# THE WEB AS A LEGACY COMPUTER SYSTEM APPLICATION INTERFACE

MICHAEL HINCHLIFFE AND ANTHONY SYMONS

***Abstract*—This paper investigates the issues of using Web pages as a user interface via the internet, to a custom legacy application. The advantages of Web page based interfaces are discussed and a typical architecture which integrates Web browsers and legacy systems is depicted. Uncertainty with the security of the legacy system is a critical issue needing resolution for enterprises who are considering internet based solutions. The security issues raised, include certificates, firewalls and security policy. A solution which includes functionality in each of these areas can be acceptable. An example of a generic Web browser/legacy application features administration, user and public interfaces. Each of these roles is expanded in the paper. Aspects of legacy system integration are described, and some of the problems of managing a suite of web pages are addressed. Finally, a number of Web browser/legacy applications are referenced.**

***Keywords*—WorldWideWeb, Legacy Systems, Business applications, Security Policy**

***Source of Publication*—Proceedings of AusWeb96: The Second Australian WorldWideWeb Conference.**

## 1 ADVANTAGES OF USING THE INTERNET

The use of the Web browser as a platform independent, legacy system front end, is still in its infancy. When existing systems are required to be made more open, that is, in terms of increasing access to newly empowered users, the use of Web browsers is one of the most attractive options.

This is because web based applications provide a common look and feel, regardless of the computing platform or geographical location.

The Internet is increasingly being seen, by the key players in the provision of the Information Superhighway, as the service delivery medium of the future [2]. Consequently web browsers are fast becoming defacto standard graphical user interfaces (GUIs) due to the following factors:-

- Ease of access is increased for individual users (with their own equipment) as they are able to bookmark important pages to gain fast access to frequently used functions.
- From the provider's viewpoint, a Web page front end is very customisable, allowing help, marketing information, operating manuals, catalogues and demonstrations to be available on demand, alongside the actual service interface.
- Functionality from the legacy system can be augmented by scripts on the server, adding extra features such as data defaults, and usage logging as desired.
- Operational management overhead can be reduced, in terms of configuration control, especially if dynamic page generation techniques described later in this paper are used.

### 1.1 TYPICAL ARCHITECTURE

The interface to the legacy system is via the Common Gateway Interface (CGI) [3]. We define dynamic pages as HTML pages generated on demand with information specific to the user or request, rather than pages which have dynamically moving pictures.

Figure 1 shows the architecture and processes required to provide a Web page based GUI to existing computer applications. We define a legacy system as any generic computer/information system to which we can provide a Web based interface. This includes traditional legacy databases, as well as new systems, such as office administration tools.

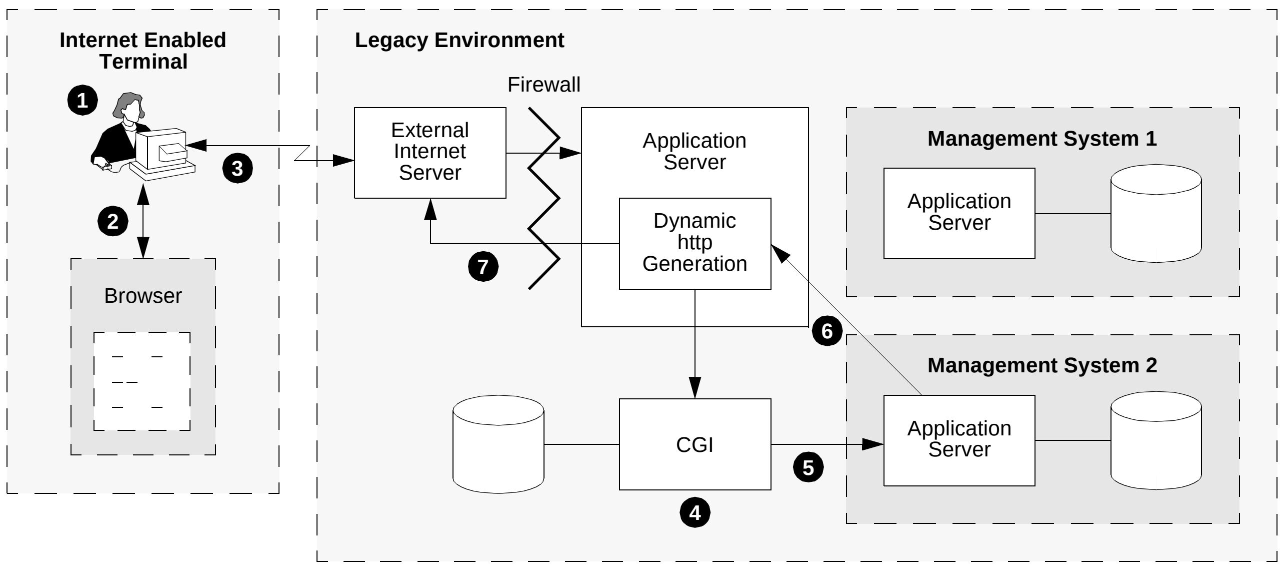

Figure 1: Legacy System Integration

The user ❶ requests a web page ❷ which interacts with a legacy system. This request is transmitted to the external web server over the internet ❸. Relevant data is extracted via CGI scripts and sent across the firewall to the application server, where a script command file is executed ❹. This re-formats the data and launches the legacy application request ❺. The response ❻ is passed through a dynamic page generation facility and *HTTP* ❼ is returned across the firewall to the external server. The new page ❷, is downloaded via the internet ❸ to the user's Web browser ❶ and the transaction is completed.

## 2 ISSUES RAISED

The major problem facing the integration of legacy systems to the Web is data integrity. This falls into three broad areas. Firstly protecting data from unauthorised users, that is, restricting access to the Web server to authorised users. Secondly protecting the data across the internet, that is, authenticating users and protection against packet sniffing. Thirdly protecting the sensitive legacy systems application data from malicious damage and/or data loss caused by erroneous or aborted transactions, that is, in short, legacy system security.

It must be noted that the security issues raised in this paper are not unique to Web/legacy integrations. Therefore, use of existing and well tested security models can be readily employed.

The first area, data protection, is essentially Web server security. There is considerable server security information available [4], which discusses ways of improving security.

The security technique most relevant to commercial use of Web pages for legacy system access, is access restriction. This includes restriction by IP address, time domain and user name/password. Typically a server can be setup via configuration files (for example, **`.htaccess`** and **`access.conf`** [5]) to restrict access to certain machines to certain users.

These methods can help to prevent attacks on the server itself. The second issue is data security over the Internet. The *HTTP* protocol sends all data unencrypted over the Internet, meaning that passwords and other sensitive data (for example, credit card numbers) are open to attacks from packet sniffers.

There exists a ‘Flock of birds’ mentality, that is, there are many eyes checking that no trouble occurs. If something does occur, then it is very likely that it will happen to someone else.

For commercial systems, this approach is too lax. A better method is to use data encryption. Secure HTTP (SHTTP or HTTPS in URLs) allows all data sent over the internet to be encrypted. As each session uses a new key, a ‘hacked’ key is only good until the session closes. SHTTP is based on the Secure Socket Layer protocol (SSL) [6]. Other methods for secure transactions include SHEN and Secure Transaction Technology (STT).

Servers which support SHTTP include Commerce Builder, Netscape Commerce, OpenMkt Secure WS and WebSTAR [7].

The third area of data integrity is the ability to recover after an erroneous transaction. Currently the log files of the Web server daemon can record the name, date and web page accessed. However, this is not enough information for site transaction management. Additional project specific logging is required to record important user data fields and transaction record numbers.

Much research and development has been made into the area of legacy system security (for example, relational database management systems provide well tested security and transaction management). Whilst highly relevant, these methods of transaction recovery are outside the scope of this paper.

## 2.1 SECURITY CERTIFICATES

Secure transport methods, such as SHTTP use public key cryptography methods, which raise the issues of key handling and trust, that is, your public key must be available, but you don't want it compromised. Security certificates provide a mechanism for increasing the level of trust.

Security certificates are a digitally signed certificate including the user's name and public key. Digital signatures ensure the security of the information by appending an encrypted summary of the message to the message itself. Thus if one byte of the message is changed, then the whole message is invalidated.

These certificates can easily be created using mathematical functions which can lead to security loop holes. Thus to increase the level of trust, Certification Authorities, such as RSA, provide a certificate signing service. These certificates include the Certification Authority's own digital signature which cannot readily be forged. To protect the clients, often browsers such as Netscape only allow connections to secure servers which have certificates signed by one of the recognised Certification Authorities.

To obtain a security certificate an application must be lodged with a Certification Authority. This application will include personal details such as name and organisation. Once this application has been verified, a digitally signed certificate is issued.

## 2.2 LEGACY SYSTEM SECURITY

A firewall [8] is used to separate the legacy system from the rest of the world. Computers inside the firewall are protected from external attack. This is achieved by having a monitored single entry/exit point for packets from the server to the legacy system. These packets are filtered to only allow specific classes of messages through to the internal network.

For computers inside the firewall who need access to external Web sites, a method to pass through the firewall is required. One method is not to filter HTTP messages. However, as HTTP includes services such as ftp, this can be dangerous.

A better method is to use an HTTP proxy server, sitting on the firewall. This proxy can forward HTTP messages between the internal network and the external server.

The CERN HTTPD [9] server can act as such a proxy, but it doesn't allow SHTTP requests to be forwarded. However, see [7] for servers which can be used as a SHTTP proxy and will then forward SHTTP messages.

## 2.3 SECURITY POLICY

For commercial applications, an enforceable security policy is essential. A security policy should include the following items.

- Enforcement of IP address restriction and password/username/time domain access on users of Web browsers.
- Non-trivial password enforcement, that is, minimum length of eight characters, etc.
- Expiration plan for passwords and accounts.
- Ensure the use of secure transport (for example, SHTTP) for all data transfers.
- Do not use default file names where possible, for example, directory access is often specified in a file with a default file name. This default should be changed in the server configuration files.
- Unusual event logging and reporting. Unusual events include repeated login failures, attempts to access ‘hidden’ files (for example, password files) and repeated occurrences of erroneous data field reception.
- Verification of input data before use. This includes checking mail aliases, field length and other situations which may indicate a user trying to gain server access via shell scripts.

- Separate file trees for internal and external Web sites.
- Move all unnecessary files out of the Web document tree so that they cannot be accessed—in particular the password files.
- Keep logged data in a secure place that is not accessible by web users.
- Use a firewall and proxy HTTPD server to isolate the legacy system from the internet.
- Enforced log-off policy to close dormant sessions.

## 3 TYPICAL APPLICATION FEATURES

No matter the actual particular functionality of an individual Web browser front end, for a legacy system, the following generic interfaces are required.

### 3.1 ADMINISTRATION INTERFACE

The application will require an administration interface, (probably web page based). This will allow the ‘application system administrator' to add or remove user accounts, enable and disable source IP addresses, modify privileges and view system performance logs, etc. on the Web server.

Additional functionality required would include web page and server management utilities. Currently tools are relatively undeveloped in this area and no means of monitoring web servers via existing management platforms exist. First generation tools are now beginning to appear, but functionality varies between server platform.

Problems with a legacy system rather than the Web server will remain difficult to detect from the server unless specific monitoring or integration utilities are developed (for example, via a Java applet).

### 3.2 USER INTERFACE

This is the service providing interface to a new or existing legacy system. With the arrival of PDAs, notebooks and wireless PC modems, as well as the increasing deployment of information kiosks, this interface will be increasingly ubiquitous. The enabling mobile technology, combined with further deregulation in 1997, increases the desirability of a Web browser based interface.

For sophisticated user interfaces, a mix of static and dynamic web page generation is required. This will allow the server to generate the user interface page (web form) at the time of the page request. A dynamically generated form can include very specific default data and/or optional fields (such as date, user name, and organisation), allowing different functionality to be exposed on the page, depending upon the requester. Performance of dynamic page generation systems currently is not an issue, as available bandwidth is the current bottleneck, and will remain so in the short term.

One advantage of dynamic web page generation systems, is that configuration management is simplified. This is because as soon as the page generation code has been updated, no HTML generation or directory management is required. Instead, when the page is next requested, the updates will be automatically included when the page is down loaded to the user's browser.

### 3.3 PUBLIC INTERFACE

This is optional, as some organisations will not make available restricted or sensitive data over public interfaces. However, for commercially offered service interfaces, a public interface, perhaps with limited functionality will be a very useful marketing tool.

A public interface can also be regarded as a restricted user interface, offering the lowest functionality. This will probably be complemented with additional marketing and pricing data and can be closely integrated with the existing public (external) web pages of an organisation.

### 3.4 LEGACY SYSTEM INTEGRATION

No matter whether the legacy system is a transaction manager, database or specific application, from the server side, the following are minimum requirements for effective integration.

### 3.4.1 APPLICATION LAUNCHING MECHANISM

Values entered into a web form need to be passed onto the legacy application, which will in be located on a different machine behind a firewall. The semantics and syntax will obviously be application specific, however as currently 80% of internet servers are Unix machines [10], it is likely that sockets will be used.

### 3.4.2 ERROR HANDLING MECHANISM

Errors can occur with the application launch, within the application, or the application may not respond after a successful launch.

The user requires a meaningful response even if the legacy system itself fails to respond. Consequently a time-out mechanism is required which traps legacy system failure. If the time-out period elapses an appropriate error message is returned to the user. Any delayed response from the legacy system will need to be handled by the server, with possible cancellation of the transaction.

If the application returns an error, then this must be passed on/expanded for the user, so that a correctly formatted query can be made. Most non-trivial applications will generate a wide variety of error messages, each of which may result in a different response for the user.

### 3.4.3 DYNAMIC RESPONSE GENERATION

Given the above situation, where there are a number of possible outcomes, successful or not, it is important that the returned user response be dynamically created by the server. This facilitates responses with specific information related to the actual requested data.

### 3.4.4 LEGACY SYSTEM APPLICATION PROGRAMMER INTERFACES (APIS)

Products are becoming available which alleviate some of the integration effort required when connecting internet pages to databases or external processes. The main focus is relational database integration, but recent developments have seen object oriented (for example, CORBA) and Network Management (GDMO) interfaces being announced in the last 2 quarters. We have recently developed JAVA applets integrating these technologies ourselves.

Another legacy integration technique used is screen scraping. Here the web server interfaces to a process which acts as a virtual terminal on the existing system. The web page user-entered data is converted to legacy application commands. Returned data is extracted and formatted into HTML before being displayed on the browser.

However, even with well defined APIs, CGI programming requires significant effort to achieve legacy system integration. Tools are essential for timely and economic development of such systems.

## 3.5 APPLICATION PAGE MAINTENANCE

Commercial sites must maintain consistency in Web pages. This is a non trivial task as it is common to have thousands of Web pages at a site, spread throughout many directories.

There are a multitude of problems facing a Web administrator. Often a sense of ordering is inherent in the pages, which leads to the concept of forward and backward links—that is, go to the next/previous page. Initially, these are easy to implement, however difficulties arise if pages are moved, inserted or deleted.

Further, the issue of standard page design and non-functional pages must be addressed. Certainly both the content and look and feel of a page are important. To have a professional and complete aura at a site, all pages benefit from a common layout, that is, same font size, background and foreground colours.

Additional features, such as company logo, common tool bars (at the top and bottom of pages) for navigation around the site and to external links, last updated field, mail to, etc., all prove useful.

These concepts of common layout are relatively easy to adhere to when only a handful of pages are required. Yet a typical site will have hundreds or thousands of pages. When these are generated by different developers and often dynamically, as a result of CGI scripts, common layout is difficult to enforce.

Often this results in pages which are non functional and boring, which leads to user frustration. This robs the site of impact and often basic hypertext functionality is not fully utilised.

A more exasperating situation for users is broken links. These are links to files which no longer exist. This can happen for a variety of reasons, such as file system or page reorganisation. Broken links reduce the user's confidence in a site and hence the appeal of the site.

## 4 EXAMPLE APPLICATIONS

A number of Web applications are described in this section. Unfortunately links to examples cannot be provided in every case, as the authors are only aware of internal implementations of these applications.

### 4.1 LIBRARY

Many Australian State library systems expect to be placed on the Web this year. These projects involve the Z39.50 standards which are concerned with information searching and retrieval techniques. This will allow web based access to national, university and electronic library catalogues which contain both printed and multimedia material—video, film clips, sound archives and cadastral (geographic) data, which previously was only available to librarians.

Simpler internal enterprise solutions involve database searching and email integration (for reservations and loan advise etc.). These applications do not necessarily integrate with existing legacy systems, although flat file searching may be integrated very easily. A public library system [11] has been implemented in New Zealand, and this demonstrates how the basic ‘legacy’ functions can be augmented with extra internet functionality.

### 4.2 OFFICE ADMINISTRATION UTILITIES

Utilities that were internal to individual organisations, such as meeting room booking, time sheets and travel/expense requests, internal phone lists etc., are often paper based. These were not computerised because the development effort required did not justify the cost saving. Also they required everyone in the organisation to be on the net, and have the same computer equipment.

Web based applications can be developed cost effectively which avoid these problems. Further, for applications local to the office intranet, security is less of an issue as employees can usually be trusted. Expect a number of in-house, public domain or cheap solutions to become available shortly.

## 5 LESSONS LEARNT

From our web/legacy integration efforts, our design goals are:

- Provide Legacy system integration.
- Consistent look amongst Web pages.
- Dynamic page generations via CGI.
- Consistent with HTML, and thus aimed at HTML literate users.
- Incorporation of security features, (such as, firewall, certificates and security policies) described earlier.

## 6 CONCLUSIONS

This paper describes the processes and some of the issues, which CiTR encountered, when we developed an web front end to existing application software. The lessons learnt in developing suites of web pages, the need for security, ease of management and integration aids focused our attention on the desirability of tools. Consequently which we have developed for use in future integration work.

## HYPERTEXT REFERENCES

**[1]** **`http://www.citr.uq.edu.au`**
CiTR's Home Page

**[2]** **`http://www.ncsa.uiuc.edu/SDG/Presentations/Stats/WebServer.html`**
Web Server Activity: Total Connections.

**[3]** **`http://www.best.com/%7Ehedlund/cgi-faq/faq.html`**
The CGI Frequently Asked Questions (FAQ) List pages.

**[4]** **`http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.htm`**
The WWW Security FAQ

**[5]** **`http://hoohoo.ncsa.uiuc.edu/docs/setup/httpd/Overview.html`**
NCSA HTTPd Server Configuration File

**[6]** **`http://www.netscape.com/info/SSL.html`**
SSL Draft specification

**[7]** **`http://www.proper.com/www/servers-chart.html`**
Web Servers Comparison

**[8]** **`http://www.auscert.org.au/Surfing_Between_the_Flags.html`**
Surfing Between the Flags: Security on the Web

**[9]** **`http://www.w3.org/hypertext/WWW/Daemon/User/Config/Overview.html`**
Syntax of CERN server configuration (rule) file

**[10]** **`http://webcrawler.com/WebCrawler/Facts/Servers.html`**
HTTP Server Distribution

**[11]** **`http://www.ccc.govt.nz/Library`**
Welcome to Canterbury Public Library

**[12]** **`http://www.hursley.ibm.com/cics/saints/index.html`**
CICS

**[13]** **`http://bscw.gmd.de`**
BSCW