Planning and Development of an Airport Group WAP Site

Alaistair Deacon, Real Time Engineering Ltd

Airport Information Services are an ideal target for WAP. Here we have a highly mobile target audience, and a range of information services and value added m-commerce opportunities that can be offered. In this case study, we will take a look at the planning and development of an International Airport Group WAP Site.

During the planning and development, we will look at other design requirements, such as the creative input, dynamic content, infrastructure, and future planned developments.

At the outset of this case study, I must point out that you should not attempt to browse WAP sites on your mobile while in the air! However, I have spoken to one airline that is considering the *possible* application of wireless technologies, such as Bluetooth, for use within the aircraft. Once we have multifunction handsets, it *may* be possible to use your mobile in the air, with the handset communicating via Bluetooth to the aircraft's satellite uplink.

Introduction

The Travel and Transport Division of Real Time Engineering (RTEL) currently provides a number of systems and solutions for the airports industry. These systems include the support and development of airport operational databases (the "on-the-day" management tools required to operate an airport) and Flight Information Display Systems (FIDS) that are used at check-in desks, gates, baggage reclaim, and all arrivals and departures displays. Linked to these systems are a number of peripheral systems, including the feeds to the television companies for the teletext pages of arrivals and departures.

When we recognized the importance of the emerging WAP technologies, the obvious application was to provide these airport information services via WAP.

There are a number of reasons why WAP is ideal for airport applications. By definition, our main target audience is mobile — that is, they are traveling and are away from their desktops. The business traveler is also typified as a high value customer, and so is suited to m-commerce offerings. Business travelers are also seen as early adopters of technology (most business travelers can be seen in the airport lounges with laptops, mobiles, and PDAs).

With these business drivers in place, RTEL pressed ahead with learning how to develop WAP services, created some early (static) prototypes of a number of services, and presented these to our clients in the airport industry. From these initial prototypes, we began the development of an airport WAP site in conjunction with one of our clients.

This paper describes the process of realizing the airport WAP site.

Planning

The planning of the airport WAP site consisted of a number of phases. These were:

- Education
- Strategy
- Product Selection

Education

The initial phase was "education". We had to educate the client as to what WAP was, what it could offer, and its limitations. This was achieved through a number of technology briefing sessions, where the team from RTEL met with the client's e-commerce team to present the capabilities of WAP.

Topics that required explanation included:

- An overview of the WAP architecture
- The details of the infrastructure required for WAP, especially details on the "ownership" of the infrastructure, with regard to who is responsible for providing the WAP gateway, and how users from any mobile network provider can access the site
- The relationship between WAP and SMS
- The capabilities of WAP
- The possible opportunities for revenue generation via WAP

By far the easiest method we found for getting the infrastructure requirements across was to use a laptop configured as an RAS server, WAP gateway, and web server all in one. We then demonstrated a WAP handset dialing into the RAS server and making a connection to a WAP site on the laptop via the WAP gateway.

To illustrate the capabilities of WAP as an application platform, we created a number of static and simple dynamic prototypes of the service that could be offered, and then led the client's team through a number of brainstorming sessions where they would propose ideas and services, and we would evaluate and advise how these could be translated to WAP.

Strategy

The second phase we went through was "strategy". We had to look at and evaluate the differing technologies that were available to provide a mobile service to passengers. This phase was carried out around February and March 2000, when it was extremely difficult to source WAP handsets in the UK and there were some questions as to the viability and the possible adoption of WAP. In hindsight now, with the easy availability of handsets, we should have persevered, but we diverted our attention briefly to SMS.

We investigated the use of a two-way SMS service to allow passengers to create SMS messages containing key words and send them to a SMS Message Center. This message center would immediately reply with an SMS message containing the required information. However, this solution would cost the airport for *every* SMS alert issued. Also, interactive SMS services are closely linked to the mobile network provider's facilities. Our client was very keen that this service should be available, in a consistent fashion, across all the mobile networks, and that a revenue stream could be generated by reverse billing to the user's handset at a premium charge per query to cover the cost of the responses.

We had a number of discussions with the various mobile networks on the viability of this, and the possibility of the reverse charging. This resulted in two main problems:

- We could not find a means by which the service could be provided in an identical fashion across all the mobile networks (due to different number translation systems, difficulty in receiving incoming SMS messages from all the networks to our server, etc.). For an international service, this would appear to be almost impossible.

- There were also large differences in the facilities provided by the mobile operators for reverse billing, making it difficult to set up the revenue stream to cover the costs.

When compared to the interactivity of WAP, the two-way SMS service seemed to be difficult to use and unattractive. We decided that we should go ahead with the development of the WAP site, and reconsider the use of SMS as an alert mechanism for the WAP service, to provide push notification of changes to flight information.

Product Selection

The last phase was "product selection". This involved understanding the mobile environment and selecting those applications and services that would best fit the WAP medium. It also included the identification of "quick wins", such as flight arrivals and connecting train times, which should be provided on an initial offering.

With the education provided, the brainstorming, and the prototype applications, our clients fully understood the capabilities of WAP applications. This allowed their e-commerce team to define their requirements for the mobile products and services to be provided.

The initial set of products and services selected for implementation were:

- About the airport
- Flight arrivals
- Flight departures
- Access to airport loyalty card points
- Train timetables
- Terminal information (which airlines use which terminals, etc.)
- Shareholder news
- Shopping information
- Useful links

Products and Services

In this section, I'll discuss further the set of products and services selected above.

About the Airport

This provides basic information on the airport operating group, and includes access to their recent press releases. The press releases were tailored from the original full press release text down to a single page per release.

Flight Arrivals

Flight arrivals were identified as one of the "quick win" applications for dynamic content. All flight information is accessible from a database maintained on the web server providing the WAP service. This includes:

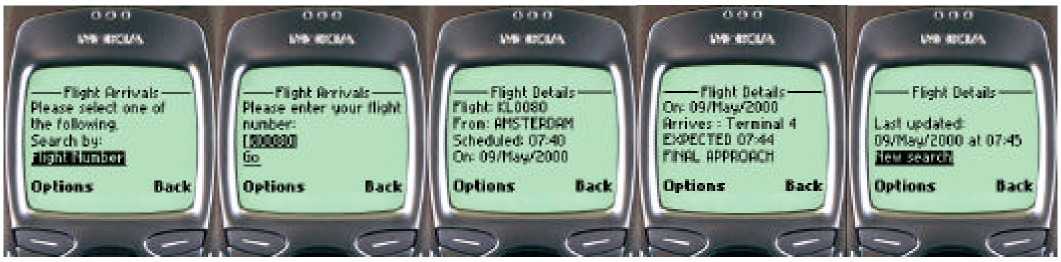
- Flight date
- Scheduled time
- Direction (arrival or departure)
- Flight number
- Airport terminal
- Route (up to four stops)
- Flight status (scheduled, estimated, landed, etc.)
- Additional information (on stand, baggage in hall, etc.)

It was anticipated that passengers would not always know the full details of the flight they were interested in. Therefore, the service was designed to allow passengers to search either by flight number or by originating airport, where both these options perform "wildcard" searches on the database. For instance, entering "BA" for the flight number would return all British Airways flights in the database. Entering "GL" is sufficient to find all flights that have "Glasgow" in any of the route points. I would add that selecting "BA" will return a lot of flights, and it is normal to search by flight number only if you actually know the correct one.

By providing a wildcard search, it is possible to have a number of flights that match the search criteria. However, if there were too many flights, the returned WML deck would overflow the memory available in a WAP handset. To overcome this, the application returns only three flights per page, with additional pages being accessed by clicking a 'more' link placed at the end of each page.

As the data for flight arrivals is derived directly from the airport's operational database, the flight status and additional information gives extremely detailed flight information. During the aircraft flight, the airline sends messages via the SITA network (basically a private network used by the airlines and airports) to provide the latest estimated arrival time. As the aircraft holds for a landing slot and lines up for final approach, it is picked up by the airport's local radar. On touchdown, and again on arrival at the stand, the control tower and the ground movement controllers update the flight status. Finally, when the baggage system delivers the baggage to the reclaim belt, the flight status is again updated.

With this level of detail, the aircraft can be tracked from estimated time to approach, final approach, touchdown, parked on stand, and baggage ready for reclaim — all via the WAP service.



```
Flight Arrivals
Please select one of
the following.
Search by:
Flight Number
Options        Back
```

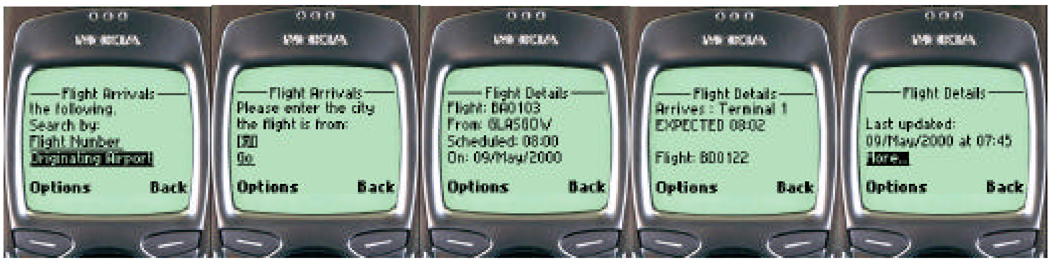
```
Flight Arrivals
Please enter your flight
number:
[illegible]
Go
Options        Back
```

```
Flight Details
Flight: KL0080
From: AMSTERDAM
Scheduled: 07:40
On: 09/May/2000
Options        Back
```

```
Flight Details
On: 09/May/2000
Arrives : Terminal 4
EXPECTED 07:44
FINAL APPROACH
Options        Back
```

```
Flight Details
Last updated:
09/May/2000 at 07:45
New search
Options        Back
```

Search by Flight Number



```
Flight Arrivals
the following.
Search by:
Flight Number
Originating Airport
Options        Back
```

```
Flight Arrivals
Please enter the city
the flight is from:
GL
Go
Options        Back
```

```
Flight Details
Flight: BA0103
From: GLASGOW
Scheduled: 08:00
On: 09/May/2000
Options        Back
```

```
Flight Details
Arrives : Terminal 1
EXPECTED 08:02

Flight: BD0122
Options        Back
```

```
Flight Details
Last updated:
09/May/2000 at 07:45
More...
Options        Back
```

Search by Originating Airport

Flight Departures

The flight departures are almost identical to the flight arrivals, except for the direction flag in the database. However, flight departures have their own unique problems that have caused some debate. This was because there are a number of factors (such as problems at the airport, with the airline, or with congestion in air traffic control) that may all contribute to a delay to a flight.

Anyone who has been involved in the aviation industry will know the complex interactions between the airport, the airline, and the air traffic control systems, with a flight often being held on the ground at the departure airport due to air traffic congestion hundreds of miles away at the destination airport. As these systems work on "slot" times, the delay situation can change rapidly by one aircraft missing their allocated slot due to a late connecting passenger, and an already-delayed flight getting moved to that earlier slot.

If the airport was to publish that a flight was late in departure, passengers may either stay away from the airport until the updated departure time, attempt to transfer to another airline or flight, or may not go to the gate for the scheduled time. If the aircraft was to get an earlier slot time due to changes in air traffic control congestion, it may not be ready due to those 'late' passengers!

In the first implementation of the WAP flight departures service, the problem of what to publish was taken away from the WAP developers, as the WAP flight departures simply displayed what the airport placed in the flight status and additional information fields of the operational database. These are used by the airport in exceptional circumstances, where it is known that there are going to be substantial delays.

However, flight departures have been identified as one of the first services that will be enhanced using "push" technology, as described in *Future Development*.

Access to Airport Loyalty Card Points

Like many airlines and airports, our airport client operates a loyalty card scheme that allows passengers to accrue points on all retail transactions, such as car parking and shopping within the airport. These points can then be redeemed against various special offers. The loyalty card scheme also provides the data required for targeted direct marketing.

This service provides passengers with news and special offers from the loyalty card scheme, and details of how to join. The service will also allow passengers to enter their loyalty card number and some form of password/PIN to access their current points balance.

However, due to the plans to provide user personalization of the WAP site, as described in *Future Development*, access to the loyalty card balance has been deferred until the WAP site personalization is complete, as this will remove any requirement to enter different usernames, card numbers, or passwords/PINs (depending on the service being accessed).

Train Timetables

Airports normally form the hub of a transport network, with trains, buses, and roads all being as important to the operation of the airport as the aircraft themselves. Our airport client also has environmental targets set in the business plan to maximize the effectiveness and use of the public transport network for getting passengers to and from the airport.

An important part of the public transport network is the regular, high speed, express train links. The WAP site allows passengers to select the airport terminal or city center train station from a list. This then presents the passenger with the next four departing train times.

Terminal Information

A large airport often has numerous terminals that are used by the different airlines. Due to the size of an airport, it is important that a passenger knows which terminal their flight operates from, as the terminals are often served by different car parks and train stations.

The terminal information service allows passengers to select their airline from a list to check the correct terminal. This list also differentiates between airlines that operate from different terminals for different destinations.

Shareholder News

This service copies the web pages for shareholder news, providing the company results, key performance indicators, and a live share price.

Shopping Information

The airport makes significant revenue from retail sales, including duty free, currency exchange, and car parking. These services are all candidates for early adoption of m-commerce applications. However, the airport already provides a call center to allow passengers to pre-order any of these services. The WAP site provides information on these pre-ordering services, as well as access to the call center number.

Useful Links

This service provides direct links to be selected: useful travel-related WAP sites, such as traffic information, news and airlines. These links will be further exploited as part of the possible partnering opportunities to service provision.

Creative Input and User Interface Prototyping

In this section, I'll look at how we worked with the airport's creative team in developing the look and feel of the user interface prototypes.

Creative Input and Design

The creative consultants used for the airport's web site were called upon to advise the e-commerce team on the style and layout of the WAP site.

Few WAP sites attempt to create any form of corporate branding, but our client had a strong desire to achieve this in some form. This was achieved by creating an introductory sequence to the WAP site, consistent navigation, and strategic graphical flashes on entry to individual products and services within the WAP site.

A common frustration with menu and link WML cards is that there are often graphics and text at the top of the card that need to be scrolled past to access the menu or links. To overcome this, the introductory WML card flashes a logo, some introductory text, and then the main menu. This is achieved with a single deck consisting of three cards that use a timer to advance to the next card. After users have selected a service from the main menu and then navigate back to the main menu, they are presented with the menu, without having to scroll past the introductory graphics and text. This is shown in the code example below:

```
<card id="home" ontimer="#page" title="Airport WAP">
   <timer value="20"/>
   <p align="center">
      <img src="logo.wbmp" alt="Airport"/>
   </p>
</card>

<card id="page" ontimer="#page1" title="Airport WAP">
   <timer value="20"/>
   <p align="left">
      Welcome to the Airport WAP Site, for all your airport and
      flight information needs.
   </p>
</card>
```

```
<card id="page1" title="Airport WAP">
   <p align="left">
      <a href="about/about.wml">About Airport</a>
   </p>
   <p align="left">
      <a href="flight/Arrivals.wml">Airport Arrivals</a>
   </p>
   <p align="left">
      <a href="points/points.wml">Check Loyalty Card</a>
   </p>
   <p align="left">
      <a href="rail/rail.wml">Rail Express Times</a>
   </p>
   <p align="left">
      <a href="terminal/terminal.wml">Which Terminal?</a>
   </p>
   <p align="left">
      <a href="shares/shares.wml">Shareholder News</a>
   </p>
   <p align="left">
      <a href="shopserv/shopserv.wml">Shopping</a>
   </p>
   <p align="left">
      <a href="#links">Useful Links</a>
   </p>
   <p align="left">
      <a href="disclaim.wml">About This Site</a>
   </p>
</card>
```

Note that Flight Departures is missing from the above menu. This service has yet to be published to the live WAP site, due to the issues described above.

Another frustration when navigating WAP sites is the lack of consistency in being able to navigate back to the main menus. Often, after accessing some pages within a WAP site, the user is forced either to enter the starting URL, or to reselect the WAP site entry bookmark after getting lost.

The `<template>` construct was used to overcome this. All the cards in a deck have the template applied to them, as shown in the code below:

```
<wml>

   <template>
      <do type="accept" label="Back" name="back">
         <prev/>
      </do>
   </template>

   ...

</wml>
```

A clean and consistent approach was required for all links. This was achieved by reviewing their wording and appearance, and ensuring that all links fitted on a single line on the most popular WAP handsets. The site was tested on as many WAP handsets and gateways as possible. This included the Nokia 7110, Ericsson R320, Motorola (Phone.com), Mitsubishi Trium, AU-Systems Palm Browser, Phone.Com UP.Simulator, Ericsson R380 Simulator, and several web-hosted WAP simulators, along with gateways from Ericsson, Nokia, and all the mobile networks in the UK providing WAP access.

To reinforce the branding of the WAP site, flash graphics were defined for all products and services that could be selected from the main menu. This was a single, relevant graphic that was flashed on entry to the product or service, in the same manner as the entry card to the WAP site.

There were a number of issues concerned with transferring the flash graphics from the creative consultants. This was mainly due to the tools (Macintoshes, Adobe Illustrator, and Photoshop) they used. To overcome this, we created a test WAP site, so that the creative consultants could e-mail us a graphic that we would convert into a WBMP for posting to the test area. These graphics could then be viewed on a WAP handset by the creative consultants. This allowed quick turnaround and iteration of the graphics, allowing the creative consultants to tailor the graphics to suit the WAP site.

Site Mapping and User Interface Prototyping

The site mapping was very much a traditional exercise, with the e-commerce team mapping out each page on lots of interlinked A1 flipcharts in a long, coffee-and-donuts-powered session.

These flipchart maps were then converted into a prototype site that was brought back to the e-commerce team, who evaluated it for usability and conformance to the creative designs set out above.

Dynamic Applications

Within the airport WAP site, there are three main dynamic applications. Early experiments with WMLScript left us disappointed with the differences that we found when accessing the site from a variety of WAP service providers. This was put down to differing levels of support for WMLScript at the WAP gateways.

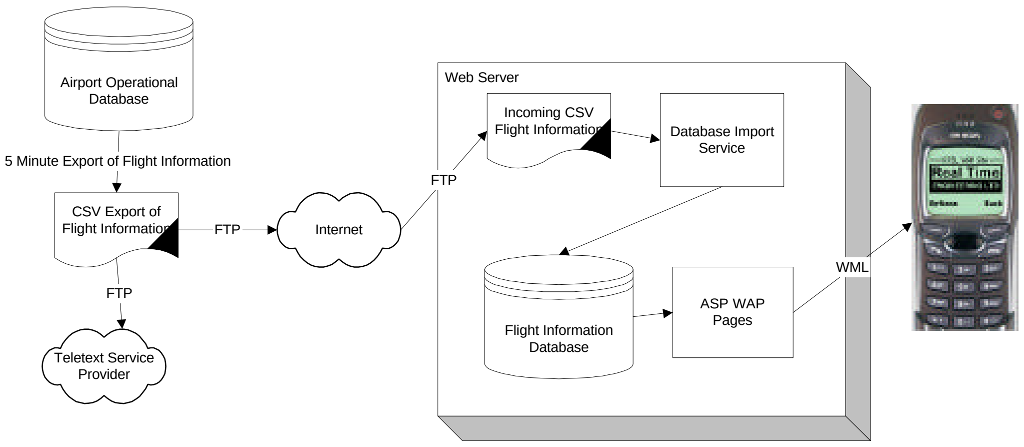
To overcome this, we decided that any dynamic applications should be achieved by scripting within our own web server, where we could maintain complete control of the WML environment generated.

As we run Windows NT as our main operating system for web servers, we decided to use ASP to generate the dynamic content.

Flight Information

The flight information consists of the flight arrivals and departures services described above.

The first task was to get the flight information from the airport's operational database to a database on the web server hosting the WAP service. Fortunately there was already a convenient mechanism in place for this. The current flight information in the airport operational database is dumped to a comma-separated file every five minutes. This file is then sent via FTP to the television companies for use within their teletext systems. To provide the flight information for the WAP site, this comma-separated file was also copied via FTP to an area on the web server hosting the WAP site. A small service written in Visual Basic was then installed on the web server to scan for this incoming file and load it into the database.


Airport Operational Database
5 Minute Export of Flight Information
CSV Export of Flight Information
FTP
Internet
FTP
Teletext Service Provider
Web Server
FTP
Incoming CSV Flight Information
Database Import Service
Flight Information Database
ASP WAP Pages
WML
Real Time

With the flight information loaded into the database, we had to create the WML cards needed for the user to be able to select the required flight information. The first deck checked for the search method:

```
<!--Search method card-->
<card id="searchmethodcard" title="Flight Arrivals" newcontext="false">
   <p>
      Up-to-the-minute flight arrival times at the Airport.
      <br/>
      Please select one of the following.
      <br/>
      Search by:
   </p>
   <p>
      <a href="arrivals1.wml#searchflight">Flight Number</a>
   </p>
   <p>
      <a href="arrivals1.wml#searchairport">Originating Airport</a>
   </p>

   <do type="accept" label="Help">
      <go href="arrivals_help.wml#searchmethod_help"/>
   </do>
</card>
```

The second deck then allowed the user to enter the selection criteria:

```
<!--Enter flight number-->
<card id="searchflight" title="Flight Arrivals">
   <p>
      Please enter your flight number:
      <input format="*M" name="flight" value="" title="Flight Num" />
   </p>
   <p>
      <anchor title="flightlink">Go
*        <go href="arrivals.asp?airport=JFK&amp; _
            flightnum=$(flight)&amp;from=&amp;counter=1" />
      </anchor>
   </p>
   <p>
      <a href="arrivals_help.wml#flight_help">Help</a>
   </p>

   <do type="accept" label="Go">
*     <go href="arrivals.asp?airport=JFK&amp; _
         flightnum=$(flight)&amp;from=&amp;counter=1" />
   </do>
   <do type="accept" label="Help">
      <go href="arrivals_help.wml#flight_help"/>
   </do>
</card>
```

```
<!--Enter source airport-->
<card id="searchairport" title="Flight Arrivals">
   <p>
      Please enter the city the flight is from:
      <input format="*M" name="source" value="" title="Origin" />
   </p>
   <p>
      <anchor title="sourcelink">Go
*        <go href="arrivals.asp?airport=JFK&amp; _
             flightnum=&amp;from=$(source)&amp;counter=1" />
      </anchor>
   </p>
   <p>
      <a href="arrivals_help.wml#source_help">Help</a>
   </p>

   <do type="accept" label="Go">
*     <go href="arrivals.asp?airport=JFK&amp; _
          flightnum=&amp;from=$(source)&amp;counter=1" />
   </do>
   <do type="accept" label="Help">
      <go href="arrivals_help.wml#source_help"/>
   </do>
</card>
```

** Note the use of the underscore '_' as a 'line continuation character'. This is purely for layout purposes; the code should occupy a single line, as it does in the accompanying material.*

The selection criteria were then passed to an ASP script that simply opens the database and formulates a SQL query from the criteria. It was important to allow the wildcard searching to minimize the user input of the selection criteria.

The ASP script then writes out the first three aircraft that match the search criteria. If there are more aircraft that match, a "more" link is created. This simply points back to the same ASP script, with an index giving the starting point in the matched flights.

First, we insert our headers and declare a few variables.

```
<%
   Option Explicit
   Response.ContentType = "text/vnd.wap.wml"
   Response.AddHeader "Cache-Control", "no-cache"
%><?xml version="1.0" ?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                     "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<%
   ' Inputs:  airport - the airport selected
   '          flightnum - flight number
   '          from - source airport
   '          counter – starting position in the recordset.
   '          counter = 1, start from first record
   '          counter = 4, start from fourth record…
   '          used for the more link.
   Dim datum, cn, sql, db, connStr, conn, rs
   Dim fn, nr, r1, r2, r3, r4, st, tm, dt, ad, te
   Dim inc
   const NFLIGHTS=3

   inc=NFLIGHTS-1
%>
<card id="results" title="Flight Details">
<p>
```

Next, open the database containing the flight information.

```
<%
   ' Change this path to point to the ODBC Datasource dsn
   connStr= "DSN=Flight"
   Set conn = server.createobject("adodb.connection")
   conn.open connStr
```

Create the root of our SQL query, formatting the dates and times as required for the output.

```
sql = "select Flight_Number, Terminal, "
sql = sql & " format(Scheduled_Date,'dd/mmm/yyyy') as Sch_Date, "
sql = sql & " Route_One, Route_Two, Route_Three, Route_Four,"
sql = sql & " format(Scheduled_Time,'HH:MM') as Sch_Time, "
sql = sql & " Status, Additional"
sql = sql & " from Flights"
```

If we are searching by airport, check all the route fields (for flights with several stops).

```
' If no flight number given, construct query to search source airport
If request.querystring("flightnum") = "" Then
  sql = sql & " where (Route_One like '" & request.querystring("from") & "%'"
  sql = sql & " or   Route_Two like '" & request.querystring("from") & "%'"
  sql = sql & " or   Route_Three like '" & request.querystring("from") & "%'"
  sql = sql & " or   Route_Four like '" & request.querystring("from") & "%')"
End If
```

Else check the flight number requested.

```
' If no source airport given, construct query to search flight number
If request.querystring("from") = "" Then
   sql = sql & " where Flight_Number like '" &
         request.querystring("flightnum") & "%'"
End If
```

Complete the SQL query and open the recordset.

```
sql = sql & " and Direction = 'A' and Airport = '" & _
      request.querystring("airport") & "'"
sql = sql & " order by Scheduled_Date, Scheduled_Time,"
sql = sql & " Flight_Number"
Set rs = conn.execute(sql)
```

Begin by looping through the recordset to get to the first flight to be returned. For instance, on the first query, counter = 1 — that is, we start returning from the first flight found. When a user selects a "more" link for the next page of flights, counter = 4 — that is, we loop through to the fourth flight before returning the flight details.

```
If Not rs.eof Then
   cn = 1
   Do While Not rs.eof
      If cn >= cint(request.querystring("counter")) And _
         cn <= cint(request.querystring("counter"))+inc Then

         ' Extract each field into local vars
         fn = rs("Flight_Number")
         dt = rs("Sch_Date")
         r1 = rs("Route_One") & ""
         r2 = rs("Route_Two") & ""
         r3 = rs("Route_Three") & ""
         r4 = rs("Route_Four") & ""
         st = rs("Sch_Time")
         tm = rs("Status") & ""
         ad = rs("Additional") & ""
         te = rs("Terminal") &""
```

Build the WML from the extracted flight information.

```
         ' Build the return string
         datum = "Flight: " & fn & "<br/>"
         If r2 = "" Then
            datum = datum & "From: " & r1 & "<br/>"
         Elseif r3 = "" Then
            datum = datum & "From: " & r1 & "<br/>"
            datum = datum & "Via: " & r2 & "<br/>"
         Elseif r4 = "" Then
            datum = datum & "From: " & r1 & "<br/>"
            datum = datum & "Via: " & r2 & ",<br/>"
            datum = datum & r3 & "<br/>"
         Else
            datum = datum & "From: " & r1 & "<br/>"
            datum = datum & "Via: " & r2 & ",<br/>"
            datum = datum & r3 & ",<br/>"
            datum = datum & r4 & "<br/>"
         End If
         datum = datum & "Scheduled: " & st & "<br/>"
         datum = datum & "On: " & dt & "<br/>"
         If request.querystring("airport")="JFK" Then
            datum = datum & "Arrives : Terminal " & te & " <br/>"
         End If
         If tm <> "" Then
            datum = datum & tm & " <br/>"
         End If
         If ad <> "" Then
            datum = datum & ad & " <br/>"
         End If
         datum = datum + "<br/>"

         ' Output return string
         response.write datum
      End If

      ' Increase the local counter
      cn = cn + 1
      rs.movenext
  Loop
  rs.close
  Set rs = Nothing
```

Now add the last updated time to give the user additional confidence that the information returned is accurate.

```
      sql = "select format(Date,'dd/mmm/yyyy hh:mm')"
      sql = sql & " as Last_Date from Lastupdate"
      Set rs = conn.execute(sql)
      If Not rs.eof Then
         dt = rs("Last_Date")

         ' Build the return string
         datum = "Last updated: " & dt & "<br/>"

         ' Output return string
         response.write datum
      End If
      rs.close
      Set rs = Nothing

      conn.close
      Set conn = nothing
```

If there are more flights that can be displayed, add a "more" link starting from the flight after the last flight displayed, along with a new search link. Otherwise, just add a new search link.

```
      ' More flights to be displayed? If so, generate "More" link and option.
      If cn > cint(request.querystring("counter"))+inc+1 Then
         response.write("<anchor title=""morelink"">More...")
         response.write("<go href=""arrivals.asp?airport=" & _
            request.querystring("airport") & "&amp;flightnum=" & _
            request.querystring("flightnum") & "&amp;from=" & _
            request.querystring("from") & "&amp;counter=" & _
            cint(request.querystring("counter"))+3 &"""/>")

         response.write("</anchor></p><p>")
         response.write("<a href=""arrivals.wml"">New search</a></p><p>")
         response.write("<do type=""accept"" label=""More..."">")
         response.write("<go href=""arrivals.asp?airport=" & _
            request.querystring("airport") & "&amp;flightnum=" & _
            request.querystring("flightnum") & "&amp;from=" & _
            request.querystring("from") & "&amp;counter=" & _
            cint(request.querystring("counter"))+3 &"""/>")
         response.write("</do>")
         response.write("<do type=""accept"" label=""New search"">")
         response.write("<go href=""arrivals.wml""/>")
         response.write("</do>")
      Else
         response.write("<a href=""arrivals.wml"">New search</a></p><p>")
         response.write("<do type=""accept"" label=""New search"">")
         response.write("<go href=""arrivals.wml""/>")
         response.write("</do>")
      End If
```

If no flights were found, then tell the user.

```
   Else
      If request.querystring("from") <> "" Then
         response.write("No flights from " & request.querystring("from") & ".")
      Else
         response.write("Flight " & request.querystring("flightnum") & _
            " does not exist.")
      End If
   End If
%>
```

Finish off the WML card.

```
<do type="prev" label="Back">
   <prev/>
</do>
</p>
</card>
</wml>
```

A problem occurred with caching of pages when checking frequently for flight updates. This was due to the WML pages generated by the ASP scripts being cached in the WAP handset. This was overcome by adding a cache control header to ensure that the page was not cached.

Train Timetables and Terminal Information

The train timetables and terminal information services use very similar solutions to the flight information shown above. The ASP scripts are included in the accompanying material.

The train timetables database is not updated in real time, as the service has a fixed timetable. However, the ASP script uses the current server date and time to retrieve the next four train times due to depart on the current day.

Infrastructure and Hosting

In this section, I'll look at how we provided users with access to the WAP site, and how it was hosted.

WAP Access

The WAP site is to be accessed by the public, so no attempt was made to provide the WAP access infrastructure, such as the RAS servers and WAP gateway. Our experience tells us that most users would not be able to enter new gateway settings into their handsets. Also, many handsets only have one set of gateway configuration settings, and these are often locked down by the mobile network provider. We anticipated that the users would access the WAP services provided by their mobile service provider and either access the site via a third party portal, or enter the site URL directly into their bookmarks.

Hosting

The WAP site is hosted on one of our own NT web servers running IIS, which is used to host a number of different web sites as well. This is connected via a firewall to a 2Mbps leased line to the Internet.

Test Test Test

As an ISO9001 registered software house, quality plays a significant part in our service provision. The importance of testing a WAP site cannot be overstressed for a number of reasons. Users are likely to be unfamiliar with the Internet, there is limited space for displaying error messages (even if a WAP gateway tries to convert an HTML 404 file not found error), and they are likely to abandon your site as there is very little opportunity for them to get past the error (unlike a web site where they may click refresh, or select another link on your site).

The WAP site should be tested with as many handsets as possible, as well as other browsers such as PDAs and web based emulators. Additionally, the WAP site should be accessed from as many WAP service providers (and their associated gateways) as we can find.

Future Development

This section outlines a number of ideas that are being proposed as enhancements to the WAP site.

User Pre-registration and Personalization

An important part of any e-commerce strategy is to maximize the market knowledge of users accessing the service. However, it is unrealistic to expect users to go through a complete registration process on a WAP site before accessing the service.

The proposed solution to obtain this user information is via user pre-registration on the associated web site. This user registration will capture the user's lifestyle and preferences before leading them to the details of the WAP site URL and information on configuring the most common handsets.

An important proposed function of web site pre-registration would be to obtain the user's credit card and billing details for any value added services and m-commerce applications that may be offered via the WAP site. It will be easier to obtain this information via a secure web site, than by asking for the users credit card details over WAP. This also helps overcome any of the associated security fears of passing credit card details over the air.

On receipt of the users pre-registration information, they would be returned a username and associated PIN number.

With the user's pre-registration information, it would then be possible to personalize the WAP service and make it easier to complete an m-commerce transaction. The issue is to make these services easy to use.

The last thing a user will want to do is enter their username and PIN every time they access the WAP site. The obvious solution to this would be for a user to enter their information once and to store this as a cookie on the WAP handset. One big problem with this is that we do not have cookie support in WAP (although some WAP Service providers are providing mechanisms for storing cookies at the WAP gateway). Additionally, if a handset was lost or stolen with the cookie information on it, then it would be possible for a thief to make unauthorized m-commerce transactions.

What is required is a simple mechanism to identify the user, with additional confirmation of their identity when committing an m-commerce transaction.

The preferred model to achieve this is to have the WAP site accessible to *all* users, without having to enter their user name and PIN. However, an option will be provided for the user to activate the WAP site personalization. This will take a user to a WAP card where they will enter their username and PIN obtained during the pre-registration process. This username and PIN will be validated against the pre-registration details, and if successful, the user will be informed that they will be redirected to a new menu that should be bookmarked.

The user will then be redirected to a URL that includes the username as some form of index, using techniques such as GUIDs (Globally Unique Identifiers). For example, the user may register at the following URL:

```
wap.airport.com/register.wml
```

They will then be re-directed to

```
wap.airport.com/index.asp?user=ABCD123456789
```

where ABCD123456789 is an identifier for the user. This is the new URL that is then stored in their bookmark.

The next time the user accesses the site via the bookmark, the site will provide the same basic services, except that this time we will know the user who is accessing the service.

What's still required, though, is some form of confirmation of the user when accessing a premium service or m-commerce transaction. This is achieved by requesting the user to re-enter their PIN when accessing these services. It is felt that a simple, 4-digit PIN should not overly inconvenience users. The username (which is passed as a parameter in the URL throughout the site), when combined with the conformation PIN, will confirm the identity of the user.

The final security barrier is the fact that no credit card details are being passed over the air, and even if the username ID and PIN were intercepted, they could only be used to order goods and services that are delivered to the user's phone or home/business address, etc.

Push Messaging

An airport is a dynamic and changing environment. Users need to be notified of any flight delays or events that may affect their travel plans. We will need to wait until the implementation of the WAP 1.2 push architecture to provide this notification via WAP; in the meantime, we have to use the existing SMS services to provide this service.

It appears to be reasonably simple to set up a link with either a mobile network operator or a third party SMS message provider to send these messages over mechanisms such as X.25 or SMTP mail. However, the problems encountered relate to the cost of providing this service.

Costs appear to vary between 4p and 10p per message sent. Initially, it seems that the airport would have to cover these costs. However, the airport would like to see this as a revenue *generation* opportunity. This can be done in a number of ways, which I will now elaborate on.

Reverse Billing

With a reverse billing solution, the cost of the SMS message would be forwarded to the receiver's mobile phone bill. This could be a premium charge per message sent, covering the costs for the SMS service provider, and providing a revenue stream for the airport.

The difficulties encountered on attempting a reverse billing solution are that agreements need to be set up with *all* the mobile network operators, as they appear to be unwilling to address reverse billing of SMS messages across the different networks. Additionally, it would appear that most of the mobile networks' billing engines are insufficiently developed to achieve this.

Charge to Credit Card

It should be possible for the airport to charge the costs of using this service to the user's credit card entered during pre-registration. However, there are significant overheads in doing this, including the possibility of generating numerous, low value credit card transactions that would be uneconomic to process. Additionally, it is anticipated that this could cause severe frustration to business travelers who would have to claim for many small credit card transactions in expenses, instead of the costs being billed back to the phone bill that would be paid for by their employer.

Sponsorship

There is an opportunity to use the SMS messages as an advertising medium, allowing an airline to sponsor the service. There are several good reasons for an airline to be interested in this opportunity. Firstly, the messages are extremely targeted to a specific traveling passenger. Additionally, if you have just received a message that your flight with airline XYZ has been cancelled or delayed, and there is a message at the bottom of the SMS from airline ABC, then the passenger may very well want to change or re-book their flight with the latter. Also, associated companies (such as car hire, etc.) may be equally interested in the sponsorship opportunities.

m-Commerce Opportunities

Within the airport environment, there are a number of m-commerce opportunities that can be identified. This includes the pre-ordering of foreign currency, duty free shopping, and car parking.

These services match well with the user pre-registration described above. Passengers can simply select the items that they wish to pre-order from the WAP site and confirm their transaction by re-entering their PIN. The duty free goods or currency is then pre-picked and prepared at the airport, ready to be picked up by the passenger. There is an additional security opportunity here, in that the passenger can actually present their credit card to be validated against the details entered during pre-registration.

Partnering Opportunities

Finally, there are significant partnering opportunities here too. The model of generating advertising revenue from banner adverts is already well established on the Web, even though many of them are either ignored or followed merely due to curiosity, with a low probability of getting a revenue transaction. However, the airport WAP site presents a very different user.

If a user hits the airport WAP site for flight arrival information, is presented with a link to a partner car hire firm, and follows that link, they are far more likely to complete a transaction — since we can already be sure that the user is traveling and interested in aircraft arrival times. They are therefore significantly more likely to take the car hire offered.

By ensuring that any partnering opportunities are tailored and specific to the traveler, the use of short "advert links" will be tolerated on the WAP site — just as long as they can be viewed as giving value to the traveler. (Please don't try to sell someone a mortgage when they are checking flight arrival times — I just don't think that would work!)