
WebTransactions Creating
W@P applications

Contents

1	<i>Introduction</i>	3
1.1	What is WAP?	3
1.2	Terminals	3
1.3	Architecture	3
1.3.1	WWW architecture	3
1.3.2	WAP architecture	4
2	<i>W@P applications with WebTransactions</i>	5
2.1	WML and WTML	5
2.1.1	Start template	5
2.1.2	Input options	5
2.1.3	Master for WML templates	5
2.2	Development with WebLab	7
2.3	Sample application	8
3	<i>Glossary and abbreviations</i>	9
4	<i>Related publications</i>	10

1 Introduction

1.1 What is WAP?

The **Wireless Application Protocol** has positioned itself at the forefront of two fast developing network technologies, the wireless data market and the Internet. It defines protocols and languages which have been specially designed for the restricted hardware configurations of mobile terminals and the narrow bandwidths of mobile networks.

The WAP standard is published by the W@P Forum (<http://www.wapforum.com>), whose members include all the major mobile network providers and mobile phone manufacturers (<http://www.wapforum.com/who/members.htm>), as well as Siemens AG. The latest approved version is WAP 1.1, though work on Version 1.2 is currently under way.

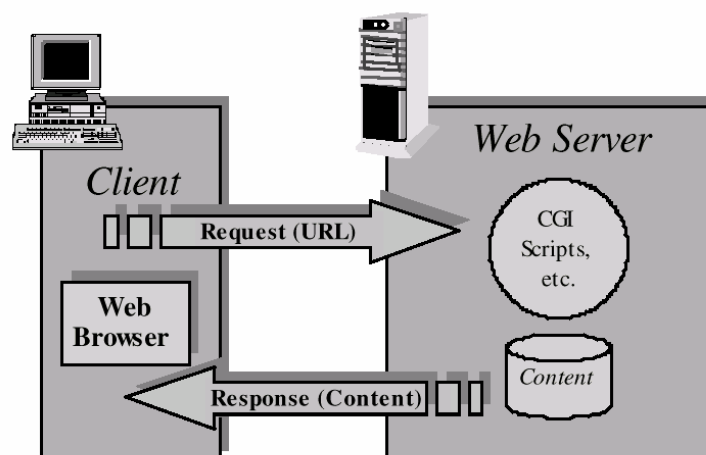
1.2 Terminals

Siemens S35i, M35i, and C35i mobile phones are all WAP-enabled. Our *IC35 – The Unifier* palmtop also comes with an infrared interface for a mobile phone, which includes a WAP1.1 browser.



1.3 Architecture

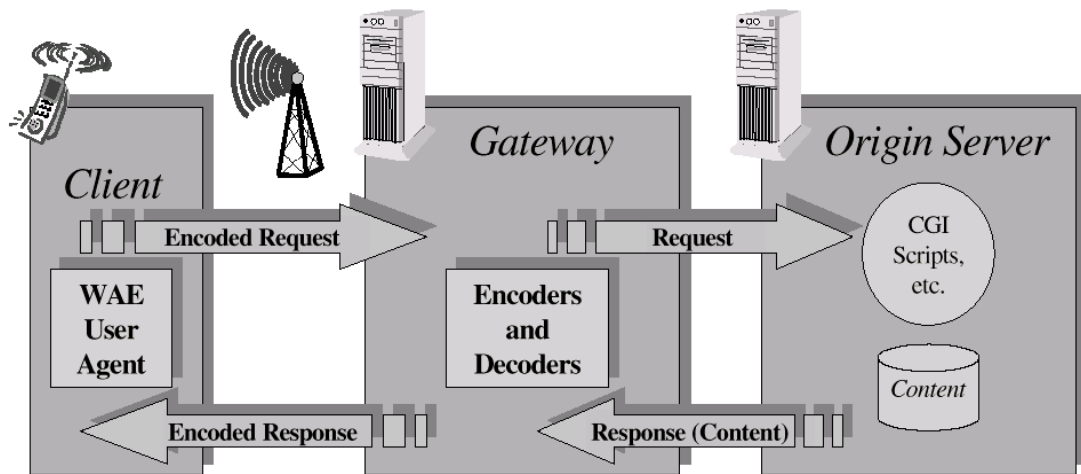
1.3.1 WWW architecture



The figure above shows the classic WWW model. The web browser issues requests to a web server, which responds by returning the desired page.

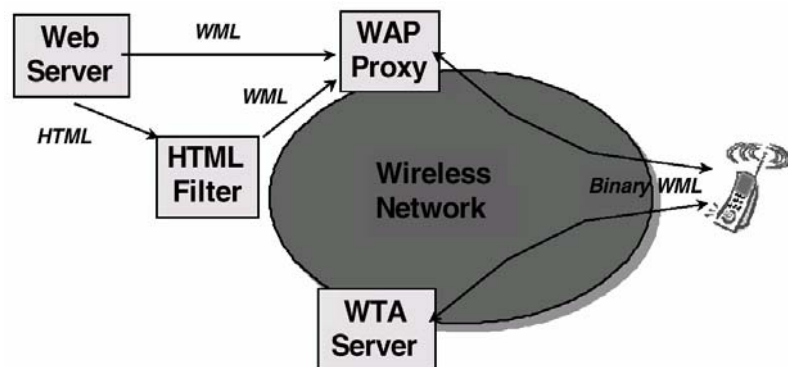
1.3.2 WAP architecture

The WAP protocol builds on the existing infrastructure with a proxy solution.



A WAP gateway converts the WTP protocol (**Wireless Transport Protocol**) used by mobile terminals to HTTP/TCP, and forwards requests to the addressed web server. This server must return the requested pages in the correct format, i.e. all content must be written in WML (**Wireless Markup Language**). The gateway compresses the response (converts it to WML byte code) and translates the HTTP/TCP protocol back into WTP.

The following scenarios can occur in a WAP network:



- A web server supplies content in WML format. Using the mobile client, these pages can be called up via the provider's gateway.
- A web server supplies content in traditional HTML format. In this case, an additional HTML filter is configured. It is addressed by the WAP proxy and converts original pages into WML. This scenario is particularly desirable, as it allows web content to be converted into WAP content quite easily. However, an HTML-to-WML filter is still not available for producing legible output on a mobile phone display which is pleasing to the eye.
- The mobile network provider supplies WAP content directly on WTA servers (**Wireless Telephony Application**). Access to various provider-specific services (e.g. T-D1 News, online banking, mailbox configuration) is thus standardized. User interfaces can be provided for these services without the mobile phone having to actually support the services.

2 W@P applications with WebTransactions

2.1 WML and WTML

2.1.1 Start template

In order to generate WML instead of HTML, all you need to do is make a minor adjustment in the start template. The variable `WT_SYSTEM.HTTP_DEFAULT_HEADER` must be supplied with the HTTP header appropriate to WML pages. All generated pages then have this header:

```
<wtOnCreateScript>
<!--
  WT_SYSTEM.HTTP_DEFAULT_HEADER = 'Content-type: text/vnd.wap.wml\n\n';
//-->
<wtOnCreateScript>
```

2.1.2 Input options

There are two input controls in WML (*input* and *select*) but, unlike HTML, there are no forms. All values entered are set in \$ variables, and must be specified explicitly with *go.* or in the link. As a functional extension, an optional format specification for input fields and hierarchical select lists is provided. `WT_SYSTEM.HREF` or `WT_SYSTEM.HREF_ASYNC` can be used in conjunction with the following trick: the “&” character must be converted to “&” using the replace method. Here is a short example from the demo:

```
<wml>
<card id="order" title="Order">
  <p>
    Quantity:
    <input type="text" name="Quantity" format="*N" size="2" maxlength="2"/>
    Customer Number:
    <input type="text" name="kdnr" format="*N" size="8" maxlength="8"/>
  </p>
  <p>
    <a href="##WT_SYSTEM.HREF.replace(/&/g, '&amp;')#&amp;command=waren">
      Back
    </a>
    <br/>
    <a
href="##WT_SYSTEM.HREF.replace(/&/g, '&amp;')#&amp;menge=$(Quantity) &amp;kdnr=$(kdnr) ">
      Send
    </a>
    <br/>
    <a href="##WT_SYSTEM.HREF.replace(/&/g, '&amp;')#&amp;command=ende">Exit</a>
  </p>
</card>
</wml>
```



2.1.3 Master for WML templates

The following master template can be used for generating the Automask and captured templates in WML:

```
<wtOnCreateScript>
<!--
  //{{WebLab(assignCommunicationObject)
  %%CommObj% = WT_HOST.%%CommObj%;
  if (%%CommObj%.WT_SYSTEM != null)
    %%CommObj%_system = %%CommObj%.WT_SYSTEM; // commu. specific system object
```

```

else
    %%CommObj%_system = WT_SYSTEM;          // global system object
    //}}
    // propagate communication object to included WTML documents ///////////////

lastLine = 1;
function taggedOutput( hostObject )
{
    if ( hostObject.StartLine != lastLine )
    {
        document.writeln('\n<br/>');
        lastLine++;
    }
    output = hostObject.HTMLValue.replace(/\$/g,"$$");
    if (hostObject.Intensity == 'Normal')
        output = '<b>' + output + '</b>';
    if (hostObject.Blinking == 'Yes')
        output = '<i>' + output + '</i>';
    if (hostObject.Underline == 'Yes')
        output = '<u>' + output + '</u>';
    document.write( output );
}

function taggedInput( hostObject )
{
    if ( hostObject.StartLine != lastLine )
    {
        document.writeln('\n<br/>');
        lastLine++;
    }
    currentLength = hostObject.Length;
    input = '<input type=' + (hostObject.Visible == 'No' ? "password" : "text" );
    input += ' name="' + hostObject.Name + ' size="' + currentLength + ' maxlength="' +
currentLength + ' value="' + hostObject.Value + '"/>';
    document.write( input );
}
//-->
</wtOnCreateScript>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
           "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="%%Format%" title="WebTransactions">
    <p>
      %%LINES DisplayAttributes=Dynamic StaticText=No TaggedInput=Enforced
      TaggedOutput=Enforced%
      <wtOnCreateScript>
        <!--
          urlext = '';
          for( element in wtInputFields )
          {
            urlext += '&' + element + '=' + element;
          }
        //-->
      </wtOnCreateScript>
    </p>
    <do type="accept">
      <go href="##WT_SYSTEM.HREF.replace(/\$/g,'&')+urlext#"/>
    </do>
  </card>

```

```

</wml>
<wtRem** Script executed after post of WML page *****>
<wtOnReceiveScript>
<!--
  //{{WebLab (processPostedData)
  %%OnReceiveCopies%
  //}}
  //{{WebLab (processHostCommunication)
  %%CommObj%.send();
  if (!WT_SYSTEM.ERROR)
    %%CommObj%.receive();
  if ( WT_SYSTEM.ERROR && WT_SYSTEM.COMMUNICATION_ERROR_FORMAT )
    setNextPage( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT );
  else
  {
    if( %%CommObj%_system.CAPTURED_FLD == "Yes" &&
        %%CommObj%_system.APPLICATION_PREFIX )
      setNextPage( %%CommObj%_system.APPLICATION_PREFIX + '@' +
                  %%CommObj%_system.FLD );
    else
      setNextPage( %%CommObj%_system.FLD );
  }
  //}}
-->
</wtOnReceiveScript>

```

2.2 Development with WebLab

To develop W@P applications under *WebTransactions*, you will need a W@P browser on the development system. Such browsers are readily available on the Internet and can be downloaded free of charge. However, as they have no OLE/DDE interfaces, connection to WebLab is not as straightforward as normal.

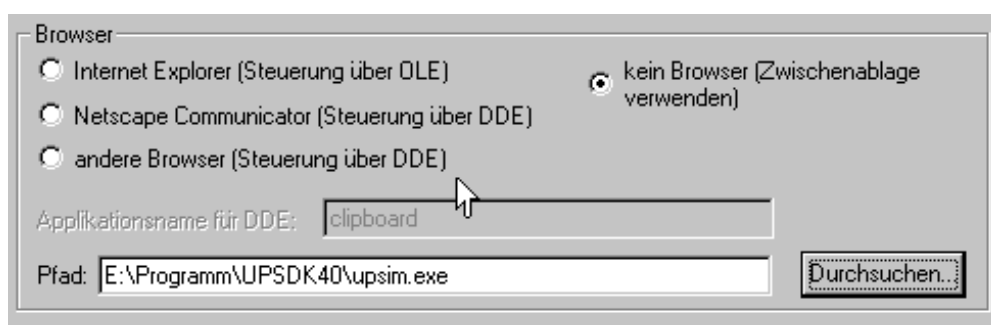
In the WebLab settings, activate "clipboard" in the browser options. This copies the URL transferred automatically to the browser when opening a session or reloading a new page to the clipboard. This string can then be inserted in the corresponding browser entry field simply by pressing Ctrl+V. When repeating a regenerate process, the browser reload function can also be used. This is because the URL for the regenerate process does not change provided the template remains the same.

If the path to the W@P browser is entered in the "Path" text box of the browser settings, this browser is called automatically when the session is opened. The URL must then be transferred manually as described above.

Procedure:

The following example uses the UP.Simulator from Phone.com (<http://www.phone.com>).

- In WebLab, choose "Options/Options". In the "Programs" tab, select "no browser (use clipboard)" and enter the UP.Simulator.exe file in "Path".



- Open a session as normal. The UP.Simulator is loaded automatically.
- Position the cursor in the "Go" field and press *Ctrl+V* followed by *Return*.



- This opens a *WebTransactions* session. In the browser, navigate to the template to be changed.
- Change the template in WebLab and choose "*Update in Browser*".
- In the UP.Simulator, mark the field "Go" again, and press *Ctrl+V* followed by *Return* to send off the inserted URL. The screen is then reloaded.
- If you make other changes to the same template and then save the template in WebLab, you can reload the screen directly in the UP.Simulator using "*Edit/Reload (F9)*".

2.3 **Sample application**

The PackAndGo archive "WAP-Demo.zip" contains a small sample tool ordering system. The most important W@P techniques used in the context of *WebTransactions* are explained in the templates.

3 Glossary and abbreviations

HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
MIME	Multipurpose Internet Mail Extensions
WScript	Web <i>Transactions</i> script (corresponds to the JavaScript language core)
WTML	Web <i>Transactions</i> Markup Language
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTA	Wireless Telephony Application
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol

4 Related publications

[1] **W@P White Paper**, Wireless Internet Today, June 1999

[2] **WAP WML, WAP Forum**, Version 16-Jun-1999

[3] **WebTransactions V3.0, Template Language**, Manual August 1999