerifier.SMTPconDisconnect;
begin
  SMTPtxt := SMTPtxt + smtp2html('[Disconnecting]',appcomments);

  //SMTPcon.DisconnectSocket;
  communicating := false;
end;

//****************************************************************
{
destructor TVerifier.Destroy;
begin
  try
    SMTPcon.Free;
  finally
    inherited;
  end;
end; }

//****************************************************************

end.
```

StringQueue.pas

```
unit StringQueue;
interface
uses
    classes, RegExpr, SysUtils;

type
  TStringQueue = class(TObject)

  private
    Data           : TStrings;
    LowerTrigger   : Integer;  // To fire events
    HigherTrigger  : Integer;
    Handled        : integer;
  public
    constructor Create();
    destructor  Kill();
    procedure   Add(Element : string);
    function    Remove(): string;
    procedure   Flush();
    function    Size(): integer;
    function    IsEmpty(): boolean;
  protected
    FOnHigherTrigger        : TNotifyEvent;
    FOnLowerTrigger         : TNotifyEvent;
    FOnEmptyTrigger         : TNotifyEvent;
    FOnNotEmptyTrigger      : TNotifyEvent;
  published
    property StringsHandled : integer      read  Handled;
    property OnHigherTrigger: TNotifyEvent read  FOnHigherTrigger
                                           write FOnHigherTrigger;
    property OnLowerTrigger : TNotifyEvent read  FOnLowerTrigger
                                           write FOnLowerTrigger;
    property OnEmpty        : TNotifyEvent read  FOnEmptyTrigger
                                           write FOnEmptyTrigger;
    // execute whenever add/remove methods called and
    // the queue is not empty
    property OnNotEmpty     : TNotifyEvent read  FOnNotEmptyTrigger
                                           write FOnNotEmptyTrigger;
end;

function  cstr(value : variant): string;
procedure Str2PlainTxt(var str : string);
procedure File2string(FileName : string; var Filecontent : string);
procedure GetFiles(APath: string; AExt: string; var AList: TStrings;
ARecurse: boolean; AShowDirs : boolean; var Status : string);

implementation

//*****************************************************************

procedure Str2PlainTxt(var str : string);
var
  i : Cardinal;
begin
  if str <> '' then
    for i := 0 to length(str) do
    begin
      if Not ( ((ord(str[i]) < 128) and (ord(str[i]) > 31)) or
         ((ord(str[i]) = 13) or (ord(str[i]) = 10)) ) then
        str[i] := chr(32);
```

```
    end;
end;

//***************************************************************

function cstr(value : variant): string;
// Casts any data type (hopefully!) to a string type
var
  temp : string;
begin
  try
    temp := value;
    cstr := temp;
  except
    on E: Exception do
      cstr := '';
      //ErrorDialog(E.Message, E.HelpContext);
  end;
end;

//***************************************************************

procedure File2string(FileName : string; var Filecontent : string);
var
  Stream  :  TMemoryStream;
begin
  Filecontent := '';
  if fileexists(Filename) then
  begin
    Stream := TMemoryStream.Create;
    try
      Stream.LoadFromFile(Filename);
      SetLength(Filecontent, Stream.Size);
      Stream.Read(Filecontent[1], Stream.Size)
    finally
      Stream.Free
    end
  end;
end;

//***************************************************************

procedure GetFiles(APath: string; AExt: string; var AList: TStrings;
ARecurse: boolean; AShowDirs : boolean; var Status : string);
var
  theExt    : string;
  searchRec : SysUtils.TSearchRec;
begin
  status := 'Searching : ' + APath;
  if APath[Length(APath)] <> '\' then
    APath := APath + '\';
  if AShowDirs then AList.AddObject(APath, Pointer(-1));
  if FindFirst(APath + '*.*', faAnyFile, searchRec) = 0 then repeat
    with searchRec do begin
      if (Name <> '.') and (Name <> '..') then
        if (Attr and faDirectory <= 0) then
        begin
          theExt := '*' + UpperCase(ExtractFileExt(searchRec.Name));
          if (AExt = '*.*') or (theExt = UpperCase(AExt)) then
            AList.AddObject(APath + searchRec.Name, Pointer(0))
        end
```

```
        else
        begin
          if ARecurse then
            GetFiles(APath + Name + '\', AExt, AList,
                     ARecurse, AShowDirs, status);
        end;
    end; {with searchRec...}
  until FindNext(searchRec) <> 0;
  SysUtils.FindClose(searchRec);
end;

//*********************************************************************

constructor TStringQueue.Create();
begin
  Data := TStringList.Create;
  Handled := 0;
end;

//*********************************************************************

destructor TStringQueue.Kill();
begin
  Data.Destroy;
end;

//*********************************************************************

procedure TStringQueue.Flush();
begin
  Data.Clear;
  Handled := 0;
  if assigned (FonemptyTrigger) then
    FOnemptyTrigger(self);
end;

//*********************************************************************

procedure TStringQueue.Add(Element : string);
begin
  data.Add(Element);
  inc(Handled);
  // If higher trigger reached, fire the event.
  if highertrigger > 0 then
    if highertrigger = data.count then
       if assigned (Fonhighertrigger) then
          FOnHigherTrigger(self);
  // If not empty fire event.
  if data.count > 0 then
    if assigned(FOnNotEmptyTrigger) then
       FOnNotEmptyTrigger(self);
end;

//*********************************************************************

function TStringQueue.Remove(): string;
begin
  If data.count <> 0 then
  begin
    result := Data.Strings[0];
    Data.Delete(0);
```

```
  end
  else
    result := '';

  // If lower trigger reached, fire the event.
  if lowertrigger >= 0 then
    if lowertrigger = data.count then
      if assigned (Fonlowertrigger) then
        FOnlowerTrigger(self);

  // If empty, fire the event.
  if 0 = data.count then
    if assigned (FonemptyTrigger) then
      FOnemptyTrigger(self);
end;

//***************************************************************

function TStringQueue.IsEmpty(): boolean;
begin
    result := (data.count = 0 );
end;

//***************************************************************

function TStringQueue.Size(): integer;
begin
    result := data.count;
end;

//***************************************************************

end.
```