

# WAP-Enabling SAS® Applications

*S. David Riba, JADE Tech, Inc., Clearwater, FL*

## ABSTRACT

WAP, Wireless Application Protocol, is an exciting new technology which has significant capabilities to extend SAS® applications across thin client interfaces to portable devices. With WAP technology, SAS applications can now be totally portable, available to end users via cell phones or Personal Digital Assistants (PDAs) such as cell phones or the PalmPilot. This presentation will explore WAP and discuss the requirements necessary to make SAS applications easily accessible to WAP-enabled devices. It is an introductory paper intended for intermediate to advanced SAS programmers.

## INTRODUCTION

As the state of technology has evolved, demands on information have increased dramatically. It is no longer acceptable to wait weeks, days, or even hours for a report. Our organizations demand instant access to information. We have moved from the mainframe, to the desktop, to the laptop. We carry our offices with us when we travel. Now, even that is not good enough. We want instant access to our data 24 hours a day, whether we are in our offices, in our homes, or on the road.

Over the past few years, cellular phones and Personal Digital Assistants (PDAs) have changed the world dramatically. We can now be in communication with the outside world at all hours of the day or night, even when on vacation. The world wide web has made access to information a global phenomenon. However, until recently cell phones and PDAs were unable to communicate with the world of information accessible over the web.

This has all changed with the advent of WAP. Wireless Application Protocol was designed to allow any WAP enabled device, primarily cell phones and PDAs, to communicate with the world wide web and access information across the web. This information could be a web page, a web search, a database query on your office computer, or even the ability to run a SAS application remotely.

Using WAP, it is now possible to enter or retrieve information from your data warehouse or other SAS

application using nothing more than a cell phone or a PDA. Since cell phones provide a very limited "window" and can only display small amounts of information at one time, WAP has only had limited success in cell phone implementations. However, with a larger "window" of information, PDAs such as the PalmPilot or Handspring Visor are much more suitable for WAP implementations.

This presentation will concentrate on PDAs as a data entry / retrieval device. It will use the Palm VILx for demonstration purposes. However, the techniques are valid for any WAP-enabled device. This paper will provide an introduction to WAP, discuss some of the key elements needed to understand WAP, and demonstrate some SAS applications can be used with WAP.

## WHAT IS WAP?

WAP is not a product. It is a technology standard that defines an industry-wide specification for developing applications over wireless communication networks. The WAP Forum (or more properly W@P Forum) was founded in the summer of 1997 by Ericsson, Nokia, Motorola, and Unwired Planet. For more information about the WAP Forum and the WAP specifications, visit [www.wapforum.org](http://www.wapforum.org)

The WAP specification defines an application framework and network protocols for wireless devices such as mobile telephones, pagers and PDAs. There are many constraints in this wireless networking environment compared to wired networks. Wireless devices must contend with:

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

Accordingly, WAP enabled devices must be able to deal with these issues. In addition, unlike desktop computers, most handheld devices have limited on-board processing capabilities. Because these are battery powered devices, they were never designed for the intensive computational capabilities of larger computers. These devices also have very limited memory capabilities, so memory intensive tasks, such as rendering 3-D graphics would be problematic.

## WEB CLIPPING OR WEB BROWSING?

One of the key problems that WAP must address is the issue of bandwidth. With the improvements in communication speeds, web pages have become more complex, have more graphics, and have generally become bigger in size. As such, they require faster communication speeds to move all of that information in a reasonable time frame. It was not too many years ago when we were communicating at speeds of 14.4 or 28.8. Now, 56k modems and DSL lines are fairly universal. Imagine viewing a typical web page with all of its graphics and content at the communication speeds of a 14.4 modem. Now imagine piping all that information to a cell phone or PDA. The transmission time to move all that information would be totally unacceptable to most people.

The other key problem that WAP addresses is the issue of "window" size. Picture a typical cell phone or PDA display. It is much smaller than your computer monitor, and has a much more limited footprint with which to display information.

There are two approaches which can be used to resolve these two problems, web clipping or web browsing. Web clipping is a technique which uses a small program that resides on the PDA to format a query that is transmitted to a server. The results of the query are extracted and returned to the PDA to be displayed. The alternative is web browsing, which uses a specially designed page that matches the smaller form factor of the device.

## WHAT ARE WEB CLIPPINGS?

Web clipping applications (Palm calls them Palm Query Applications) are small compiled programs that are loaded onto the PDA device. These applications create a query that is sent from the device to an internet server. The web clipping is an HTML page that answers the query and is returned to the PDA device. These clippings can either be static pages of information or small pages generated by a CGI script from the user's query.

In order to create Palm Query Applications (PQAs), Palm provides a Web Clipping Application Builder which runs on Windows 95, Windows 98, Windows NT, and Mac OS (System 7.x or better). The Web Clipping Application Builder is used to compress and compile web clipping applications.

While PQAs and clippings can achieve similar results to web pages, CGI scripts, or Java servlets, there are constraints imposed by the limitations of handheld devices. Besides the smaller display area and the slower transmission speeds, developers

must also consider that the cost of airtime for connecting a handheld to the wireless network is significantly higher than land-based alternatives. As such, web clipping applications must be designed to minimize transmission time, better utilize server processing capabilities, and to format output specifically to the type of device requesting the information. For example, it would be impractical to download an entire database to a PDA. However, it is possible to generate queries on the PDA to extract and display information from a server based database. Another example would be using a PDA to update a server based database. In this case, it would be more practical to have the application gather and validate information on the PDA, and then transmit the entire record to the server with instructions to update the server based database.



## WHAT ABOUT WEB BROWSING?

While it is possible to view existing HTML documents over the wireless web, for the reasons cited above it is impractical to do so in most instances. It would be preferable to design web pages or query output specifically for the limitations of the PDA or cell phone interface. As stated above, it is essential to consider a clean interface optimized for the limited viewing window and the relatively slower transmission speeds.

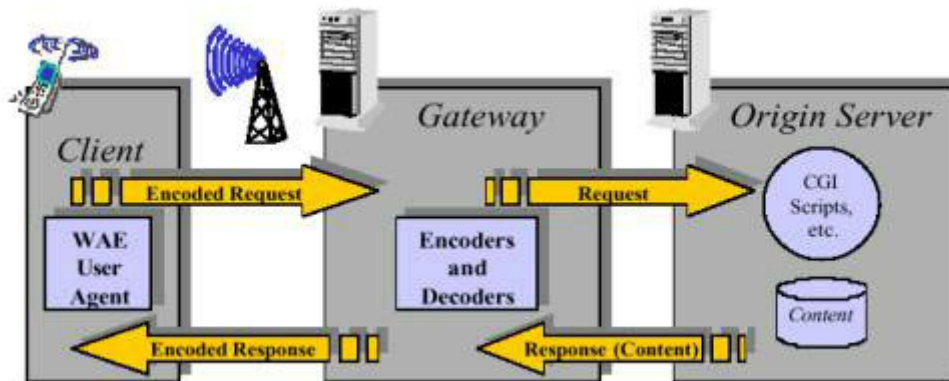
However, once those limitations are considered, you are not limited to just browsing an existing web page. The SAS programmer can develop applications that accept input from the PDA, process that input, and return the results back to the PDA. The rest of this paper will look at the elements of WAP and the various methods that SAS can use WAP to perform these tasks.

## OVERVIEW OF WAP ARCHITECTURE

WAP functions in an environment that has been designed as a combination of the World Wide Web and Mobile Telephonics technology. The Wireless Application Environment (WAE) closely follows the World Wide Web model. All content is specified in formats that are similar to the standard Internet formats. Content is transported using standard protocols in the WWW domain and an optimized HTTP-like protocol in the wireless domain. WAE has borrowed from WWW standards including authoring and publishing methods wherever possible. The WAE architecture allows all content and services to be hosted on standard Web origin servers that can incorporate proven technologies such as CGI. All content is located using WWW standard URLs.

The Wireless Application Environment assumes the existence of gateway functionality to encode and decode data transferred between the mobile client and the server. The purpose of encoding content delivered to the client is to minimize the size of data sent to the client over-the-air as well as to minimize the computational energy required by the client to process that data.

Below is a generalized overview of the Wireless Application Environment used by WAP.



Client applications reside on a WAP enabled device such as a cell phone or PDA. These devices send queries thru a gateway server, such as Palm's PALM.NET, which then forwards the query to the destination server. Responses are routed back through the gateway server and returned to the original querying device.

The Wireless Application Environment includes support for the following functionality:

- Wireless Markup Language (WML) – a lightweight markup language, similar to HTML, but optimized for use in hand-held

mobile terminals, such as mobile telephones and PDAs

- WMLScript – a lightweight scripting language, similar to JavaScript™
- Wireless Telephony Application (WTA, WTAI) – definitions for telephony services and programming interfaces
- Content Formats – a set of well-defined data formats, including images, phone book records and calendar information

It is important to understand that, like the Internet model WAP is based on, the client device (either cell phone or PDA) only needs a micro-browser, while all content and applications are hosted on Web servers.

## WHAT IS WML?

Most web publishing has traditionally been done using HTML, HyperText Markup Language. Recently, XML, or eXtensible Markup Language, was introduced as a foundation for other specialized Markup Languages. While HTML is intended to describe page layout issues such as fonts and formatting, XML is designed specifically to manage the page content. One of the specialized languages

derived from XML is WML (Wireless Markup Language). WML has been designed to address the needs of communicating with the wireless web via handheld devices.

In addition, like the relationship of HTML to JavaScript™, WML has a lightweight scripting language called WMLScript.

A WML-aware browser is required to read and display WML-based content. There are several commercial and shareware WML browsers currently available.

At this time, to the best of the author's knowledge, Microsoft's Internet Explorer 5.50 does not support XML or WML. However, it appears that Netscape Navigator 6.0 might support WML. However, this has not been tested or confirmed.

## WML BASICS

Instead of Web pages, the wireless world uses the metaphor of decks consisting of individual cards. Here is an example of a static WML deck:

```
<?xml version="1.1"?>
<wml>
  <card>
    <p>
      Welcome to WML world!
    </p>
  </card>
</wml>
```

It is possible to create applications based only on static WML decks, but real-time information can only be provided to users by creating dynamic WML decks. This can be achieved by using CGI, the Java language, ASP, or any scripting language that can query databases dynamically and format the output to conform to WML standards. In effect, dynamic WML programming is the key to database-driven wireless connections.

WML is based on XML, and is specifically designed for use on narrowband devices, such as cell phones, pagers, or PDAs. If you have an understanding of HTML, you should have a solid foundation to understand WML.

The WML specification includes four major functional areas:

- ⇒ Text Presentation and Layout (formatting, text attributes, image support)
- ⇒ Deck card organizational metaphor (all information is organized into a collection of cards and decks)
  - ⇒ Cards specify one or more units of user interaction (menus, text screens, data entry fields)
  - ⇒ Decks, which are groupings of cards (similar to an HTML page)
- ⇒ Inter-card navigation and linking (navigational links between cards, anchored links, or scripts)
- ⇒ String parameterization and state management (variables can be used in place of strings and substituted at run-time)

Since WML is an XML language, it inherits the XML document character set, currently the Universal Character Set (Unicode 2.0).

XML (and WML) is designed to describe not only the formatting but also information about the content of a part of the page. It does this with *element* tags and

*attribute* tags. Element tags specify the markup and structural information about a WML deck. They may contain a start tag, content, and an end tag. They have one of two structures:

⇒ With content: **<tag> content </tag>**

⇒ Without content: **<tag/>**

Unlike HTML, which is only concerned with document formatting and display issues, XML and WML documents are self-describing. The element tags describe content and they can be read and understood by countless applications. Effective use of these element tags makes the document effectively an “intelligent” document.

The second structure above, element tags without content, reference tags such as the BREAK tag, which would appear in a WML document as **<BR/>**. There is no “content” or other information worth passing to an application when a line break is called for, so this construction defines the break, but identifies it as a non-content tag.

WML attribute tags define additional information about an element. Specifically, they define information about an element that is not part of the element’s content. Attribute tags are always specified in the start tag of an element. For example,

⇒ **<tag attr="abcd">**

Attribute names are XML names and they are case sensitive. All attribute values must be quoted with either single or double quotation marks. Character values can be included in an attribute value

**<WML>** is one of the elements. It defines the start of a WML deck, which can be considered similar to the **<BODY>** tag in HTML. It defines the beginning and ending of the web page. **</WML>** is the ending tag for this element. The content between the **<WML>** and **</WML>** tags is called a *deck*. The entire deck is loaded from the server and cached in the device’s memory. The only time that it is necessary for the device to connect back to the server is when the user requests a new deck.

The **<CARD>** tag defines the beginning of a new card. Every deck must contain at least one card, but typically decks contain many cards. Navigating between cards in a deck does not require a connection back to the server.

In the example below, there is a hyperlink defined in the first card which, when selected, points to the second card. Users navigate between cards in a

deck using links in the card definition, such as the following:

```
<wml>
  <card>
    <p>
      <do type="accept">
        <go href="#card2"/>
      </do>
      Hello world!
      This is the first card ...
    </p>
  </card>

  <card id="card2">
    <p>
      This is the second card.
    </p>
  </card>
</wml>
```

When this deck is loaded, the first card is automatically displayed. The first card contains the text "Hello World! This is the first card ..." and a link to the second card. When the user selects the link defined in the DO element, the second card is displayed. Only one card can be viewed at a time.

The Document Prologue is the header that must exist in all valid WML files. It is based on the XML header declaration, and looks like this:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
"-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org
/DTD/wml13.dtd">
```

A Document Prologue header must be included at the beginning of every WML file. Part of the header definition is the version of XML (1.0) and the URL location for the Document Type Definitions (DTD)

For a complete description of the WML specification, the reader is referred to the Wireless Application Protocol Wireless Markup Language Specification. It is available from the WAP Forum, online at [www.wapforum.org](http://www.wapforum.org).

## SAS WIRELESS TECHNOLOGIES

The current release of SAS software supports WAP in several different ways. Version 8.02, which has just been released, provides the following support for WAP applications:

- ⇒ WML output destination in ODS
- ⇒ SAS IntrNet
- ⇒ AppDev Studio 2.0

The Output Delivery System now includes a WML output destination to create WML files directly from a SAS program. SAS IntrNet has new features to handle deployment issues of WAP applications. AppDev Studio Release 2.0 has been enhanced to ease the development of WAP applications. Each of these areas will be discussed below.

### New ODS Output Destination - WML

Just introduced in Release 8.02 of the SAS System, the Output Delivery System (ODS) now includes support for WML as an output destination. The WML output destination is still experimental in Release 8.02.

The syntax for the WML output destination is consistent with the other output destinations:

**ODS WML file = 'output-filename.wml' ;**

Consistent with ODS syntax, once the WML destination is opened, it will remain open until closed. All output generated will be sent to the WML file until the destination is closed with the statement:

**ODS WML close ;**

Without ODS, WML can still be written the old-fashioned way with a DATA \_NULL\_ step and PUT statements. Below is a simple Hello World from the SAS Sample Library:

```
data _null_ ;
  file _webout;
  old = appsrv_header('Content-type',
'text/vnd.wap.wml');
  put '<?xml version="1.0"?>';
  put '<!DOCTYPE wml PUBLIC
"-//WAPFORUM//DTD WML 1.1//EN"@;
  put ' "http://www.wapforum.org
/DTD/wml_1.1.xml">';

  put '<!-- A Simple Hello World Deck
-->';
  put '<wml>';
  put '<card>';
  put '<p>';
  put 'Hello Wireless World!';
  put '</p>';
  put '</card>';
  put '</wml>';
run;
```

However, using the WML output destination in ODS the same program would look like this:

```
ods wml file=wapdemo ;
data _null_ ;
  file print ;
  put 'Hello Wireless World !' ;
```

```
run;
ods wml close ;
```

ODS creates a file that looks like this:

```
<?xml version="1.0"
encoding="windows-1252"?>
<!DOCTYPE wml PUBLIC
"-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org
/DTD/wml_1.1.xml">

<wml>
<template>
  <do type="accept"
label="Back">
    <prev/>
  </do>
</template>
<card>
</card>
<card id="IDX">
<p>The SAS System</p>
<p>
<table columns="1">
<tr><td>Hello Wireless World !
</td></tr>
</table>
</p>
</card>
</wml>
```

Used in this way, WML can be used for either static reports or as part of more dynamic applications.

### SAS IntrNet

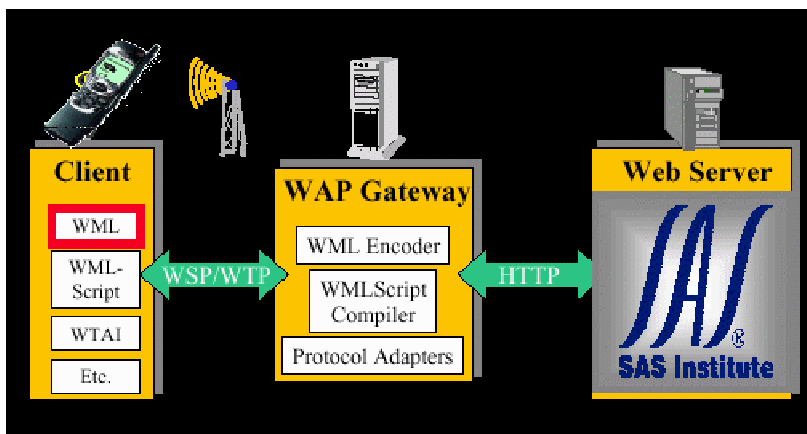
The Application Dispatcher has been updated to include support for several new entry types and output destinations, include XML and WML.

### AppDev Studio 2.0

AppDev Studio is the primary development environment within SAS for creating Java-based applications for wireless devices. Just introduced in Release 2.0, AppDev Studio now has WML Transformation Beans integrated into the webAF development environment. These Beans allow for the automatic generation of Java Servlets and JSP (Java ServerPages) that serve up WML content.

In addition, AppDev Studio 2.0 now includes "Intelligent Page" TransformationBeans. These iPage beans are intended to create output pages in either WML, HDML, or HTML for WAP devices.

With the new additions to AppDev Studio 2.0, it is now possible for the SAS developer to create thin-client applications that communicate between a wireless device and SAS running on a webserver.



### DOING WAP - THE SAS® WAY !

The above illustration depicts the WAP environment and how SAS fits into that environment. Static or dynamic SAS applications can run on a variety of client devices, such as cell phones, pagers, or PDA's. They can communicate across the WAP gateway to web and data servers located virtually anywhere. For example, a SAS IntrNet webserver can run a SAS query against a corporate data warehouse, and return the results to the requesting client device.

With the increased emphasis on Wireless Technologies in SAS Version 8.02, SAS software developers now have a variety of tools at their disposal to create applications for the wireless web. From the simple to use WML output destination in ODS to the robust set of WML classes in AppDev Studio 2.0, it is now possible to generate fully portable SAS applications.

The range of applications possible across the wireless web is limited only by the SAS developer's imagination and talents. Many different types of remote wireless applications are now possible, from simple updates to a SAS dataset through remote queries to a data warehouse to a CEO's morning reports delivered to a cell phone or PDA.

One interesting footnote is that, even though cellular phones are not allowed to be used while an airplane is in flight, WAP-enabled PDAs such as the Palm Pilot or Handspring Visor **can** be used in flight. This leads to the interesting, but untested, potential to use a PDA while enroute to a destination to communicate with a WAP server and run SAS programs remotely while en route to a final destination.

## CONCLUSION

This paper has attempted to provide a foundation in WAP, the Wireless Application Protocol. With an understanding of WAP and WML, the Wireless Markup Language, it is possible to create thin-client SAS-based applications that are accessible anytime, and virtually anywhere. SAS Wireless Technologies have simplified the task of developing and deploying these types of applications.

## TRADEMARK INFORMATION

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

## REFERENCES

Lee Min En, Laurence "Introduction to the Wireless Application Protocol (WAP) Architecture"  
<http://www.fit.qut.edu.au/DataComms/itn540/gallery/a100/lee/INTROD~1.HTM#intro>

Palm Computing, Inc. "Web Clipping Developer's Guide" <http://www.palm.com/devzone>

SAS Institute, Inc., *SAS® webAF Help, Version 2.0*  
Cary, NC. SAS Institute Inc., 2001

Torr, Mark and Benson, Cory "Wireless Information Delivery Applications" Proceedings of the Twenty-Fifth Annual SAS® Users Group International Conference, Cary, NC: SAS Institute Inc., 2000.

Web ProForum, "Web ProForum Tutorial: WAP",  
<http://www.webproforum.com/wap/topic01.html>

The WAP Forum, <http://www.wapforum.org>

The WAP Forum, "Wireless Application Protocol Wireless Markup Language Specification Version 1.3" 2000 <http://www.wapforum.org>

## AUTHOR

The author may be contacted at:

S. David Riba  
**JADE Tech, Inc.**

P O Box 4517  
Clearwater, FL 33758  
(727) 726-6099

INTERNET: [dave@JADETEK.COM](mailto:dave@JADETEK.COM)

This paper is available for download from the author's Web site:

[HTTP://WWW.JADETEK.COM/PUBS.HTM](http://WWW.JADETEK.COM/PUBS.HTM)

## AUTHOR BIO

S. David Riba is CEO of JADE Tech, Inc., a SAS Institute Quality Partner who specializes entirely in applications development, consulting and training in the SAS System. Dave has presented papers and assisted in various capacities at SUGI, SESUG, NESUG, MWSUG, SCSUG, WUSS and PharmaSUG.

Dave is an unrepentant SAS bigot. His major areas of interest are efficient programming techniques and applications development using the SAS System. His SAS software product specialties are SAS/AF and FRAME technology, SAS/EIS, SAS/IntrNet, AppDev Studio, and CFO/Vision. Dave is a SAS Certified Professional.

Dave is the founder and President of the Florida Gulf Coast SAS Users Group. He chartered and served as Co-Chair of the first SouthEast SAS Users Group conference, SESUG '93, and served as President of the Executive Council of the SouthEast SAS Users Group. He also served on the Executive Council of CONSUG, the Consultant's SAS Users Group. His first SUGI was in 1983, and he has been actively involved in both SUGI and the Regional SAS User Group community since then.

Dave is currently the Co-Chair of SSU 2001, the combined SouthEast and SouthCentral SAS Users Group conference to be held in New Orleans in August, 2001.

