Abstract

This documentation contains information about the Wireless Application Converter (WAC) for WXES 3181. It is divided into four chapters. These chapters are Chapter 1: Introduction, Chapter 2: Literature Review & Analysis, Chapter 3: Methodology, Chapter 4: System Design, Chapter 5: System Implementation, Chapter 6: System Testing and Chapter 7: System Evaluation. The objective of this documentation is to provide the reader with an overview of the purpose of developing WAC, the architecture and functionality of it.

Chapter 1: Introduction provides the introduction for readers about the objectives of developing WAC. It covers the objectives, development scopes, targeted users, development limitations and a proposed schedule to finish this project.

Chapter 2: Literature Review & Analysis provides information on several terms that are related to this project. These terms are taken from the current wireless technology field. Besides, there are two systems described in detail that use the similar approach which are the reference for WAC. This chapter also provides information about the technology require to develop WAC, which include development software, wireless hardware, protocol for wireless network, file storage and server for WAC.

Chapter 3: Methodology describes the methodology used to develop WAC. The descriptions include an introduction of method and the reason for using it. Included in this chapter as well, are the requirements of the system. These are the system functionalities, hardware and software requirements.

Chapter 4: System Design explains how the system is designed. It covers the system architecture, system modules, system functionality design and module functionality design. Explanation on system architecture covers a brief introduction on the chosen architecture, how this architecture can be implemented in WAC, the advantages for choosing it and the disadvantages. Explanations on other parts of this chapter cover the descriptions on the functions involved, the data flowing and how the functions integrating with each others.

Chapter 5: System Implementation explains how the system is developed. It covers the coding approach, scripting language and development tool used to develop each module.

Chapter 6: System Testing explains the testing done on the system. This chapter covers the type of testing being implemented, testing approach and testing resources. It also covers the changes that had been done on the system.

Chapter 7: System Evaluation is the final chapter. It covers the problems encountered during the development and the relevant solutions, evaluation by the end user, pros and cons of the system, future enhancements and also knowledge and experience gained from the development. This chapter will conclude the details about Wireless Application Converter.

Compliment

First, I would like to thank for my supervisor, Miss Rafidah Mohd. Noor for willing to supervise me in this final year thesis. I would like to express my gratitude for her guidance in preparing this documentation as well on the system design.

Secondly, I would like to thank Mr. Andrew Khoo, Director of Handisplay (M) Sdn. Bhd for his valuable advice in pursuing research in wireless field. I would like to thank Mr. Rachmat Hartono, Senior Software Engineer of BCL Technologies. Under his guidance during the industrial training periods, I am able to learn the technique that is helpful in developing WAC.

Finally, I would like to express my gratitude to all my friends. For their willingness in sharing information and teaching, I am able to gather some information related to my development of WAC.

Table of Contents

Table Reference

Figure Reference

Chapter 1: Introduction

Now a day, the wireless technology is advancing in tremendous pace. Wireless technology is no longer only for wireless communication: surfing internet, voice messaging, picture messaging and traffic navigation are evolving from theory to practical.

Technologies such as General Packet Radio Services (GPRS), Wireless Application Protocol (WAP) and I-MODE (only available in Japan) have been invented to make full use of the advantages of wireless communication. Community defined the current wireless technology as 2.5 G and it is moving towards the 3 G era.

Under the influence of wireless wave, several researchers and developers are currently running research on implementing wireless technology that emphasized the use of server. This technique is to overcome certain constraint of the wireless devices such as lower processing capability that make it unable to view a variety of multimedia content.

There are some products already available in the market that implementing this technique, such as AvantGo. The server is responsible for pre-processing before sending back the result to the client.

Wireless Application Converter (WAC) is a system that enables wireless devices such as PDA and web-enabled phone to view the same internet content within a wireless network.

It is also implementing the server-side technique, where all the pre-processing is done at the server before displaying the result at the client. For a wireless devices to view internet content, the content must be built in the format that supported by the devices, such as WAP. Through WAC, the devices are able to view the content

although it is not written in the supported format because the pre-processing procedure of WAC will eventually change the content format to the one that supported by the devices.

Furthermore, WAC provides real-time internet browsing ability, which means the requesting, pre-processing and result display are done once the users make the request.

1.1. Objectives

The objective of WAC is to enable information sharing in a form of internet content between wireless device and desktop computer within a wireless network. For example in a corporate company, the staff can have the latest information about today meeting schedule or event with their handheld devices as long as there is wireless connectivity between the device and the server.

Besides, it is a mean in providing a cross platform support for different wireless devices. Wireless devices are different in some aspects, such as the operating system, processing power and supported format. WAC is able to overcome this problem by building the content into the format supported by the wireless devices without the need to modify the original content.

Finally, it is to overcome the constraints of wireless devices, such as processing power, low bandwidth and small display screen. The pre-processing of WAC is responsible to optimize the content to be viewed at the wireless devices.

1.2. Scopes

For the initial start, the development of WAC will focus on converting HTML to Wireless Application Protocol (WAP).

The reason for choosing WAP is because most of the wireless devices have this feature. These devices are ranging from handheld computer such as Personal Digital Assistant (PDA) to mobile phone.

In Malaysia, most of the telecommunication companies offer the WAP service, such as Telekom and Maxis Mobile. With this service, the users are able to connect to the internet with their mobile phone every time they wish to.

Besides WAP, the other consideration will be the operating system. Operating system does affect the performance of the handheld devices.

Currently, Microsoft Pocket PC and Palm OS are the two major operating systems that have been used on handheld computer. In comparison, Pocket PC is more powerful than Palm OS in processing power, multimedia features and functionalities.

The ideal operating system for development purpose will be Palm OS. It is because most of the features that supported by it is supported by Pocket PC. In contrast the features supported by Pocket PC cannot be supported by Palm OS.

For the operating environment, WAC will operate under Microsoft Windows platform. The reason for choosing this platform is because of easy development. There are a lot of Windows based resources and building tools available which could help in ease the development progress.

1.3. Targeted Users

The targeted users will be the internet content administrative from the corporate world and home users.

For the corporate users, they are able to use this system for information sharing as well as data query through internet.

The reason for choosing them as the targeted users is because mobility is one of the features that they needed, as this can enable them to connect and retrieve data from any where within the wireless network without being constraint by cabling.

For home users, they could use this system to connect to the internet and surf their favourites web site within a wireless network.

This system is very simple because it only needed a normal desktop computer as server and wireless LAN to work. It is very convenient to own these things now a day and it is affordable for home user.

Besides, the system provides the home user an alternative to connect to internet without the need to sit down every time in front of the desktop computer.

1.4. Project Limitations

The development progress might encounter some limitations. Currently, there are few limitations for the initial start. These limitations are time, lack of references and internet standard.

Developing WAC is very time consuming because there are a lot of considerations during the development process. These considerations are:

1) Types of devices need to be supported, such as Personal Digital Assistant or mobile phone.
2) Types of content that can be converted which is related to HTML tag.
3) Types of protocol to be supported, such as WAP.
4) Processing structure.

Since there is not much time is allocated, WAC will focus on WAP and Palm OS.

Furthermore, there are not many references out there about this kind of system. Currently this kind of system is still under research. Although there are

references out there, some of it are only theory and some of it even using different approach, like artificial intelligent, which is difficult to implement

Besides, the internet standard is also an issue to be considered. There is no standard define how the internet content should be. It might be pure HTML script, mixture of HTML script and JavaScript, or totally JavaScript. It is impossible to come out with a solution that could fix into all of the mentioned above. For the development purpose, it will focus on certain web site that is either HTML or mixture with JavaScript.

1.5. Schedule (Gantt chart)

The schedule in developing WAC is shows by the Gantt Chart in Table 1.1 below.

Table 1.1: Gantt Chart for WAC development duration

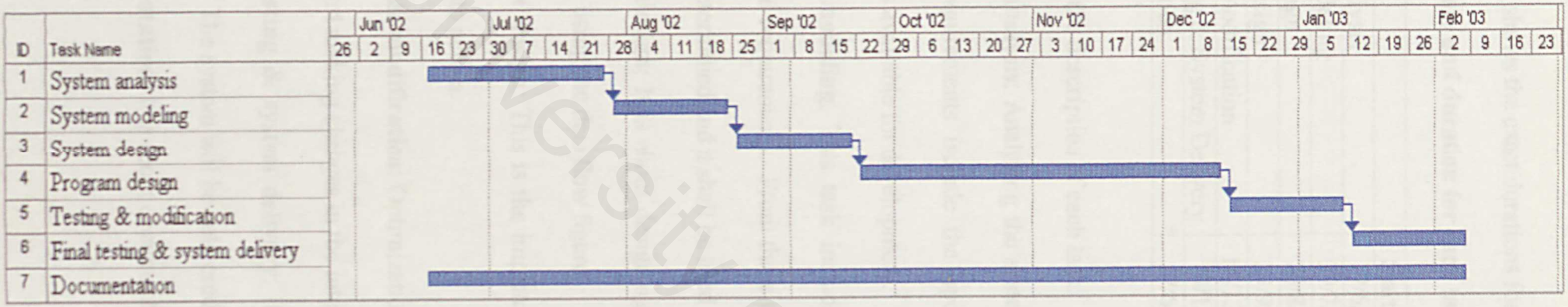


Table 1.2 shows the exact durations for each task mentioned in Table 1.1.

Table 1.2: Date and duration for each task

ID	Task Name	Start Date	End Date	Duration (days)
1	System Analysis	17/6/2002	26/7/2002	30
2	System Modeling	29/7/2002	23/8/2002	20
3	System Design	26/8/2002	20/9/2002	20
4	Program Design	23/9/2002	13/12/2002	60
5	Testing & Modification	16/12/2002	10/1/2003	20
6	Final Testing & System Delivery	13/1/2003	7/2/2003	20
7	Documentation	17/6/2002	7/2/2003	170

Below is the description of each task:

1. **System analysis**: Analyzing the system and capture the basic requirements. These requirements include the operating environment, architecture and resources available for development.
2. **System modeling**: This task includes capturing the functional and non-functional requirements. From these requirements, the specification of WAC can be determined and it shall be used for initial start of the development.
3. **System design**: It is about designing the internal process and data flow of WAC by using the data flow figure.
4. **Program design**: This is the implementation state where the programming comes into action.
5. **Testing & modification**: Optimization of WAC is done at this state through testing and making changes in the internal process.
6. **Final testing & system delivery**: It is the final state of the development process. The system will be delivered after it is fully tested.
7. **Documentation**: It is about documenting the process of each task.

Chapter 2: Literature Review & Analysis

2.1. Wireless

Wireless is a way of using the radio-frequency spectrum or electromagnetic waves for transmitting and receiving voice, data and video signals for communications. The signals are transmitted without physical conductor, but through air or water.

The radio-frequency spectrum is divided into eight ranges, called bands, each regulated by government authorities. These bands are rated as follow:

1. **Very Low Frequency (VLF: 3 KHz – 30 KHz)**: For long-range radio navigation and submarine communication.
2. **Low Frequency (LF: 30 KHz – 300 KHz)**: For long-range radio navigation and radio beacons or navigational locator.
3. **Middle Frequency (MF: 300 KHz – 3 MHz)**: For AM radio, maritime radio, radio direction finding (RDF) and emergency frequencies.
4. **High Frequency (HF: 3 MHz – 30 MHz)**: For international broadcasting, military communication, long-distance aircraft and ship communication, telephone, telegraph and facsimile.
5. **Very High Frequency (VHF: 30 MHz – 300 MHz)**: For VHF television, FM radio, aircraft AM radio and aircraft navigational aid.
6. **Ultrahigh Frequency (UHF: 300 MHz – 3 GHz)**: For UHF television, mobile telephone, cellular radio, paging and microwave links.
7. **Superhigh Frequency (SHF: 3 GHz – 30 GHz)**: For terrestrial and satellite microwave and radar communication.

8. **Extremely High Frequency (EHF: 30 GHz – 300 GHz)**: For radar, satellite, and experimental communication.

The radio wave transmission is propagate in five different ways as describe below:

1. **Surface Propagation**: The radio waves travel through the lowest portion of the atmosphere, surrounding the earth. The signals are at the lowest frequencies. The signals transmitted will move in all direction, following the curvature of the earth. The greater the amount of power in the signal, the further distance it can travel.
2. **Tropospheric Propagation**: This propagation way can be done in two way, that is directing the signals in a straight line from antenna to antenna (line-of-sight) or broadcast at an angle into the upper layers of troposphere and reflected back to earth's surface. The first way require the transmitter and the receiver to be within line-of-sight distances, while the second way allows greater distance to be covered.
3. **Ionospheric Propagation**: Higher-frequency radio waves are transmitted upward into the ionosphere where they reflected back to earth. Eventually, the radio wave will change direction and speed up when travel from troposphere into ionosphere. It has a greater range of transmission covered with lower power output.
4. **Line-of-Sight Propagation**: Very high frequency signals are transmitted in straight lines directly from antenna to antenna. The antennas must be facing each other and either tall enough or close enough for not to be affected by the curvature of the earth.

5. **Space Propagation**: It utilizes satellite relays in place of atmospheric refraction. Signal is received by an orbiting satellite, which rebroadcasts the signal to the intended receiver back on the earth. Basically, it is a line-of-sight transmission with an intermediary, the satellite. The distance for transmission is the greatest compare to others propagation ways.

2.2. Wireless Application

Wireless application is specially builds for the wireless devices such as the mobile phone.

An article written by Christoffer Andersson says that wireless application is divided into two categories: client-server and client resident applications.

Client-server: WAP is the first example of client-server applications. Most of the applications that we see are WAP applications or using similar technologies (web clipping, I-MODE). Most of these applications are browser-like menus with the content on a remote server. WAP and ultra-light clients are very appealing, as they don't require any skills from the user. There is no need to download, install or configure each application. The updates are done at the central server, and therefore instant to the user.

Client resident applications: It is an application that is always ready for connection. User will be able to connect to the application anytime they want and only paying for the content they have retrieved. Stock trading application is an example. It is constantly online, giving users a level of interactivity that is beyond what we get with browsing.

2.3. Converter

Converter is a device, process or application that transforming a format from one to another. Digital-to-analog converter is an example of converter. It is use to transform the signal from digital format to analog format.

WAC is considered to be a converter. It is responsible in converting the HTML format web page to WAP format page. It is done in real-time basis; once the user request for a HTML page, WAC will convert it to WAP page within a few second.

2.3. Wireless Network Environment

Wireless Network Environment is an environment of network components, which are link together with the use of wireless technology. Wireless Local Area Network and Bluetooth are examples of the environment.

Wireless Local Area Network (WLAN) is the new implication of network technology that allows the devices such as desktop computer, laptop, handheld computer and PDAs to be connected together wirelessly. It is like an intranet but without cabling.

WLAN is gaining its acceptance in the market now a day. The reason for this is because WLAN offer lower price but for transmission speed ranging from 1 to 54 Mbps depending on the protocol it used. Furthermore, WLAN can be working in a reasonable range and does not need line-of-sight requirement.

The benefits of WLAN is that it is able to reach out to area where is not possible for wiring. It also helps in reduce the future cost for office re-layout and re-location as it is able to complement the current wired network. Besides, WLAN is

more cost effective for infrastructure project and even in generate new revenue for Hotspot operator.

There are three types of deployment network models for WLAN. These models are Small Office Home Office (SOHO), Enterprise and Hotspot. A SOHO requires only sharing information and resources within a small office, which usually consist of a printer, a personal computer and a notebook. Compare to wired network, WLAN is more cost effective because it reduces the cost for wiring, especially when the re-layout has to take place.

Enterprise model is more complex than SOHO, where it needs to connect many computers, usually more than 50, to the internal server for information sharing and internet connectivity. It requires reasonable security feature to ensure the integrity and avoid unauthorized access. The architecture of this model can be deployed together with the current telecommunication technology such as GPRS and Virtual Private Network for accessing the resources in the office remotely.

Hotspot is mainly used by consumer for email, web surfing, Virtual Private Network, voice over IP service and multimedia entertainment. It requires intermediate service providers between the internet and the wireless access point. These service providers are responsible in providing the services as mentioned above.

Bluetooth is a computing and telecommunications industry specification that describes how mobile phones, computers, and PDAs can easily interconnect with each other and with home and business phones and computers using a short-range wireless connection.

With Bluetooth, users will be able to buy a three-in-one phone that can double as a portable phone at home or in the office, get quickly synchronized with

information in a desktop or notebook computer, initiate the sending or receiving of a fax, initiate a print-out, and, in general, have all mobile and fixed computer devices be totally coordinated. It requires that a low-cost transceiver chip be included in each device. The transceiver transmits and receives in a previously unused frequency band of 2.45 GHz that is available globally (with some variation of bandwidth in different countries).

In addition to data, up to three voice channels are available. Each device has a unique 48-bit address from the IEEE 802 standard. Connections can be point-to-point or multipoint. The maximum range is 10 meters. Data can be exchanged at a rate of 1 megabit per second (up to 2 Mbps in the second generation of the technology).

Furthermore, it has a frequency hop scheme which allows devices to communicate even in areas with a great deal of electromagnetic interference. Besides, it also has built-in encryption and verification that enhance the security features.

WAC will operate inside Wi-Fi connectivity. The reason for it will be discussed in Section 2.11.4 Wireless Protocol.

2.5. Wireless Application Protocol (WAP)

The WAP (Wireless Application Protocol) is a specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for internet access as well as intranet. It is a standard defined by the WAP forum which consists of Nokia, Motorola, Eriksson and Phone.com.

WAP is developed for the wireless devices which have several constraints as follow:

- Less powerful processing unit
- Less memory
- Restricted power consumption
- Smaller displays
- Different input devices (e.g. a phone keypad)

The wireless network as well has the following constraints:

- Less bandwidth
- More latency
- Less connection stability
- Less predictable availability

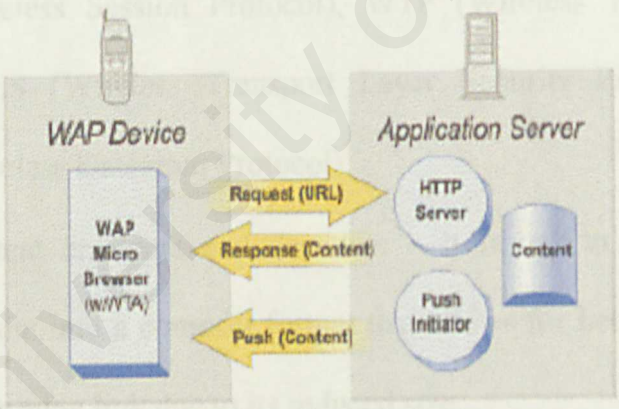


Figure 2.1: WAP Programming Model

The WAP programming model is actually the World Wide Web (WWW) programming model with a few enhancements. In WWW programming model, the client request for a URL from the application server. The application server will response to the client together with the content.

WAP programming model is optimizing the WWW programming model to match with the characteristic of the wireless environment. Figure 2.1 shows the WAP programming model. Instead of web browser like the WWW, the WAP device has the micro browser with telephony support, called the WTA. At the WAP device, it will request a URL from an application server. Like the WWW model, the HTTP server response with the content. In addition to the content response, the server will push the content to the WAP device as well with the Push Initiator at the server.

In order to optimize and enhance the connection between the wireless domain and the WWW, WAP utilize the proxy technology. The WAP proxy provides functions as follow:

- Protocol Gateway – A gateway to translate request from a wireless protocol stack. E.g., the WAP 1.x stack (from top to bottom) – WSP (Wireless Session Protocol), WTP (Wireless Transport Protocol), WTLS (Wireless Transport Layer Security Protocol) and WDP (Wireless Datagram Protocol).
- Content encoders and decoders – It is use to translate the WAP content into a compact format that allows for better utilization of the underlying link due to its reduced size.
- User agent profile management – Describe the client capabilities and personal preferences are composed and presented to the applications.
- Caching proxy – To improve the perceived performance and network utilization by maintaining a cache of frequently accessed resources.

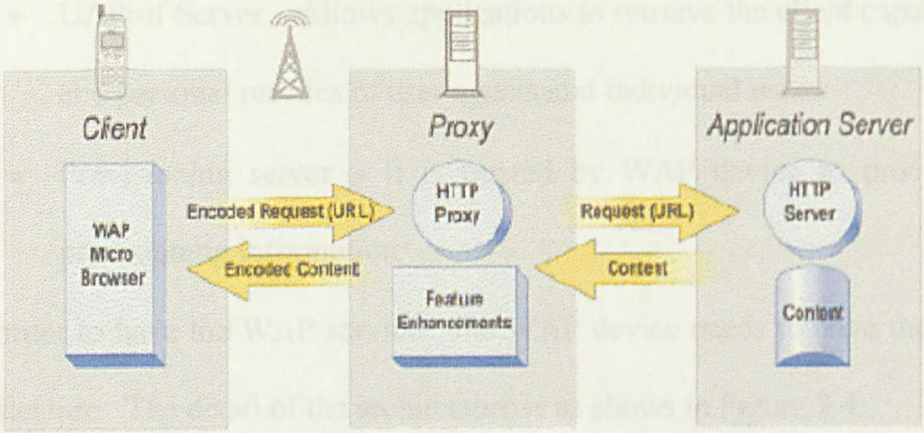


Figure 2.2: WAP Proxy

Figure 2.2 shows the WAP proxy. This infrastructure is meant to ensure the users are able to access a wide variety of internet content and application.

WAP architecture also includes supporting servers that provide services to device, proxies and applications as needed. These services are often specific in function but are of general use to a wide variety of applications. The supporting servers defined by the WAP forum are as below:

- PK1 Portal – As shown in Figure 2.3. It is to allow devices to initiate the creation of new public key certificates.

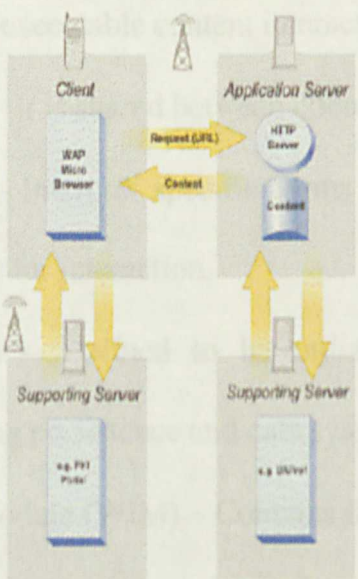


Figure 2.3: PK1 Portal

- UAProf Server – Allows applications to retrieve the client capabilities and personal profiles of user agents and individual users.
- Provisioning server – It is trusted by WAP device to provide its provisioning information.

In order to have the WAP services, the WAP device needs to have the WAP client architecture. The detail of the architecture is as shows in Figure 2.4:

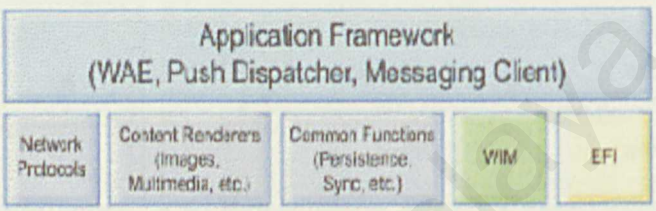


Figure 2.4: WAP Client Architecture

- Application framework – Provides the device execution environment for the WAP applications. WAP applications are consist of markup, script, style sheets and multimedia content. WAP Application Environment (WAE) defines the structure of various form of executable and non-executable content interaction.
- Network protocols – It is shared between client and the server.
- Content renderers – Interpret specific forms of content and present them to the end user for interaction.
- Common functions – Defined to be utilized by the application framework, including persistence and data synchronization.
- Wireless Identity Module (WIM) – Contains the identity of the device and cryptographic means to mutually authenticate WAP devices and servers.

- External Functionality Interface (EFI) – Provides the mechanism to access external functions that are embedded or attached to the devices.

For the WAP architecture, it is designed in a stack layer as shown in Figure 2.5. It is designed this way to provide a scaleable and extensible application development environment for mobile communication devices.

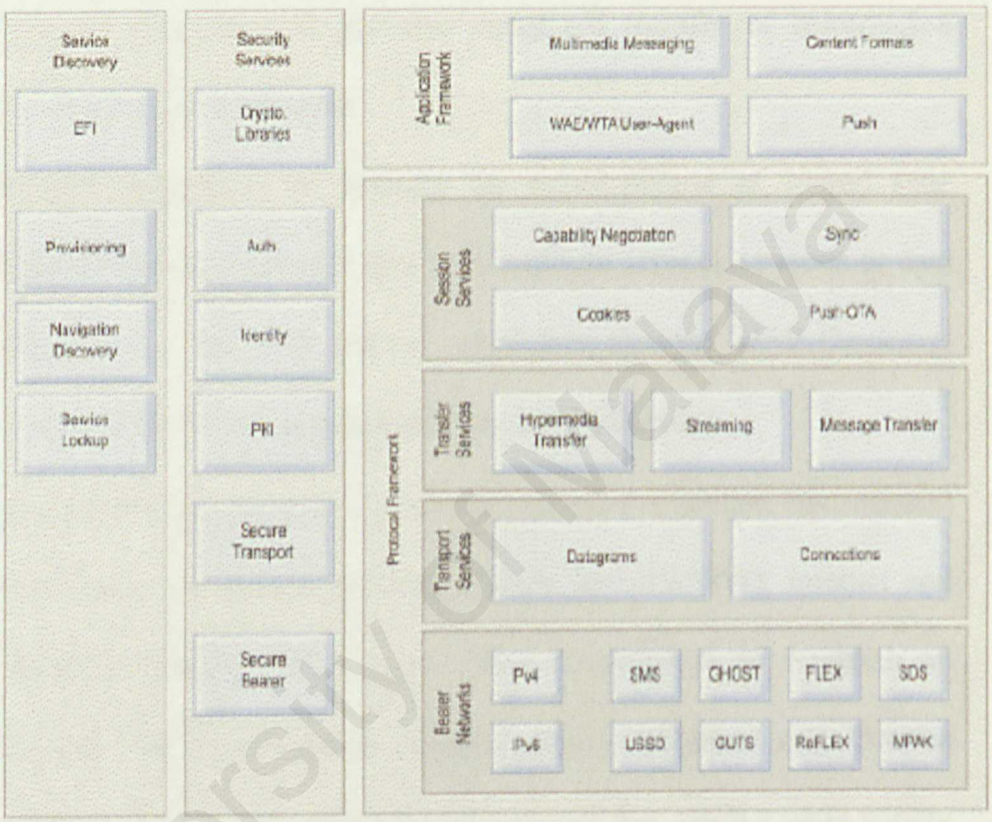


Figure 2.5: WAP Architecture

Each layer has a set of functions and/or services to other services and applications through a set of well-defined interfaces. It is also accessible by the layers above.

The explanation for each component in the WAP architecture is as follow:

- Bearer Networks – Protocols at this layer are designed to compensate for tolerate varying levels of services offer by the bearer, such as short message service, circuit switched data and packet data.

- Transport Services – Transport unstructured data across the underlying bearer networks. It provides a set of consistent services to the upper layer protocols and maps those services to the available bearer services. It also creates a common abstraction that is consistent across all bearers.
- Transfer Services – It provides structured transfer of information between network elements.
- Session Services – It provides the establishment of shared state between network elements that span multiple network requests or data transfers, i.e. the Push session establishes that the WAP Device is ready and able to receive pushes from the Push Proxy.
- Application Framework – A general-purpose application environment based on a combination of World Wide Web (WWW), internet and mobile telephony technologies. The objective is to establish an interoperable environment which allow operators and service providers to build applications and services that reach a wide variety of different wireless platforms in an efficient and useful manner.
- Security Services – It is a fundamental part of the WAP architecture and its services can be found in many layers. The general security facilities offered are privacy, authentication and non-repudiation.
- Service Discovery – It is a fundamental part of WAP architecture and the services provided can be found at many layers. The services include External Functionality Interface, Provisioning, Navigation Discovery and Service Lookup.

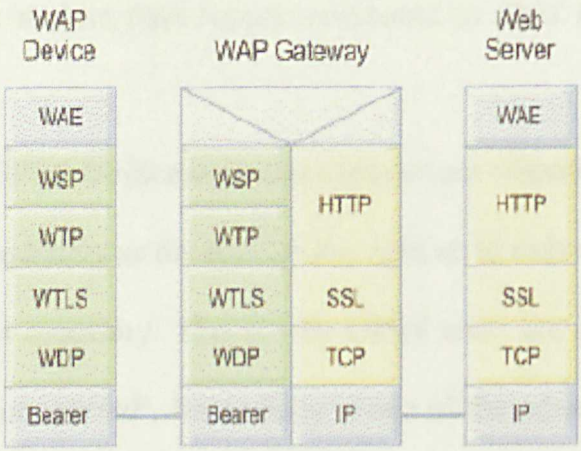


Figure 2.6: Example of WAP 1.x Gateway

As the services in the WAP stack can be provided using different protocols based circumstances, there are more than one possible stack configurations. Figure 2.6 is one of the examples that show the protocol stacks for the original WAP architecture.

2.6. General Packet Radio Service (GPRS)

GPRS is a non-voice value added service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service. It has several unique features which can be summarized as below:

Speed: Theoretical maximum speeds of up to 171.2 kilobits per second (kbps) are achievable by using all eight timeslots at the same time. This is about three times as fast as the data transmission speeds possible over today's fixed telecommunications networks and ten times as fast as current Circuit Switched Data services on GSM networks. By allowing information to be transmitted more quickly, immediately and efficiently across the mobile network, GPRS may well be a

relatively less costly mobile data service compared to SMS and Circuit Switched Data.

Immediacy: GPRS facilitates instant connections whereby information can be sent or received immediately as the need arises, subject to radio coverage. No dial-up modem connection is necessary. This is why GPRS users are sometimes referred to be as being "always connected". Immediacy is one of the advantages of GPRS (and SMS) when compared to Circuit Switched Data. High immediacy is a very important feature for time critical applications such as remote credit card authorization where it would be unacceptable to keep the customer waiting for even thirty extra seconds.

New Applications: GPRS facilitates several new applications that have not previously been available over GSM networks due to the limitations in speed of Circuit Switched Data (9.6 kbps) and message length of the Short Message Service (160 characters). GPRS will fully enable the Internet applications used on desktop from web browsing to chat over the mobile network. Other new applications for GPRS, profiled later, include file transfer and home automation- the ability to remotely access and control in-house appliances and machines.

Service Access: To use GPRS, users specifically need a mobile phone or terminal that supports GPRS (existing GSM phones do NOT support GPRS) and a subscription to a mobile telephone network that supports GPRS. The use of GPRS must be enabled for that user. Some mobile network operators may allow automatic access to the GPRS while others will require a specific opt-in. Besides, users need to have knowledge of how to send and/ or receive GPRS information using their specific model of mobile phone, including software and hardware configuration (this creates a customer service requirement). Also, it needs to have destination to send or receive information through GPRS. Whereas with SMS this was often another

mobile phone, in the case of GPRS, it is likely to be an Internet address, since GPRS is designed to make the Internet fully available to mobile users for the first time. From day one, GPRS users can access any web page or other Internet applications-providing an immediate critical mass of uses.

2.7. Personal Digital Assistant (PDA)

PDA is a term for any small mobile handheld device that provides computing and information storage and retrieval capabilities for personal or business use, often for keeping schedule calendars and address book information handy.

The term handheld is a synonym. Many people use the name of one of the popular PDA products as a generic term. These include Hewlett-Packard's Palmtop and 3Com's PalmPilot.

Most PDAs have a small keyboard. Some PDAs have an electronically sensitive pad on which handwriting can be received. Apple's Newton, which has been withdrawn from the market, was the first widely sold PDA that accepted handwriting.

Typical the PDA include functions like schedule, address book storage, retrieval and note entering. Many applications have been written for PDAs. Increasingly, PDAs are combined with telephones and paging systems.

Operating system for PDAs is varying from their manufacturer. It might use the operating system developed by its own manufacturer or from other company. The most common operating systems in-used are Microsoft Pocket PC and Palm OS.

2.8. 2.5 Generation (2.5 G)

2.5 G is a term used to describe the state of wireless technology and capability usually associated with General Packet Radio Services (GPRS) - that is, between the second and third generations of wireless technology.

The second generation or 2G-level of wireless is usually identified as Global System for Mobile (GSM) service and the third generation or 3G-level is usually identified as Universal Mobile Telecommunication Service (UMTS).

Most of the telecommunication companies in Malaysia provide the 2.5G services, such as Telekom, Maxis Communication and DiGi.

2.9. 3rd Generation (3G)

The next generation of wireless technology beyond personal communication services is called the 3 G.

The World Administrative Radio Conference assigned 230 megahertz of spectrum at 2 GHz for multimedia 3G networks. These networks must be able to transmit wireless data at 144 kilobits per second at mobile user speeds, 384 kbps at pedestrian user speeds and 2 megabits per second in fixed locations.

The 3G mobile communication standards and technologies will enable communication using voice, text, images and video. The following examples illustrate the possibility with 3G:

- 3G is being on a train and watching clips from your favorite soap
- 3G is being out and sending images back to headquarters
- 3G is using your phone to take holiday pictures to instantly send to friends at home
- 3G is using your phone for a video conference in a taxi

Multimedia Messaging (MMS) is one of the 3G services that combine imaging with mobility using exciting new content and high quality displays.

The first 3G networks launched in Japan in 2001 have already proven the possibilities, encouraging operators elsewhere to build their own 3G networks. In Malaysia, two of the telecommunication companies Telekom and Maxis Communication have started to provide 3G services starting this year, after they have succeeded in getting the license.

GPRS, EDGE, WCDMA and UMTS - the technologies leading up to 3G - may fascinate and surprise people, but the services and applications they enable will be easy and fun to use.

2.10. Two Similar Systems

2.10.1 Handisplay Smart Screen Creator

Smart Screen Creator is a server side technology that is developed to enable wireless devices to view online internet content based on real-time basis. It is developed by Handisplay (M) Sdn. Bhd.

It provides a cross platform support for different kind of wireless devices, range from PDA to WAP enabled phone. In order for Smart Screen Creator to work, it needs only a server and wireless network connectivity. Figure 2.7 shows the architecture of it.

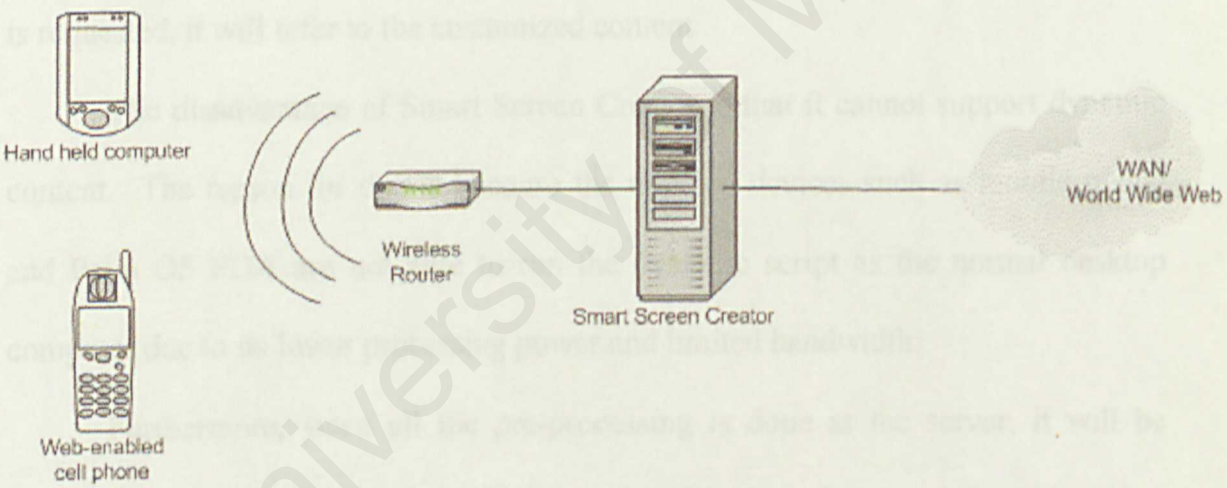


Figure 2.7: Architecture of Smart Screen Creator

First, the wireless devices will need to make connection to the server by simply type in the URL address. After the connection, the wireless devices or the client now able to surf the internet as they do using internet explorer on a desktop computer.

When the client makes the request, Smart Screen Creator will fetch the appropriate content from the internet. It will do some pre-processing before sending the content to the client.

Smart Screen Creator pre-processing is responsible to make the content suit for various types of wireless devices. It is able to detect what type of device make the request, what is the format used by the device and then change the content into the format that the device supported.

Another uniqueness about Smart Screen Creator is it capability in customizing the content. It provides a user interface at the server which enabled the administrative to view the pre-processed content. It allows the administrative to choose the content they want, save the customization and every time the same URL is requested, it will refer to the customized content.

The disadvantage of Smart Screen Creator is that it cannot support dynamic content. The reason for this is because the targeted devices such as mobile phone and Palm OS PDA are not able to run the dynamic script as the normal desktop computer due to its lower processing power and limited bandwidth.

Furthermore, since all the pre-processing is done at the server, it will be overloaded with heavy workload if there are many users log on at the same time. This situation can cause the performance and reliability going down.

Overall, Smart Screen Creator is one of the creative technologies available. Due to its ability in centralizing the process, different devices with different format can be used to log on to the internet and share information within the wireless network. Although it is still under development, it is a good alternative in providing information with mobility.

2.10.2. AvantGo Mobile Internet

The AvantGo Mobile Internet is a service that provides free interactive and personalized content and applications to handheld device or Internet-enabled mobile phone real-time via wireless connection or desktop synchronization.

AvantGo enable seamlessly transition between wireless and offline modes to browse the websites on mobile devices or select from the AvantGo channel for up-to-date news and events.

Basically, AvantGo channel is another web site. It is a simplified version HTML web site that is able to be viewed on mobile devices. To use AvantGo Mobile Internet service, one must first register and configure an account on the server. This account contains information about certain detail, such as username, password and the subscribed channels.

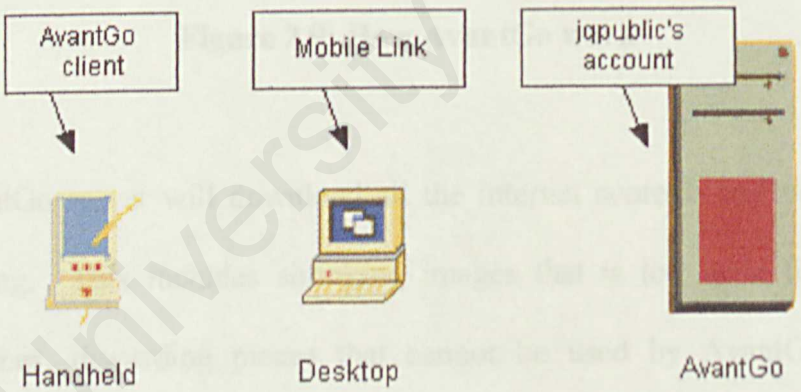


Figure 2.8: Component needed

AvantGo require a web browser to and a mini web server to be installed on the mobile devices operating system, such as Palm OS and Windows CE. It also requires a mobile link to be installed on desktop computer. It enables connection to AvantGo servers whenever synchronization is done. Figure 2.8 shows the component needed.

When synchronizing the handheld device, Mobile Link will take over and connect to AvantGo server. At first, the server will look at what channels is subscribed. Then, it will download those pages from the WWW. In many cases, these sites are distinct areas that contain pages optimized specifically for AvantGo channels. Figure 2.9 shows how it works.

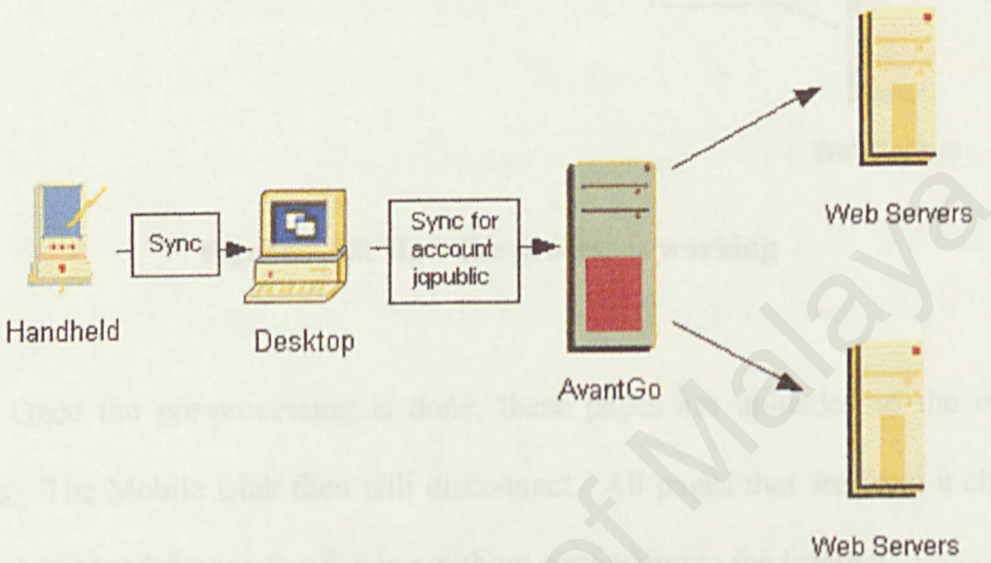


Figure 2.9: How AvantGo work

AvantGo server will download all the internet contents and performs some pre-processing. This includes shrinking images that is too large for display on mobile devices, discarding pieces that cannot be used by AvantGo client and compressing the rest of the HTML. Figure 2.10 shows the process.

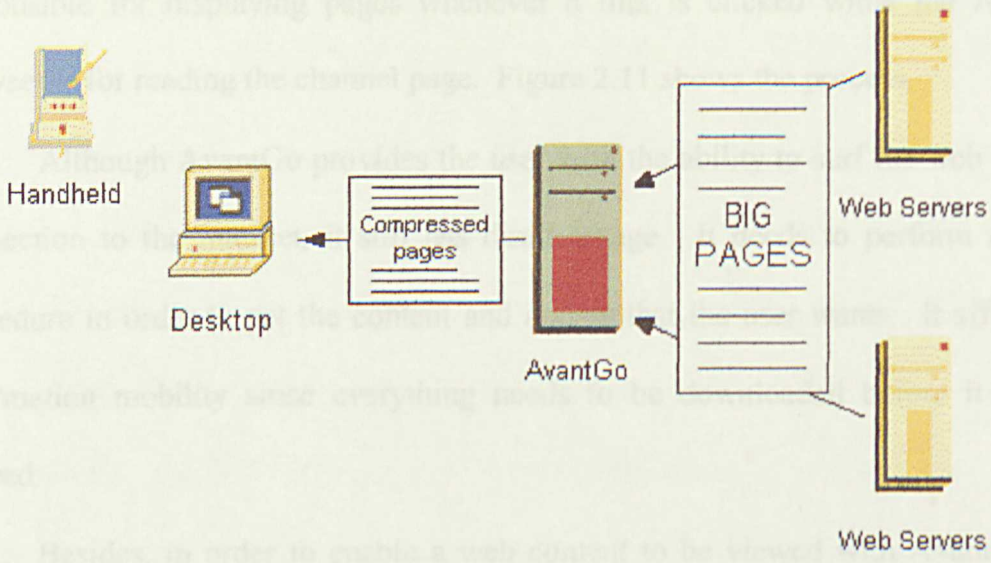


Figure 2.10: How the process is working

Once the pre-processing is done, these pages are uploaded to the mobile devices. The Mobile Link then will disconnect. All pages that are from a channel need to be uploaded once for viewing without connection to the Internet.

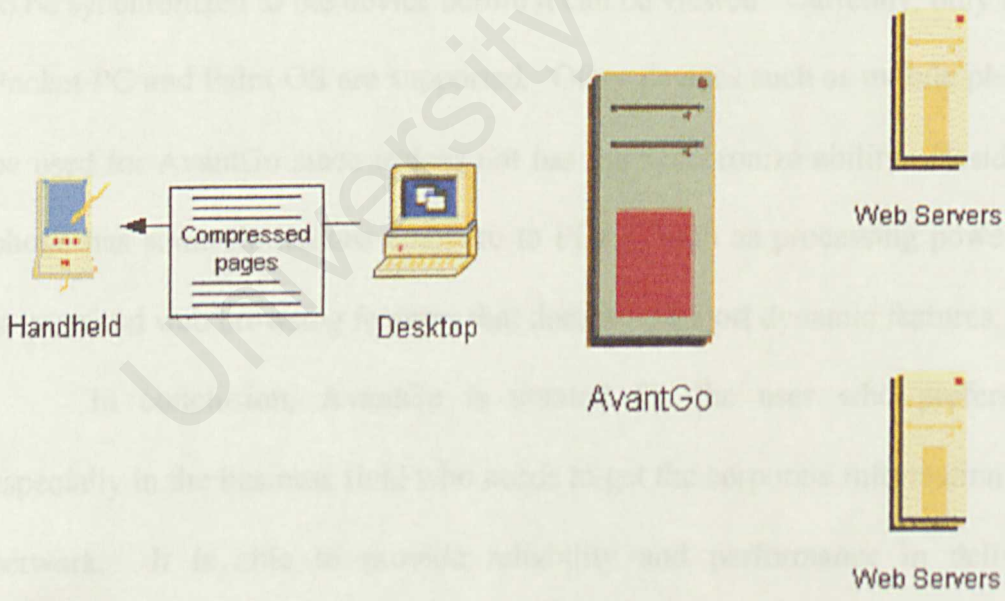


Figure 2.11: Desktop to client process

After the pages are uploaded to the mobile device, it is ready for browsing with the AvantGo client browser. The mini web server that mentioned earlier is

responsible for displaying pages whenever a link is clicked while the AvantGo browser is for reading the channel page. Figure 2.11 shows the process.

Although AvantGo provides the user with the ability to surf the web without connection to the internet, it still has disadvantage. It needs to perform a lot of procedure in order to get the content and update that the user wants. It affects the information mobility since everything needs to be downloaded before it can be viewed.

Besides, in order to enable a web content to be viewed with AvantGo, the web page must be constructed with the format predefined by it. If the web site is not build in AvantGo format, than it would not be able to be viewed. This is very cumbersome, as the web developer need to learn the format and redo everything in order to fit the format for mobility purpose.

Furthermore, it is device independent. As mentioned before, the content need to be synchronized to the device before it can be viewed. Currently, only PDAs with Pocket PC and Palm OS are supported. Other devices such as mobile phone cannot be used for AvantGo since it does not has the synchronize ability. Besides, mobile phone has some limitations compare to PDAs, such as processing power, memory storage and web browsing features that does not support dynamic features.

In conclusion, AvantGo is created for the user who prefers mobility especially in the business field who needs to get the corporate information within the network. It is able to provide reliability and performance in delivering the information, as the content is well organized at the server or desktop computer before passing it to the wireless device. As the number of AvantGo channel is increasing, it is very likely that it will become a standard in wireless field.

2.11. Technology

2.11.1. Software for WAC

2.11.1.1. Active Server Page (ASP)

ASP is a technology created by Microsoft. It is use to create dynamic internet content with the script contain inside an ASP page execute at the server. Like normal internet content, ASP page can be viewed on normal web browser, such as Internet Explorer. The content written in ASP is identified by the extension .asp.

In order to run an ASP page, the server must have either Internet Information System that comes with Windows 2000/Server or Personal Web Manager for Windows 98.

As part of the development, ASP will be used to create the interface that acts as a browser to surf the internet, create the content and then display it. Through the web-based interface created by it, user is able to interact with WAC to make request and view the output.

Further information about ASP can be found in the appendix A.

2.11.1.2. Component Object Model (COM)

COM is an interface specification for reusable software components. It is identifies by a unique ID. Each COM has its own functionality which can be used by other programs, web applications or by other COM object.

COM provides an interface that allows the program to access the embedded COM functions. It can be written in any programming languages such as C++ or Java, as long as the programming tools have the ability to implement the COM interface. Further information can be found in the appendix B.

For WAC, COM will be written using Visual C++. It is responsible for the underlying core processing, like the content formatting and information storing. This COM object will cooperate with the ASP interface in providing the information needed to create the output.

2.11.2. Operating System for WAC

The operating system that will be used for WAC development is Windows 2000 professional edition or server edition.

The reason for using Windows 2000 is because it has the Internet Information System which is very easy to be used to setup a web server and the available resources that support the use of ASP and COM.

2.11.3. Architecture of WAC

The architecture for WAC is Client-Server Wireless Local Area Network with access to the internet. This architecture can be as simple as a computer which acts as a server and a wireless device which is the client.

WAC will be running on the server. The client will first call for the WAC at the server to establish a connection. After a connection has been established, the client is able to request for information it requires. Further explanation will be provided in Chapter 3.

2.11.4. Wireless Protocol: Wi-Fi (802.11b)

Wi-Fi is a popular term for a high-frequency wireless local area network (WLAN). It is rapidly gaining acceptance in many companies as an alternative to a wired LAN. It can also be installed for a home network.

Wi-Fi is specified in the 802.11b specification from the Institute of Electrical and Electronics Engineers (IEEE) and is part of a series of wireless specifications together with 802.11, 802.11a, and 802.11g. All four standards use the Ethernet protocol and CSMA/CA (carrier sense multiple access with collision avoidance) for path sharing.

The 802.11b (Wi-Fi) operates in the 2.4 GHz range offering data speeds up to 11-megabits per second. The modulation used in 802.11 has historically been phase-shift keying (PSK). The modulation method selected for 802.11b is known as complementary code keying (CCK), which allows higher data speeds and is less susceptible to multipath-propagation interference.

2.11.5. Wireless Hardware

Wi-Fi compliant router or access point is needed to setup the network for WAC. It serves as a terminal between the wireless devices and server.

Router is a simple device that capable of doing certain specific tasks. It acts as a station on a network, which it has the addresses and links to two or more networks at the same time. It relays the packets among multiple interconnected networks with the used of the routing protocol. The routing protocol is a routing algorithm that allows the router to determine the shortest path or the best routing to send the packets to the destination.

Access point is a device that allows multiple users to connect to a wireless network at the same time. It is like a terminal that provides an open connection for any devices to a network. Unlike router, it does not have any addresses and links of other networks or routing protocol because it is not design to passing data between multiple networks.

2.11.6. WACluster

WACluster is a file storage that keeps the information of requested internet content. The information contain the address of the internet content, the total segmentation for that web content and the displaying sequence of the segments.

When WAC is servicing a request, the information inside WACluster will be referred first. If there is information of that particular internet content, it will be used to construct the display. If not, the internet content will be retrieved from the World Wide Web and will go through the processing module of WAC.

A file will be generated for each web's content to store its information. Each file will be identified by a unique ID number which is generated by the COM part of WAC.

2.11.7. WAC Server

The WAC server is considered as a web server. It is setup using Internet Information System (IIS) which is available with Windows 2000.

Internet Information System is a management console comes with Windows 2000 that enables information publishing over the internet.

The reason for choosing Internet Information System for the web server is because it has the Active Server Page snap-in that enable internet content built with this technology to be displayed, which is used to create the interface of WAC.

Furthermore, Internet Information System provides the easy implementation in setup the web server with a friendly user interface in a few steps. Further information is available in appendix C.

3.1. System Analysis

The modeling technique used to develop WAC is called transformation model. Process of this method is shows in Figure 3.1.

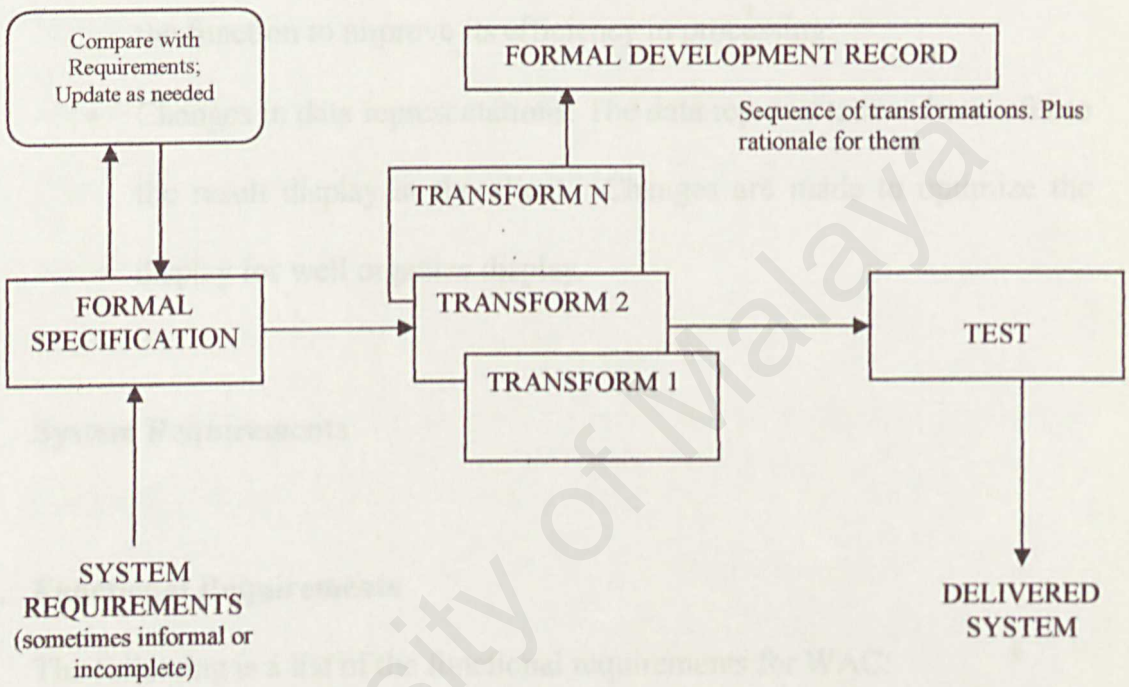


Figure 3.1: Transformation Model

Transformation model applies a series of transformation to change a specification into a deliverable system. It tries to reduce the opportunity for error by eliminating several major development steps. The transformations can include changing the data representations, selecting algorithms, optimizing and compiling.

As many paths can be taken from the specification to the delivered system, the sequence of transformations and the decisions they reflect are kept as a formal development record. This approach must have a formal specification expressed precisely in order for the transformations to operate.

The reasons for choosing this modeling technique for WAC development are as follow:

- Changes in selecting the pre-processing algorithm: The chosen algorithm might need to be change when it cannot used to handle majority of the internet content.
- Function optimizing: Either new features or changes will be made to the function to improve its efficiency in processing.
- Changes in data representations: The data representations here refer to the result display at the client. Changes are made to optimize the display for well organize display.

3.2. System Requirements

3.2.1. Functional Requirements

The following is a list of the functional requirements for WAC:

1. WAC must able to receive request from the wireless device. It will retrieve the requested internet content from the World Wide Web.
2. Identifying the format supported by the wireless devices. The formats refer to is either WAP or HTML.
3. Analyzing the content to check whether each element is suitable for converting to other format or not.
4. Divide the content into several segments. Each segment will become an independent WAP page when it is displayed on a wireless device.
5. Converting the content's elements inside each segment into WAP format.

6. Creating WAP page for each segment and organizing the sequence of each created WAP page to be displayed accordingly.
7. Create a text file which contains the information from the pre-processed content. This text file will be stored inside WACBase for future reference when the same content is requested.

3.2.2. Non-Functional Requirements

The following is a list of the non-functional requirements for WAC:

1. Check the requested web site to see whether it requires secure connection like Secure Socket Layer.
2. Secure connection need to be provided if it is needed.
3. Image processing. The image must be processed to fit the display mode of the wireless device.

3.3. Hardware Requirements

The basic hardware requirement for WAC is a desktop computer and a PDA.

The desktop computer is where the WAC will be installed. It plays the role as the server in the whole system. The specification for the desktop computer is as follow:

- 400 MHz processing speed.
- 256 MB memory.
- 40 GB Hard Disk Drive for storage.

The PDA is used to test the output of WAC. The requirement for it is the operating system preferable to be a Palm OS. This hardware is optional, because there is simulator that can be downloaded for development testing.

3.4. Software Requirements

Windows 2000 professional edition or server edition will be used as the operating system for WAC. Windows 2000 has the Internet Information System that enables the server to be setup easily with a few steps.

Furthermore, WAC is operating inside a network environment. Windows 2000 has the features that enables network setup can be done easily. It has the ability to automatically detect the network configuration and set it up according to the appropriate protocol.

Besides Windows 2000, Palm OS simulator is needed for testing purpose. This simulator will be used to test the output result of WAC. It is an alternative to own the real device.

3.5. Conclusions

Basically, WAC is responsible in retrieving the internet content, performing the internal processing and then converting the content to WAP format before sending it to the client. Compare to the available system in chapter 2, WAC is simpler because it performs less complex processing.

Although it is simple, it still has some constraints. Some web sites implement the security features such as SSL which is quite tricky. In order to view it, WAC must have a security feature like certificate authority to enable it to retrieve the content from the secure site. As a result, some of the web site might not be able to be displayed.

Besides the mentioned above, the internet content is quite difficult in a sense that there is no standard that define the structure and language that should be used.

As a result, it is very difficult to determine the appropriate internal logic of the pre-processing operation.

For the reason mentioned above, Windows 2000 is used as the platform for WAC for reasons of available resources, easy implementations and user-friendly development environment. There are library files, classes or other resources in a form of dynamic link library. These resources are predefined to perform certain task that can be used as part of WAC. Although it might not fully solve the constraints mentioned, but at least it is able to handle them until certain state.

Chapter 4: System Design

4.1. System Architecture Design

Architecture for WAC is client-server. It is implemented as shows in Figure 4.1.

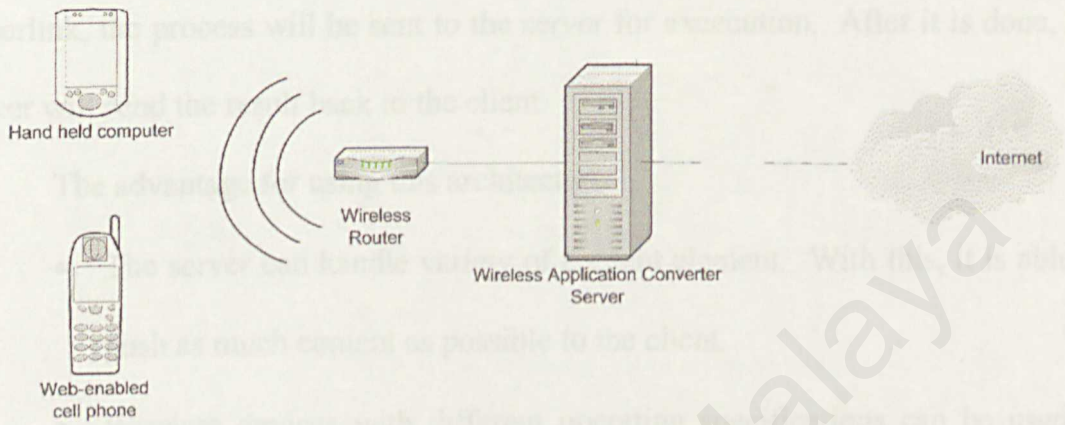


Figure 4.1: WAC Architecture

As shown in Figure 4.1, the WAC server is connected with the client's wireless device through the use of wireless router. The server must has a connection to the internet in order to retrieve the client requested content.

For the client to make request, it must establish a connection with the server. After the connection has been setup, client can start to make its request. The requested content will be pre-processed at the server before sending the end result back to the client.

It is considered as a three tiered client-server. A three tiered client-server is a technique that the data manipulations and logic implementations are executed at the server. The client is only responsible for representing the data without the need to perform any processing.

In this architecture, WAC is the server for the three tiered client-server. It is responsible in serving the request, retrieve the requested internet content, perform the processing and send the end result to the client.

The wireless device, which is the client, is only responsible to show the end result. When the user interact with the displayed content, such as accessing a hyperlink, the process will be sent to the server for excecution. After it is done, the server will send the result back to the client.

The advantage for using this architecture:

- The server can handle variety of content element. With this, it is able to push as much content as possible to the client.
- Wireless devices with different operating specifications can be used to surf the internet regardless their differences as the underlying processes are done at the server. It is an advantage to overcome certain constraint of wireless devices, such as lower processing power.
- The system is scalable in supporting different wireless protocol. For long term, WAC can serves different kind of wireless protocol other than WAP, such as I-MODE. It can be done by adding the specific processing for that particular protocol at the server.

Although the three tiered client-server has the advantages as mentioned above, but it has disadvantage as well. As the processing is concentrated at the server, it will become a burden if there are a lots of clients need to be served at the same time. Therefore, the server needs to have higher processing capabilities and larger storages to accommodate the requests.

4.2. System Module

WAC is divided into three modules. These modules are shown in Figure 4.2.

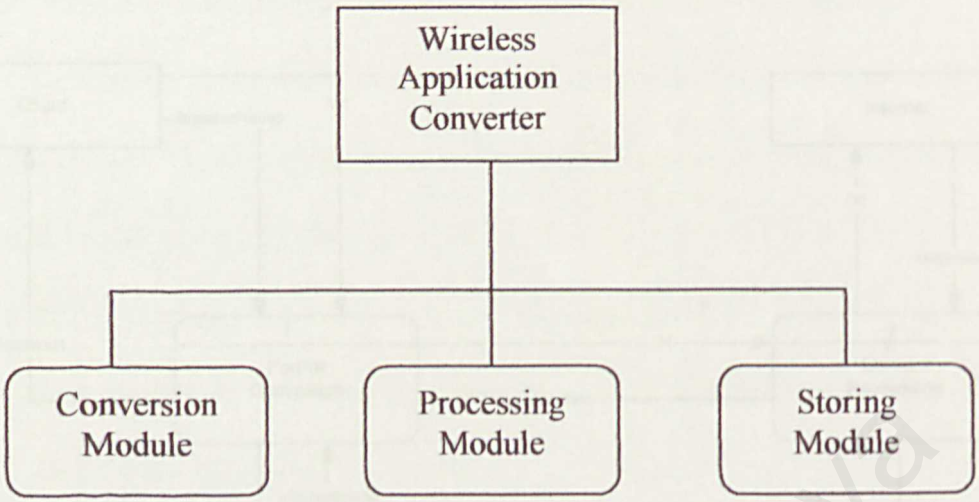


Figure 4.2: WAC system module

Processing Module: This module is the core for the pre-processing of WAC. The functions of this module include retrieving and analyzing the internet content, storing the necessary content and divide it into several segments.

Storing Module: This module is responsible for storing the pre-processed web content information in a form of text file, which will be stored in WACBase. It works closely with Processing Module because the information it needs is produced by that module.

Conversion Module: This module is responsible for converting the requested URL content into the format that supported by the wireless device. The functions of it are making request on client behalf to retrieve the content from the internet, get the pre-processed information from either WACBase or from Processing Module, and check for the client supported format, creating the content to be viewed by using the pre-processed information and sending the content back to the client.

4.3. System Functionality Design

Figure 4.2 shows the system functionality design for WAC.

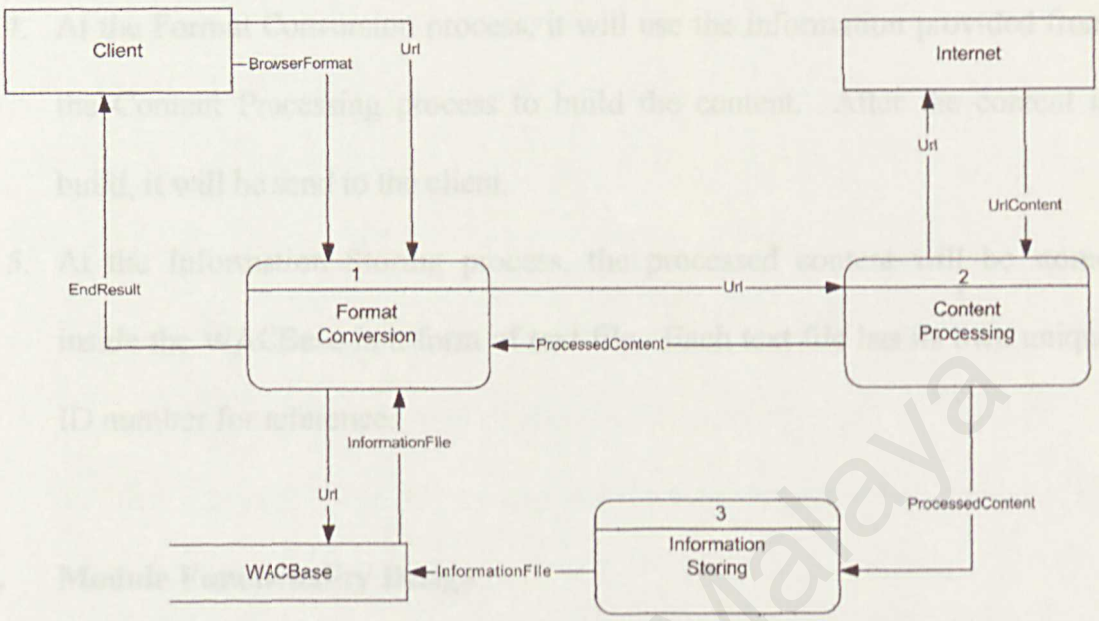


Figure 4.3: WAC system functionality design

The tree main processes represent the modules that have been mentioned earlier: Format Conversion represents the Conversion Module; Content Processing represents the Processing Module; Information Storing represents the Information Storing.

Explanation for the data flow is as follow:

1. First, the client sends the URL request together with its browser information to the Format Conversion process.
2. The Format Conversion process will first search for the similar URL information. If the information is available, it will retrieve it and use the information provided to build the content and sends the end result to the client. If the information is not available, it will send the URL to the Content Processing process.

3. At the Content Processing process, it will retrieve the respective URL content from the internet and processed it. The processed information will be sent to the Format Conversion process and the Information Storing process.
4. At the Format Conversion process, it will use the information provided from the Content Processing process to build the content. After the content is build, it will be send to the client.
5. At the Information Storing process, the processed content will be stored inside the WACBase in a form of text file. Each text file has its own unique ID number for reference.

4.4. Module Functionality Design

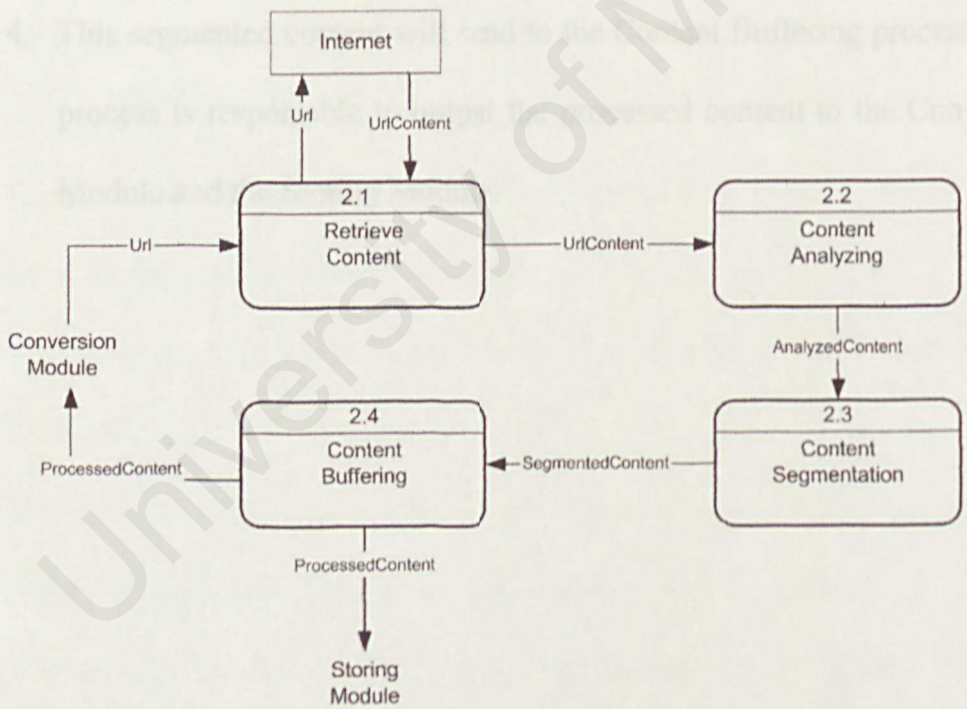


Figure 4.4: Processing Module functionality design

This part is the explanation of each module's functionality. It explains the processes involved inside each module that are cooperating in performing the module's task. Figure 4.4 shows the functionality design of Processing Module.

Processes involve in this module are Retrieve Content, Content Analyzing, Content Segmentation and Content Buffering.

The data flow for this module is as follow:

1. First, it receives the URL content from the Conversion Module. Then, the Retrieve Content process will use the URL and retrieve the content from the internet.
2. Then, the content will send to the Content Analyzing Process for analyzing. This process is responsible in scanning the content's elements and determines the needed elements by filtering.
3. The analyzed content is send to Content Segmentation process for division into segments.
4. This segmented content will send to the Content Buffering process. This process is responsible to output the processed content to the Conversion Module and the Storing Module.

For the Storing Module functionality design, it is shows in Figure 4.5.

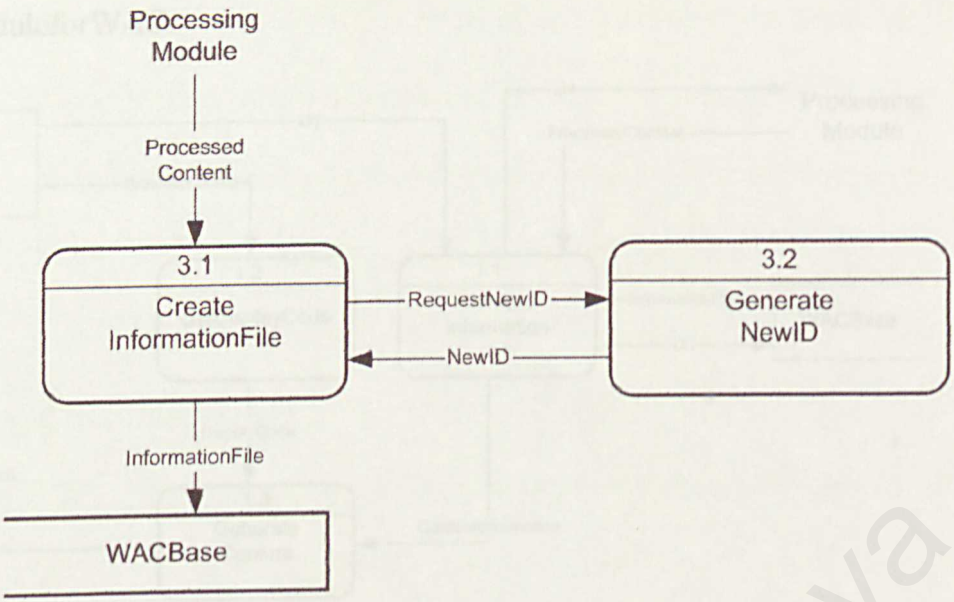


Figure 4.5: Storing Module functionality design

The processes include in this module are Create Information File and Generate New ID. Following explain the data flow for this module:

1. First, Processing Module will send the processed content to the Create Information File process in this module.
2. Then, Create Information File process will request for a new file ID from the Generate New ID process.
3. Create Information File process will create a information file with the information obtained from processed content and the new ID.
4. The information file created will be stored inside the WACBase for future reference if the same URL is requested.

Figure 4.6 shows the Conversion Module functionality design. It is the finalmoduleforWAC.

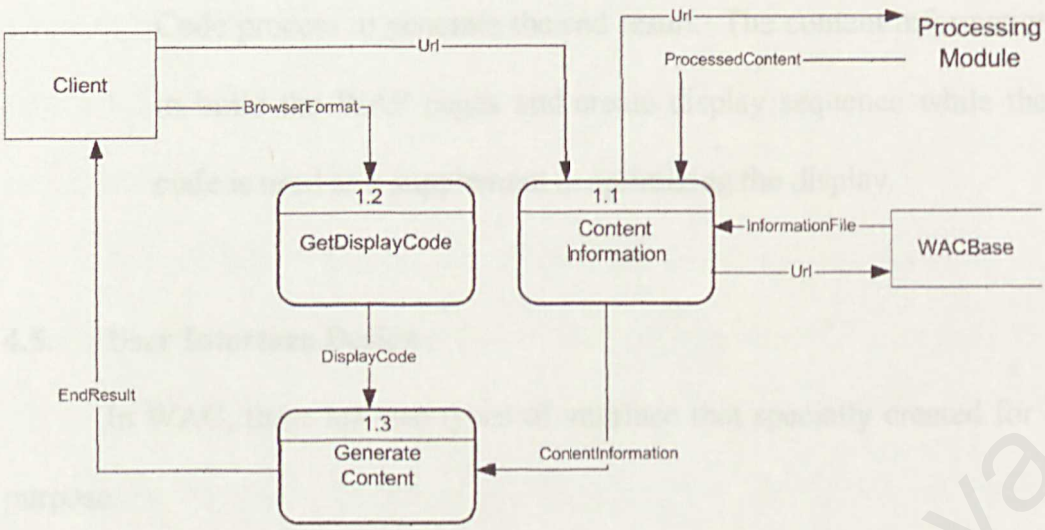


Figure 4.6: Conversion Module functionality design

The processes involved in this module are Content Information, Get Display Code and Generate Content. The following explain the data flow for this module:

1. First, the module will receive the requested URL and browser format from the client.
2. Content Information process will receive the URL and retrieves the appropriate content information. At first, it will check the WACBase with the URL for the similar content. If the checking return true, than the appropriate information file will be retrieved from WACBase; if not, the URL will be sent to Processing Module. The Processing Module will return the related processed content for generating the end result.
3. At the same time, the Get Display Code process will get browser format information to generate the appropriate display code. This display code will be used by the next process in producing the end result for client.

4. Finally, the Generate Content process will get the content information from Content Information process and display code from Get Display Code process to generate the end result. The content information is used to build the WAP pages and create display sequence while the display code is used as a supplement in optimizing the display.

4.5. User Interface Design

In WAC, there are two types of interface that specially created for different purpose.

For the wireless device, the interface is simple with just only a query column for the user to make the request. This interface is shows in Figure 4.7.

Figure 4.7: User interface for wireless device

A complex interface is created at the server. It allows user to make request and to see the structure of the end result that will be displayed at the client. The structure is about the segmentation of the requested content, the converted elements and also the display sequence.

This interface is build by using the Active Server Page. It will detect the request make from either the server it selves or from the client. After the detection, it will load the appropriate interface before the user can make request.

4.6. Conclusion

WAC is designed to be a three tiered client-server architecture. This architecture enables devices with different format to retrieve the internet content regardless to the knowledge of the underlying processing. Besides, it is also expandable as new support for new devices can be added to it.

In exchange, it can be overwhelmed by too many requests at the same time. Therefore it needs computer with higher processing power and storage to do the job.

WAC is divided into three modules: Processing Module, Conversion Module and Storing Module. These are the basic modules that have been identified. In future, new module can be added to enhance the performance.

Since the devices make request has the possibility to be either HTML or WAP, therefore the interface is designed in a way that suit these two formats. One is a simple interface for display at the wireless device while the other is a more complex interface build for improvement purpose that will be showed at the server.

Overall, WAC is designed in this way to support different kind of devices that come with different format. Although it only supports WAP and certain functions at the development states, there might be considerations in putting in more features that support variety of format or provide new functionality such as editing.

Chapter 5: System Implementation

5.1. Introduction

System implementation is the next step in realizing the system. It is a process that transforming each requirement that had been phased out during the design stage into executable program codes.

In the design stage, WAC is consists of three modules. The coding approach is based on these modules, which mean each module will have different way of implementation. For example, the processing module is the core of WAC. Since it requires real-time processing power, therefore it will have to be written using real-time capable scripting language, such as C++. On the other hand, the Conversion module involved displaying output on different devices, based on the type of web browser they used. Since IIS is used, therefore the possible implementation used is ASP. Further explanation will be done in the following section.

5.2. Processing Module

Processing module is the core of WAC. It does all the internal processing, such as read the source stream, identify the elements, store the elements using data structure approach and passes the elements to the Conversion module for further display processing.

Before going into further details, here is a brief explanation about how the Processing Module works. As shown in Figure 5.1, the first step of what it does is to open the source file (in HTML format) specified by the received variables and store it in stream form. Then, it will process the stream by identify the HTML elements, the name of the tags, the tags' properties and the corresponding text content. The

elements will be pushed into a link list. After that, the link list will be passed to the Conversion module for further processing.

```
FILE *fSrc;
fSrc = fopen( sSrcLocation.c_str(), "r" );
char ch;
while( (ch = fgetc(fSrc)) != EOF )
{
pszHtml += ch;
}
CNode *pNode = new CNode;
COutProcess *pOut = new COutProcess;
string sOut = "";
// ProcessNode( pszHtml, pNode );
// if( pNode->HasChild() )
// pNode->GetChildNode(sOut);
// pOut->ProcessOutput(pNode, m_sOutPath, m_sID); //Using linked pointer.
SpecificPush( pszHtml, pNode );
if(!m_bWap)
pOut->SpecificOutputProcess(pNode, m_sOutPath, m_sID, m_sDomainUrl);
//Pushing
all to one.
else
pOut->WapOutput(pNode, m_sOutPath, m_sID, m_sDomainUrl); //Pushing
all to one (WAP);
if( pOut->CreateIndexFile(m_sOutPath, m_sID) )
{
/*
Index file being created.
```

Figure 5.1: WAC – Processing Module Sample

5.2.1. Processing Module: Coding

The Processing Module is part of the WAC component, WAC.dll. It is a Component Object Module (COM) created using Visual C++. The object-oriented approach is used to program this module.

Object-oriented programming is an approach where it is using object. An object is an entity with its own properties and functions. An object can be used to perform several processes and it can be reused. The reason for choosing this approach is because the objects defined can be reused throughout the whole program.

Since the function is already defined in an object, all we need to do is just calling the object and ask it to perform that particular function. It helps in reducing the need to code that particular function, which will make work more difficult.

The language used to code this module is C++. It is coded using Microsoft's Visual C++. Visual C++ is a very useful development tool. Besides compiling program codes, it has several features that aid the developer in developing the application according to their needs. At first, Visual C++ will create the workspace that suits the application. For example, if the application is a COM, then the developer will specify the specification. After that, Visual C++ will generate the COM workspace, with all the necessary basic codes served as blueprint. The developer will only need to add in the codes they wanted to make a complete COM. After it is complete, Visual C++ will help to build the COM, and then it is ready to be used.

C++ is a scripting language that supports object-oriented programming. Each object is identified by the keyword class. Inside each class, there will be several functions/methods and properties. These functions/methods and properties can be categorized into three groups: private, public or protect. Private specified that only the member functions of the class can access the functions/properties in this category. Public category functions/properties can be accessed outside of the class. Protect category functions/properties only allow the member functions of the class it selves or its sub class to access.

There are 3 classes in this module: CNode, CTag and CProperty. CNode is a class that performs operations on the source stream. It identifies the HTML elements: the name, the associate properties and the content. After it manages to

identify the elements, it will store them into a stack. This stack will be passed to the Conversion module for further processing.

CTag is a class that stores information of an HTML element. It will store the name of the tag, the properties of the tag and the content associated with the tag. It has functions that will return the information stored upon request.

CProperty is a class where it stores information of the element's properties. It will store the name of the property and the associate value. Like CTag, it also has functions that will return the information stored when it is requested.

5.3. Conversion Module

The conversion module is responsible for constructing the end-result display. It is separated into two parts: the internal processing, which is part of the WAC.dll, and the displaying process, which is part of the ASP scripting.

The internal processing will use the stack produced by the processing module to construct the output with a mathematical method. At first, the method will determine the maximum number of elements should be placed inside an output. After determine the number, it will construct the output by writing the elements into a text file. The number of text files is determined by the maximum number of elements. If the maximum number is reached, than the following elements will be written into a new text file.

5.3.1. Conversion Module: Coding

The internal processing is written using the object-oriented approach. There is only one class defined, COutProcess. COutProcess contains methods and properties that responsible for the display processing. The processing method uses

the mathematical approach as described above. After the processing is done, COutProcess will create an index file that contains the link to each processed output.

The displaying process, which is part of the ASP, is responsible in displaying the output. The approach used is structure programming. There are several functions inside the ASP that working together to perform the task. First, it needs to know what is the browser that making the request. If it is a WAP browser, the displaying process will construct the output according to the WAP standard. If not, it will construct the output according to the HTML standard.

The conversion module will provide the user with a web based interface. There are two types of interface, which are written in WML and HTML. The reason for having two types of interface is because the mobile devices, such as cell phone or Palm PDA is using WAP standard, while other devices such as Pocket PC is using HTML standard. As mentioned before, the displaying process of this module will detect what is the browser used by the user. It is determined by a server variable called User Agent, which is included inside the HTTP request header.

5.4. Storing Module

The storing module is responsible in storing the processed files. It is not using any database for storing purpose. Instead, it stores all the processed files into one folder, called WACluster. WACluster is the main folder that stores all processed files for the requested web contents. The processed files are store according to the requested web content.

At first, when the user requesting for a web content through WAC, the request will be given an identification number (ID). WAC will create a folder inside WACluster, using the ID as the folder name. This folder will be used to store all the

data needed for that particular request. Such data are the source file and the processed files.

5.4.1. Storing Module: Coding

The management of WACluster and its contents are done within the ASP. There is one object that can be used to perform this task, the FileSystemObject. It is a scripting object that used at the server to access the file system. It allows user to:

- Get and manipulate the information about all of the drives in the server.
- Get and manipulate information about all of the folders and sub-folders on a drive.
- Get and manipulate information about all of the files inside of a folder.

It can be used to perform any task on the file system aside from setting security information.

The identification number is generated by a COM object called genuid. It is a dynamic link-library, which can be used to generate a long string of number. The reason to generate a long number is to avoid the possibility of unauthorized access to the files inside the folder. Each time it will generate a different string of number when it is called to avoid collision among the folders.

5.5. Complement Component Object

There is one COM called HtmTear, which is not included inside the design module. It is a standalone COM that responsible in making connection with the requested server, retrieved the source file and store the source file inside the WACluster.

HtmTear is written using Microsoft Foundation Class (MFC). It is a predefined library written in C++. Writing MFC application is different from ordinary C++ application, for it has its own structure and its own syntax, which is totally different from the conventional C++ structure. There are many classes inside MFC that help the developer to develop application faster, especially Windows based application.

The classes that used by HtmTear is called CInternet. This class is used to perform internet connection using the HTTP protocol. Although it does not perform as well as normal web browser, but it is useful for its ability in retrieving data from destination web server.

When HtmTear create the internet connection, it will retrieve the data from the targeted web server. The data will be written into a buffer. After the writing process is finish, HtmTear will create a source file of that particular request, using all the information given such as the folder identification number inside WACluster and the source file name. The data inside the buffer will be written to that file and after that, it is ready for further processing.

5.6. System setup

After the system being implemented, the next step is system setup. WAC consists of a collection of ASP files, HTML files and three dynamic-link library COMs.

The ASP files and HTML files need to be stored inside one folder, which act as the root for IIS to refer. The root folder will be called wwwroot. After storing the files, the next step will be configuring the IIS to point to this root folder.

Inside IIS management console, expand the tree until it reaches the default web site. Right click on the default web site then click the properties. Inside the properties dialog, point to Home Directory section. Set the local path to where the wwwroot is located, for example C:\Handisplay\Inetpub\wwwroot. After setting the root, the next step is to set the default file. The default file is the file that will be referred to when a web application is accessed. Go to the Document tab under the same dialog box, add a new document called index.asp and move the file name to the highest location. The file name at the highest location will have the priority to be referred first. After it is done, click OK to come out from the dialog box.

The next step is to register the COM. In order to use a COM, it needs to be register into the registry. When a COM is called, Windows will refer to the registry to find the match. If the match is found, Windows will use the information obtained from the registry to call the COM.

First, store all the COM into Windows system folder, C:\WINNT\system32. After storing the COM, open the command prompt. Then, in the command prompt, type the regsvr32 [COM name] (i.e. regsvr32 WAC.dll) to register the COM. There will be a dialog prompted stated that either the registration is success or fail. One COM can only be registered at a time, therefore this process need to be repeated until all COM had been successfully registered.

5.7. Summary

The system implementation of WAC is done based on the modules it has. Each module uses different implementation approaches. These approaches are the coding approach, the scripting language used and the development tool.

To put it in a nut shell, the two major coding approaches used are object-oriented and structure programming. All COM components are written using the object-oriented approach, while the ASP is written using structure programming approach.

The scripting languages used are C++, VBScript, HTML and WML. C++ is used to code the COM, while VBScript, HTML and WML are used to code ASP. The development tool used to develop COM is Visual C++, while Visual InterDev is used to develop ASP web based application.

Chapter 6: System Testing

6.1. Introduction

System testing is a series of identification process, where the purpose is to discover the weaknesses. These weaknesses will be used to improve the system performance. However, not all weaknesses can be corrected due to the outcome such as non-standardized internet content structure.

The following section will explain how the system testing is done on WAC. For a brief introduction, the testing is done on each module. These testing are unit testing, module testing, and integration testing as well as real-world testing. These testing are done either during or after the system implementation.

6.2. Types of Testing

As mentioned before, there are several types of testing being done for WAC. These testing are unit testing, module testing, and integration testing as well as real-world testing.

6.2.1. Unit Testing

Unit testing is the first type of testing being implemented. It is performed on each basic function either during or after system implementation. It is done on each individual function. The purpose of unit testing is to make sure the function is working as required and reduce the error rate of the module performance.

This testing is done to examine and review the code of each function. At first, the function is constructed into an executable program. The testing resource for the function is specified within the code it selves. The resource might be a simple

string or a simple file. Extra codes are added into the main code for debugging purpose. This way, the bug and error can be easily identify and help to ease the debugging and correction process.

6.2.2. Module Testing

Module testing is the next testing stage after unit testing. It combines all units under the same module for testing. In other word, it is an integration test among the units. Although the units have been tested, module testing will make sure the integration among the units are working as required. If there are errors during the testing, each unit will need to be tested again to identify the causes and correct it.

Module testing is done by combining the related functions of a module into one executable program. The testing resource is either the combination of all units testing resources or a new test file. It is specified inside the executable program.

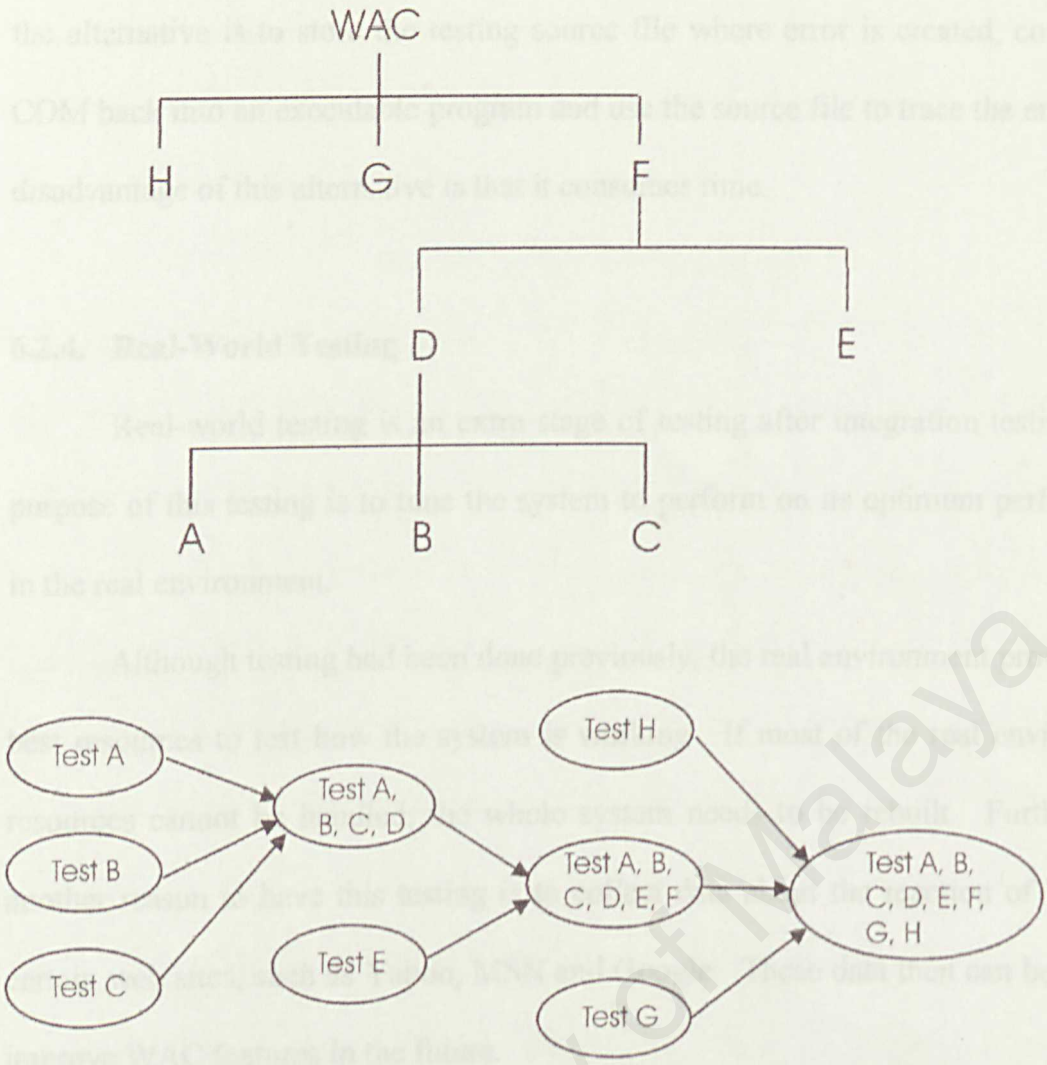
Module testing consumes more time compare to unit testing. The reason is because the size of the program, where it is the combination of all units and will consume more time in debugging and correction. Furthermore, it is very difficult to trace the error as well due to the complexity of the code, although extra code had been added.

After the module testing had been done successfully, the next step is to perform the integration test. Before performing the integration test, some of the modules need to be converted into other format. If the module is COM, than it need to be converted into a dynamic link-library file, in order for other module, which is in ASP, to communicate with it.

6.2.3. Integration Testing

Integration testing is performed by combining all three modules together. These three modules are working together to perform the required functionality of WAC. Integration testing is done to make sure that each module, after being combined, will still perform as it is required.

Integration testing uses the approach of Bottom-Up method. This method stated that every unit would be tested individually. After each unit being tested, the testing advances to the next level, where it involves combination of several units. The process continue until it reaches the highest level, where the combination of all tested units to perform the final test. Figure 6.1 shows the Bottom-Up approach used for WAC.



Legend

A: CNode
B: CTag
C: CProperty
D: WAC.dll
E: WACTear.dll
F: Processing Module
G: Conversion Module
F: Storing Module

Figure 6.1: WAC – Bottom-Up Testing

Error tracing will become more difficult every time it goes up one level. If error occurred in the highest level, it will be very hard to trace the error. Fortunately, ASP has this ability where it will inform where the error is and allow us to correct it. On the other hand, if the error is inside the COM it selves, it will be even more difficult, as dynamic link-library it selves is not an executable program. Therefore,

the alternative is to store the testing source file where error is created, convert the COM back into an executable program and use the source file to trace the error. The disadvantage of this alternative is that it consumes time.

6.2.4. Real-World Testing

Real-world testing is an extra stage of testing after integration testing. The purpose of this testing is to tune the system to perform on its optimum performance in the real environment.

Although testing had been done previously, the real environment provides the best resources to test how the system is working. If most of the real environment resources cannot be handled, the whole system needs to be rebuilt. Furthermore, another reason to have this testing is to collect data about the reaction of WAC to certain web sites, such as Yahoo, MSN and Google. These data then can be used to improve WAC features in the future.

6.3. Testing Resources

Table 6.1 below shows the testing resources being used for each type of testing. The local.htm file is in the appendixes.

Table 6.1: Testing Resources

Testing Type	Types of Resources	Example
Unit	Simple HTML String	<html><head></head><body></body></html>
Module	HTML String	<html><head><title></title></head><script></script><body></body></html>
Integration	Simple Internet Content	http://ocalhost/local.htm
Real-World	Internet Content	http://www.yahoo.com, http://www.google.com

6.4. Changes Done

There is one modification done on the Processing Module. At first, Processing Module uses tree data structure method to store the elements. After the testing had been done, this method is not suitable in doing the job due to the complexity of the algorithm it selves and the structure of the internet content.

An example of a HTML tree data structure is shown in Figure 6.2 below. Starting from the root, there is HTML, then follow by its children nodes, HEAD and BODY. Inside each tag, there are other HTML elements, such as META, TITLE, SCRIPT, PARAGRAPH and HEADING.

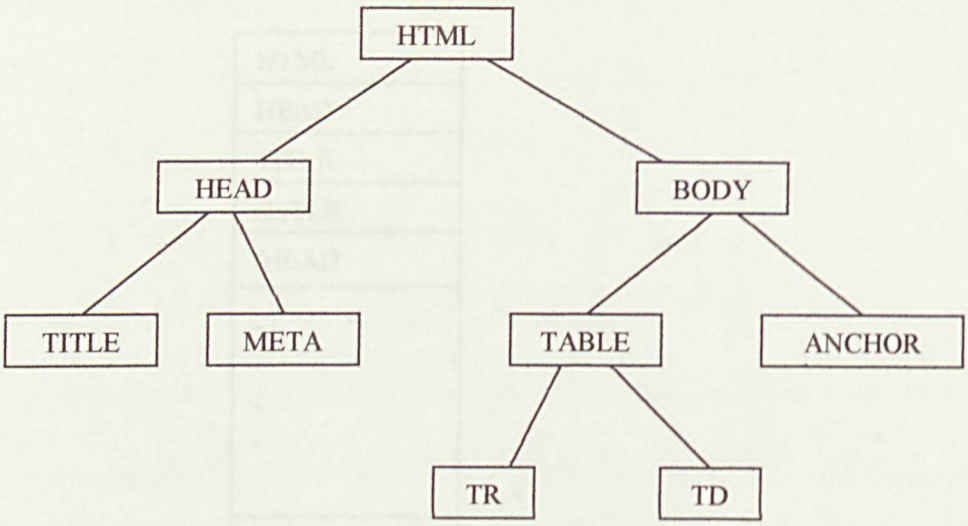


Figure 6.2: WAC – Tree Structure

During the testing, the tree structure is suitable for smaller uncomplicated internet content. But when it comes to much more complicated internet content, it cannot handle the processing properly. If the tree is very big, it is very difficult to traverse to and forth to obtain the information from each node.

After careful consideration, it is better to use a stack structure to store the HTML elements. The reason is because it is easier to implement in Visual C++ by using Standard Template Library (STL), compare to tree structure where we have to define our own algorithm. Figure 6.3 shows an example of stack structure.

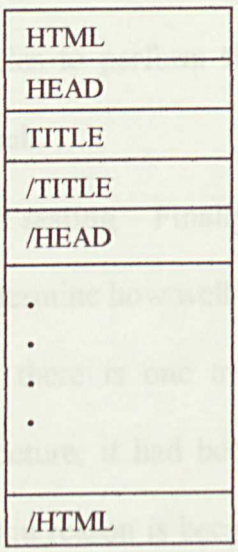


Figure 6.3: WAC – Stack Structure

Since STL provides functionality to manipulate the stack, the information stored in the stack can be easily obtained. Processing Module will be able to determine the wanted and unwanted element by specifying the condition. Although it requires the developer to hard code the specification of the condition, the ease-of-use is the reason why it is used.

6.5. Summary

System testing is needed to make sure the system is performing as it is required. It is to determine whether modification need to be done on the current coding and algorithm used, as well as to collect data for future enhancement.

There are four stages of testing. They are:

- **Unit testing** – It is responsible in testing the basic functions.
- **Module testing** – It combines functions that belong to a module for further unit testing.

- **Integration testing** – After each module being tested, the next stage is to combine all modules to perform the integration testing. This testing uses Bottom-Up approach.
- **Real-environment testing** – Finally, the system will be tested inside the real environment to determine how well it can perform.

After the testing, there is one modification done to Processing Module. Instead of using tree structure, it had been change to stack structure to store the HTML elements. The main reason is because stack is easier to manipulate with the help of Standard Template Library compare to tree, where developer need to come out with their own algorithm.

System testing is very important, as it helps to recognize the pros and cons of the internal implementation. It consumes time to perform testing on each component. However, it helps to reduce the error rate and makes the system works better when it is implemented in the real environment.

Chapter 7: System Evaluation

7.1. Introduction

System evaluation is a process taken to measure the achievement of the developed system. The evaluation process will determine whether the system have fulfilled the requirements, achieved the goals and the worth to advance.

This chapter will explain the evaluation done by the end user and the developer. It will cover the problems encountered during or after the implementation, the strengths and the constraints, and future enhancements.

7.2. Problems Encountered

Problems are encountered during the system implementation and testing stage. Such problems are categorized into developer's skill, development resources, and raw materials.

7.2.1. Developer's skill

The problems encountered here are during the system implementation. Such problems are as explain below.

Lack of programming experience – When developing WAC, the problem faced is the programming skill and experience. As mentioned in previous chapter, WAC is developed using COM and ASP. Although having experience in writing C++ program, but when it comes to write COM, there are difficulties as writing COM is totally a new expose. The COM can only be written by using the wizard available in the development tool, but cannot write it start from scratch, for COM has its own way of writing.

Beside COM, MFC is also a problem, especially in writing the HtmTear COM. As mentioned before MFC is a set of predefined classes, which aid developer in building Windows application. Like COM, there is a special way of writing MFC. As I never expose to MFC before, there are difficulties during writing using MFC. Besides MFC, the standard template library (STL), which is a predefined set of template library that help in writing data structure application is a problem, for I have to know which type of STL should be used, what are the proper syntax, how it is used and what are the complement header need to be included.

The only way to solve these programming skill problems is by reading relevant programming resources, either from the internet or books. Although the online resources are there, it stills not enough because a novice developer might not be able to understand it, as most of the resources assumed that the users have experience in such field.

7.2.2. Development resources

The development resources here refer to hardware and software. WAC is developed to be used in a wireless environment. Devices such as wireless network devices and a mobile device such as PDA are needed. Since the costs to have such devices are expensive, especially a PDA, therefore it is hard to test WAC in the real environment. Therefore, the only way to test WAC is by using simulator, which has similar functions of a PDA.

The software problems are the simulators used for testing purpose. Although there are varieties of simulators can be used, but the configurations of each simulator are different. Some might even affect the whole operating system, such as the

Microsoft's Embedded Studio Pocket PC 2002 simulator, that could cause the performance of the whole operating system decrease.

7.2.3. Raw Materials

The raw materials refer to the internet contents. The reason for saying that internet contents can cause problem is because so far there is no standard that defined how internet content should be built.

The major problem here is that not all internet contents can be processed correctly by WAC, as the internal processing algorithm can only suit some standard. There is no way WAC can hundred percent handle all internet content if there is no standard being initialized. The only thing that can be done is to optimize WAC to handle as much internet content as possible.

7.3. Evaluation by end users

Although WAC being tested successful by developer him selves, it needs to be tested by the end user as well. The reason to have other people to do the testing is to get opinion the feed back on how the system being perform, how user friendly it can be and how well it can meet the end user requirement.

Several feed back from the people who are using WAC said that it is a very innovative system, where it allows different platform to view the same internet content, while optimizing the content to suit different mobile device.

Since WAC can only support text streaming, several users found out that it is not suitable to be used to surf the internet. Despite of that, WAC can be utilized to retrieve information within a local area network, especially for corporate mobile

users, where they will be able to use WAC to view the documents inside there company file server by using their mobile devices when they are on the run.

The feed back also stated that the other disadvantage of WAC is that it needs to be installed manually on the server, where everything can become troublesome and error prone. They request that in the future, there will be a way to install WAC automatically without the need to perform manual installation.

7.4. System Strengths

Cross-platform viewing - WAC provides the ability to view the same internet content in different devices. It helps internet content developer especially in constructing different internet content that suit different devices, where developer does not need to learn specific internet content development language or technology.

Web-based interface – WAC provides a simple user interface that suits different types of mobile devices' format. The interface serves as a portal for mobile users to access WAC. As long as there is network connection, mobile users will be able to access WAC from anywhere.

Display optimization – WAC is able to divide the input into several segments. Each segment will become an individual content. It enables small screen devices such as web enabled phone to view the document without the need to scroll too much.

7.5. System Constraints

Variety internet standard – Although W3C has defined XHTML as the standard in writing internet content, there are still many people who do not use this standard, where they still kept with HTML standard. There is no precision in writing internet content with HTML, where developer can use either capital letter or small letter throughout the whole content. Such problem can effect the internal processing of WAC, where it is very difficult to identify each element inside the content.

Multimedia content – Currently, WAC can only support text-based internet content. It cannot supports multimedia content, such as sound and video images. It can only support image file to a certain extend and it only applicable for Pocket PC PDA. It will only provide text support for WAP devices.

Dynamic content – Dynamic contents are written using different languages such as Javascript. WAC does not support dynamic contents because it does not have the ability to compile or to decode Javascript.

HTML elements – There are several HTML elements that cannot be supported by WAC. Such elements are Form, Anchor, Frame and elements that defined styles and format of templates, such as Style, Span, Div and Link. It can only support basic text formatting elements, such as Heading, Break and Paragraph.

7.6. Future Enhancements

7.6.1. Support form query and hyperlink

One of the enhancements that need to be done is to support querying and redirection. There are many internet contents such as Yahoo and Google that support query. If query can be supported, search engine such as Yahoo can be viewed on

mobile devices such as WAP phone, where the user will be able to search the web from there.

7.6.2. Text editing

WAC can provide a user interface that allows administration level users to generate different format content from a single text format file. This enhancement helps the user to create different format content without the need to learn the knowledge in constructing each type of content.

There are several things that need to be considered in implementing this enhancement. First, the format of the text file needs to be specified; if there is no format within the text file, it will be very hard for WAC to identify the internal elements. Second, the increasing text file might overload the server storage and workload; WAC users need to make sure there is enough space to store the increasing text file. They can either delete the unused file or use a file server to store the files. Thirdly, since the input will be a text format file, WAC needs to have security features in avoiding these files being altered by unauthorized users. One of the ideas is to have a database that contains record of the user that has permission in modifying the content.

7.7. Knowledge and Experience Gained

There are a lots of experience gained during the development of WAC. First, the development process it selves is a very good experience, where I could expose to how a system is develop, what are the steps and procedures involved.

The programming skills that I have had been improving during the implementation. Although I have learnt the skills from the courses that I have attended, it is very different when trying to implement it inside an application. There are several new things that I have learnt, such as ASP and COM. I learnt both of it during the industrial attachment program. There are difficulties when I try to implement it within WAC but with trial and error, I managed to come out with a functional application.

The other important aspect that I have learnt is to come out with a plan. Planning is very important as it helps to recognize the stages involved for the development. A proper planning will aid developers to progress in each stage, make sure the objective of each stage is fulfilled, and finally reach the overall objective and able to finish the work within the planned period.

7.8. Summary

System evaluation is a process of measuring the achievement of the developed system, whether it meets the predefined specifications and criteria.

There are several problems encountered during the development, which can be categorized into Developer's Skill, Development Resources and Raw Materials. Developer's Skill is about the programming skill and experience. Development Resources is about the development infrastructure, such as the development tools and testing equipments. Raw Materials are about the input that will be used to test WAC, such as the internet content.

The feed back from the end user after their evaluation stated that WAC is a very innovative system. Since it does not support multimedia contents, it is not

suitable to use for surfing the internet, but it has the potential to be used for corporate purposes.

The strengths of WAC are allowing cross-platform viewing, web-based interface for easy access and display optimization for different mobile devices. The constraints of WAC are unable to support multimedia contents, dynamic content, and some HTML elements such as Anchor, Frame, Form and other style formatting related elements.

Future enhancements of WAC are allowing text editing and support for form query and redirection. Text editing will allow user to generate different format content from the same text file. While support for form query and redirection will enable search engine such as Google to be operate on mobile devices.

The experiences gained during the development of WAC are programming skill, proper planning and the expose to the development process.

7.9. Conclusion

WAC is an application that enable different types of mobile devices to access the WWW same WWW content. It is able to convert the format of the WWW content into the format supported by the mobile devices. For example, if the user is a WAP device, WAC will generate WML content from the original WWW content, without modifying the original files.

The objectives of WAC are information sharing among different wireless devices through the utilization of wireless network, providing a cross platform to allow different wireless devices to view the same internet content and to optimize the generated output to fit the screen resolution of each different wireless device. These objectives have been fulfilled after the system being tested.

Although WAC cannot support multimedia content at the moment, it is very useful in distributing word document through the utilization of wireless network. Since it is a web-based application, user can access their WWW documents using their mobile devices, as long as there is network connectivity. It can help the corporate users especially, where they are always on the move.

There are several enhancements for WAC. From these enhancements, two had been mentioned as both have the priority compare to the others. Such enhancements are document editing, perform form query and redirection. Other enhancements would be like enable support for dynamic content, but due to the difficulty level and the limited resources, it will take time to solve.

There is a potential for WAC in the future, as technologies are advancing, mobile devices are becoming even more sophisticated. Such improvement will benefit WAC, as multimedia content can be pushed into the devices, and helps to reduce unnecessary internal processing.

Although there are new scripting standard coming out, such as XML that support multiple types of format, the advantage of WAC is that it can produce many format of content, such as WML and HTML 3.0 (Pocket PC), from a single HTML file. It helps to reduce the need to learn new technique in reconstructing the content to fit different mobile devices format as the learning process consumes time.

Finally, the vision of WAC is that it will be able to serve as a starting platform for research and development purposes in this field. It is not a new technology for there are many companies are venturing in this field, such as AvantGo and Handisplay. Hope that in the future, there will be a variety of such application available.

Appendix A: Active Server Pages (ASP)

Microsoft Active Server Pages (ASP) is a server-side scripting environment that enables developer to create and run dynamic, interactive web server applications. With ASP, developer is able to combine HTML pages, script commands, and COM components to create interactive web pages or powerful web-based applications, which are easy to develop and modify.

The server-side scripts written in ASP are an easy way to begin creating complex, real-world web applications. It provides a compelling solution in storing HTML form information into a database, personalize web sites according to visitor preferences or use different HTML features based on the browser. For example, previously to process user input on the web server, one will have to learn language such as Perl or C to build a conventional Common Gateway Interface (CGI) application. However, ASP is able to collect HTML form information and pass it to a database using simple server-side scripts embedded directly inside the HTML document. If a person is already familiar with scripting languages such as Microsoft VBScript or Microsoft JScript (Jscript is the Microsoft implementation of the ECMA 262 language specification), he or she will have only little trouble in learning ASP.

Since ASP is designed to be language-neutral, a person who is skilled at a scripting language such as VBScript, JScript, or PERL, he or she is already know how to use Active Server Pages. Furthermore, ASP pages allow developers to use any scripting language for which a COM compliant scripting engine had been installed. ASP comes with VBScript and JScript scripting engines, but developers can also install scripting engines for PERL, REXX, and Python, which are available through third-party vendors.

ASP is a flexible way for back-end web applications programmer in creating web applications. Besides adding scripts to create an engaging HTML interface for the application, it allows developer to include the COM components as well. The logical operations behind the application can be encapsulated into a reusable module that can be called from ASP scripts, from another component or from another program.

If you develop back-end Web applications in a programming language, such as Visual Basic, C++, or Java, you will find ASP a flexible way to quickly create Web applications. Besides adding scripts to create an engaging HTML interface for your application, you can build your own COM components. You can encapsulate your application's business logic into reusable modules that you can call from a script, from another component, or from another program.

A server-side script begins to run when a browser requests an .asp file from the web server. The web server then calls ASP, which processes the requested file from top to bottom, executes any script commands, and sends a web page to the browser. Because the scripts run on the server rather than on the client, the web server does all the work involved in generating the HTML pages sent to browsers. Server-side scripts cannot be readily copied because only the result of the script is returned to the browser. Users cannot view the script commands that created the page they are viewing.

Appendix B: Component Object Model (COM)

Defining COM

COM is a platform–independent, distributed, object–oriented system for creating binary software components that can interact. COM objects can be created with a variety of programming languages, such as C++. Object–oriented languages, such as C++, provide programming mechanisms that simplify the implementation of a COM object. These objects can be within a single process, in other processes, or even on remote machines.

To understand COM—and therefore all COM–based technologies—it is crucial to bear in mind that it is not an object–oriented language, but a standard. Nor does COM specify how an application should be structured. Language, structure and implementation details are left to the software developer.

COM does specify an object model and programming requirements that enable COM objects—also called COM components, or sometimes simply object—to interact with other objects. They can have been written in other languages and may be structurally quite dissimilar. That is why COM is referred to as a binary standard—it is a standard that applies after a program has been translated to binary machine code.

COM Language Requirements

The only language requirement for COM is that code is generated in a language that can create structures of pointers, either explicitly or implicitly, call functions through pointers. Object–oriented languages such as C++ and Smalltalk® provide programming mechanisms that simplify the implementation of COM objects.

Languages such as C, Pascal, Ada, Java, and even BASIC programming environments can create and use COM objects.

COM Objects

COM defines the essential nature of a COM object. Generally, a software object is made up of a set of data and the functions that manipulate the data. A COM object is one in which access to an object's data is achieved exclusively through one or more sets of related functions. These function sets are called *interfaces*, and the functions of an interface are called *methods*. Further, COM requires that the only way to gain access to the methods of an interface be through a pointer to the interface.

Besides specifying the basic binary object standard, COM defines certain basic interfaces that provide functions common to all COM–based technologies. It also provides a small number of API functions that all components require. COM has now expanded its scope to define how objects work together over a distributed environment, such as the digital car environment, and added security features to ensure system and component integrity.

Additional information about COM can be found on the Microsoft Developer web site (http://msdn.microsoft.com).

Appendix C: Internet Information Server (IIS)

IIS is a group of internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows 2000 Server operating systems. With IIS, Microsoft includes a set of programs for building and administering web sites, a search engine, and support for writing web-based applications that access databases. Microsoft points out that IIS is tightly integrated with the Windows NT and 2000 Servers in a number of ways, resulting in faster Web page serving.

A typical company that buys IIS can create pages for Web sites using Microsoft's Front Page product (with its WYSIWYG user interface). Web developers can use Microsoft's Active Server Page (ASP) technology, which means that applications - including ActiveX controls - can be imbedded in Web pages that modify the content sent back to users. Developers can also write programs that filter requests and get the correct Web pages for different users by using Microsoft's Internet Server Application Program Interface (ISAPI) interface. ASPs and ISAPI programs run more efficiently than common gateway interface (CGI) and server-side include (SSI) programs, two current technologies (However, there are comparable interfaces on other platforms).

Microsoft includes special capabilities for server administrators designed to appeal to Internet service providers (ISPs). It includes a single window (or "console") from which all services and users can be administered. It's designed to be easy to add components as snap-ins that you didn't initially install. The administrative windows can be customized for access by individual customers.

Features

Internet Information Services 5.0 has many new features to help Web administrators to create scalable, flexible Web applications.

- **Security**
- **Administration**
- **Programmability**
- **Internet Standards**

Security

- **Digest Authentication:** Digest authentication allows secure and robust authentication of users across proxy servers and firewalls. In addition, Anonymous, HTTP Basic, and integrated Windows authentication (formerly known as Windows NT Challenge/Response authentication and NTLM authentication) are still available.
- **Secure Communications:** Secure Sockets Layer (SSL) 3.0 and Transport Layer Security (TLS) provide a secure way to exchange information between clients and servers. In addition, SSL 3.0 and TLS provide a way for the server to verify who the client is *before* the user logs on to the server. In IIS 5.0, client certificates are exposed to both ISAPI and Active Server Pages, so that programmers can track users through their sites. Also, IIS 5.0 can map the client certificate to a Windows user account, so that administrators can control access to system resources based on the client certificate.
- **Server-Gated Cryptography:** Server-Gated Cryptography (SGC) is an extension of SSL that allows financial institutions with export versions of IIS to use strong 128-bit encryption. Although SGC capabilities are built into IIS 5.0, a special SGC certificate is required to use SGC.

- **Security Wizards:** Security wizards simplify server administration tasks.
    - The Web Server Certificate Wizard simplifies certificate administration tasks, such as creating certificate requests and managing the certificate life cycle.
    - The Permissions Wizard makes it easy to configure Web site access by assigning access policies to virtual directories and files. The Permissions Wizard can also update NTFS file permissions to reflect these Web access policies.
    - The CTL wizard helps you configure your certificate trust lists (CTLs). A CTL is a list of trusted certification authorities (CAs) for a particular directory. CTLs are especially useful for Internet service providers (ISPs) who have several Web sites on their server and who need to have a different list of approved certification authorities for each site.
- **IP and Internet Domain Restrictions:** You can grant or deny Web access to individual computers, groups of computers, or entire domains.
- **Kerberos v5 Authentication Protocol Compliance:** IIS is fully integrated with the Kerberos v5 authentication protocol implemented in Microsoft Windows 2000, allowing you to pass authentication credentials among connected computers running Windows.
- **Certificate Storage:** IIS certificate storage is now integrated with the Windows CryptoAPI storage. The Windows Certificate Manager provides a single point of entry that allows you to store, back up, and configure server certificates.

- **Fortezza:** The U.S. government security standard, commonly called Fortezza, is supported in IIS 5.0. This standard satisfies the Defense Message System security architecture with a cryptographic mechanism that provides message confidentiality, integrity, authentication, and access control to messages, components, and systems. These features can be implemented both with server and browser software and with PCMCIA card hardware.

Administration

- **Restarting IIS:** Now you can restart your Internet services without having to reboot your computer.
- **Backing Up and Restoring IIS:** You can back up and save your metabase settings to make it easy to return to a safe, known state.
- **Process Accounting:** Provides information about how individual Web sites use CPU resources on the server. This information is useful in determining which sites are using disproportional high CPU resources or which might have malfunctioning scripts or CGI processes.
- **Process Throttling:** You can limit the percentage of time the CPU spends processing out-of-process ASP, ISAPI, and CGI applications for individual Web sites. In addition, misbehaving processes can be stopped and restarted.
- **Improved Custom Error Messages:** Now administrators can send informative messages to clients when HTTP errors occur on their Web sites. Also includes detailed ASP error processing capabilities through the use of the 500-100.asp custom error message. You can use the custom errors that IIS 5.0 provides, or create your own.
- **Configuration Options:** You can set permissions for Read, Write, Execute, Script, and FrontPage Web operations at the site, directory, or file level.

- **Remote Administration:** IIS 5.0 has Web-based administration tools that allow remote management of your server from almost any browser on any platform. With IIS 5.0, you can set up administration accounts called Operators with limited administration privileges on Web sites, to help distribute administrative tasks.
- **Terminal Services**: Terminal Services is a feature of Windows 2000 that allows you to run 32-bit Windows applications on terminals and terminal emulators running on personal computers and other computer desktops. Terminal Services allows virtually any desktop to run applications on the server. This enables you to remotely administer Windows 2000 services such as IIS as if you were at the server console, including administration from older legacy PCs, or even non-PC devices such as UNIX workstations with compatible client software. (Non-Windows-based client devices require third-party add-on software.)
- **Centralized Administration:** Administration tools for IIS use the Microsoft® Management Console (MMC). MMC hosts the programs, called snap-ins that administrators use to manage their servers. You can use IIS snap-in from a computer running Windows 2000 Professional to administer a computer on your intranet running Internet Information Services on Windows 2000 Server.

Programmability

- **Active Server Pages:** You can create dynamic content by using server-side scripting and components to create browser-independent dynamic content. Active Server Pages (ASP) provides an easy-to-use alternative to CGI and ISAPI by allowing content developers to embed any scripting language or

server component into their HTML pages. ASP provides access to all of the HTTP request and response streams, as well as standards-based database connectivity and the ability to customize content for different browsers.

- **New ASP Features:** Active Server Pages has some new and improved features for enhancing performance and streamlining your server–side scripts.
- **Application Protection:** IIS 5.0 offers greater protection and increased reliability for your Web applications. By default, IIS will run all of your applications in a common or *pooled* process that is separate from core IIS processes. In addition, you can still *isolate* mission-critical applications that should be run outside of both core IIS and pooled processes.
- **ADSI 2.0:** In IIS 5.0, administrators and application developers will have the ability to add custom objects, properties, and methods to the existing ADSI provider, giving administrators even more flexibility in configuring their sites.

Internet Standards

- **Standards Based:** Microsoft Internet Information Services 5.0 complies with the HTTP 1.1 standard, including features such as PUT and DELETE, the ability to customize HTTP error messages, and support for custom HTTP headers.
- **Multiple Sites, One IP Address:** With support for host headers, you can host multiple Web sites on a single computer running Microsoft Windows 2000 Server with only one IP address. This is useful for Internet service providers and corporate intranets hosting multiple sites.

- **Web Distributed Authoring and Versioning (WebDAV):** Enables remote authors to create, move, or delete files, file properties, directories, and directory properties on your server over an HTTP connection.
- **News and Mail:** You can use SMTP and NNTP Services to set up intranet mail and news services that work in conjunction with IIS.
- **PICS Ratings:** You can apply Platform for Internet Content Selection (PICS) ratings to sites that contain content for mature audiences.
- **FTP Restart:** Now File Transfer Protocol file downloads can be resumed without having to download the entire file over again if an interruption occurs during data transfer.
- **HTTP Compression:** Provides faster transmission of pages between the Web server and compression-enabled clients. Compresses and caches static files, and performs on-demand compression of dynamically generated files.

Appendix I: Coding – ASP

1. File: index.asp

```
<%
dim strHttpUserAgent
dim nDevType '1 = html 4.0; 2 = html 3.0; 3 = wap;
strHttpUserAgent = Request.ServerVariables("HTTP_USER_AGENT")
if InStr(strHttpUserAgent, "Mozilla") > 0 then
        Response.Redirect "Redirection.asp"
else
        Response.Redirect "wap.asp"
nDevType = 3
end if
%>
```

2. File: html.asp

```
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipntg/THEME.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipntg/GRAPH0.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipntg/COLOR0.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipntg/CUSTOM.CSS" VI6.0THEME="Sumi Painting">
</head>
<body rightmargin=0 topmargin=0 leftmargin=0 marginwilth="0" marginheight="0">
<table border=0 width = "103%" height="100%" bgcolor=deepskyblue cellspacing=2 cellpadding=4>
<tr>
<td width=187>
<IMG style="WIDTH: 200px; HEIGHT: 86px" height=10src=images/logo.jpg width=187 >
</td>
<td align=left>
<form action="script/Interface.asp" method="get" target="_parent">
<input type="hidden" name="header" value="http://">
<input type="text" name="url">
<br><br>
<input type="reset" value="Clear">
<input type="submit" value="Go ">
</form>
</td>
</tr>
</table>
</body>
</html>
```

3. File: html\index.asp

```
<!--
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<link REL="stylesheet" TYPE="text/css" HREF="../_Themes/sumipntg/THEME.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="../_Themes/sumipntg/GRAPH0.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="../_Themes/sumipntg/COLOR0.CSS" VI6.0THEME="Sumi Painting">
<link REL="stylesheet" TYPE="text/css" HREF="../_Themes/sumipntg/CUSTOM.CSS" VI6.0THEME="Sumi Painting">
</head>
-->
<%
dim strBodyFrame
strBodyFrame = "../script/body.asp?i=" & Request.QueryString("i")
%>
<script>
</script>
<frameset rows="30%, *" cols="*" scroll="NO">
<frame name="upper" src=../html.asp noresize scrolling="NO">
<frame name="body" src="<%=strBodyFrame%>">
</frameset>
<!--
</html>
```

```
-->
```

4. File: wap\index.asp

```
<!--#include file="../script/setting.asp"-->
<!--#include file="../script/ProSupport.asp"-->
<%
Response.ContentType = "text/vnd.wap.wml"
%>
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta http-equiv="Cache-Control" content="max-age=0"/>
</head>
<%
dim strID
dim strFullOutPath
dim strID
strID = Request.QueryString("i")
if len(strID) <> 0 then
strFullOutPath = strOutPath & strID & "/i.htm"
dim objFso, objOut
set objFso = server.CreateObject("Scripting.FileSystemObject")
if objFso.FileExists(strFullOutPath) then
call DisplayPage(3, strOutPath, strID)
else
Response.Write "<card id=""main"" label=""Menu"">" & vbCrLf
Response.Write "<p>There is no menu</p>" & vrCrLf
Response.Write "</card>" & vbCrLf
end if
end if
%>
</wml>
```

5. File: script\Display.asp

```
<!--#include file="setting.asp"-->
<html>
<head>
<title>WAC-FlexDisplay</title>
</head>
<body>
<%
dim strID
dim strFile
dim strLocation
dim strIndex
dim nPrev
dim nNext
dim nTotal
strID = Request.QueryString("i")
strFile = Request.QueryString("f")
strIndex = Request.QueryString("in")
nPrev = Request.QueryString("prev")
nNext = Request.QueryString("next")
nTotal = Request.QueryString("total")
if len(strID) > 0 and len(strID) > 0 then
strLocation = strOutPath & strID & "\" & strFile
dim objFso, objSrc
set objFso = server.CreateObject("Scripting.FileSystemObject")
set objSrc = objFso.OpenTextFile(strLocation, 1, false -2)
do while not objSrc.AtEndOfStream
Response.Write objSrc.ReadLine() & vbNewLine
loop
objSrc.close()
set objFso = nothing
Response.Write "<br><hr><br>"
if strIndex = "first" then
Response.Write "<a href=""display.asp?i=" & strID
Response.Write "&f=" & nNext & "i.htm"
Response.Write "&in=mid"
```

```
Response.Write "&prev=1"
Response.Write "&next=3"
Response.Write "&total=" & nTotal
Response.Write """>Next</a>"
elseif strIndex = "last" then
Response.Write "<a href=""display.asp?i=" & strID
Response.Write "&f=" & nPrev & "i.htm"
Response.Write "&in=mid"
Response.Write "&prev=" & nPrev-1
Response.Write "&next=" & nTotal
Response.Write "&total=" & nTotal
Response.Write """>Previous</a>"
elseif strIndex = "mid" then
Response.Write "<a href=""display.asp?i=" & strID
Response.Write "&f=" & nPrev & "i.htm"
if nPrev = 1 then
Response.Write "&in=first"
Response.Write "&next=" & nPrev+1
Response.Write "&total=" & nTotal
else
Response.Write "&in=mid"
Response.Write "&prev=" & nPrev-1
Response.Write "&next=" & nNext-1
Response.Write "&total=" & nTotal
end if
Response.Write """>Prev</a>   "
Response.Write "<a href=""display.asp?i=" & strID
Response.Write "&f=" & nNext & "i.htm"
if nNext = nTotal then
Response.Write "&in=last"
Response.Write "&prev=" & nPrev+1
Response.Write "&total=" & nTotal
else
Response.Write "&in=mid"
Response.Write "&prev=" & nPrev+1
Response.Write "&next=" & nNext+1
Response.Write "&total=" & nTotal
end if
Response.Write """>Next</a>"
end if
else
Response.Write "Oops, no data available"
end if
%>
</body>
</html>
```

6. File: script\body.asp

```
<!--#include file="setting.asp"-->
<!--#include file="ProSupport.asp"-->
<%
dim str_ID
dim strScript
strScript = "Display.asp"
str_ID = Request.QueryString("i")
if not str_ID = "" then
call DisplayPage(1, strOutPath, str_ID, strScript)
else
%>
<html>
<head>
<title>WAC</title>
</head>
<body>
<p>There is no display</p>
</body>
</html>
<%
end if
%>
```

7. File: script\Interface.asp

```
<!--#include file="setting.asp"-->
<%
dim strUserAgent
dim strUrl
dim strID
dim strUrlHeader
dim nDevType
dim nVersion
dim nVerPos
strUserAgent = Request.ServerVariables("HTTP_USER_AGENT")
if InStr( strUserAgent, "Mozilla") <> -1 then
nVerPos = InStr( strUserAgent, "/")
if nVerPos > 0 then
nVersion = mid( strUserAgent, nVerPos+1, 1)
if nVersion >= 4 then 'IE or compatible
nDevType = 1
elseif nVersion < 4 then 'Pocket IE or compatible
nDevType = 2
end if
end if
else
nDevType = 3 'Wap devices
end if
strID = ""
if nDevType = 1 or nDevType = 2 then
strUrl = Request.QueryString("Url")
strUrlHeader = Request.QueryString("header")
elseif nDevType = 3 then
strUrl = Request.QueryString("url")
end if
%>
<!--#include file="Process.asp"-->
```

8. File: Mobile.asp

```
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipnt/THEME.CSS" VI6.0THEME="Sumi Painting"><!--
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipnt/GRAPH0.CSS" VI6.0THEME="Sumi Painting">-->
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipnt/COLOR0.CSS" VI6.0THEME="Sumi Painting"><!--
<link REL="stylesheet" TYPE="text/css" HREF="_Themes/sumipnt/CUSTOM.CSS" VI6.0THEME="Sumi Painting">-->
</head>
<body>
<table border=0 width="100%" height="100%" bgcolor=#0098c6 cellspacing=2 cellpadding=4>
<tr>
<td style="color:white;font-style:italic;" valign="center" align=center><b>ENTER URL</b></td>
</tr>
<tr>
<td valign=center align=center>
<form action="script/Interface.asp" method="get">
<input type="hidden" name="header" value="http://">
<INPUT type="text" name="url"><br><br>
<input type="reset" value="clear">   
<INPUT type="submit" value="Go">
</form>
</td>
</tr>
</table>
</body>
</html>
```

9. File: script\Process.asp

```
<!--#include file = "ProSupport.asp" -->
<%
dim strDomainUrl
dim strFullOutPath
dim objFso, objFolder
set objFso = server.CreateObject("Scripting.FileSystemObject")
```

```
if InStr(strUrl, "http://") = 0 then
strDomainUrl = strUrlHeader & strUrl
elseif InStr(strUrl, "http://") > 0 then
strDomainUrl = strUrl
end if
if len(strUrl) > 0 then
if len(strID) > 0 then
strFullOutPath = strOutPath & strID
else
if strID = "" then
strID = GetID()
end if
strFullOutPath = strOutPath & strID
set objFolder = objFso.CreateFolder(strFullOutPath)
end if
if objFso.FolderExists(strFullOutPath) then
if CreateHTML( strDomainUrl, strFullOutPath, strOutPath,strSrcFile, strID) = true then
if nDevType = 1 then
Response.Redirect "../html/index.asp?i=" & strID
elseif nDevType = 2 then
Response.Redirect "body.asp?i=" & strID
else'if nDevType = 3 then
Response.Redirect "WapBody.asp?i=" & strID
end if
end if
end if
set objFso = nothing
'call the error page
end if
%>
```

10. File: script\ProSupport.asp

```
<%
function GetID()
GetID = "" ' set default return value to nothing.
Dim objUID
Dim sID
Set objUID = Server.CreateObject("genuid.GenUIDObj")
sID = objUID.UID
Set objUID = nothing
GetID = CStr(sID) ' setting the return value to new ID.
end function
'*****************************************************************
function GetTotal(strSrcLocation)
dim objFso, objTotal
dim nTotal
dim strTemp
nTotal = 0
strTemp = ""
set objFso = server.CreateObject("Scripting.FileSystemObject")
set objTotal = objFso.OpenTextFile(strSrcLocation, 1, false, -2)
' Response.Write "GEt total"
do while not objTotal.AtEndOfStream
strTemp = strTemp + objTotal.ReadLine()
nTotal = nTotal + 1
loop
GetTotal = nTotal
strTemp = ""
' Response.Write nTotal
' Response.End()
objTotal.close()
set objFso = nothing
end function
'*****************************************************************
function DisplayPage(nDevType, strOutPath, strID, strScript)
dim objFso, objOut
dim strSrcLocation
dim nPos
dim nTotal
nPos = 1
nTotal = 0
strSrcLocation = strOutPath & strID & "\i.txt"
nTotal = GetTotal(strSrcLocation)
set objFso = server.CreateObject("Scripting.FileSystemObject")
```

```
' set objTotal = objFso.OpenTextFile(strSrcLocation, 1, false, -2)
set objOut = objFso.OpenTextFile(strSrcLocation, 1, false, -2)
if nDevType = 2 or nDevType = 1 then 'html
Response.Write "<html>" & vbNewLine
Response.Write "<head>" & vbNewLine
Response.Write "<title>WAC-Mobile View</title>" & vbNewLine
Response.Write "</head>" & vbNewLine
Response.Write "<body>" & vbNewLine
' do while not objTotal.AtEndOfStream
' nTotal = nTotal + 1
' loop
' objTotal.close()
' Response.write nTotal
' Response.End()
do while not objOut.AtEndOfStream
Response.write "<a href="""
Response.write strScript & "?i=" & strID
Response.write "&f="
Response.write objOut.readLine()
if nPos = nTotal then
Response.Write "&in=last&prev=" & (nPos-1)
Response.Write "&total=" & nTotal
elseif nPos = 1 then
Response.Write "&in=first&next=" & (nPos+1)
Response.Write "&total=" & nTotal
else
Response.Write "&in=mid&prev=" & (nPos-1)
Response.Write "&next=" & (nPos+1)
Response.Write "&total=" & nTotal
end if
Response.write """>"
Response.write "Page " & nPos
Response.write "</a><br>" & vbNewLine
nPos = nPos + 1
loop
'process the file
Response.Write "</body>" & vbNewLine
Response.Write "</html>"
elseif nDevType = 3 then
dim strWmlOut
strWmlOut = ""
dim nWapPos
nWapPos = 1
do while not objOut.AtEndOfStream
strWmlOut = "<a href="""
strWmlOut = strWmlOut & strScript & "?i=" & strID & "&amp;f="
strWmlOut = strWmlOut & objOut.readLine()
if nWapPos = nTotal then
strWmlOut = strWmlOut & "&amp;in=last&amp;prev=" & (nWapPos-1)
strWmlOut = strWmlOut & "&amp;total=" & nTotal
elseif nWapPos = 1 then
strWmlOut = strWmlOut & "&amp;in=first&amp;next=" & (nWapPos+1)
strWmlOut = strWmlOut & "&amp;total=" & nTotal
else
strWmlOut = strWmlOut & "&amp;in=mid&amp;prev=" & (nWapPos-1)
strWmlOut = strWmlOut & "&amp;next=" & (nWapPos+1)
strWmlOut = strWmlOut & "&amp;total=" & nTotal
end if
strWmlOut = strWmlOut & """>"
strWmlOut = strWmlOut & "Page " & nWapPos
strWmlOut = strWmlOut & "</a><br/>" & vbNewLine
Response.Write strWmlOut
nWapPos = nWapPos + 1
loop
end if
objOut.close()
set objFso = nothing
end function
'*************************************************************************
function CreateHTML( strDomainUrl, strFullOutPath, strOutPath, strSrcFile, strID)
CreateHTML = true
'First, get the content from the URL.
'if RequestPage(strUrl, strOutFile, nOutType, forminfo,formmethod, strID) = true then
dim strSrcLocation
strSrcLocation = strOutPath & strID & "\" & strSrcFile
if RequestPage(strDomainUrl, strSrcLocation, strID) = true then
```

```
'then, process the page.
' if ProcessFilter(strSrcLocation, strID, strSrcFile) = true then
dim strAgent
strAgent = request.ServerVariables("HTTP_USER_AGENT")
dim objWac
set objWac = server.CreateObject("WAC.FlexDisplay")
if InStr(strAgent, "Mozilla") > 0 then
objWac.WmlProcess = false
else
objWac.WmlProcess = true
end if
call objWac.FlexProcDisp(strDomainUrl, strSrcFile, strOutPath, strID)
set objWac = nothing
' end if
else
CreateHTML = false
Response.Write "Cannot be processed"
exit function
end if
end function
'*************************************************************************
function RequestPage(strDomainUrl, strSrcLocation, strID)
dim objTear
dim strValue
' set objTear = server.CreateObject("Tear.HtmTear")
' RequestPage = objTear.RequestInternet(strDomainUrl, strSrcLocation)
set objTear = server.CreateObject("WACTear.WacHtmTear")
objTear.UserAgent = Request.ServerVariables("HTTP_USER_AGENT")
' objTear.FollowRedirect = false
' objTear.ContentType = request.ServerVariables("HTTP_CONTENT_TYPE")
' objTear.Accept = request.ServerVariables("HTTP_ACCEPT")
' for each item in request.ServerVariables
' if item <> "HTTP_USE_AGENT" and item <> "HTTP_CONTENT_TYPE" and item <> "HTTP_ACCEPT" then
' strValue = request.ServerVariables(item)
' objTear.AddHeader item, strValue
' ' end if
' next
RequestPage = objTear.GetPage(strDomainUrl,2,"",strSrcLocation,"","")
set objTear = nothing
end function
'*************************************************************************************
%>
```

11. File: Redirection.asp

```
<%
dim strUserAgent
dim nVerPos
dim nVersion
strUserAgent = Request.ServerVariables("HTTP_USER_AGENT")
nVerPos = InStr(strUserAgent, "/")
nVersion = mid(strUserAgent,nVerPos+1, 1)
if nVersion >= 4 then
Response.Redirect "../wac/html/index.asp"
elseif nVersion < 4 then
Response.Redirect "mobile.asp"
end if
%>
```

D:\WAC\WAC_Local\script\setting.asp 1

```
<%
' variables defined for web url
' this include directory
strOutDir = "WACluster"
strSrcFile = "src.txt"
strOutFile = "display.htm"
'*******************************
strOutPath = "C:\" & strOutDir & "\"
dim objSet, objFoldSet
set objSet = server.CreateObject("Scripting.FileSystemObject")
if objSet.FolderExists(strOutPath) = false then
set objFoldSet = objSet.CreateFolder(strOutPath)
end if
set objSet = nothing
```

```
%>
```

12. File: wap.asp

```
<%
Response.ContentType = "text/vnd.wap.wml"
%>
<%
dim strLocation
dim strScriptFile
dim strServerName
strServerName = Request.ServerVariables("SERVER_NAME")
strScriptFile = "Interface.asp"
strLocation = "http://" & strServerName & "/wac/script/" & strScriptFile
%>
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta http-equiv="Cache-Control" content="max-age=0"/>
</head>
<card id="main" title="WAC">
<p align="center">
Welcome to WAC
</p>
<do type="accept" label="Search">
<go href="#search"/>
</do>
</card>
<card id="search" title="WAC">
<p align="center">Enter Url<br/>
<input name="url" value="http://" type="text"/>
</p>
<do type="accept" label="Go">
<go href="<%=strLocation%>" method="get">
<postfield name="url" value="$(url)"/>
</go>
</do>
</card>
</wml>
```

13. File: script\WapBody.asp

```
<!-- #include file="ProSupport.asp"-->
<!-- #include file="setting.asp"-->
<%
Response.ContentType = "text/vnd.wap.wml"
'Response.AddHeader "Pragma", "no-cache"
'Response.AddHeader "Cache-Control", "no-cache, must-revalidate"
%>
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="main" title="WAC">
<p>
<%
dim str_ID
dim strScript
strScript = "WapProcess.asp"
str_ID = Request.QueryString("i")
if not str_ID = "" then
call DisplayPage(3, strOutPath, str_ID, strScript)
else
Response.Write "No File exists"
end if
%>
</p>
<do type="option" label="Back">
<prev/>
</do>
</card>
</wml>
```

14. File: script\WapProcess.asp

```
<!--#include file="setting.asp"-->
<%
Response.ContentType = "text/vnd.wap.wml"
%>
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="Page" title="WAC">
<p>
<%
dim strID
dim strFile
dim strLocation
dim strIndex
dim nPrev
dim nNext
dim nTotal
strID = Request.QueryString("i")
strFile = Request.QueryString("f")
strIndex = Request.QueryString("in")
nPrev = Request.QueryString("prev")
nNext = Request.QueryString("next")
nTotal = Request.QueryString("total")
'Response.Write strID & "<br/>"
'Response.Write strFile
if len(strID) > 0 and len(strID) > 0 then
strLocation = strOutPath & strID & "\" & strFile
' Response.Write "here"
dim objFso, objSrc
set objFso = server.CreateObject("Scripting.FileSystemObject")
set objSrc = objFso.OpenTextFile(strLocation, 1, false -2)
do while not objSrc.AtEndOfStream
Response.Write objSrc.ReadLine() & vbNewLine
loop
objSrc.close()
set objFso = nothing
if strIndex = "first" then
Response.Write "<a href=""WapProcess.asp?i=" & strID
Response.Write "&amp;f=" & nNext & "i.htm"
Response.Write "&amp;in=mid"
Response.Write "&amp;prev=1"
Response.Write "&amp;next=3"
Response.Write "&amp;total=" & nTotal
Response.Write """>Next</a>"
elseif strIndex = "last" then
Response.Write "<a href=""WapProcess.asp?i=" & strID
Response.Write "&amp;f=" & nPrev & "i.htm"
Response.Write "&amp;in=mid"
Response.Write "&amp;prev=" & nPrev-1
Response.Write "&amp;next=" & nTotal
Response.Write "&amp;total=" & nTotal
Response.Write """>Previous</a>"
elseif strIndex = "mid" then
Response.Write "<a href=""WapProcess.asp?i=" & strID
Response.Write "&amp;f=" & nPrev & "i.htm"
if nPrev = 1 then
Response.Write "&amp;in=first"
Response.Write "&amp;next=" & nPrev+1
Response.Write "&amp;total=" & nTotal
else
Response.Write "&amp;in=mid"
Response.Write "&amp;prev=" & nPrev-1
Response.Write "&amp;next=" & nNext-1
Response.Write "&amp;total=" & nTotal
end if
Response.Write """>Prev</a>"
Response.Write "<a href=""WapProcess.asp?i=" & strID
Response.Write "&amp;f=" & nNext & "i.htm"
if nNext = nTotal then
Response.Write "&amp;in=last"
Response.Write "&amp;prev=" & nPrev+1
Response.Write "&amp;total=" & nTotal
else
```

```
Response.Write "&amp;amp;in=mid"
Response.Write "&amp;amp;prev=" & nPrev+1
Response.Write "&amp;amp;next=" & nNext+1
Response.Write "&amp;amp;total=" & nTotal
end if
Response.Write """>Next</a>"
end if
else
Response.Write "Oops, no data available"
end if
%>
</p>
<do type="option" label="Back">
<prev/>
</do>
</card>
</wml>
```

15. File: wap\index.asp

```
<!--#include file="../script/setting.asp"-->
<!--#include file="../script/ProSupport.asp"-->
<%
Response.ContentType = "text/vnd.wap.wml"
%>
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta http-equiv="Cache-Control" content="max-age=0"/>
</head>
<%
dim strID
dim strFullOutPath
dim strID
strID = Request.QueryString("i")
if len(strID) <> 0 then
strFullOutPath = strOutPath & strID & "/i.htm"
dim objFso, objOut
set objFso = server.CreateObject("Scripting.FileSystemObject")
if objFso.FileExists(strFullOutPath) then
call DisplayPage(3, strOutPath, strID)
else
Response.Write "<card id=""main"" label=""Menu"">" & vbCrLf
Response.Write "<p>There is no menu</p>" & vrCrLf
Response.Write "</card>" & vbCrLf
end if
end if
%>
</wml>
```

Appendix E: Coding – Visual C++

1. File: FlexDisplay.cpp

```
// FlexDisplay.cpp : Implementation of CFlexDisplay
#include "stdafx.h"
#include "WAC.h"
#include "FlexDisplay.h"
#include<atlconv.h>
#include<string.h>
using namespace std;
using namespace std;
////////////////////////////////////////////////////////////////////
// CFlexDisplay
STDMETHODIMP CFlexDisplay::FlexProcDisp(BSTR bstrDomainUrl, BSTR bstrSrcFile, BSTR bstrOutPath
, BSTR bstrID, BSTR *pbPageIndex)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_sDomainUrl = OLE2T(bstrDomainUrl);
m_sOutPath = OLE2T(bstrOutPath);
m_sSrcFile = OLE2T(bstrSrcFile);
m_sID = OLE2T(bstrID);
string sSrcLocation = "";
sSrcLocation = m_sOutPath + m_sID + "\\" + m_sSrcFile;
CString pszHtml;
FILE *fSrc;
fSrc = fopen( sSrcLocation.c_str(), "r" );
char ch;
while( (ch = fgetc(fSrc)) != EOF )
{
pszHtml += ch;
}
CNode *pNode = new CNode;
COutProcess *pOut = new COutProcess;
string sOut = "";
// ProcessNode( pszHtml, pNode );
// if( pNode->HasChild() )
// pNode->GetChildNode(sOut);
// pOut->ProcessOutput(pNode, m_sOutPath, m_sID); //Using linked pointer.
SpecificPush( pszHtml, pNode );
if(!m_bWap)
pOut->SpecificOutputProcess(pNode, m_sOutPath, m_sID, m_sDomainUrl); //Pushing
all to one.
else
pOut->WapOutput(pNode, m_sOutPath, m_sID, m_sDomainUrl); //Pushing
all to one (WAP);
if( pOut->CreateIndexFile(m_sOutPath, m_sID) )
{
/*
Index file being created.
*/
}
fclose(fSrc);
delete pOut;
delete pNode;
return S_OK;
}
/*************************************************************************/
1
/*************************************************************************/
CNode* CFlexDisplay::GetPreviousNode(CNode* pCurrentNode, string sTemp)
{
/*
Get Nodes: Get the previous nodes.
*/
if( sTemp != pCurrentNode->GetTagName() )
return GetPreviousNode( pCurrentNode->GetParentNode(), sTemp );
else if( sTemp == pCurrentNode->GetTagName() )
return pCurrentNode;
}
/*************************************************************************/
/*************************************************************************/
```

```
void CFlexDisplay::PushNode(CNode* pCurrentNode, CNode* pTempPushNode)
{
/*
Pushing Nodes: Push nodes into structure.
*/
pCurrentNode->AddChilde(pTempPushNode);
pCurrentNode->SetContent("CHILD");
while( pTempPushNode->HasParent() && pTempPushNode->GetParentNode() != pCurrentNode)
{
pTempPushNode = pTempPushNode->GetParentNode();
pCurrentNode->AddChilde( pTempPushNode );
pCurrentNode->SetContent("CHILD");
}
}
/*************************************************/
/*************Generic push function, not used************************/
void CFlexDisplay::ProcessNode(const char *pszHtml, CNode *pCurrentNode)
{
const char* pszProcHtml = new char;
const char* pszBackup = new char;
char ch;
string sTemp;
string sName;
string sProperty;
string sPropValue;
string sContent = "";
CNode *pTempNode = new CNode;
CNode *pTempPushNode = new CNode;
pszBackup = pszHtml;
while( (ch = *(pszHtml++)) != NULL )
{
if( ch == '<' )
{
if( (ch=*(pszHtml++)) == '/' )
{
/*
Push Node: Create the tree structure.
*/
while( (ch=*(pszHtml++)) != '>' )
sTemp += tolower(ch);
--pszHtml;
if( sTemp == pCurrentNode->GetTagName() )
{
if( pCurrentNode->HasParent() )
{
(pCurrentNode->GetParentNode())->AddChilde(pCurrentNode);
(pCurrentNode->GetParentNode())->SetContent("CHILD");
pCurrentNode = pCurrentNode->GetParentNode();
}
}
else
{
2
//Recursive process to get the parent node.
pTempPushNode = pCurrentNode;
pCurrentNode = GetPreviousNode( pCurrentNode, sTemp );
PushNode( pCurrentNode, pTempPushNode);
}
sTemp = "";
}
else if( ch == ' ' )
continue;
else
{
--pszHtml;
/*
Tag Name Processing: Get the name of current Tag.
*/
while( (ch=*(pszHtml++)) != '>' )
{
if( ch == ' ')
break;
else
sName += tolower(ch);
}
if( !sName.empty() )
```

```
{
if( !(pCurrentNode->GetTagName().empty()) )
{
pTempNode->SetTagName(sName);
pTempNode->SetParentNode(pCurrentNode);
pCurrentNode = pTempNode;
pTempNode = new CNode;
}
else
{
pCurrentNode->SetTagName(sName);
}
if( ch == ' ' )
{
/*
Property Processing: Get the properties for current tag.
*/
while( (ch = *(pszHtml++)) != '>')
{
if( ch == '=' )
{
while( (ch = *(pszHtml++)) != '>')
{
if(ch == ' ')
break;
else
sPropValue += ch;
}
pCurrentNode->SetTagProperties(sProperty, sPropValue);
sPropValue = "";
sProperty = "";
--pszHtml;
}
else if( isalpha(ch) )
sProperty += ch;
}
--pszHtml;
}
sName = "";
}
}
}
else if( ch!= '<' && ch!='>' )
3
{
/*
Content Processing: Getting the Text content.
*/
--pszHtml;
while( (ch=*(pszHtml++))!='<' )
sContent += ch;
if( !sContent.empty() )
{
pCurrentNode->SetContent(sContent);
}
--pszHtml;
sContent = "";
}
}
}
/*********************************************************************/
/***************Push function*****************************************/
void CFlexDisplay::SpecificPush(const char* pszHtml, CNode *pSpecificPush)
{
/*
html---head
| |----meta
| |----title
| |----script (optional)
| |----style (optional)
| |----link (optional)
|
|---body
| |----table
| | |-------tr
| | |---td
```

```
|||----standard element
||
||----standard element
|
|---script(optional, not included)
*/
const char* pszProcHtml = new char;
const char* pszBackup = new char;
char ch;
string sTemp;
string sName;
string sProperty;
string sPropValue;
string sContent = "";
CNode *pTempNode = new CNode;
CNode *pTempCloseNode = new CNode;
// CNode *pTempPushNode = new CNode;
pszBackup = pszHtml;
while( (ch = *(pszHtml++)) != NULL )
{
if( ch == '<' )
{
if( (ch=*(pszHtml++)) == '/' )
{
/*
Closing tag: If it's the same, with current node,
skip it; If not, push the temp node into the structure
and skip the closing tag.
*/
while( (ch=*(pszHtml++)) != '>' )
sTemp += tolower(ch);
sTemp = "/" + sTemp;
if( !sTemp.empty() )
4
{
if( pTempCloseNode->GetTagName().empty() )
{
pTempCloseNode->SetTagName(sTemp);
if( !pTempNode->GetTagName().empty() )
{
pSpecificPush->AddChilde(pTempNode);
pTempNode = new CNode;
}
pSpecificPush->AddChilde(pTempCloseNode);
pTempCloseNode = new CNode;
}
sTemp = "";
}
}
else if( ch == '!' )
{
/*
Filtering the comment tag.
*/
char chNext;
// string sEndComment = "";
if( (ch=*(pszHtml++)) == '-' )
{
if( (ch=*(pszHtml++)) == '-' )
{
do
{
// pszHtml++;
ch = *pszHtml;
chNext = *(++pszHtml);
}while( !((ch == '-') && (chNext == '>')) );
}
}
}
else
{
--pszHtml;
/*
Tag Name Processing: Get the name of current Tag.
*/
while( (ch=*(pszHtml++)) != '>' )
```

```
{
if( ch == ' ')
break;
else
sName += tolower(ch);
}
if( sName == "script" )
{
/*
Filtering the script tag.
*/
string sEnd = "";
while( (ch=*(pszHtml++)) != NULL )
{
if( ch == '<' )
{
if( (ch=*(pszHtml++)) == '/' )
{
while( (ch = *(pszHtml++)) != '>' )
sEnd += tolower(ch);
if( sEnd == "script" )
{
if( (ch = *pszHtml) == '"')
{
sEnd = "";
5
continue;
}
else
{
--pszHtml;
break;
}
}
else
{
sEnd = "";
}
}
}
}
sName = "";
}
else
{
if( !sName.empty() )
{
if( !(pTempNode->GetTagName().empty()) )
{
pSpecificPush->AddChilde(pTempNode);
pTempNode = new CNode;
pTempNode->SetTagName(sName);
}
else
{
pTempNode->SetTagName(sName);
}
if( ch == ' ' )
{
/*
Property Processing: Get the properties for current tag.
*/
while( (ch = *(pszHtml++)) != '>')
{
if( ch == '=' )
{
while( (ch = *(pszHtml++)) != '>')
{
if(ch == ' ')
break;
else
sPropValue += ch;
}
pTempNode->SetTagProperties(sProperty, sPropValue);
sPropValue = "";
sProperty = "";
```

```
--pszHtml;
}
else //if( isalpha(ch) )
sProperty += tolower(ch);
}
--pszHtml;
}
sName = "";
}
}
}
}
else if( ch == '\n' || ch == '\r')
continue;
else if( ch != '<' && ch !='>' )
6
{
/*
Content Processing: Getting the Text content.
*/
--pszHtml;
while( (ch=*(pszHtml++))!='<' )
sContent += ch;
if( !sContent.empty() )
{
pTempNode->SetContent(sContent);
}
--pszHtml;
sContent = "";
}
}
delete pTempNode;
delete pTempCloseNode;
}
/*************************************************************/
/***************To get short domain**********************/
string CFlexDisplay::GetShortDomain(string sDomainUrl)
{
string sShortDomain;
char ch;
for(int i=0;i<sDomainUrl.size();i++)
{
ch = sDomainUrl.at(i);
sShortDomain += ch;
if( i > 7 && ch == '/' )
break;
}
return sShortDomain;
}
/*************************************************************/
/*********To get long domain*******************************/
string CFlexDisplay::GetLongDomain(string sDomainUrl)
{
string sLongDomain;
int nPoint;
char ch;
for(int i=sDomainUrl.size();i>0;i--)
{
ch = sDomainUrl.at(i);
if( ch == '/' )
{
nPoint = i;
break;
}
}
for(i = 0; i< nPoint+1; i++)
{
sLongDomain += sDomainUrl.at(i);
}
return sLongDomain;
}
/*************************************************************/
STDMETHODIMP CFlexDisplay::FlexWapDisplay(BSTR bstrDomainUrl, BSTR bstrSrcFile, BSTR bstrOutPa
th, BSTR bstrID, BSTR *pbPageIndex)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
```

```
7
// TODO: Add your implementation code here
return S_OK;
}
STDMETHODIMP CFlexDisplay::get_WmlProcess(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_bWap;
return S_OK;
}
STDMETHODIMP CFlexDisplay::put_WmlProcess(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_bWap = newVal;
return S_OK;
}
```

2. File: Node.cpp

```
// Node.cpp: implementation of the CNode class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "resource.h"
#include "Node.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
CNode::CNode()
{
m_pParentNode = NULL;
// m_sName = "";
// m_pTag = new CTag;
}
CNode::~CNode()
{
}
/************************/
bool CNode::HasContent()
{
// return m_pTag.HasContent();
return (!m_sContent.empty());
}
bool CNode::HasTagProperty()
{
return m_pTag.HasProperties();
}
void CNode::SetContent(string sContent)
{
m_sContent.push_back(sContent);
// m_pTag.SetSequence(iSeq);
}
void CNode::SetParentNode( CNode* pNode )
{
m_pParentNode = pNode;
}
CNode* CNode::GetParentNode()
{
```

```
return m_pParentNode;
}
bool CNode::HasParent()
{
return (m_pParentNode != NULL);
}
bool CNode::HasChild()
{
return (m_sChildNode.size() > 0);
}
string CNode::GetTagName()
{
return m_pTag.GetName();
}
void CNode::SetTagName(string sName)
{
m_pTag.SetName(sName);
}
void CNode::AddChilde(CNode* pChildNode)
{
m_sChildNode.push_back(pChildNode);
}
/*****************************************/
/*********Get Child Node, not used********/
/*
void CNode::GetChildNode(string &sOut)
{
vector<string>::iterator p1;
unsigned int i = 0;
sOut += "<" + this->GetTagName();
if( this->HasTagProperty() )
this->GetTagProperties(sOut);
sOut += ">";
if(this->HasContent())
{
for(p1 = m_sContent.begin();p1<m_sContent.end();p1++)
{
if( (*p1) == "CHILD" && this->HasChild() )
{
sOut += "\n";
if( m_sChildNode.at(i)->HasChild() || m_sChildNode.at(i)->HasContent() )
{
m_sChildNode.at(i)->GetChildNode(sOut);
}
i++;
}
else
sOut += (*p1);
}
}
sOut += "</" + this->GetTagName() + ">\n";
}*/
/*********************************/
/*********************************/
void CNode::DisplayContent()
{
vector<string>::iterator p2;
if( this->HasContent())
{
for(p2 = m_sContent.begin();p2 != m_sContent.end(); p2++)
{
//viewing the content.
}
}
```

```
}
/*****************************/
/*************************************************************/
void CNode::SetTagProperties(string sProperty, string sPropValue)
{
m_pTag.SetProperties(sProperty, sPropValue);
}
void CNode::GetTagProperties(string &sOut, string sShortDomain, string sLongDomain)
{
m_pTag.GetProperties(sOut, sShortDomain, sLongDomain);
}
unsigned int CNode::GetNodeSize()
{
return m_sChildNode.size();
}
/*****************************/
/*****************************/
/*
CNode* CNode::GetBodyNode()
{
vector<CNode*>::iterator p1;
bool bFound = false;

for(p1 = m_sChildNode.begin(); p1 != m_sChildNode.end(); p1++)
{
if( (*p1)->GetTagName() == "body" )
{
bFound = true;
return *p1;
break;
}
}
if(bFound == false)
return NULL;
}
*/
CNode* CNode::GetNode(unsigned int i)
{
return m_sChildNode.at(i);
}
string CNode::GetContent(unsigned int i)
{
return m_sContent.at(i);
}
unsigned int CNode::GetContentSize()
{
return m_sContent.size();
}
/*************************************/
/*************************************/
/* This function is not used
void CNode::DisplayChild()
{
vector<CNode*>::iterator p1;
string sProperty = "";
for(p1 = m_sChildNode.begin(); p1 != m_sChildNode.end(); p1++)
{
if( (*p1)->HasTagProperty() )
{
(*p1)->GetTagProperties(sProperty);
sProperty = "";
}
if( (*p1)->HasContent() )
{
```

```
(*p1)->DisplayContent();
}
}
}
*/
string CNode::GetSpecificContent()
{
string sContent = "";
vector<string>::iterator p2;
if( this->HasContent())
{
for(p2 = m_sContent.begin();p2 != m_sContent.end(); p2++)
{
sContent += (*p2);
}
}
return sContent;
}
/***********************************/
```

3. File: OutProcess1.cpp

```
// OutProcess1.cpp: implementation of the COutProcess class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "resource.h"
#include "OutProcess1.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
COutProcess::COutProcess()
{
}
COutProcess::~COutProcess()
{
}
/*function is not used
void COutProcess::ProcessOutput(CNode* pMainNode, string szOutPath, string szID)
{
// size of page: 800 x 600 dpi
// number of segment per page: 4 segment( each segment will be transformed into individual li
nked pages)
// number of nodes per page: size/4
unsigned int nSegment = 4;
unsigned int nNodesPerPages;
unsigned int nContentSize;
unsigned int nPage = 0;
char tmp[10];
FILE *fOut;
string sOut = "";
string sLocation = "";
string sContent = "";
CNode* pBodyNode = new CNode;
CNode* pCurrNode = new CNode;
pBodyNode = pMainNode->GetBodyNode();
if(pBodyNode)
{
nContentSize = pBodyNode->GetContentSize();
if( nContentSize > 4 )
nNodesPerPages = nContentSize/4;
else if( nContentSize < 4 )
nNodesPerPages = nContentSize;
else if( nContentSize == 4 )
nNodesPerPages = 4;
```

```
unsigned int nMax;
static unsigned int nContentNav;
static unsigned int nChildNav;
nContentNav = 0;
nChildNav = 0;
nMax = nNodesPerPages;
if( nMax >= 4 )
{
for( unsigned int i=0; i< 4; i++)
{
for( ; nContentNav < pBodyNode->GetContentSize(); nContentNav++ )
{
sContent = pBodyNode->GetContent(nContentNav);
if( (nContentNav%nNodesPerPages) == 0 )
break;
else
{
sOut += "<html>\n";
sOut += "<head>\n</head>\n\n";
sOut += "<body>\n";
if( sContent == "CHILD")
{
pCurrNode = pBodyNode->GetNode(nChildNav++);
pCurrNode->GetChildNode(sOut);
}
else
{
sOut += sContent + "\n";
}
sOut += "\n</body>\n";
sOut += "\n</html>";
nPage++;
_itoa(nPage, tmp, 10);
sLocation = szOutPath + szID;
sLocation += "\\";
sLocation += tmp;
sLocation += "i.htm";
fOut = fopen(sLocation.c_str(), "w");
fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
sOut = "";
sLocation = "";
}
}
}
m_nPages = nPage;
}
else if( nMax < 4 )
{
for( ; nContentNav < nMax; nContentNav++)
{
sOut += "<html>\n";
sOut += "<head>\n</head>\n\n";
sOut += "<body>\n";
sContent = pBodyNode->GetContent(nContentNav);
if( sContent == "CHILD" )
{
pCurrNode = pBodyNode->GetNode(nChildNav++);
pCurrNode->GetChildNode(sOut);
}
else
{
sOut += sContent;
}
sOut += "\n</body>\n";
sOut += "\n</html>";
nPage++;
sLocation = szOutPath + szID;
sLocation += "\\";
_itoa(nPage, tmp, 10);
sLocation += tmp;
sLocation += "i.htm";

fOut = fopen(sLocation.c_str(), "w");
fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
```

```
sOut = "";
sLocation = "";
}
m_nPages = nPage;
}
}
}
*/
bool COutProcess::CreateIndexFile(string szOutPath, string szID)
{
bool bCondition;
string sLink;
string sLocation;
string sIndex;
sLocation = szOutPath + szID;
sIndex = sLocation + "\\i.txt";
FILE *fIndex;
fIndex = fopen(sIndex.c_str(), "w");
if(fIndex)
{
if( m_nPages > 0 )
{
for(unsigned int i = 0; i < m_nPages; i++)
{
fprintf(fIndex,"%di.htm\n", i+1);
// fprintf(fIndex, "\n");
}
bCondition = true;
}
else
bCondition = false;
}
else
bCondition = false;
fclose(fIndex);
return bCondition;
}
void COutProcess::SpecificOutputProcess(CNode *pMainNode, string szOutPath, string szID, strin
g sDomainUrl)
{
unsigned int nSegment = 4;
unsigned int nNodesPerPages;
unsigned int nPage = 0;
unsigned int nNodeSize;
string sProperties = "";
char tmp[10];
char ch;
FILE *fOut;
m_sDomainUrl = sDomainUrl;
string sOut = "";
string sLocation = "";
string sContent = "";
CNode* pCurrNode = new CNode;

nNodeSize = pMainNode->GetNodeSize();
if( nNodeSize > 4 )
nNodesPerPages = 10;
else if( nNodeSize <= 4)
nNodesPerPages = nNodeSize;
unsigned int nMax;
static unsigned int nContentNav;
static unsigned int nChildNav;
nContentNav = 0;
nChildNav = 0;
nMax = nNodeSize;
if( nMax >= 4 )
{
for( unsigned int i=0; nContentNav < nMax; i++)
{
for( ; nContentNav < nMax; nContentNav++ )
{
pCurrNode = pMainNode->GetNode(nContentNav);
nChildNav++;
if( pCurrNode->GetTagName() == "html" || pCurrNode->GetTagName() == "head"
||
pCurrNode->GetTagName() == "title" || pCurrNode->GetTagName() == "body
```

```
" ||
pCurrNode->GetTagName() == "meta" || pCurrNode->GetTagName() == "table
" ||
pCurrNode->GetTagName() == "tr" || pCurrNode->GetTagName() == "td" ||
pCurrNode->GetTagName() == "tbody" || //pCurrNode->GetTagName() == "im
g" ||
pCurrNode->GetTagName() == "span" || pCurrNode->GetTagName() == "style
" ||
pCurrNode->GetTagName() == "div" || //pCurrNode->GetTagName() == "stro
ng" ||
pCurrNode->GetTagName() == "frame" || pCurrNode->GetTagName() == "fram
eset" ||
(strncmp(pCurrNode->GetTagName().c_str(), "/", 1)) == 0 )
{
nChildNav--;
continue;
}
else if( pCurrNode->GetTagName() == "br" )
sContent += "<" + pCurrNode->GetTagName() + ">";
else if( pCurrNode->GetTagName() == "ul" || pCurrNode->GetTagName() == "ol
" )
ch = '*';
else if( pCurrNode->GetTagName() == "li" )
{
sContent += "<p>";
if( pCurrNode->HasContent() )
{
sContent += ch + pCurrNode->GetSpecificContent();
sContent += "\n";
}
sContent += "</p>\n";
}
else
{
// sContent += nChildNav + ":" + nContentNav;
sContent += "<" + pCurrNode->GetTagName();
if( pCurrNode->HasTagProperty() )
{
pCurrNode->GetTagProperties(sProperties, GetShortDomain(), GetLong
Domain());
sContent += sProperties;
sProperties = "";
}
sContent += ">\n";
if( pCurrNode->HasContent() )
{
sContent += pCurrNode->GetSpecificContent();

sContent += "\n";
}
sContent += "</" + pCurrNode->GetTagName();
sContent += "><br>\n";
}
if( nChildNav == nNodesPerPages )
{
nContentNav++;
nChildNav = 0;
break;
}
}
if( sContent == "" )
{
sOut = "";
sLocation = "";
break;
}
sOut += sContent;
nPage++;
_itoa(nPage, tmp, 10);
sLocation = szOutPath + szID;
sLocation += "\\";
sLocation += tmp;
sLocation += "i.htm";
fOut = fopen(sLocation.c_str(), "w");
fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
```

```
sOut = "";
sLocation = "";
sContent = "";
}
m_nPages = nPage;
}
else if( nMax < 4 )
{
for( ; nContentNav < nMax; nContentNav++)
{
pCurrNode = pMainNode->GetNode(nContentNav);
if( pCurrNode->GetTagName() == "html" || pCurrNode->GetTagName() == "head" ||
pCurrNode->GetTagName() == "title" || pCurrNode->GetTagName() == "body" ||
pCurrNode->GetTagName() == "meta" || pCurrNode->GetTagName() == "table" ||
pCurrNode->GetTagName() == "tr" || pCurrNode->GetTagName() == "td" ||
pCurrNode->GetTagName() == "tbody" || //pCurrNode->GetTagName() == "img" |
|
pCurrNode->GetTagName() == "div" || pCurrNode->GetTagName() == "span" ||
pCurrNode->GetTagName() == "style" ||
pCurrNode->GetTagName() == "frame" || pCurrNode->GetTagName() == "frameset
" ||
(strncmp(pCurrNode->GetTagName().c_str(), "/", 1)) == 0 )
{
sOut = "";
continue;
}
else if( pCurrNode->GetTagName() == "br" )
sContent += "<" + pCurrNode->GetTagName() + ">";
else if( pCurrNode->GetTagName() == "ul" || pCurrNode->GetTagName() == "ol" )
ch = '*';
else if( pCurrNode->GetTagName() == "li" )
{
sContent += "<p>";

if( pCurrNode->HasContent() )
{
sContent += ch + pCurrNode->GetSpecificContent();
sContent += "\n";
}
sContent += "</p>\n";
}
else
{
//sOut += "smaller";
sOut += "<" + pCurrNode->GetTagName();
if( pCurrNode->HasTagProperty() )
{
pCurrNode->GetTagProperties(sProperties, GetShortDomain(), GetLongDoma
in() );
sOut += " "+ sProperties;
sOut += "\n";
sProperties = "";
}
sOut += ">";
if( pCurrNode->HasContent() )
{
sOut += pCurrNode->GetSpecificContent();
}
sOut += "</";
sOut += pCurrNode->GetTagName();
sOut += "><br>\n";
}
nPage++;
sLocation = szOutPath + szID;
sLocation += "\\";
_itoa(nPage, tmp, 10);
sLocation += tmp;
sLocation += "i.htm";
fOut = fopen(sLocation.c_str(), "w");
fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
sOut = "";
sLocation = "";
}
m_nPages = nPage;
}
```

```
else
{
m_nPages = 0;
}
}
string COutProcess::GetLongDomain()
{
int nPos;
char ch;
string sTemp = "";
nPos = m_sDomainUrl.rfind('/');
if(nPos > 0 )
{
for( int i=0; i<nPos+1; i++)
{
ch = m_sDomainUrl.at(i);
sTemp += ch;
}
}
else

sTemp = m_sDomainUrl + '/';
if( !sTemp.empty() )
return sTemp;
else
return "No Domain specified";
}
string COutProcess::GetShortDomain()
{
int nPos;
char ch;
string sTemp = "";
nPos = m_sDomainUrl.find('/', 7);
if( nPos > 7 )
{
for(int i=0;i<nPos;i++)
{
ch = m_sDomainUrl.at(i);
sTemp += ch;
}
}
else
sTemp = m_sDomainUrl;
if( !sTemp.empty() )
return sTemp;
else
return "No Domain";
}
void COutProcess::PutWap(BOOL bWap)
{
m_bWapProcess = bWap;
}
void COutProcess::WapOutput(CNode *pMainNode, string szOutPath, string szID, string sDomainUrl
)
{
unsigned int nSegment = 4;
unsigned int nNodesPerPages;
unsigned int nPage = 0;
unsigned int nNodeSize;
string sProperties = "";
char cUl;
char tmp[10];
FILE *fOut;
m_sDomainUrl = sDomainUrl;
string sOut = "";
string sLocation = "";
string sContent = "";
CNode* pCurrNode = new CNode;
nNodeSize = pMainNode->GetNodeSize();
if( nNodeSize > 4 )
nNodesPerPages = 5;
else if( nNodeSize <= 4)
nNodesPerPages = nNodeSize;
unsigned int nMax;
static unsigned int nContentNav;
static unsigned int nChildNav;
```

```
nContentNav = 0;
nChildNav = 0;
nMax = nNodeSize;

if( nMax >= 4 )
{
for( unsigned int i=0; i< 4; i++)
{
for( ; nContentNav < nMax; nContentNav++ )
{
pCurrNode = pMainNode->GetNode(nContentNav);
nChildNav++;
if( pCurrNode->GetTagName() == "html" || pCurrNode->GetTagName() == "head"
||
pCurrNode->GetTagName() == "title" || pCurrNode->GetTagName() == "body
" ||
pCurrNode->GetTagName() == "meta" || pCurrNode->GetTagName() == "table
" ||
pCurrNode->GetTagName() == "tr" || pCurrNode->GetTagName() == "td" ||
pCurrNode->GetTagName() == "tbody" || pCurrNode->GetTagName() == "th"
||
pCurrNode->GetTagName() == "style" || pCurrNode->GetTagName() == "img"
||
pCurrNode->GetTagName() == "span" || pCurrNode->GetTagName() == "div"
||
pCurrNode->GetTagName() == "frame" || pCurrNode->GetTagName() == "fram
eset" ||
(strncmp(pCurrNode->GetTagName().c_str(), "/", 1)) == 0 )
{
nChildNav--;
continue;
}
else if( pCurrNode->GetTagName() == "ol" || pCurrNode->GetTagName() == "ul
" )
cUl = '-';
else if( pCurrNode->GetTagName() == "li" )
{
sContent += "\n<br/>";
sContent += cUl;
if( pCurrNode->HasContent() )
{
sContent += pCurrNode->GetSpecificContent();
sContent += "<br/>\n";
}
}
else if( pCurrNode->GetTagName() == "br" )
sContent += "<br/>";
else
{
sContent += "\n";
if( pCurrNode->HasContent() )
{
sContent += pCurrNode->GetSpecificContent();
}
}
if( nChildNav == nNodesPerPages )
{
nContentNav++;
nChildNav = 0;
break;
}
}
if( sContent == "" )
{
sOut = "";
sLocation = "";
break;
}
sOut += sContent;
nPage++;
_itoa(nPage, tmp, 10);
sLocation = szOutPath + szID;

sLocation += "\\";
sLocation += tmp;
sLocation += "i.htm";
```

```
fOut = fopen(sLocation.c_str(), "w");
fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
sOut = "";
sLocation = "";
sContent = "";
}
m_nPages = nPage;
}
else if( nMax < 4 )
{
for( ; nContentNav < nMax; nContentNav++)
{
pCurrNode = pMainNode->GetNode(nContentNav);
if( pCurrNode->GetTagName() == "html" || pCurrNode->GetTagName() == "head" ||
pCurrNode->GetTagName() == "title" || pCurrNode->GetTagName() == "body" ||
pCurrNode->GetTagName() == "meta" || pCurrNode->GetTagName() == "table" ||
pCurrNode->GetTagName() == "tr" || pCurrNode->GetTagName() == "td" ||
pCurrNode->GetTagName() == "tbody" || pCurrNode->GetTagName() == "th" ||
pCurrNode->GetTagName() == "style" || pCurrNode->GetTagName() == "img" ||
pCurrNode->GetTagName() == "span" || pCurrNode->GetTagName() == "div" ||
pCurrNode->GetTagName() == "frame" || pCurrNode->GetTagName() == "frameset
" ||
(strncmp(pCurrNode->GetTagName().c_str(), "/", 1)) == 0 )
{
sOut = "";
continue;
}
else if( pCurrNode->GetTagName() == "ol" || pCurrNode->GetTagName() == "ul" )
cUl = '-';
else if( pCurrNode->GetTagName() == "li" )
{
sContent += "\n<br/>";
sContent += cUl;
if( pCurrNode->HasContent() )
{
sContent += pCurrNode->GetSpecificContent();
sContent += "<br/>\n";
}
}
else if( pCurrNode->GetTagName() == "br" )
sContent += "<br/>";
else
{
sContent += "\n";
if( pCurrNode->HasContent() )
{
sContent += pCurrNode->GetSpecificContent();
}
}
nPage++;
sLocation = szOutPath + szID;
sLocation += "\\";
_itoa(nPage, tmp, 10);
sLocation += tmp;
sLocation += "i.htm";
fOut = fopen(sLocation.c_str(), "w");

fprintf(fOut, "%s", sOut.c_str());
fclose(fOut);
sOut = "";
sLocation = "";
}
m_nPages = nPage;
}
else
{
m_nPages = 0;
}
}
```

4. File: Property.cpp

```
// Property.cpp: implementation of the CProperty class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "resource.h"
#include "Property.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
CProperty::CProperty()
{
}
CProperty::~CProperty()
{
}
/************function to set value*************/
void CProperty::SetProperty(string sProperty)
{
m_sProperty = sProperty;
}
void CProperty::SetPropValue(string sPropValue)
{
m_sPropValue = sPropValue;
}
/*****************************************/
```

5. File: Tag.cpp

```
// Tag.cpp: implementation of the CTag class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "resource.h"
#include "Tag.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
CTag::CTag()
{
}
CTag::~CTag()
{
}
/*****************************************/
void CTag::SetContent(string sContent)
{
m_sContent.push_back(sContent);
}
/*****************************************/
/*****************************************/
void CTag::SetName(string sName)
{
m_sName = sName;
}
/*****************************************/
/*************************************************************/
void CTag::SetProperties(string sProperty, string sPropValue)
{
CProperty *pTemp;
pTemp = new CProperty;
pTemp->SetProperty(sProperty);
```

```
pTemp->SetPropValue(sPropValue);
m_pProperties.push_back(pTemp);
}
/*************************************************************/
/*************************************************************/
void CTag::GetProperties(string &sOut, string sShortDomain, string sLongDomain)
{
/*
To get all the properties of a tag.
*/
vector<CProperty*>::iterator p1;
string sTempProperty;
string sTempPropValue;
for(p1 = m_pProperties.begin(); p1 != m_pProperties.end(); p1++)
{
sTempProperty = (*p1)->GetProperty();
sTempPropValue = (*p1)->GetPropValue();

if( (*p1)->GetPropValue().at(0) == '"' )
sTempPropValue = (*p1)->GetPropValue().substr(1, ((*p1)->GetPropValue().length()-2) )
;
else
sTempPropValue = (*p1)->GetPropValue();
if( sTempProperty == "onclick" || sTempProperty == "onsubmit" ||
sTempProperty == "onload" || sTempProperty == "onmousedown" )
{
sTempProperty = "";
sTempPropValue = "";
continue;
}
if( sTempProperty == "src" )
{
if( sTempPropValue.at(0) == '/' )
{
/* if the first char is a slash*/
sTempPropValue = sShortDomain + sTempPropValue;
}
else if( sTempPropValue.find("http://") == 0 )
sTempPropValue = sTempPropValue;
else
{
/* if the first char is neither 'h' of http nor slash */
sTempPropValue = sLongDomain + sTempPropValue;
}
}
else
sTempPropValue = sTempPropValue;
/* else if( sTempProperty == "href" )
{
if( sTempPropValue.at(0) == '/' )
{
/* if the first char is a slash
sTempPropValue = sShortDomain + sTempPropValue;
}
else if( sTempPropValue.at(0) == 'h' )//use the string search to search for the exist
ent of http://
{
/*no process
}
else
{
/* if the first char is neither 'h' of http nor slash
sTempPropValue = sLongDomain + sTempPropValue;
}
}
*/
sOut += " " + sTempProperty + "=";
sOut += '"' + sTempPropValue + '"';
sTempProperty = "";
sTempPropValue = "";
}
}
/*********************************/
/*********************************/
bool CTag::HasProperties()
{
```

```
return (m_pProperties.size()>0);
}
bool CTag::HasContent()
{
return (!m_sContent.empty());
}
/*******************************/
```

6. File: WAC.cpp

```
// WAC.cpp : Implementation of DLL Exports.
// Note: Proxy/Stub Information
// To build a separate proxy/stub DLL,
// run nmake -f WACps.mk in the project directory.
#include "stdafx.h"
#include "resource.h"
#include <initguid.h>
#include "WAC.h"
#include "WAC_i.c"
#include "FlexDisplay.h"
CComModule _Module;
BEGIN_OBJECT_MAP(ObjectMap)
OBJECT_ENTRY(CLSID_FlexDisplay, CFlexDisplay)
END_OBJECT_MAP()
class CWACApp : public CWinApp
{
public:
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CWACApp)
public:
virtual BOOL InitInstance();
virtual int ExitInstance();
//}}AFX_VIRTUAL
//{{AFX_MSG(CWACApp)
// NOTE - the ClassWizard will add and remove member functions here.
// DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CWACApp, CWinApp)
//{{AFX_MSG_MAP(CWACApp)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
CWACApp theApp;
BOOL CWACApp::InitInstance()
{
_Module.Init(ObjectMap, m_hInstance, &LIBID_WACLib);
return CWinApp::InitInstance();
}
int CWACApp::ExitInstance()
{
_Module.Term();
return CWinApp::ExitInstance();
}
////////////////////////////////////////////////////////////////////
// Used to determine whether the DLL can be unloaded by OLE
STDAPI DllCanUnloadNow(void)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
return (AfxDllCanUnloadNow()==S_OK && _Module.GetLockCount()==0) ? S_OK : S_FALSE;
}
////////////////////////////////////////////////////////////////////
// Returns a class factory to create an object of the requested type
STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{

return _Module.GetClassObject(rclsid, riid, ppv);
}
////////////////////////////////////////////////////////////////////
// DllRegisterServer - Adds entries to the system registry
STDAPI DllRegisterServer(void)
```

```
{
// registers object, typelib and all interfaces in typelib
return _Module.RegisterServer(TRUE);
}
////////////////////////////////////////////////////////////////////////
// DllUnregisterServer - Removes entries from the system registry
STDAPI DllUnregisterServer(void)
{
return _Module.UnregisterServer(TRUE);
}
```

7. File: WACTear.cpp

```
// WACTear.cpp : Implementation of DLL Exports.
// Note: Proxy/Stub Information
// To build a separate proxy/stub DLL,
// run nmake -f WACTearps.mk in the project directory.
#include "stdafx.h"
#include "resource.h"
#include <initguid.h>
#include "WACTear.h"
#include "WACTear_i.c"
#include "WacHtmTear.h"
CComModule _Module;
BEGIN_OBJECT_MAP(ObjectMap)
OBJECT_ENTRY(CLSID_WacHtmTear, CWacHtmTear)
END_OBJECT_MAP()
class CWACTearApp : public CWinApp
{
public:
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CWACTearApp)
public:
virtual BOOL InitInstance();
virtual int ExitInstance();
//}}AFX_VIRTUAL
//{{AFX_MSG(CWACTearApp)
// NOTE - the ClassWizard will add and remove member functions here.
// DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CWACTearApp, CWinApp)
//{{AFX_MSG_MAP(CWACTearApp)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
CWACTearApp theApp;
BOOL CWACTearApp::InitInstance()
{
_Module.Init(ObjectMap, m_hInstance, &LIBID_WACTEARLib);
return CWinApp::InitInstance();
}
int CWACTearApp::ExitInstance()
{
_Module.Term();
return CWinApp::ExitInstance();
}
////////////////////////////////////////////////////////////////////////
// Used to determine whether the DLL can be unloaded by OLE
STDAPI DllCanUnloadNow(void)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState());
return (AfxDllCanUnloadNow()==S_OK && _Module.GetLockCount()==0) ? S_OK : S_FALSE;
}
////////////////////////////////////////////////////////////////////////
// Returns a class factory to create an object of the requested type
STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{

return _Module.GetClassObject(rclsid, riid, ppv);
}
```

```
////////////////////////////////////////////////////////////////////
// DllRegisterServer - Adds entries to the system registry
STDAPI DllRegisterServer(void)
{
// registers object, typelib and all interfaces in typelib
return _Module.RegisterServer(TRUE);
}
////////////////////////////////////////////////////////////////////
// DllUnregisterServer - Removes entries from the system registry
STDAPI DllUnregisterServer(void)
{
return _Module.UnregisterServer(TRUE);
}
```

8. File: WacHtmTear.cpp

```
// WacHtmTear.cpp : Implementation of CWacHtmTear
#include "stdafx.h"
#include "WACTear.h"
#include "WacHtmTear.h"
#include "TearData.h"
#include "direct.h"
#include "io.h"
#include <atlconv.h>
////////////////////////////////////////////////////////////////////
// CWacHtmTear
BOOL CWacHtmTear::_Init()
{
return m_info.Init();
}
int CWacHtmTear::_MakeDir(LPCSTR pszFileName)
{
const int SUCCESS = 0;
const int ERR_NOT_FULL_PATH = 1;
const int ERR_MAKE_DIR = 2;
TCHAR szDrv[_MAX_DRIVE];
TCHAR szDir[_MAX_DIR];
TCHAR szFName[_MAX_FNAME];
TCHAR szExt[_MAX_EXT];
TCHAR szPath[_MAX_PATH];
_splitpath(pszFileName, szDrv, szDir, szFName, szExt);
if(strlen(szDrv) == 0)
return ERR_NOT_FULL_PATH;
//make directory name including drive name.
sprintf(szPath, "%s%s\\0", szDrv, szDir);
if(_access(szPath, 06) == -1)//path exists? access permission? using ANSI C errno.
{
TCHAR szTmp[_MAX_PATH];
LPSTR pos = strchr(szPath, '\\');
do
{
pos = strchr(pos+1, '\\');
if(pos != NULL)
{
strcpy(szTmp, szPath);
szTmp[pos - szPath] = '\0';
}
else if(szPath[strlen(szPath) - 1] != '\\')
strcpy(szTmp, szPath);
else //no more.
return 0;
if(_access(szTmp, 0) == -1) //see if it already exists.
{
//make directory
if(_mkdir(szTmp) == -1)
return ERR_MAKE_DIR;
}
}while(pos != NULL);
}
return SUCCESS;
}
HRESULT CWacHtmTear::_ReportError(LPCSTR psz, HRESULT hReturnVal, HRESULT hErNum)
{
USES_CONVERSION;
```

```
Error(T2OLE(psz), GetObjectCLSID(), hErrNum);
return hReturnVal;
}
STDMETHODIMP CWacHtmTear::_StrToVarArray(LPCSTR psz, VARIANT *pvar)
{

SAFEARRAYBOUND sabBound;
SAFEARRAY *psaArray = NULL;
HRESULT hResult;
VARIANT varElem;
VariantInit(&varElem);
V_VT(&varElem) = VT_UI1;
hResult = VariantClear(pvar);
if(FAILED(hResult)) return hResult;
V_VT(pvar) = VT_ARRAY | VT_VARIANT;
sabBound.cElements = strlen(psz);
sabBound.lLbound = 0;
psaArray = SafeArrayCreate(VT_VARIANT, 1, &sabBound);
if(!psaArray) return S_FALSE;
VARIANT HUGEP *pvarArray;
hResult = SafeArrayAccessData(psaArray, (void HUGEP* FAR*)&pvarArray);
if(FAILED(hResult))
{
SafeArrayDestroy(psaArray);
return hResult;
}
for(int i=0; i<(int)strlen(psz); i++)
{
V_UI1(&varElem) = (unsigned char)psz[i];
pvarArray[i] = varElem;
}
SafeArrayUnaccessData(psaArray);
V_ARRAY(pvar) = psaArray;
return S_OK;
}
STDMETHODIMP CWacHtmTear::GetPage(BSTR bstrUrl, short nMethod, BSTR bstrPayload, BSTR bstrFilenam
e, BSTR bstrUsername, BSTR bstrPassword, VARIANT_BOOL *pbResult)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pbResult = (VARIANT_BOOL)0x0000;
LPSTR pszUrl = OLE2T(bstrUrl);
LPSTR pszPayload = OLE2T(bstrPayload);
LPSTR pszUsername = OLE2T(bstrUsername);
LPSTR pszPassword = OLE2T(bstrPassword);
LPSTR pszFileName = OLE2T(bstrFilename);
if(strlen(pszUrl) == 0) return _ReportError("Invalid URL", E_FAIL, 11);
if(strlen(pszFileName) == 0) return _ReportError("Invalid filename", E_FAIL, 12);
if(strlen(pszPayload) == 0) pszPayload = NULL;
if(strlen(pszUsername) == 0) pszUsername = NULL;
if(strlen(pszPassword) == 0) pszPassword = NULL;
int nVerb = CHttpConnection::HTTP_VERB_GET;
if(nMethod == 1) nVerb = CHttpConnection::HTTP_VERB_POST;
else if(nMethod==2) nVerb = CHttpConnection::HTTP_VERB_GET;
if( 0 != _MakeDir(pszFileName))
return _ReportError("Cannot create specified directory", E_FAIL, 13);
//create the output file.
FILE *pFile = fopen(pszFileName, "w+b");
if(!pFile) return _ReportError("Cannot create specified file", E_FAIL, 14);
CWacTearProc tearObj;
tearObj.Init(m_info);

BOOL bResult = tearObj.RequestPage(pFile, pszUrl, nVerb, pszPayload, pszUsername, pszPassword
);
m_info.m_strHeaders = tearObj.GetHeaders();
m_info.m_nStatusCode = tearObj.GetStatusCode();
fclose(pFile);
if(!bResult) return _ReportError(tearObj.GetLastError(), E_FAIL, m_info.m_nStatusCode);
*pbResult = (VARIANT_BOOL)0xFFFF;
return S_OK;
}
//DEL STDMETHODIMP CWacHtmTear::qwerqwer()
//DEL {
//DEL AFX_MANAGE_STATE(AfxGetStaticModuleState())
//DEL
```

```
//DEL // TODO: Add your implementation code here
//DEL
//DEL return S_OK;
//DEL }
STDMETHODIMP CWacHtmTear::AddHeader(BSTR bstrHeaderName, BSTR bstrHeaderValue, VARIANT_BOOL *pbRe
sult)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
LPSTR pszHeaderName = OLE2T(bstrHeaderName);
LPSTR pszHeaderValue = OLE2T(bstrHeaderValue);
if(strlen(pszHeaderName) > 0 && strlen(pszHeaderValue) > 0)
{
m_info.m_strHeaderExtra += pszHeaderName;
m_info.m_strHeaderExtra += ": ";
m_info.m_strHeaderExtra += pszHeaderValue;
m_info.m_strHeaderExtra + "\r\n";
*pbResult = (VARIANT_BOOL)0xFFFF;
}
else
*pbResult = (VARIANT_BOOL)0x0000;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Accept(BSTR *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strAccept);
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Accept(BSTR newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strAccept = OLE2T(newVal);
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Cache(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())

// TODO: Add your implementation code here
*pVal = m_info.m_bCache;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Cache(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bCache = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_ConnectionTimeout(long *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_nConnectionTimeout / 1000;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_ConnectionTimeout(long newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_nConnectionTimeout = newVal * 1000;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_ContentType(BSTR *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strContentType);
return S_OK;
```

```
}
STDMETHODIMP CWacHtmTear::put_ContentType(BSTR newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strContentType = OLE2T(newVal);
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Cookies(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_bCookies;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Cookies(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bCookies = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_FollowRedirect(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here

*pVal = m_info.m_bFollowRedirects;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_FollowRedirect(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bFollowRedirects = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_ForceReload(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_bForceReload;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_ForceReload(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bForceReload = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_HttpVersion(BSTR *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strHttpVersion);
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_HttpVersion(BSTR newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strHttpVersion = OLE2T(newVal);
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_IgnoreInvalidCertDate(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_bIgnoreInvalidCertDate;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_IgnoreInvalidCertDate(BOOL newVal)
```

```
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bIgnoreInvalidCertDate = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_IgnoreInvalidCN(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())

// TODO: Add your implementation code here
*pVal = m_info.m_bIgnoreInvalidCN;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_IgnoreInvalidCN(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bIgnoreInvalidCN = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Port(long *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_nPort;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Port(long newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_nPort = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Proxy(BSTR *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strProxy);
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Proxy(BSTR newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strProxy = OLE2T(newVal);
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_ReceiveTimeout(long *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_nReceiveTimeout/1000;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_ReceiveTimeout(long newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_nReceiveTimeout = newVal * 1000;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_Referrer(BSTR *pVal)

{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strReferrer);
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_Referrer(BSTR newVal)
{
```

```
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strReferrer = OLE2T(newVal);
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_SendClientCertificate(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_bSendClientCertificate;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_SendClientCertificate(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bSendClientCertificate = newVal;
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_StatusCode(long *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_nStatusCode;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_StatusCode(long newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
return S_OK;
}
STDMETHODIMP CWacHtmTear::get_TrustUnknownCA(BOOL *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
*pVal = m_info.m_bTrustUnknownCA;
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_TrustUnknownCA(BOOL newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
m_info.m_bTrustUnknownCA = newVal;

return S_OK;
}
STDMETHODIMP CWacHtmTear::get_UserAgent(BSTR *pVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
*pVal = A2BSTR(m_info.m_strUserAgent);
return S_OK;
}
STDMETHODIMP CWacHtmTear::put_UserAgent(BSTR newVal)
{
AFX_MANAGE_STATE(AfxGetStaticModuleState())
// TODO: Add your implementation code here
USES_CONVERSION;
m_info.m_strUserAgent = OLE2T(newVal);
return S_OK;
}
```

Appendix F: Creating ATL COM with Visual C++

The steps below show how to create an ATL COM object using the Visual C++ ATL COM wizard.

1. In the Visual C++ development tool, go to File→New.

2. In the Projects, select ATL COM AppWizard (Figure 8.1). Type in the name for the project and specify the location to store the project's files.

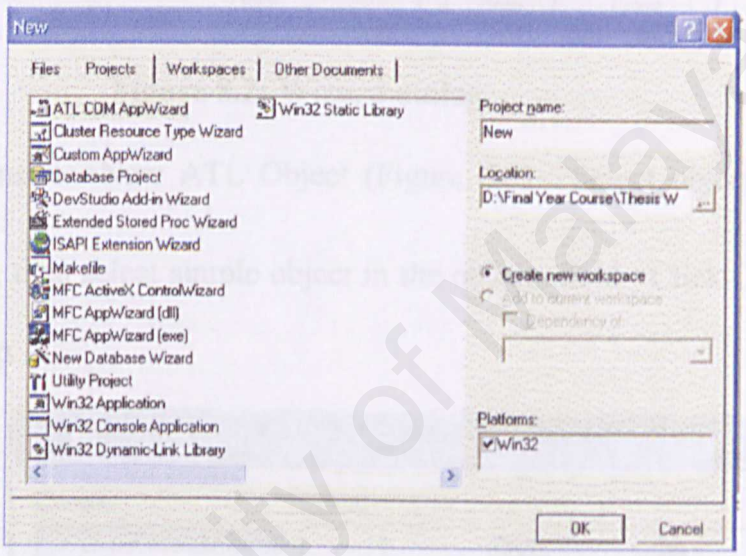


Figure 8.1: New Project

3. In the wizard dialog box (Figure 8.2), there are several options here. The ATL COM can be either in services, dynamic link-library or executable files. The example is tick on dynamic link-library and also support MFC. After finish, the necessary files will be generated.

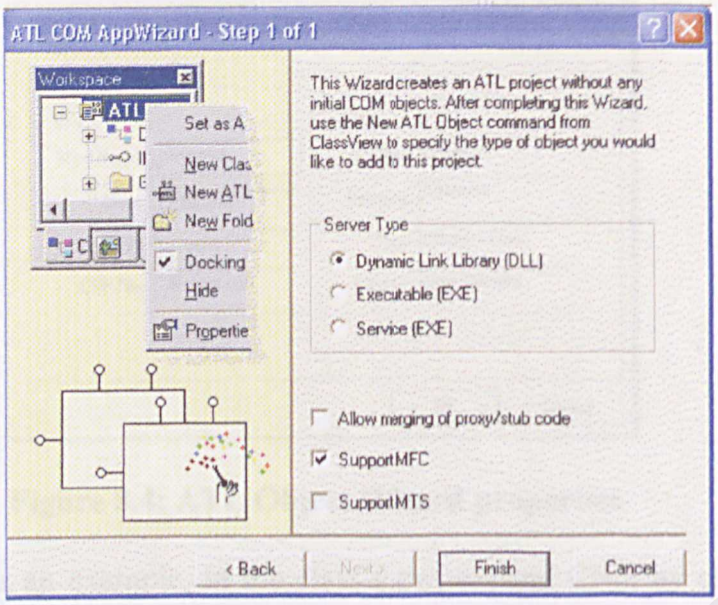


Figure 8.2: Wizard dialog

4. Next, go to Insert→New ATL Object (Figure 8.3). Select objects in the category field, then select simple object in the objects field. Click OK to go to the next step.

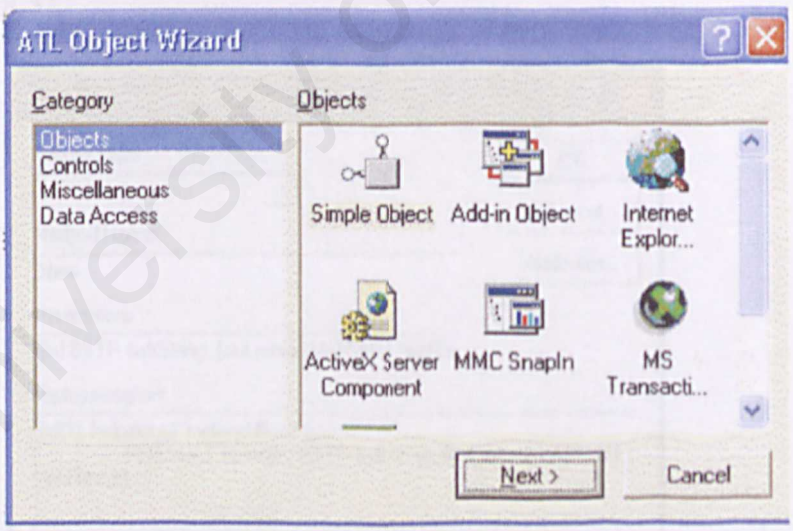


Figure 8.3: ATL Object Wizard

5. In the name field (Figure 8.4), type in the name for the object. When you type in the name, the other fields will automatically be filled.

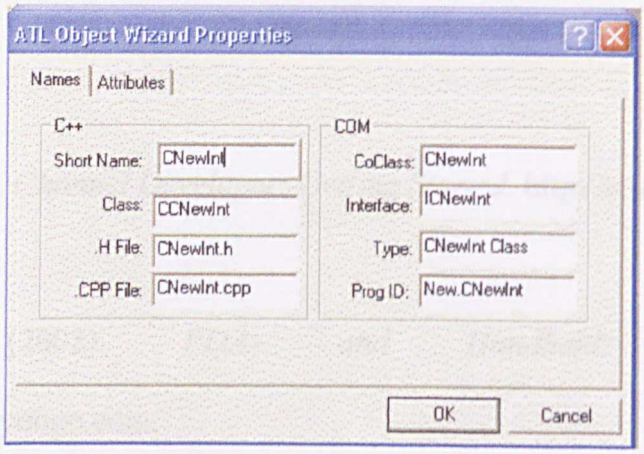


Figure 8.4: ATL Object Wizard properties

6. Using CNet as an example, in the class view, expand CNet by click on the plus sign. Then, right click on INet. There will be a dialog box prompted out that allow you to add in method and property for this object. There are specifics format on writing the method and also the property. We will not go further detail about it. Below is an example of adding a method into the object (Figure 8.5).

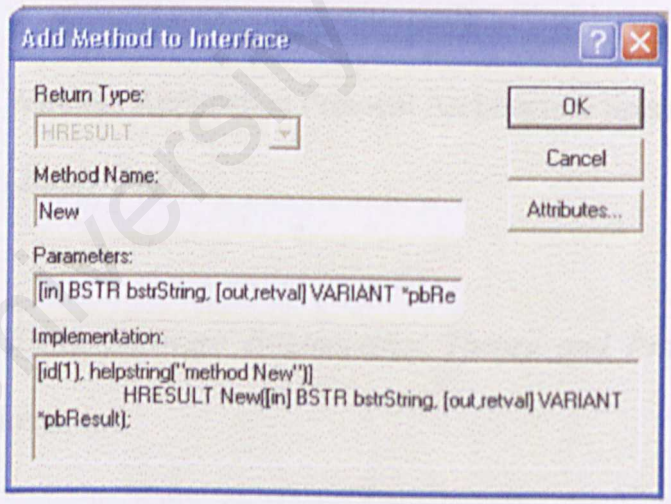


Figure 8.5: Add Method to Interface

7. For further information, please refer to the MSDN Library or visit http://msdn.microsoft.com.

Reference

AvantGo, Inc. (2002). *Channel Developer: Getting Started.* http://www.avantgo.com

Chris, Tull. (2002). *PDAs and Handheld: Introduction.* http://www.anywhereyougo.com.

Nokia, Inc. (2001). *What is 3rd Generation.* http://www.nokia.com

Rachmat, Hartono. (1999). *Smart Screen Creator.* Handisplay (M) Sdn. Bhd.

Simon, Buckingham. (2000). *What is General Radio Packet Service.* Mobile Lifestream Ltd. http://www.mobilelifestreams.com

WAP Forum. (2001). Wireless Application Protocol Architecture Specification. *WAP Architecture Version 12-July-2001.*

Shari, L. Pfleeger. (2001). *Software Engineering: Theory and Practice.* 2nd ed. Prentice Hall International, Inc.

Behrouz, A. Forouzan. (2001). *Data Communications and Networking.* 2nd ed. McGraw-Hill.

Microsoft Developer Network. (2002). *MSDN Library.* Microsoft.

Bibliography

AnyWhereYouGo.com. (2002). *i-Mode: Introduction.*

http://www.anywhereyougo.com.

WIRELESS APPLICATION CONVERTER

Wireless Application Converter User Manual

WIRELESS APPLICATION CONVERTER

WAC: USER MANUAL

Table of Content

Table of Figures

WAC: USER MANUAL

Table of Content

Table of Figures

1. Introduction

Wireless Application Converter (WAC) is a server-side application that enables different mobile devices with different format to view the same web document. It has the ability to convert the web document into different format supported by the mobile devices, such as WML and Pocket IE. The conversion is done without modifying the original document.

Before embark on the installation process, here is a brief explanation of the WAC architecture. The architecture of WAC is as shown in Figure 1. By using a centralized server, where WAC is installed, it will act as a "middle man", where it will retrieve information from the requested URL address, generate the output and send it back to the clients.

The different of WAC with a proxy server is that it is not permanent and compulsory to go through it, as the clients can use it whenever it is needed. It can be done because WAC is web-based and it only response when the clients request for its services.

1.1. Objectives

- To allow information sharing among different wireless devices with the utilization of wireless network.

- To provide cross platform that support different wireless device format, where every devices will be able to view the same internet content without the need to modify the original content.
- To optimize the output that will fit the screen resolution of the wireless device.

1.2. System Features

WAC can only support text based document. The following elements are not supported.

- Form
- Query
- Table
- Image (not supported by WAP only)
- Audio streaming
- Video streaming
- Hyperlink / redirection

2. About the manual

This manual consists of three parts. The first part explains the hardware and software requirements, follow by second part, which are the installation procedures. The final part is about how to use or interact with WAC from different devices.

There are figures in this user manual as well. It helps to aid the user in the installation process, as well as the operation / usage. Such visual aid can help the user to understand the procedure clearly.

3. System Requirements

3.1. Server

- CPU: Intel Pentium II /AMD K6II 166 MHz or above
- Memory: 128 MB (256 Recommended)
- Storage: 2 GB
- Operating System: Microsoft Windows (2000 Professional / Server Family / XP Professional)
- Others: Internet Information Services 4.0 or above, Internet Explorer 4.0 or above

3.2. Client

- PDA: Pocket PC with Wireless LAN Card
- Mobile Phone: WAP enabled

3.3. Network environment

- Wireless Network Access Point (802.11b, Wi-Fi)
- GPRS LAN Access Point (required for Mobile phone)

4. Installation Guide

4.1. DLLs registration

1. Copy all the DLL extension files to the windows system 32 folder. Example for Windows 2000: C:\WINNT\SYSTEM32\.
2. There are 3 DLL extension files. They are:
    a. genuid.dll
    b. WAC.dll
    c. WACTear.dll
3. The DLL need to be registered in order for WAC to access it.

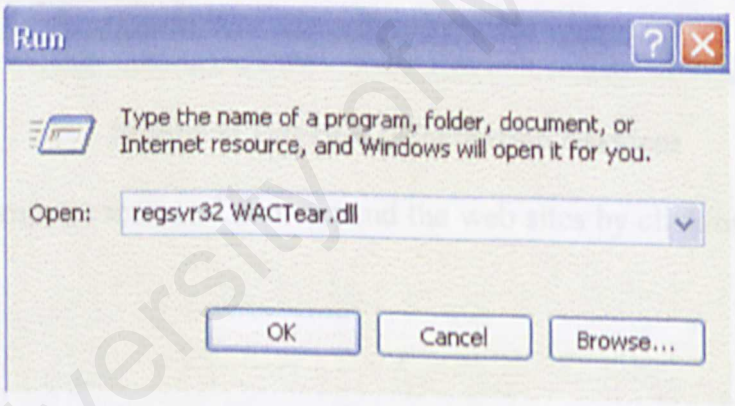


Figure 1: Registering DLL

4. Go to Start->Run. In the Run dialog box (Figure 1), type in regsvr32 [file name] to register the DLLs. Example: regsvr32 genuid.dll. Perform this procedure to register all DLLs.

4. Installation Guide

4.1. DLLs registration

1. Copy all the DLL extension files to the windows system 32 folder. Example for Windows 2000: C:\WINNT\SYSTEM32\.
2. There are 3 DLL extension files. They are:
    a. genuid.dll
    b. WAC.dll
    c. WACTear.dll
3. The DLL need to be registered in order for WAC to access it.

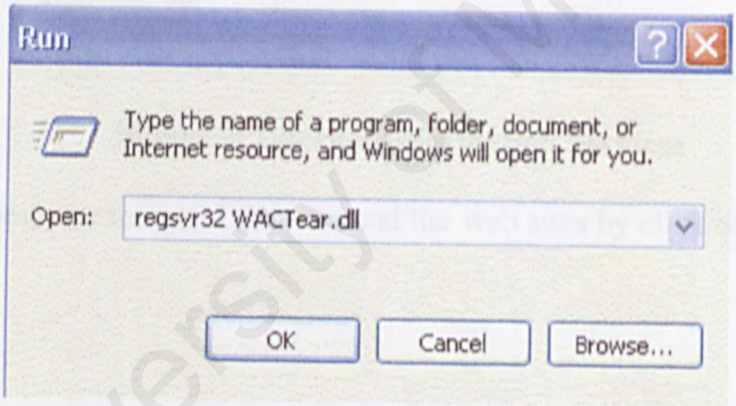


Figure 1: Registering DLL

4. Go to Start->Run. In the Run dialog box (Figure 1), type in regsvr32 [file name] to register the DLLs. Example: regsvr32 genuid.dll. Perform this procedure to register all DLLs.

4.2. WAC virtual directory setup

1. Copy the WAC folder from the source disc to directory C:\.
2. Go to Control Panel->Administrative Tools, double click on Internet Services Manager icon to open the Internet Information Services console (Figure 2).

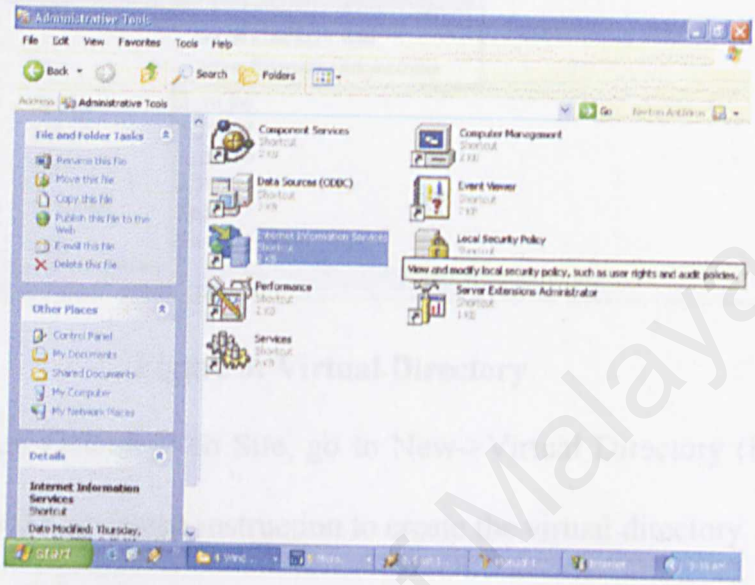


Figure 2: Internet Information Services

3. Expand the computer icon and then expand the web sites by click on the plus sign beside.

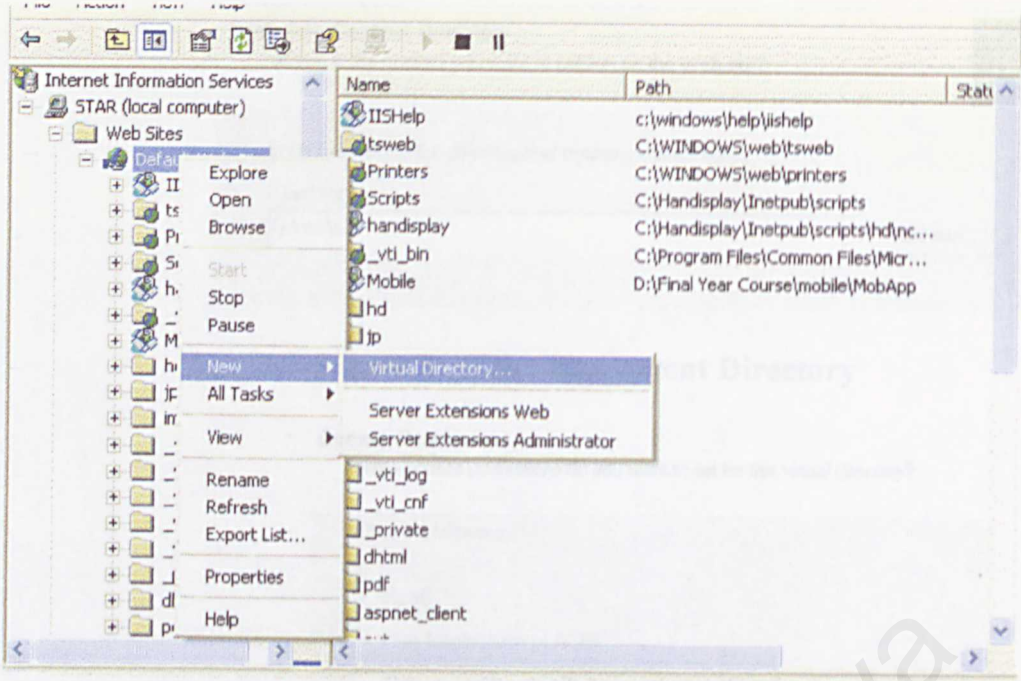


Figure 3: Virtual Directory

4. Right click on Default Web Site, go to New->Virtual Directory (Figure 3). Then, follow the on screen instruction to create the virtual directory.

   a. Type in WAC in the Virtual Directory Alias dialog (Figure 4).

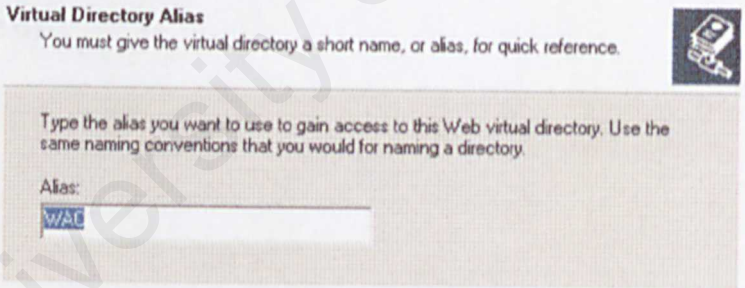


Figure 4: Virtual Directory Alias

   b. Type in or browse to the directory where WAC is stored (Figure 5). Example: C:\WAC.

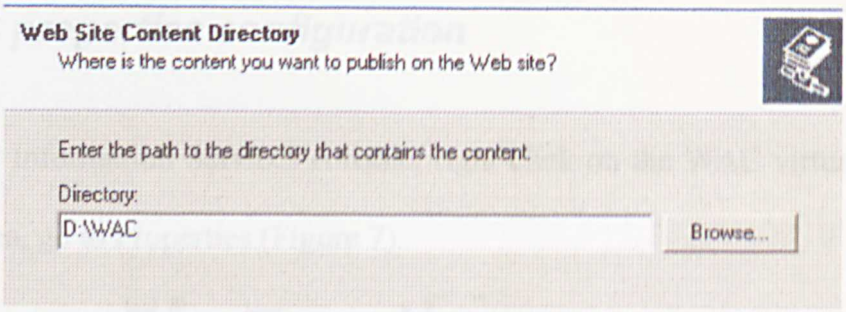


Figure 5: Web Site Content Directory

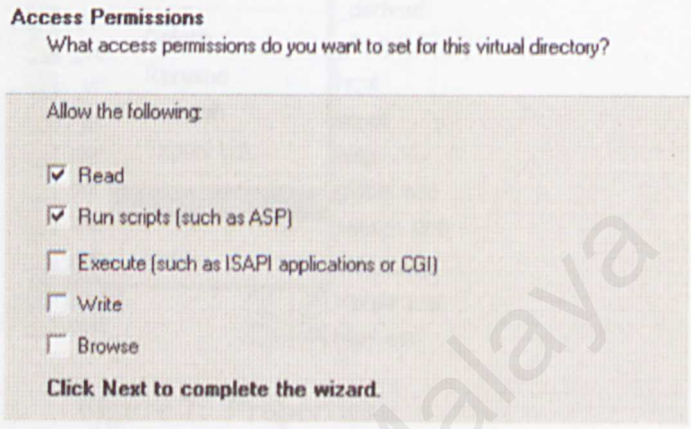


Figure 6: Access Permission

c. In the Access Permission, make sure Read and Run Script are ticked (Figure 6).

4.3. WAC properties configuration

1. In the Internet Information Services console, right click on the WAC virtual directory. Then, go to Properties (Figure 7).

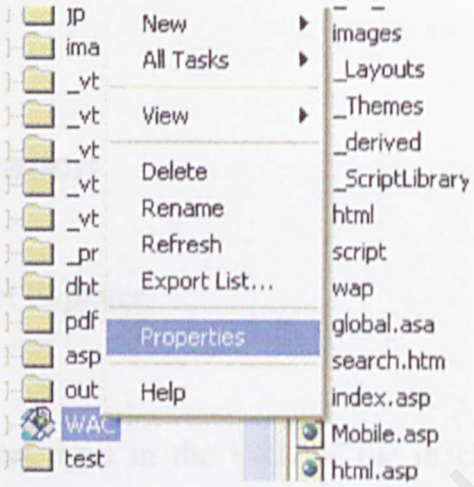


Figure 7: Properties

2. In the Properties dialog box, tab on the Documents tab (Figure 8).

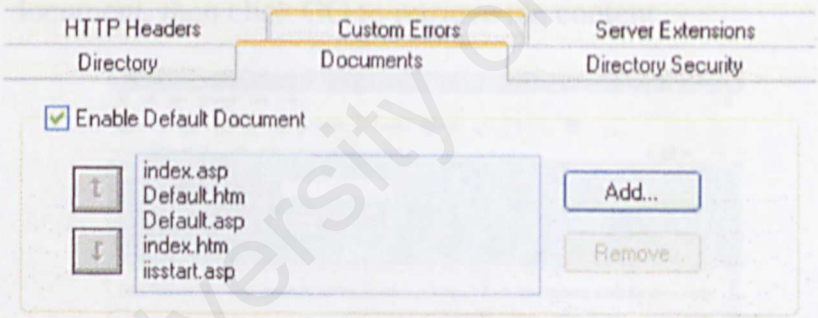


Figure 8: Documents

3. Under the Documents dialog box, make sure the Enable Default Documents option is ticked.
4. If the index.asp is not in the list, we have to add it in order it to be accessed automatically when we access WAC. Click the Add button. A dialog box will appear.
5. In the dialog box, type in index.asp, then click OK.

6. The index.asp file is added into the list. Highlight the index.asp, click on the Up Arrow key beside the dialog box to move index.asp to the highest position. Figure 8 is the example of the result.
7. Click OK to save the changes and exit the Properties dialog.

4.4. WAC Operation

Accessing WAC from Internet Explorer

1. Open Internet Explorer, type in the URL of the machine where WAC is installed. Example: http://192.168.168.103/wac.
2. A web based interface will be showed. Type in the URL of the required WWW document, then click GO to retrieve the content.

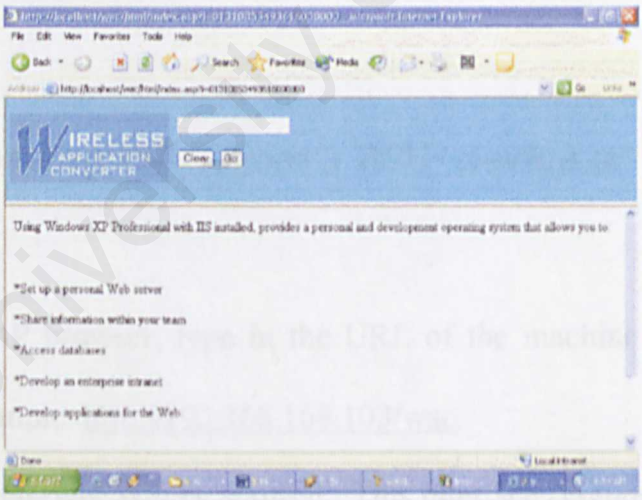


Figure 9: Result of Internet Explorer

3. Figure 9 is an example of the output.

4.5. Accessing WAC from Pocket PC

1. Open Pocket Internet Explorer, type in the URL of the machine where WAC is installed. Example: http://192.168.168.103/wac.
2. A web based interface will be showed. This interface (Figure 10) is specially optimized for Pocket PC screen resolution.

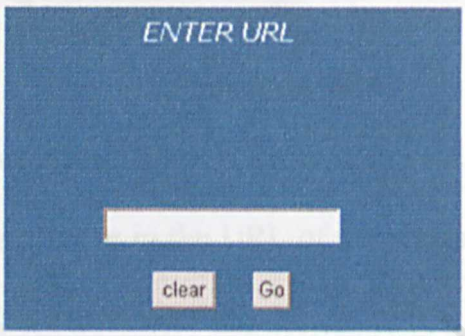


Figure 10: Pocket IE

3. In the address bar, type in the URL of the required WWW document, then click GO to retrieve the content

4.6. Accessing WAC from a WAP device or WAP simulator

1. Open any WAP browser, type in the URL of the machine where WAC is installed. Example: http://192.168.168.103/wac.
2. A web based interface will be showed. This interface (Figure 11) is specially optimized for WAP devices.

Figure 11: WAP interface

3. Go to Search, it will be redirected to the searching page.
4. From the searching page, type in the URL of the required WWW document, then click GO to retrieve the content.

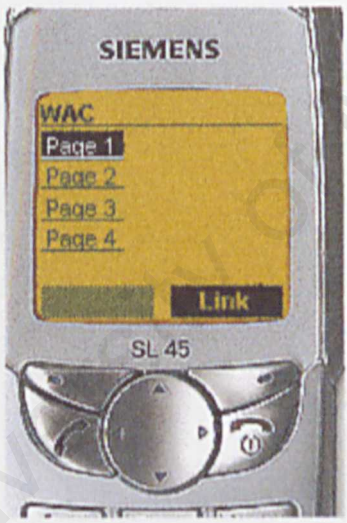


Figure 12: WAP Output

5. Figure 12 is an example of the output.

5. Troubleshooting

There might be problem encounter during the operation. If there is any problem encountered, please do not hesitate to contact our technical staff at husin_d@hotmail.com.