

Notification Service for e-commerce Systems

By

Johan Sandberg
2003-10-23

Examiner: Vladimir Vlassov

Abstract

The objective of this thesis was to construct and analyze scalability of a system that handles notification messages between customers and dealers in an e-commerce system. The communication methods analyzed in this thesis are e-mail, fax and SMS. The SMS functionality was analyzed, but not implemented due to cost reasons.

The system contains a web-based administration GUI, which allows a system administrator to set user preferences, extract information from log database, and to create invoice data.

An analysis of how system scalability affects delivery time pointed out that the main limitation was the time consumption used for connecting and sending a fax.

Security problems with unencrypted GET requests were identified, and the suggested solution was to have an encrypted login page.

Advantages and drawbacks for different platforms, scripting languages and third party software were investigated. The software selected for building the application was a Windows server with SQL Server and ASP scripting language.

Table of Contents

1. Introduction	5
1.1. Project Background	5
1.2. Problem Definition	5
1.3. Report Layout	6
2. Existing System	7
2.1. Summary	7
2.2. System Overview	7
2.3. Three-tier System Architecture	8
2.4. Technologies	9
2.5. Users Interface	9
2.6. Dealers Interface	12
2.7. Administrative Interface	13
2.8. Use Cases	14
2.9. Database	16
3. Introduction to Currently Used Technologies	17
3.1. Servers	17
3.2. Programming Languages	18
3.3. Objects and Components	19
4. Notifications	21
4.1. E-mail	21
4.2. Fax	21
4.3. SMS	23
5. Notification System Design	24
5.1. Notification use cases	24
5.2. System Design	25
5.3. Security Considerations	26
5.4. Sending Notifications	26
5.5. Fax Object	27
5.6. Administrating the System, Overview	28
5.7. Preferences	29
5.8. Statistics	30
5.9. Database Table Structure	31
6. Hardware Installation	33

6.1. Preparations	33
6.2. Installing Fax Proxy Object.....	33
6.3. Installation Verification	33
7. <i>System Integration</i>	34
8. <i>Performance Evaluation</i>	35
8.1. Expected Performance	35
8.2. Delays.....	35
9. <i>Conclusions</i>	37
<i>References</i>	38
<i>Appendix A</i>	39
<i>Appendix B</i>	40
FaxServer	40
FaxDoc	40
<i>Glossary</i>	41

1. Introduction

1.1. Project Background

The background of this thesis stems from the fact that the use of modern Internet based communication is not very extensive in some business areas. One example is dealers of snowmobiles. An increasing number of dealers realize the convenient power the Internet can provide i.e. e-mail. Even today, many dealers do not take advantage of it. Therefore this thesis was done with focus on developing an Internet based system that interacts with users that don't use e-mail.

This thesis has been done in cooperation with Scandinavian TradeCenter AB. Therefore system development was done for primary use with the existing TradeCenter e-commerce system. Also a large portion of the thesis is dedicated to investigation and analysis of the existing e-commerce system.

1.2. Problem Definition

An e-commerce system need the ability to send information to users not connected to the Internet. This problem occurs in those business areas where the frequency of computer usage is low, in example machinery dealers.

A decade ago, the most common text based communication where fax machines. They are still common, and almost every company today uses at least one fax machine to communicate with customers. As an example one can look at the hotel business, where until recently fax was the best way to make a room reservation. Nowadays there are many sites on the Internet offering room reservations online.

One should be aware that this is a decreasing problem, as more and more users turn to e-mail communication. However, this is a slow process and it tends to slow down as the remaining non e-mail users, are in general very conservative.

The massive use of e-mail has brought us one big drawback, SPAM. SPAM highly reduces communication efficiency, much like noisy or cross-hearing telephone conversations. Much of this depends on the fact that it is basically free to send e-mail. With faxes there are a lot more costs involved since it is a service using a dedicated connection in the telephone system with connection and traffic fees.

The problem in this thesis was divided into four parts:

1. Investigation of the existing system

All parts of the existing e-commerce system needed to be investigated and documented. The system investigation included overall system-, database- and code design. Since the system is very complex and contains a lot of advanced functionality that is not public information, there might be some gaps in the documentation. Generalizations have been used to describe functionality, and the code has in some cases been described in pseudo language.

2. Design an implementation strategy for a fax notification service

The implementation strategy contains some non-public information, especially on the existing system side. However, the fax notification service has a public interface that can be used by any application. This has made the new notification system highly available for other applications.

3. Design and implement a fax notification service

The design and implementation have been documented for future references. The actual source code is not public, but for academic purposes a skeleton of the code has been included in this thesis.

4. Performance evaluation of the fax notification service

The new fax notification service was performance tested and also theoretically described. This was important since faxes are physically very limited due to long delivery time. A fax takes about 1.5 minutes to send, which naturally leads to the conclusion that the 3600 minutes of a day is the main limitation of system performance. However, server performance was also tested, in order to make sure that queuing buffers could hold enough information. The sending limitation is also affected by economical decisions, since sending might be rerouted to non prime-time hours, effectively reducing active sending time to 12 hours per day.

1.3. Report Layout

The first three sections explain how the existing system was built and the technologies that have been used.

In section number four the system notification service is explained.

Section number five describes how the new notification service was designed.

Section six and seven describe the software and hardware implementation.

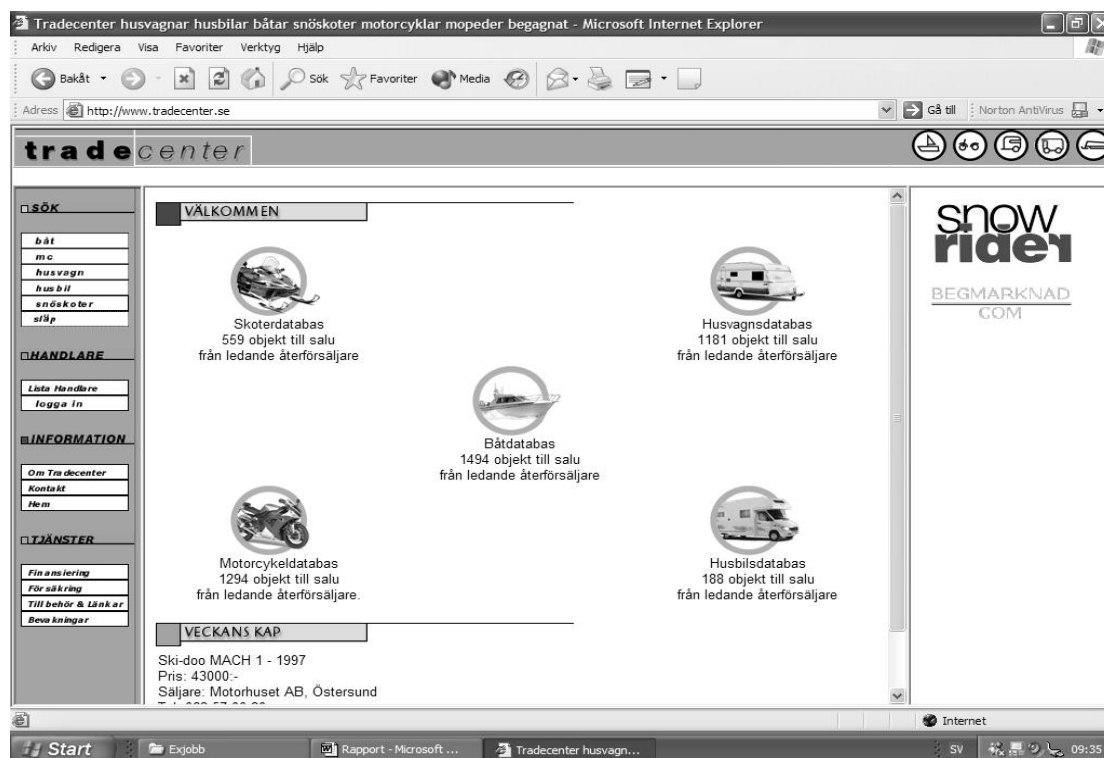
Section number eight contains a system performance evaluation.

Section nine gives conclusions of the thesis work.

2. Existing System

2.1. Summary

TradeCenter is an Internet web based advertises management system for dealers of recreational vehicles such as RV's, caravans, snowmobiles and boats. The system offers people to find vehicles of interest by using a search engine. For dealers the system offers an easy and user-friendly platform for registering of objects for sale on the Internet. See Picture 1: GUI front page for system GUI.



Picture 1: GUI front page

2.2. System Overview

The system consists of three user interfaces one for buyers, one for dealers and one for system administration. All user interfaces are web based, designed to function with Internet Explorer, and contains both static and dynamic pages.

The interfaces are hosted on an Internet Information Server, running on a Windows NT Server. For data storage an SQL server is accessed through an ODBC connection. The web server also runs a statistic server for analysis of web traffic, user behavior and search engine keywords.

For direct database access a DBMS system can be used. The most common are SQL query analyzer and SQL server manager, provided by Microsoft Corporation.

For a graphical layout of system structure, see Figure 1: System overview.

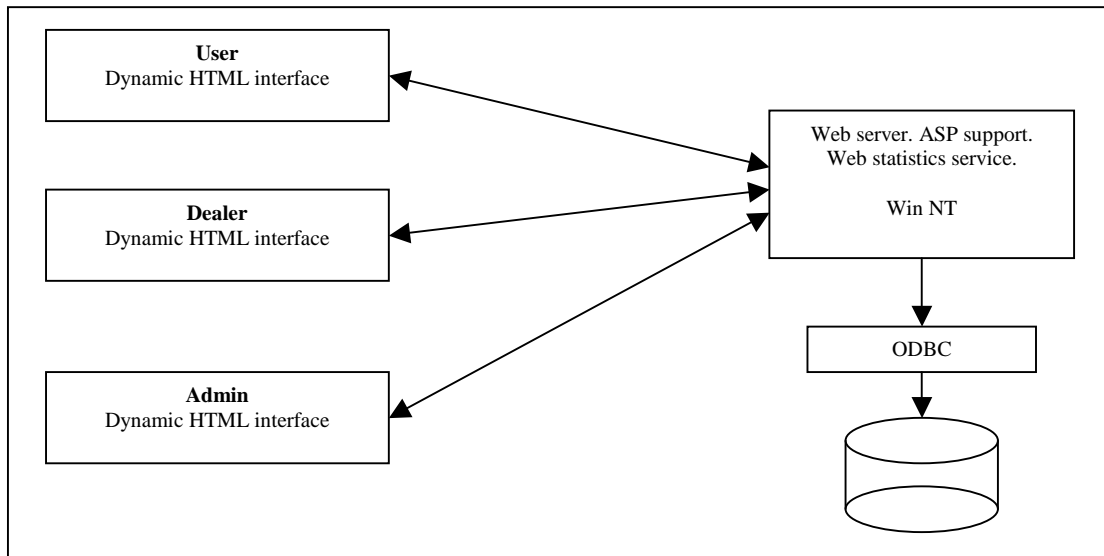


Figure 1: System overview

2.3. Three-tier System Architecture

The TradeCenter e-commerce system is based on a three-tier architecture, which makes it easy to implement system changes and performance optimizations for distinct parts of the system.

Three-tiers

The 1st tier is the user front GUI based on HTML pages that can be accessed from any web browser on a computer connected to the Internet (Picture 1: GUI front page). The GUI has been designed to provide system front-end implementations for other sites as well as for the TradeCenter site. In example external companies that are interested in providing their customers with a vehicle database can use a TradeCenter front-end implementation with a graphical design that fits into their WebPages. Vehicle manufacturers and magazines are typical users of this function. A GUI can have any graphical layout as long as the required parameters are included in the HTML forms used to submit data.

The 2nd tier consists of the web service and the dynamic WebPages with built in system specific functions. A web server supports two operations, POST and GET requests. By extending the web server with a scripting module such as ASP, it becomes possible to create a dynamic web page content. ASP scripting language has access to many easy to use functions and objects. Calculations and text modifications are made on the fly, based on data provided by either a GET request or a GET request in combination with database access. Database accesses are done through an ODBC connection object.

The 3rd tier is the data storage, powered by a MS SQL server.

Other System Services

Other services are also running in the system. None of these services are part of the user, dealer or administrative system interaction. There are currently two other system services running one for exporting and one for importing data as text files. These services are not visible outside the system and are architecturally considered as a part of the internal system functionality. Since they are implemented specifically for

different requirements, they have no user interfaces. Normal operations for these services are writing and reading text files from predefined locations.

Database Jobs

Other stand-alone services are the database jobs. They are responsible for maintaining the internal database structure, verifying field data and updating “once a day” counters. These jobs are also considered to be an internal system function and are handled by the built in MS SQL server agent.

2.4. Technologies

The TradeCenter system is built mainly on ASP scripting language for WebPages. Some pages do also contain java scripts for easier manipulation of WebPages, operations such as changing text onClick, bringing browser windows into focus and printing functions. Code layout is based on HTML with inline ASP, and ASP that writes HTML code.

Other technologies are used for more advanced operations, such as file import and data modification. Currently one import function takes advantage of the Java StringTokenizer class for easy and powerful text processing.

Export scripts are written in Visual Basic code for convenient command line execution. This also makes scheduling with Windows NT built in “at” command easy.

All file transfers between the system and external sources are accomplished through the standard FTP protocol with a control text file for unattended operation, also initiated by the built in “AT” command.

2.5. Users Interface

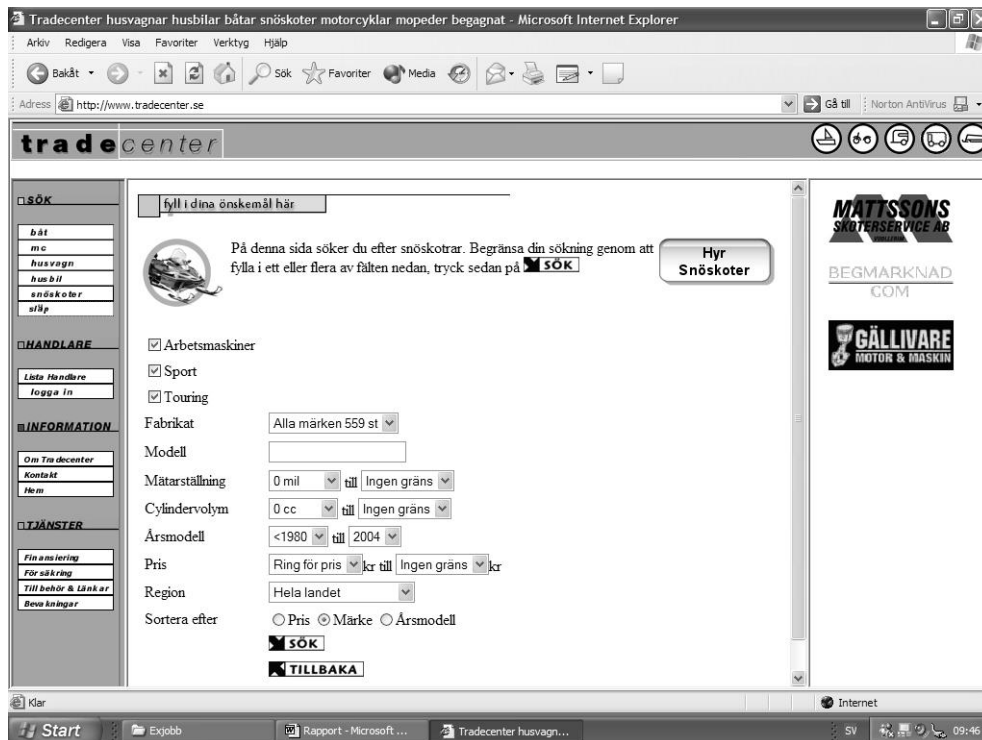
The system offers users to search for vehicles with individual criteria such as weight, engine size, production year or price. Once specified, the system can search the database to find matching objects, and present them in a structured way for the user. The user can then get more detailed information about any of the matching objects. If a user is interested in an object, it is possible to fill out a form with contact information and submit a note, so that the dealer of the object can contact the user.

Front Page

The front page of the system (Picture 1: GUI front page) contains an easy to understand environment showing types of vehicles the user can search for in the system. Other information, such as contact information and financial information is also available in the front page.

There are also objects from a selected weekly dealer shown in the front page. Dealers can purchase this space for any week.

The main purpose of the front page is to give a graphical interface that allows users to select any machinery type, and then redirect the user to the search page.



Picture 2: Search page

Search Page

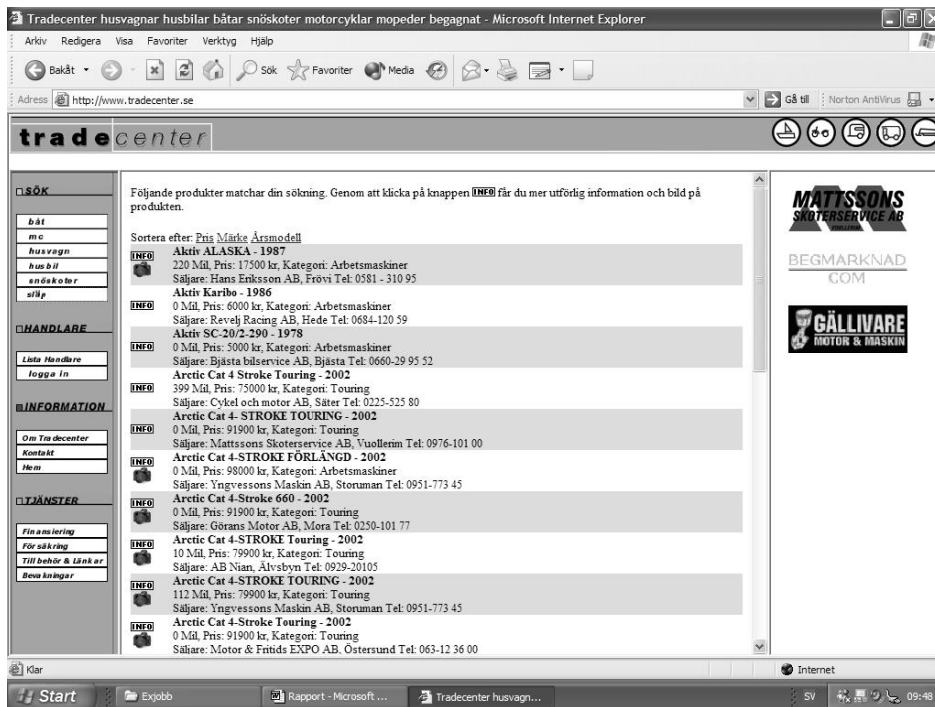
The search page (Picture 2: Search page) is called with the parameter machineryid, where machinery id is defined in the database. The search page has a different look depending on which number the parameter machineryid holds. This allows the system to have different predefined selections for different machineries. One example of differences is the price interval, which for example differs a lot between motorcycles and boats. Other customizations are the prefix and unit of the attr1 to attr6 fields. Attr1 to 6 are defined in the database.

When the search page is submitted, all parameters are passed on to the result list page.

Result List

The result list (Picture 3: Result list) contains all matching objects for the criteria the user has selected on the search page. If there are many hits, the page will divide itself into pages containing 25 hits per page. When this happens, the dynamic page can create a reference to itself with the parameter offset, which tells the server side processing how many hits to skip before starting to print on the HTML page.

The role of the result list is to create a link to the object information page with the unique object identity as a parameter. The parameter is called objectid.



Picture 3: Result list

Object Information

The object information page (Picture 4: Object information) shows pictures and detailed information about an object, based on the parameter objectid. Here it's also possible to fill out an interest notification for the current object.



Picture 4: Object information

Interest Notification

If the user presses the button for sending an interest notification (Picture 5: Interest notification), he is asked to fill out some contact information and press "submit". A

notification is thereby sent to the dealer of the current object via e-mail. This option is only available for dealers with e-mail.



Picture 5: Interest notification

2.6. Dealers Interface

Dealers can logon to the TradeCenter system, using an individual username and password, to get access to a special dealer interface that allows them to work with objects in the system. For example a dealer can add or remove objects, create and print storage lists or view different types of statistics.

Welcome Page

When a dealer logs on to the system, a welcome page is shown with links to different functions. On this page there is also a possibility for the system administrator to add information intended for dealers only.

The welcome page contains an overview of the dealer's stock value and quick buttons for showing and printing storage lists with or without thumbnail pictures.

Price Statistics

This function allows dealers to show price statistics for objects traded through the TradeCenter system. The function shows a list of average prices divided into rows with distinct machinery fabrication, model and production year.

Object Working Page

This page contains all items the current dealer has active in the system. It is possible to delete one or more objects, create price signs, change object information and create offers with trade-in information. The list also contains information on the number of hits for each object.

Add Object Page

This page can only be accessed once a dealer has logged on to the system. It shows a form that has a different layout depending on the calling parameter machineryid. This allows changing the prefix and unit for the fields attr1 to attr6, which are defined in the database. It also allows the loading of predefined machinery fabrications from the database to minimize typing errors. The form is submitted to the server with the HTML form encType parameter set to multipart/form-data, which allows the sending of images.

Once the page is submitted a verification text is shown to the dealer. URLs back to the main page are also shown.

2.7. Administrative Interface

The system includes an administrative part, used only by TradeCenter staff for managing user accounts, administrating banner advertisements and some statistics. Only privileged users can access the administrative interface.

Dealer Administration

The system administrator can add a new dealer by filling out a form with the dealer's company information. This information is then used to inform users of how to contact the dealer of an object. TradeCenter also uses the information for contacting dealers and sending invoices.

All dealers that are members of the TradeCenter system are shown in an overview list. This list allows the system administrator to easily find a dealer and access detailed information by clicking on a row. Once detailed information is shown, the administrator can update or add missing information. It is also possible to delete a dealer and all associated objects.

Activity Overview

This administrative option allows the TradeCenter system administrator to conveniently, find active, and inactive dealers in the system. The information is presented in a list with each row containing a dealer's company name, the number of active objects in the system and number of days since the last object was registered.

Statistics

The statistics option allows the system administrator to see how many new customers that have been registered into the system, number of objects registered and removed. All statistics are presented per day.

Advertisement Campaigns

The TradeCenter banners advertisement system is built by using numbers for identification of pages in the system. For example the page with snowmobiles is called page number 8, which also is the machineryid for snowmobiles. This makes it possible to assign banners to one or more pages in the system, and also to target selected user groups.

When adding a new banner campaign, the administrator names the campaign, enters a URL to the banner and assigns pages where to display the banner.

By listing all active campaigns, the administrator can get information about how many clicks that has been done for each campaign. It is also possible to end campaigns by clicking on a remove button. All information is then removed from the database.

2.8. Use Cases

This section describes typical use cases for the TradeCenter system. Use cases are object search and interest notification

Object Search

The most common use case of the system is a user connecting to the web page, choosing a machinery type of interest and seeking objects with some search criterias. Eventually the user gets a list with matching objects. From this list it is possible to view more detailed information about an object. This is another use case, not shown here.

First the user fills out a form with search criteria's such as boundaries for production year or price. Once these values meet the user requirements for vehicles the parameters are sent to the server through a standard POST request, containing the form data.

When the server receives the request, server side processing is initiated. First the values of different fields are investigated for possible typing errors and hacker attempts such as entering a single-quote character in free text areas for query manipulation. If the data is uncorrupted an SQL query is built, based on data from the users form. Once this query is constructed, the server executes it through the ODBC driver and thereafter the query is run in the database. The result from the database query is then put into a recordset. If the recordset contains any rows the server starts to generate a dynamic HTML page with a list of objects matching the users search criterias. If the recordset is empty, the server creates a HTML page with a "no hits" message. The data path can be seen in Figure 2: Data path for use case object search.

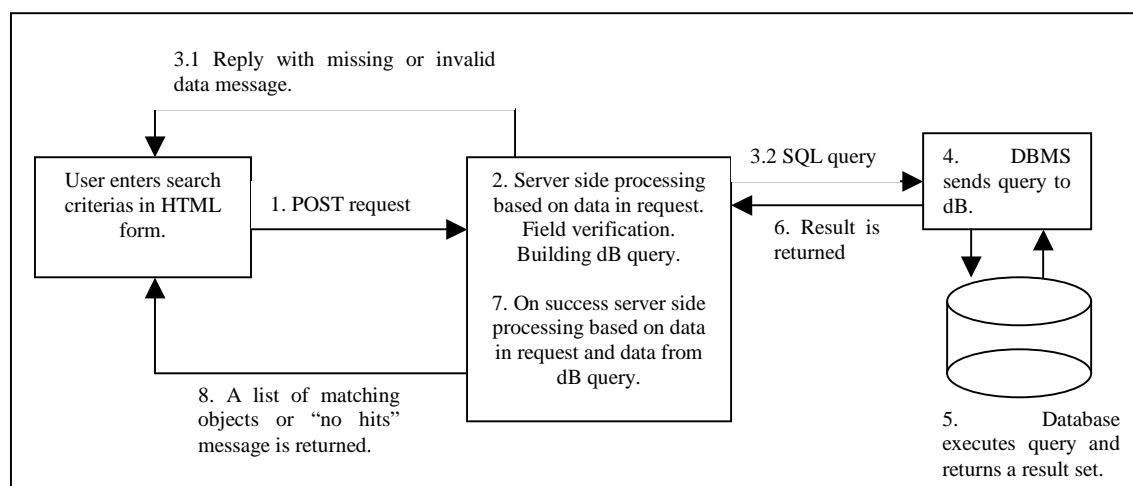


Figure 2: Data path for use case object search

Interest Notification

When a user has found an object of interest, it is possible to fill out an interest notification and send it to the dealer.

This use case begins with a HTML page requesting user contact information, such as telephone number or e-mail address. When the user posts the form to the server, field validation is done to see if any relevant data is missing and to some extent, data verification is done. If any relevant data is corrupt or missing, a reply is sent to the user with error messages containing information on which fields are missing or are corrupt. The user then has to repeat this process until the form is accepted. This form also contains a hidden field with the unique object identity for later identification of the object.

When the input data is accepted, server side processing is started and the server constructs an e-mail message with the built in mail service CDONTS. First a database query is done, requesting the dealer information about the current objects. This information contains the dealers e-mail address, which is put into the e-mail receiver field. Second a message is built based on the user input. The message contains information about how to contact the user and what object he is interested in. Finally the e-mail is send with the CDONTS send command and is forwarded to the SMTP server. When this stage is completed a reply is sent to inform the user that a message has been sent.

The e-mail is now located in the SMTP server sending queue. Once sent it can either be accepted by the dealers e-mail server or for some reason rejected. Usual reasons for errors when sending e-mails are that the receiving user has exceeded his e-mail quota. If this, or an earlier sending error, such as DNS lookup occurs, an error message is send to TradeCenters e-mail server with information on the sending error.

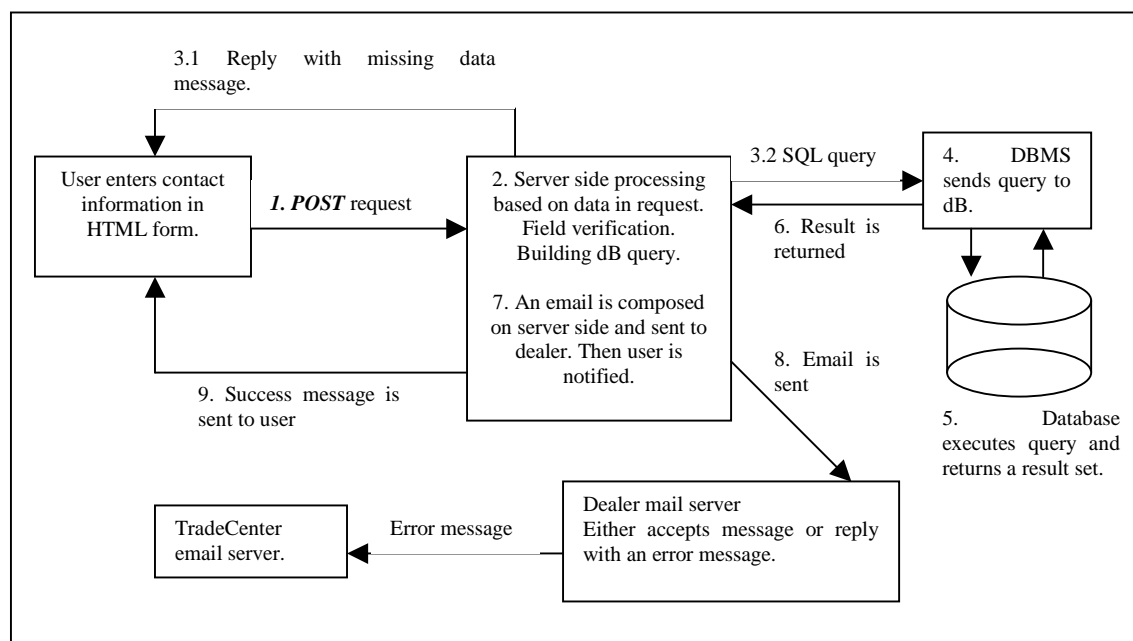


Figure 3: Data path for use case interest notification

Since e-mail is an asynchronous way of communication it is not possible to know when or if a dealer reads the notification. See Figure 3: Data path for use case interest notification for a detailed view.

2.9. Database

The TradeCenter database contains 15 tables. An investigation and documentation has been done on the database structure. The results are not published in this report.

3. Introduction to Currently Used Technologies

This chapter describes the technologies that are currently used in the TradeCenter system. The intention of this chapter is to facilitate the understanding of system descriptions for readers not familiar with the used technologies.

3.1. Servers

Servers are software daemons that run on a dedicated server machine. They are responsible for providing standardized functionality. In this section, there are descriptions of server daemons used by the TradeCenter e-commerce system, how they function and what they are supposed to do.

Windows NT Server

The Windows NT server provides high performance and reliability. The platform is somewhat outdated, but still provides enough functionality and performance. However the platform is no longer supported actively by Microsoft, which means that no more service packs will be released. This means that in the near future an upgrade will probably be made to the latest Windows 2003 platform.

SQL Server

The database management system (DBMS), used by the system is a MS SQL Server 7.0 [3], which also will be replaced with the new SQL Server 2000. This upgrade is transparent to the system, since all database accesses are done through an ODBC driver and are written in the standard SQL language.

MS SQL Server provides reliable storage space and can be scaled to support extremely large databases. This can prove useful, as system storage needs tend to increase. The data reliability can also be increased with MS SQL Server, simply by adding more physical servers. In short both storage and reliability increase can be achieved by using hardware clusters, a technology that has gained much recognition lately.

Web Server

The web server that hosts the system is the built in IIS in Windows NT 4.0. This server is also getting outdated and will be upgraded to IIS 6.0 [5], when the system is upgraded to Windows server 2003.

Since a web server by definition does nothing but transmitting files from a hard drive to a client using HTTP protocol, one can easily realize the limitations of static pages (standard HTML pages). This means that users cannot have personal pages, unless the system administrator creates and stores a page for every individual user. This would be too time consuming and also use a lot of system resources. To overcome this limitation, dynamical pages are used. A dynamic page is created on the fly as it is requested. The dynamic page gathers information from databases and then generates a user-customized page.

IIS includes an application server for dynamic pages. This application, called ASP can in real-time compile script code, access system functions and objects. ASP can be compared to the Unix world by PHP[8] and JSP[9], which are similar technologies.

SMTP Server

The SMTP server used by the system for sending e-mail is an MS Exchange server. This server can be used until the system is upgraded, when it will be replaced with a more cost effective server. All the system needs is a SMTP compatible mail server and an ASP component that allows dynamic pages to send e-mail. Currently the Windows standard CDONTS [1][2] technology is used to send e-mail, and it might not be available on the newer platforms. This implies that the code may have to be adapted, or a CDONTS proxy has to be created.

3.2. Programming Languages

This section gives a short introduction to the programming languages that are used in the system. Some basic syntactic examples are given in the context.

ASP, Active Server Pages

ASP is a technology from Microsoft Corporation®. It's a powerful scripting language that can be used either to write HTML text or to be included in HTML pages.

Below is an example of what is referred as inline and writing ASP code.

```
Example: ASP writing HTML
Response.write "<BR>This is a line on a page<BR>"
Example: ASP included in a HTML page
<BR><%Response.write "This is line on a page"%><BR>
```

ASP has build in objects for handling database access through ODBC drivers. It also provides powerful recordset objects for writing loops in order to create dynamic tables with large data amounts. Dynamic web pages are generated by an ASP page and can be customized for every user. In the TradeCenter system, dynamic pages are used to display information from the database.

Below is an example of a database access through an ODBC driver that put the result in a record set for displaying on a web page.

```
`Create an empty record set
Set rs=Server.CreateObject("ADODB.Recordset")
`Create a connection object
Set conn=Server.CreateObject("ADODB.Connection")
`Open the connection
conn.Open "DSN=testserver;UID=testuser;PWD=secret"
`Select something from the database
query="SELECT prices FROM products"
`Run the query and put the result in the recordset
rs.open query, conn
`Loop through entire record set
while not rs.EOF
    `write text to web page
    Response.write "Price: " & rs("price")
`Move to next line in record set
rs.movnext
wend `End of loop
```

The above example shows how easy and powerful dynamic pages can be, when used in conjunction with databases. This is the basis of all modern Internet sites.

Java

Java is used in one of the system's import functions. An import file is a large text file containing fields separated by semicolons. Import of data can be useful if information already exists in another system. By automatically importing data the user is liberated from doing unnecessary double work.

The reason for building file import tools in Java is that Java contains ready to use classes for string manipulation. Typically an import is done by reading a row from a text file and then splitting it into tokens, using an instance of the StringTokenizer class. After the row has been split into tokens, each token is type and length checked using basic operations. Finally the JDBC driver is used to access the database and an update procedure is done. JDBC works in the same way as Microsoft ODBC technology, which acts like a proxy between the code and the database.

Visual Basic

Visual Basic has a similar code structure as ASP with only small differences, which makes it easy to understand both ASP and Visual Basic. Visual Basic files are named with the extension VBS, and can be executed from the command line without any pre compilation on a Windows system. Because of these properties, VBS has been chosen for handling data exports from the system. Typically an export script connects to the database, gathering data and putting it in a semicolon separated text file. The text file is then either collected or sent to a customer. Export operations are scheduled using the Windows built in scheduler.

Scheduler

Many systems have a need for automatically launching scripts or programs in a predefined manner. Windows has a built in command line utility for this, called AT. It takes a program or script name, time and weekdays as arguments and then schedules the application or script for running accordingly to the parameters passed.

3.3. Objects and Components

This section describes some of the objects and components used in the system.

CDONTS

CDONTS is an ASP component that can be used when the server has an SMTP installation running. The component makes it easy to send e-mail from ASP pages by a simple method calling of the CDONTS component. CDONTS allows sending of plain text messages, HTML messages and mail attachments. It is a powerful component but the sending of e-mail can be limited by SMTP routing restrictions that are not visible to the CDONTS component. For more information, see [1][2].

Below is an example of CDONTS usage:

```
Set email=Server.CreateObject("CDONTS.NewMail") 'Create the object
email.To= to@tradecenter.se 'Set recipients email address
```

```
email.From=    from@tradecenter.se    `Set    senders    email  
address  
email.Subject = "Sending a mail from ASP" `Set message  
subject  
email.Body = "Message text here" `Text in message body  
email.Send `Send the message
```

This technique has the drawback that no active follow-up of the message is done. The system does not know what happened during the delivery. There is a possibility that the message ends up as undeliverable. Since most e-mail servers tries to deliver a message for five days a real-time system would have to wait a long time for confirmation. However, having automatic returned mail analysis done by a server daemon can solve the problem.

ASP Smart Upload

ASP Smart upload is a free to use component provided by the company Advantys. The component makes it easy to handle file transfers when using HTML components of type file. Functions that are included in this component are, in example transferring and controlling one or more files, restricting file size, database access and more. For complete component description, see homepage (www.aspsmart.com).

4. Notifications

Notifications are sent to dealers when a customer has found an interesting object by filling out the form with contact information. Once this form is completed the dealer of the object will be contacted, preferably by e-mail since this is the most cost effective way of communication. However, in some cases dealers do not have access to e-mail. Since many dealers of recreational vehicles that use fax machines and cellular phones rather than e-mail, the system must be extended with functionality that supports communications by fax and SMS. This makes it easier to establish a contact between customer and dealer.

4.1. E-mail

At the present e-mail is the only way for a user to online send an interest notification to a dealer.

Existing Solution

When a user completes the form, the system creates an email with data from the form and data for the current object from the database. The e-mail body is composed by adding string data into a variable and then put it in the CDONTS mail object. Dealers e-mail address is read from the dealer's table.

Improvement Suggestions

E-mails are created in real-time when the user sends his request. This is not necessary a more optimal system should store e-mails and do the sending of e-mails during off peak time, when other user interference is small. This solution has the drawback that dealers might have to wait up to 24 hours before receiving notifications. However, the amount of notifications has to be quite large, before any considerable impact is noticed, and therefore there is no need for changing the e-mail functionality.

4.2. Fax

Fax machines are present in almost every office around the world. They are getting obsolete because of e-mail and other computer communication. But as with all other technology evolutions, there are stages to take before completely migrating to a new communication method. Dealers of recreational vehicles are frequent users of fax machines.

Desired Functionality

The system should create and send a fax to the dealer that looks like the e-mail body. The fax component should be easy to integrate in the current TradeCenter system. It should also offer administrative settings such as price per fax.

Off the Shelf Components

There exists much software for all common platforms on the open market. To reduce costs of development, a component was used that had similar functionality as the CDONTS mail objects. For example the system only need to set a parameter with the fax body and another with the receiving phone number. The object model also makes upgrading and bug fixing easy.

Manufacturers of fax software are, among others Symantec, Mdaemon and SMS/FAX. There are also many small companies, which make all kinds of special fax implementations. However, software costs ranges from 100\$ and more. Since off the shelf software often contains a lot of unnecessary functionality, and only the most basic is needed, these are not an interesting option.

Solution in Java

Java offers no ready to go solutions, but the Java communications option contains functionality for communicating through comports. This means that all the fax modems AT commands must known, and good system portability would be difficult to achieve. Java requires too much raw coding for fax hardware, without buying any off the shelf component. Java was not considered an alternative for the notification system.

For more information on Java, see Sun Microsystems homepage, java.sun.com.

Java Advantages and Drawbacks

Java offers the free environment advantage, which is important nowadays, when application development must be cost effective. If the fax service had been developed in Java, it could easily have been transferred between different hosting platforms. This is a basic concept of Java. It gives the option of trading the system to other interested companies, who can run it on any platform of their choice.

A major drawback of a java implementation is the lack of key functionality for faxing in the standard J2SE. Also, the current web-hosting environment is not optimized for Java applications, which could cause interruptions in the existing system.

Solution in ASP

There are a lot of existing components for sending fax from ASP pages but to keep costs reduced the ASP platform can use the FAX object that comes with the Windows built in fax service. This makes the solution portable to any windows platform having a FAX modem connected and the fax service installed.

For more information on ASP, see the Microsoft homepage, www.microsoft.com

ASP Advantages and Drawbacks

The most obvious advantage with an ASP implementation is that the current system is built on structure, and the hosting platform is adapted for running ASP applications. Also, the new ASP .NET technology makes it possible to implement a lot more advanced functions in languages such as C#. Since the current system is running ASP there are no major integration problems.

ASP has the drawback that it is bound to the Windows platform with the fax service objects installed, leading to less inter platform compatibility.

Selected Technology

The investigation and arguments discussed above results in the choice of ASP as language for the fax implementation. Since ASP scripting language does not require type declaration, the fax service object cannot be used directly. To solve this problem

a proxy object has to be created and implemented using COM technology. (See appendix BAppendix B).

4.3. SMS

SMS, short message service, is a modern way of communication. An SMS is a short text message that is sent to a cellular phone. The message size is limited to 255 characters, in standard. However, it is possible to send multipart messages if more information needs to be sent. Lately the MMS technology has begun to replace SMS as communication for cellular phones. Since the notification service only uses text messages, there is no need to look at the MMS technology.

Desired Functionality

After a user has filled out a form the dealer of the object should receive a SMS with necessary information.

Service Providers

In order to send SMS, a service provider or SMS sending company must be contacted for a service agreement. Once this is done, an SMS is sent via the Internet to a dynamic page at the service provider. The dynamic page is called with parameters that hold the receiving phone number and message text.

Implementation Notes

In the first version of the notification system, the SMS functionality will not be implemented. This has been decided due to economical reasons.

5. Notification System Design

The notification system is divided into two parts one for sending notification and another for administrating the system.

5.1. Notification use cases

The notification service use cases are e-mail-, fax- or SMS message delivery. All use cases start when a customer has found an interesting object and decides to contact the dealer.

Use Case Fax Notification

When a notification is sent via fax, the system first gets the fax number where to send the fax from the database. Second a fax object is created and all parameters are set, such as receiver and message text. The server side script then submits the fax object to the fax service, which is responsible for delivering the fax message to the assigned number. This can take some time because of busy telephone lines or any other delay. The actual sending of a fax is also quite time consuming with modem handshaking and data transfer. See Figure 4: Use case sending a fax notification, for a detailed data path.

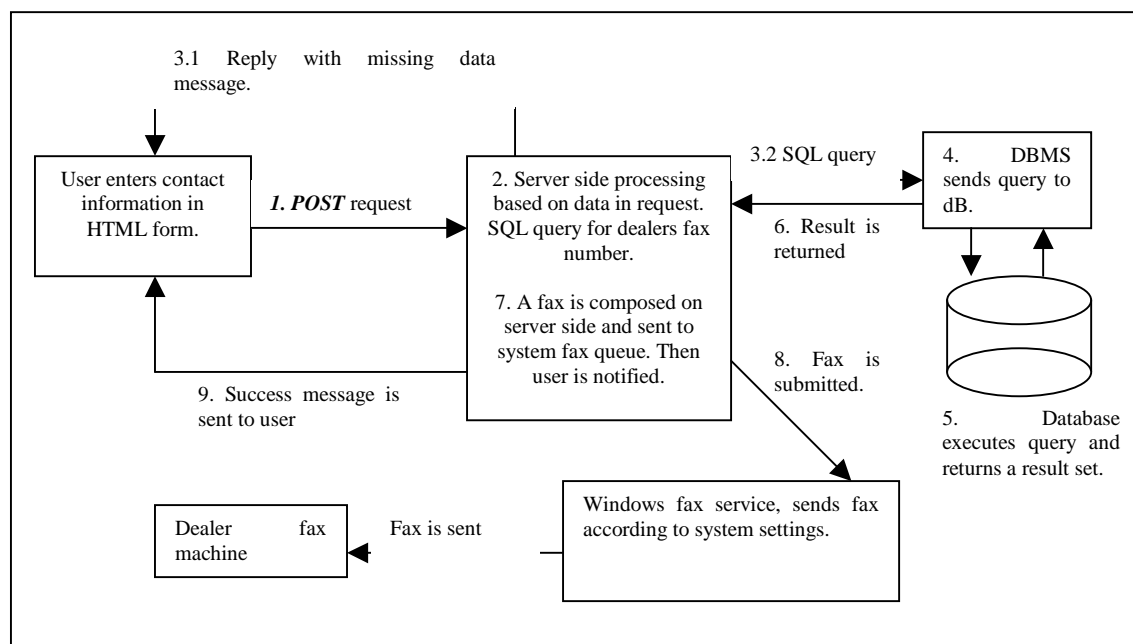


Figure 4: Use case sending a fax notification

Use Case SMS Notification

When a notification is sent via SMS, the system first checks for the cellular number the dealer wants the notification sent to. This information is gathered from the database in the same way as in the fax use case. Once the server side script has obtained all information, a get request is sent to the cellular service provider. The get request is a standard URL with the parameters necessary to send the SMS. The service provider is then responsible for delivering the message to the dealers phone.

Once again, it is not useful to do a real time follow up of the message delivery, since there are many possible delays when sending an SMS. See Figure 5: Use case sending a SMS notification for a detailed data path.

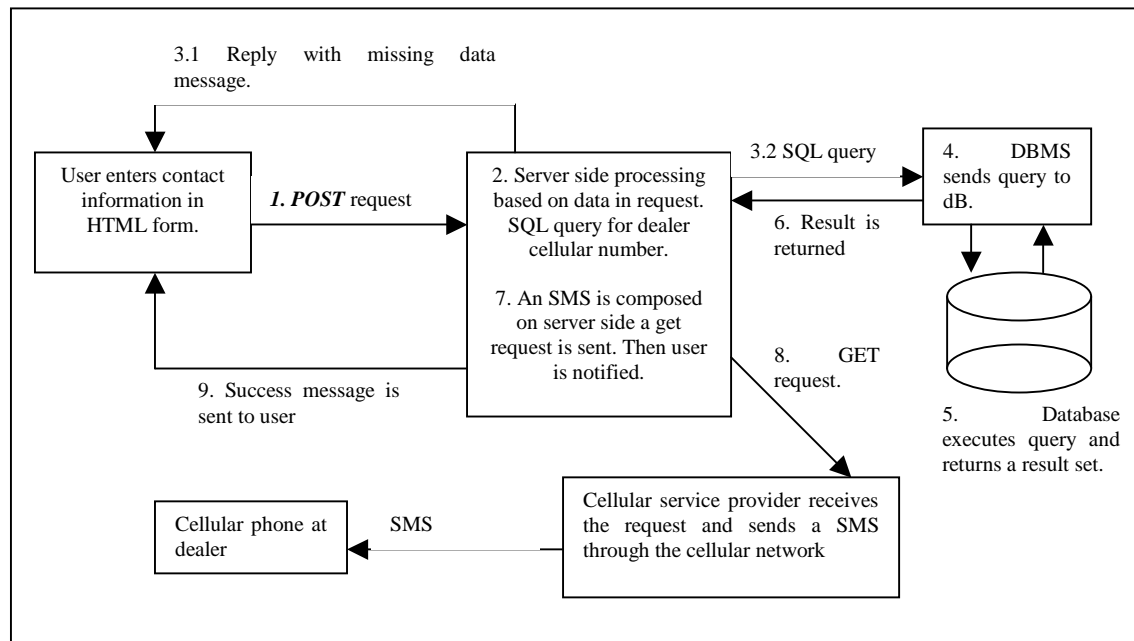


Figure 5: Use case sending a SMS notification

5.2. System Design

The notification system is developed as a stand-alone service from the TradeCenter e-commerce system. This means that upgrades and system maintains are invisible to the users of the notification system, in this case TradeCenter e-commerce system. Since the implementation of a SMS notification service was cancelled, the rest of the thesis is focused on a fax notification system.

For convenient usage of the fax notification system, access is done through GET requests. For example, sending a fax can be done by requesting the following URL:

<http://www.tradecenter.se/sendFax.asp?userId=1&password=secret&reciever=08123456&filename=http://www.tradecenter.se/testfax.txt>

This method makes the entire fax notification service invisible to the user, since everything that needs to be done is to send some parameters to an Internet page. The solution is much like the sending of SMS through Internet service that some cellular service providers offer.

In order for the system to function, the system must contain functionality for validating the parameters, check the user name and password and finally get the file for faxing.

For field validation, the script performs some standard checks, such as checking the input fields for single-quote characters.

In order to identify users, a user database has to be created since this is a stand alone system. Simply performing an SQL query that select a row with both username and password can identify users:

```
SELECT * FROM Users (NOLOCK) Where UserID=1 AND Password=secret
```

If the return set is empty, no user with matching userid and password is found. The reason for this could be that the user either did not exist, the userid is invalid or the password is invalid.

Once a user identity has been verified, the scripting page must download the file, specified by a standard URL. When the file has been placed in a temporary folder on the server computer, it can be submitted to the fax service component.

Finally the script creates a fax proxy [see section 5.5] object, and assigns the correct parameters and file name. The fax proxy object assures that the fax is submitted to the fax queue via the Windows built in fax service. If submission to outgoing fax queue is successful, the scripting page responds to the user by an OK message. If submission to outgoing fax queue fails, the user is notified of this by custom script output.

5.3. Security Considerations

The convenient method described above, contains little security since the password is sent in clear text. However, both the notification system and the client system are hosted on the same computer, which means that the GET request is never routed through any public network where listeners might exist.

If the system is scaled and external users might want to use it, security must be considered. This is important since the fax service is probably going to be an interesting target for people who want to send free faxes.

Security problem can be located in two places one is when sending the password in clear text over the Internet. Having an encrypted login page for external users can solve this problem. All communication thereafter should be encrypted to ensure minimal information leakage.

The other security breach is at the client computer. Almost every web browser stores requested URLs for some time. A hacker can use this and simply lookup the GET request, which contains both username and password. To solve this problem one can set no cache parameters remote for client browsers, but that does not ensure that there is no caching done at the client side. Also client firewalls might contain caching daemons. A better approach would be to have the user client doing encrypted POST requests, much as in the previous security problem.

5.4. Sending Notifications

The code was implemented in one ASP page as follows, first a database lookup is done to see how to contact a dealer and then code for the appropriate way of

communication is executed, in pseudo code it would look something like the pseudo code below:

```

Create message body
open  "SELECT  notification  FROM  dealers  WHERE
dealerid=deleaerid for this object"
if notification = "email" then
    Execute existing email code
end if
if notification = "fax" then
    Execute fax code
    If send ok, insert into log file
    Else print error message
end if
if notificatation = "sms" then 'not implemented in this
version.
    Execute sms code
end if
if notification <> "sms" or "fax" or "email" then
    Print error message
End if

```

Once the page above has completed, the user has either received an ok- or an error-message. If notification was sent properly, the system now has the fax in its outbound queue. The systems fax service then sends the fax to the appropriate fax number accordingly to local system fax policies. See Figure 6: Overview of fax notification execution path for a graphic layout of execution.

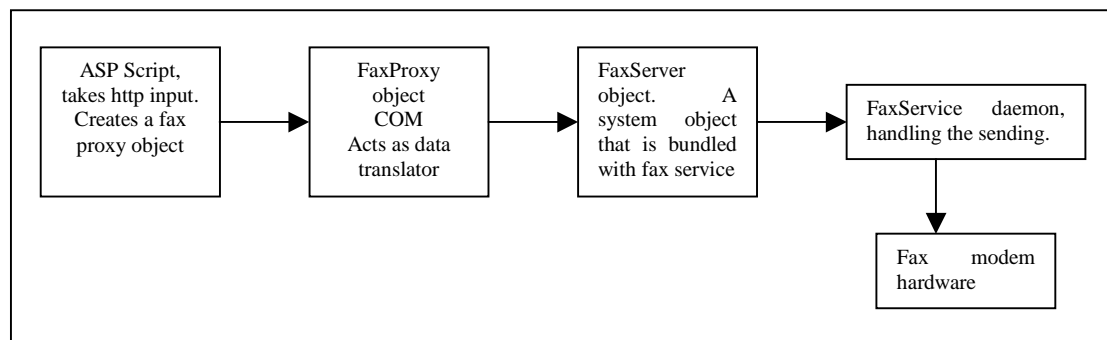


Figure 6: Overview of fax notification execution path

5.5. Fax Object

Microsoft Windows 2000 Fax service object [6] is not compatible with weakly typed scripting language such as ASP. In order to use this service, a COM proxy object has to be designed. This did not require type declared input, and could therefore be used from ASP pages. On the other hand, input data had to be checked, and before calling the built in fax service object, type declaration had to be done within the proxy object.

The input parameters that the object has to handle in this case are a recipients fax number, and a text file to send. All other parameters, such as sender company and fax number can be coded into the component. This reduces the possibility of errors but also the ability to use the object flexibly. In order to achieve a stabile object with some flexibility it will have some extra basic parameters implemented, such as the ones mentioned above.

An outline of the code skeleton for the fax proxy object is shown below. The code has been stripped from all input validation and error checking algorithms, but it shows how the object is intended to function:

```
Public Function Fax(FileName As Variant, FaxNumber As Variant)
    Set FS = CreateObject("FaxServer.FaxServer")
    FS.Connect ("\\localhost")
    Set FDoc = FS.CreateDocument(FileName)
    FDoc.FaxNumber = FaxNumber
    FDoc.Send
    Set FDoc = Nothing
    Set FS = Nothing
End Function
```

As shown above, the proxy object creates a FaxServer.FaxServer object, which is Windows built in communication object for the fax service, which manages all faxes in the system.

The proxy connects to the fax service machine, in this case the localhost. When connecting to the fax machine, an error could be raised, if for example the system administrator temporarily has disabled the fax service.

Next the proxy creates a FaxDocument [7], which is the object to work with when sending faxes through the FaxServer object. Calling the corresponding method of the FaxServer object creates the FaxDocument object. The creation method also takes a file name as argument. This file contains the content of the fax. When creating the FaxDocument an error could be raised, if the filename is incorrect or the file is corrupt.

The proxy sets the receiver's fax number by calling the method FaxNumber of FaxDocument. It then submits the fax to the server by calling the Send method. These methods could both raise errors, since the connection to the server might have been broken.

Finally setting the used variables to nothing performs a clean up.

The real code contains even more error checking and field validation, that returns user-friendly messages in order to make debugging easier. However, most error checking has already been done in the ASP script to make error displaying easy.

5.6. Administrating the System, Overview

The administrative part of the system was integrated in an environment, similar to the existing TradeCenter administration system. From the main page in the administration system there are buttons for viewing statistics, and setting prices for the fax service. There is also a function for adding, removing or editing user accounts.

In order to increase security when administrating the system, a secure server requiring encrypted communication must be used. This had not been decided at the time of this implementation, but is important to consider in the future.

5.7. Preferences

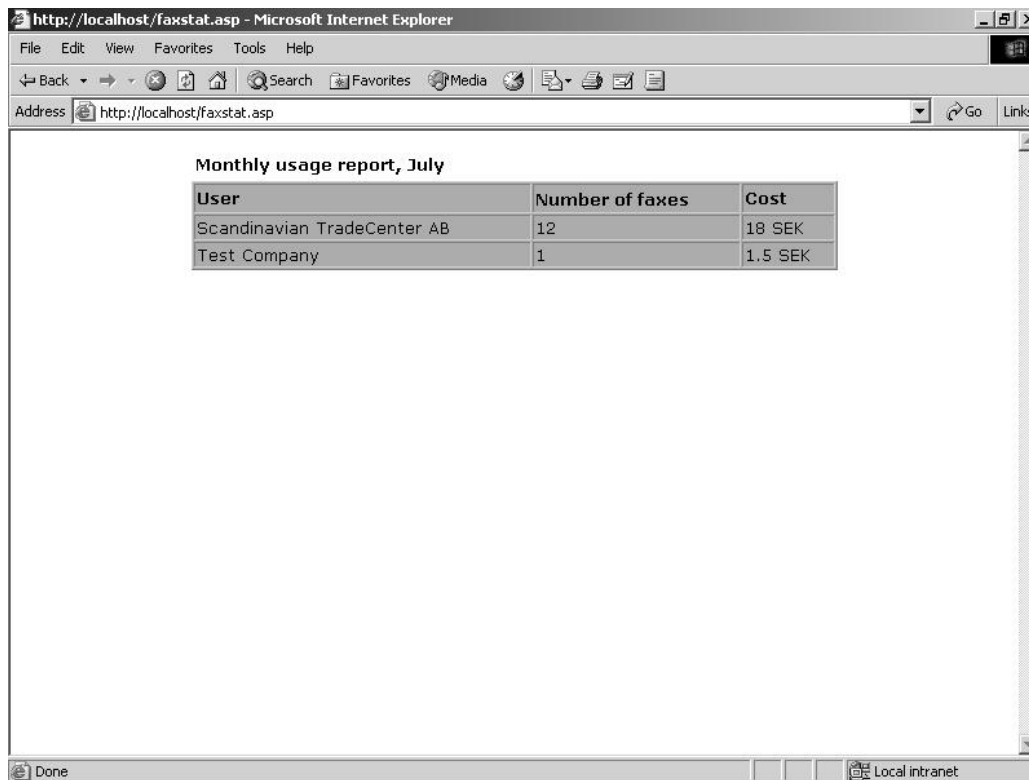
The system has one parameter, which can be affected by the system administrator. The parameter was named price per fax, an integer value that holds the cost for sending a fax. Since there is no requirement of having prices based on fax length, it is sufficient with one fixed price. The system is not required to handle differentiated fax lengths, because all faxes is based on the same layout and the fax length is always one page.

The price per fax is a global variable, which means that all users pay the same price for sending a fax. If the system is extended to support intercontinental users or differentiated prices, a more complex pricing strategy is needed. The most important aspects are the sizes of faxes, destination country and peak or off peak sending time.

In order to make the system more cost effective, parameters for setting the preferred sending times can be implemented. This allows the system to automatically set a correct sending time for the fax, in example off peak. Sending time options are also available in the built in Windows fax service and can be changed by the system administrator. This can, on the other hand, affect the entire hosting platform and is difficult to use. The option was not implemented in this version of the system, but is important to considered in future versions.

To limit costs, a parameter can be created, that limits the number of faxes a user can send. For example, setting a limit of ten faxes per day. This was not implemented in this version, but can be considered for future development of the system.

5.8. Statistics



Monthly usage report, July

User	Number of faxes	Cost
Scandinavian TradeCenter AB	12	18 SEK
Test Company	1	1.5 SEK

Picture 6: Monthly summary of system usage

When sending a fax, the system adds an entry to a log file. Every entry in the log file holds information on date, time, recipient and the cost, valid at the time. The cost is stored for every event, since it can change at any time and make all log entries useless. An alternative is to save the historical price changes, and calculate the total price as a function, that checks every log entry with the corresponding price for that date.

Important statistics for the system administrator to view are total costs for each customer every month, total number of faxes sent, distribution of sending times during the days, weeks and months. See Picture 6: Monthly summary of system usage for a real view of the statistics GUI.

Below are queries for gathering different kinds of statistics. The queries are written in ordinary SQL language. Parameters that must be passed on to the ASP scripting page, creating the queries, are `startdate` and `enddate`. Naturally the ASP page does error checking on all input parameters and checks if the date format corresponds to the one used in the database. If dates are misunderstood between scripting page and the database, the result could contain information about i.e. the 11 of December instead of the 12 of November.

Total costs for different customers in a date interval

```
SELECT DISTINCT(CustomerID), COUNT(*) as NumFaxes FROM  
FaxLog (NOLOCK) WHERE Date between 'startdate' AND  
'enddate' GROUP BY CustomerID
```

Distribution of sending times during one day in a certain date interval

```
SELECT DISTINCT(DatePart(HOUR, SendTime)), COUNT(*) as  
NumFaxe FROM FaxLog (NOLOCK) Date between 'startdate' AND  
'enddate' Group By SendTime
```

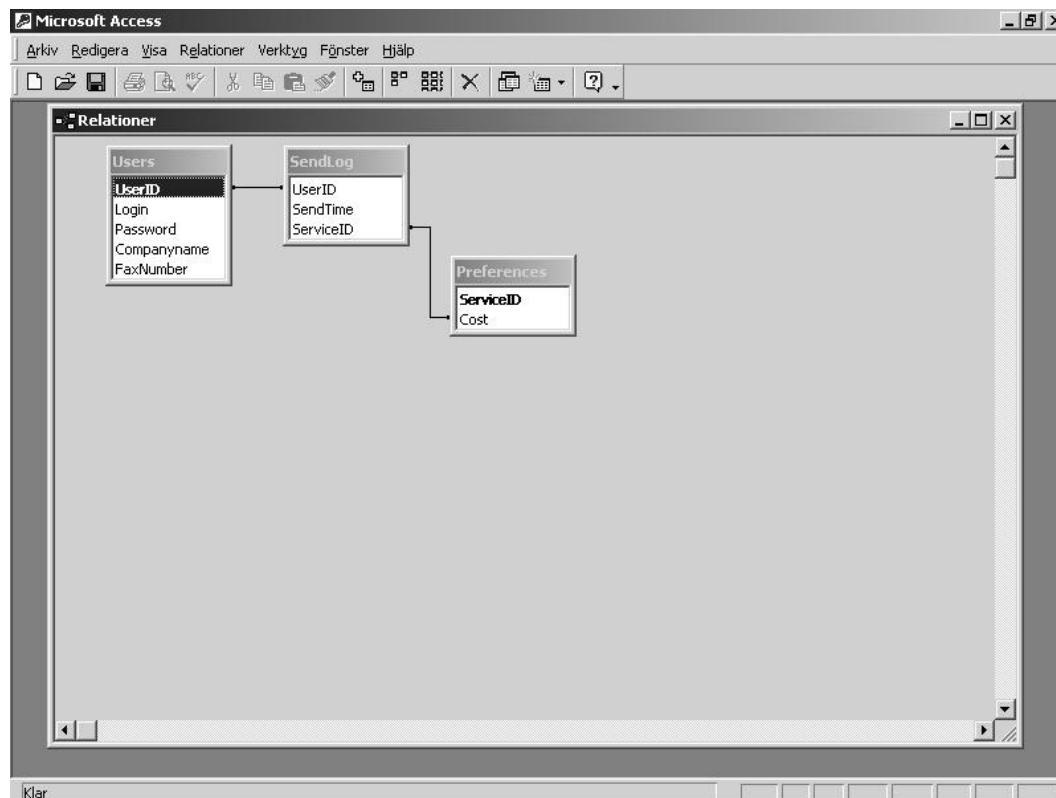
Distribution of sending times during one month in certain date interval

```
SELECT DISTINCT(DatePart(Day, SendTime)), COUNT(*) as  
NumFaxe FROM FaxLog (NOLOCK) Date between 'startdate' AND  
'enddate' Group By SendTime
```

Once the number of sent faxes is known, a simple ASP page collects this information, and the parameters in the preferences table. From this information, a cost report is created by simply multiply the number of faxes with the cost per fax.

5.9. Database Table Structure

The database (see Picture 7: Notification database design) contains tables for user data, log data and system preferences. If the system would support multiple ways of notification, a table for service definitions must be added in order to allow the system to separate SMS and fax entries.



Picture 7: Notification database design

Using an integer user identity, enables fast indexing and reduces the amount of space needed to reference a log entry with a user. Below in Table 1 is a line from the users table.

UserID	CompanyName	Address	ZipCode	City
1	Test company	Test road	123456	Testcity

Table 1: Users table

The Users table holds contact and billing information about customers, every customer is also associated with a UserID of integer type.

The preferences table holds global system information. This version of the system does not contain any global information but prices. Clearly this means that only two columns was needed, one to identifying the service (fax or SMS) and another for the price.

Since the system only function with fax notification the preferences table was reduces to a one-line table. Having tables with only one line in a database is not very useful and the cost parameter can therefore be stored in an external file or be hard coded into the scripting pages, not done in this implementation. For future scalability, this table was implement. See Table 2 for an example row from the preferences table.

ServiceID	Cost
Fax	1.5
Email	0

Table 2: Preferences

The log table holds entries for date and time, using the datetime data type in SQL, the recipient number and the UserID of the sender.

Log tables tend to grow quite large and a strategy for archiving sending events can be worked out. Since the log file will be used to create invoices, it is important that it is stored long enough to verify old invoices. See Table 3 for an example of the log file.

SendTime	Recipient	CustomerId	Cost
2003-01-01 22:01	+46-8-12345	1	3

Table 3: Log file

Further optimizing the log file size can by done, either by minimizing the length of the recipients telephone number or coding the date and time in a numerical or smaller date format. The MS SQL build in date time data type uses 8 bytes of storage space.

When minimizing the recipient's telephone number, one should consider the system usage. Since all calls will be domestic for the TradeCenter e-commerce system, there is no need for storing the international prefix and country code.

Another strategy for reducing is to store a reference to a fax job instead of the recipient's number. This could be done by a simple integer job identity, of 4 bytes. However, this also means that a copy of the fax, or some other job information, has to be stored and activated on the server.

6. Hardware Installation

The server was extended with a faxmodem in order to send faxes. When installing a faxmodem, precautions had to be taken, since the server was operating in live mode.

Adding a modem to a computer was not a big issue, but since the system had no physical storage limitations, an external modem was used. This eliminated the need to power down and physically open the server computer, and downtime was zero minutes.

Other hardware installations, such as telephone lines and power cable for the modem does not affect the server and is not to be considered as a problem.

Software installation could cause a reboot or an error installation that could hang the entire system. These problems are minimal on a robust platform as Windows 2000, but were not left without attention. Saving backup of running server daemons prevented data loss if the system had to be forced down.

6.1. Preparations

The server hosting the faxmodem is running Windows 2000 as operating system, and the modem installed was an external US Robotics Faxmodem.

To avoid unexpected shut downs of the system during the installation, the modem was test installed on a separate computer running Windows 2000. Windows could automatically detect and locate new drivers for this modem, and fax, communications proved reliable during testing. Since there were no need to take the server offline when adding a modem and installing software, the total downtime was zero for this moment.

6.2. Installing Fax Proxy Object

Adding the fax proxy component to the server COM environment did not affect any other applications running. Windows 2000 contains a manager for handling this operation. However, if a component is removed and reinstalled, a utility that reinitializes the components must be used, otherwise the system might not detect new methods that have been added.

6.3. Installation Verification

To verify that the server now can handle faxes, the built in fax service in Windows was tested manually, by sending a fax from a text file in notepad. Since no errors occurred, the software and hardware upgrade of the server was successful.

Creating a test ASP page, that submits a text file for sending, verified the implementation of the fax proxy object on the server.

7. System Integration

The integration of the new system was transparent to system users since the form that is filled out is independent of how dealers are contacted. Once the form is submitted, an ASP page connects to the database and checks which way of communication the dealer of the current object prefers. Based on the preferences, the ASP page executes different code segments for sending e-mail, fax or SMS. For the fax service, a request is done for the fax notification service script page, implemented on the server with the fax modem.

The integration was very facile, since a stand-alone solution was applied for the notification system. Future upgrades will also be easily achieved, since the system is separated from the TradeCenter e-commerce system.

Another advantage of a separate system is that it can easily be converted into a multi user platform and provided for other e-commerce companies.

8. Performance Evaluation

The system performance depends on how many fax modems the server has connected. A standard computer has two comports, but can be extended with equipment, such as Digiport®, to support 32 or more comports. In Windows 2000, the standard fax service does not support remote connection, which means that the application must run on the computer with the fax modems.

The limiting factor for sending faxes is on the connection side, rather than on the submission side. The time it takes for the script to verify field inputs and submit the fax to the servers queue is insignificant, when compared to the time it takes to send the fax. The performance evaluation is therefore focused on the time it takes to send a fax. Sending a fax adds up to a few time consuming events:

1. Initializing the modem
2. Getting dial tone
3. Dial the number
4. Await answer
5. Handshaking
6. Transmission
7. Termination
8. Resetting the modem

The eight steps above are all needed in order to send a fax and make the modem ready for the fax next in queue. However, one could also analyze the time it takes from submission of a fax until the recipient has received it. If this is done, step seven and eight can be neglected.

8.1. Expected Performance

Testing showed that it takes about 70 seconds to send a fax, with the notification message. This measure starts when the ASP script is called and ends at the completion of step eight above.

With a one modem server this means, it is not advisable to exceed 0.86 faxes per minute, which is the service rate of the system. This is sufficient for the TradeCenter e-commerce system, and also enough to support external customers.

8.2. Delays

A system with random submission of faxes always has the probability of a message delayed due to random events.

It is possible that some faxes cannot be sent due to busy recipients. This causes a fax to be delayed by the time it takes to do modem initialization, dialing, getting busy tone, disconnecting and resetting the modem.

In the TradeCenter e-commerce system it was decided to have a 20 % change of message delay due to random events. Delays caused by busy recipients were neglected.

If one looks at the probability that a delay will occur (see Figure 7: Probability of delay), it is clear that the one modem curve exceeds 20 % chance of delay at about 0,25 faxes per minute (4 minutes between faxes). This is enough to support the expected arrival rate.

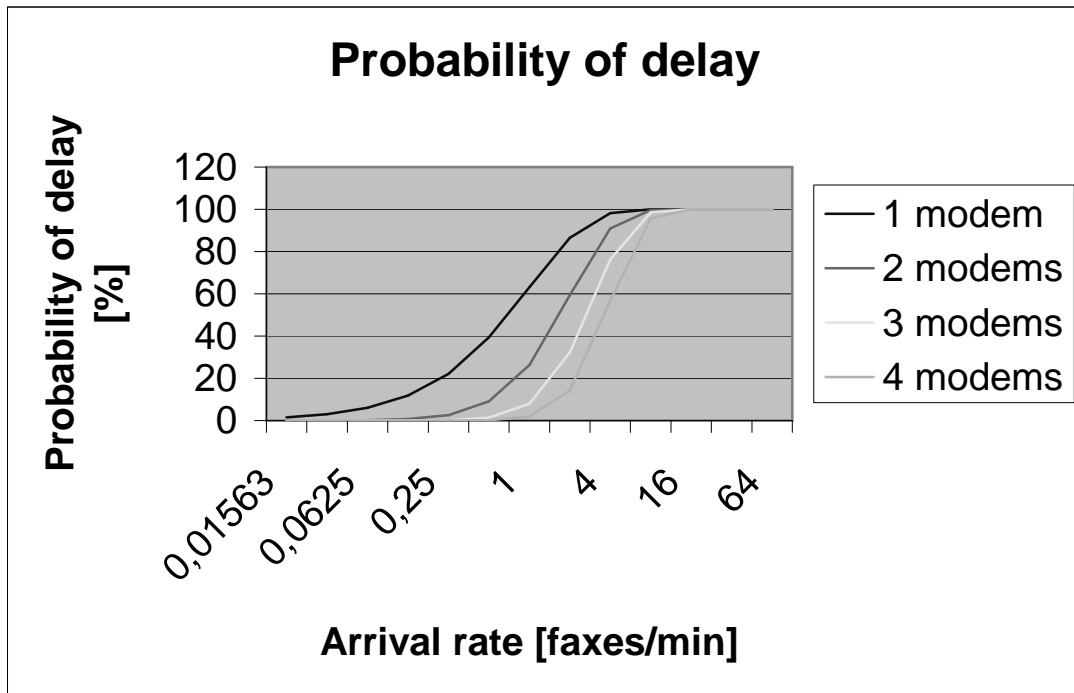


Figure 7: Probability of delay

The raw data used to create Figure 7: Probability of delay is listed in Table 4, in appendix A.

To calculate the probability of a delay, the service time, and the arrival rate must be known.

The sending time, 70 seconds, gives the sending rate as 60/70 (0.86) faxes/min.

In order to produce a graph, the arrival rate is changed from 0.015 faxes/min to 64 faxes/min.

The probability is plotted as a Poisson function in Microsoft Excel, with the service time, and arrival rate as parameters. Microsoft Excel contains functionality for Poisson process calculations.

9. Conclusions

The notification system has been developed as a stand-alone service. This means that it can be used in other applications. It also makes system scaling easier, since the users do not know how the messages are delivered.

A stand-alone system makes software upgrading easy, since there is no need for bringing down the entire e-commerce system. This is important when building systems that are to be used for many years.

System performance is highly limited by the time elapsed to send a fax. The only way to get around this limitation is to add more faxmodems. However, the intension when integrating a fax solution into an e-commerce system is to support a decreasing but important client group, the non e-mail users. It is probably not necessary to connect more than one modem to the server.

References

- [1] Introduction to CDO for NTS, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cdo/html/denali_cdo_for_nts_library.asp, 2003-03-20
- [2] HOWTO: User CDONTS to Collect and Mail Information From a User, <http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q186/2/04.ASP&NoWebContent=1>, 2003-04-15
- [3] SQL Server Homepage, <http://www.microsoft.com/sqlserver>, 2003-05-30
- [4] Vlad Vlassov, lecture notes on Three-tier architecture, 2003
- [5] <http://www.microsoft.com/iis>, 2003-05-30
- [6] FaxServer Object (Visual Basic), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fax/faxlegacyvb_836n.asp, 2003-08-01
- [7] FaxServer Object (Visual Basic), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fax/faxlegacyvb_9tb3.asp, 2003-08-01
- [8] PHP: Hypertext Preprocessor, <http://www.php.net>, 2003-09-01
- [9] The Source for Java Technology, <http://www.java.sun.com>, 2003-09-01

Appendix A

	Service time faxes / min	0,86	1,71	2,57	3,43
Arrival rate new faxes / min	Time between fax min	Probability of delay %			
0,02	64,00	1,55	0,01	0,00	0,00
0,03	32,00	3,08	0,05	0,00	0,00
0,06	16,00	6,06	0,19	0,00	0,00
0,13	8,00	11,75	0,72	0,03	0,00
0,25	4,00	22,12	2,65	0,22	0,01
0,50	2,00	39,35	9,02	1,44	0,18
1,00	1,00	63,21	26,42	8,03	1,90
2,00	0,50	86,47	59,40	32,33	14,29
4,00	0,25	98,17	90,84	76,19	56,65
8,00	0,13	99,97	99,70	98,62	95,76
16,00	0,06	100,00	100,00	100,00	99,99
32,00	0,03	100,00	100,00	100,00	100,00
64,00	0,02	100,00	100,00	100,00	100,00

Table 4: Probability of delay

Appendix B

FaxServer

The FaxServer object [6] is used to manage a connection with the Windows fax service. The FaxServer object gives a lot of information about the settings, such as number of retries for the fax service. The only method used for this notification application is the connect method for connecting to a fax server, i.e. localhost or some machine with fax modems, and the CreateDocument method.

The CreateDocument method of the FaxServer object creates an instance of the class FaxDoc, described in 9.2, it takes a filename with fax content as argument.

FaxDoc

The FaxDoc class [7] holds information such as sender, receiver and content. In this notification application, only the properties recipient's fax number and company name are set. The method Send of the FaxDoc class is used to submit the fax to the fax service via a FaxServer connection, given by the object that created the FaxDoc instance.

The file name, used by FaxServer to create a FaxDoc instance, can be any Windows registered file type. The notification application only uses .TXT files, which are by default handled by notepad.

Glossary

ASP – Active Server Pages

DBMS – Database Management System

DNS – Domain Name Service

FTP – File Transfer Protocol

GUI – Graphical User Interface

HTML – Hypertext Mark up Language

IIS – Internet Information Service

PHP – Hypertext Preprocessor

SMS – Short Message Service

SMTP – Simple Mail Transfer Protocol

SQL – Standard database query language

JSP – Java Server Pages

SPAM – Unwanted e-mail