Specification

WML Specification – Wireless Internet Gateway

Document number: ST1745519

Revision: E

Contents

Document number: ST1745519
Revision: E

1 Introduction

This document specifies the subset of Wireless Markup Language (WML) that the WIG server supports from version 2.0. An extension to WML is also specified.

The WIG server supports WML according to the WAP WML specification Version 1.1, ref. [1]. For backward compatibility reasons, a subset of WAP WML version 1.0 [4] is also supported.

Document number: ST1745519
Revision: E

2 References

Ref.	Title	Document No.
[1]	WAP WML Specification Version 1.1. http://www.wapforum.org	
[2]	HTML-WML Specification - WIG	17429012
[3]	"GSM 11.14, SIM Application Toolkit Interface", version 7.3.1, Release 1998	
[4]	WAP WML Specification Version 1.0. http://www.wapforum.org	

2.1 Revision history

Rev. Comments
E WigSendServerSM can't include variables.
D Removed and added quotation marks
C Added description of parameter used for mobile to content provider plug-ins. Added info about reserved parameters in URL.
B Enhanced description of plug-in syntax.
A Updated for release 2.0 of Wireless Internet Gateway

3 WML

The WIG server supports a subset of WML, see reference [1], required to address the WIB functionality.

3.1 WML Syntax

3.1.1 The WML page

A WML page is either stored on a web server, or stored in compressed form on the SIM as a startup file for the WIB client.

The WML page can either be written on 8-bit ASCII format, which is default, or on UTF-8 format (Unicode). If a page is written in Unicode, that is specified in the WML prologue. (See chapter 3.3.3)

3.1.2 Entities

Entities are used to specify characters in the document character set either which must be escaped in WML or which may be difficult to enter in a text editor. WML text can contain numeric or named character entities. The WIG server supports the following predefined named entities.

All entities begin with an ampersand and end with a semicolon.

Entity	Character
&amp;	&
&lt;	<
&gt;	>
&quot;	“

3.1.3 Elements

Elements may contain a start tag, content and an end tag. Elements have one of two structures:

<tag/>

or

<tag> content </tag>.

3.1.4 Attributes

Attributes specify additional information about an element and are always specified in the start tag of an element. For example,

<tag attr="abcd"/>

All attribute values, i.e. arguments, must be quoted using double quotation marks (").

Ref. [1] contains additional information about the attributes described in this specification, for instance which ones are mandatory and which ones are optional.

3.1.5 Variables

Variables can be used in the place of strings and are substituted at run-time with their current values. Anywhere the variable syntax is legal, an '$' character indicates a variable substitution. For example,

My first name is $(FIRSTNAME).

A variable can be set for instance by the **`setvar`**, **`input`** and **`select`** elements, see the element definitions below. A variable can have a value equal to the empty string ("").

3.2 WML Script Syntax

In some of the calls to perform a command on the WIB, WML Script Syntax in a "GO HREF" is used, i.e. the call looks like a call to a WML Script subroutine. Although the "GO" tag is used, no message is sent to the server, as the routine is stored locally on the SIM. The general syntax for this is:

Example:

<go href ="http://www.acrosswireless.com#functionname(arg1,,arg2,arg3)"/>

Different commands require a different function name with different numbers of arguments. The arguments are used for both input and output data. The name of the function defines which command in the WIB the function will be translated to.

Later in the chapters where this syntax is used the "argument" column includes a description of the argument passed in the function. The "argument value" column is what is actually included in the call.

Argument	**Argument value**	
arg1	Description of what to send	M
arg2	…	O
…	…	

All arguments must be included in the call, but some of them can be empty. If the last column in the following tables indicates if the attribute is M-mandatory there must be a value included. If the column indicates O-optional the argument can be empty, but don't forget that is has to be included with a comma ','. All argument values can be engrossed by single quotes (‘) but if spaces or comma ‘,’ is to be included in the argument value, the value must be engrossed by simple quotes (‘). Double quotes (“) are not allowed in the argument values.

An argument value can include a variable which are substituted at run-time with its current value.

In the following example assume that myname is set to John. The userdata will then have the value “Hello John!”

```
<wml>
<input title=”Please enter his name” name="myname"/>
<go href="http://www.across.se#wigSendSM('Hello $(myname)!',,, '0706754321',
'+46705008999')"/>
</wml>
```

In the following example no variable substitution is made because the parantheses are missing. The userdata will have the value “Hello $myname!”

```
<wml>
<input title=”Please enter his name” name="myname"/>
<go href="http://www.across.se#wigSendSM('Hello $myname!',,, '0706754321',
'+46705008999')"/>
</wml>
```

Examples where this syntax is used is in the plug-in command and in the Sim Application Toolkit commands.

3.3 Elements

The order of elements in a WML document is significant since it will be interpreted in sequence by the WIB.

3.3.1 Prologue

A WML deck may contain an XML declaration and a document type declaration.

Example:

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//xxxxxx"
                    "http://www.acrosswireless.com/xxxxx">
```

3.3.2 Text

Text that has no tag is classified as pure text and is displayed to the user. The user acknowledges text with the OK button. A blank line splits the text in two paragraphs, that are displayed separately to the user.

In some cases, a text paragraph is automatically associated to a following INPUT or SELECT tag, see below.

The character < is forbidden in the text, as this is taken for the start of a tag.

If the `encoding=”utf-8”` attribute is set on the WML page, the text contains characters contained in the Unicode character set (see below).

3.3.3 Unicode

WML pages can be encoded either with 8-bit ASCII, which is default, or with UTF-8 character encoding. If a page is written in UTF-8, it should start with a prologue containing the attribute

```
encoding=”utf-8”
```

This will result in that all text strings that are sent between the WIG server and the SIM are encoded with UCS2. UCS2 uses two bytes for every character, i.e. double the space that is needed when using the GSM default alphabet.The picture shows an example of a UTF-8 WML page:

```
<?xml version="1.0"? encoding="utf-8">
<wml>
歡迎!你想要喝甚麼?
<select key="DRINK">
<option>可樂</option>
<option>雪碧</option>
<option>芬達</option>
</select>
<go href="http://webserver/path/some_page.asp?drink=$(DRINK)"/>
</wml>
```

The example shows how Unicode can be used for the texts that are to be input and output on the telephone, and for the contents of variables. It also shows that the Unicode variable contents can be passed to the content provider as a parameter to the GO HREF command. However, the Western character set is always used for WML tags, variable names, attribute names, most attribute values, and URIs.

The whole URL in "go href", including the query string, must be Western characters. However, the contents of the variables that are passed in the query string can be Unicode, e.g. in the example above, the contents of the variable DRINK is Unicode.

The WML element defines a WML document and encloses all information in the document. The WML element requires a start and an end tag.

<wml> content </wml>

Attribute	Argument
xml:lang	Language in which the document is written. Not currently used for anything by the WIG server.

3.3.4 Input

The input element defines an input field where the user may enter information.

<input/>

The following attributes and arguments are supported

Attribute	Argument
name	Name of variable to set
type	text(default) password, meaning that the text is not echoed on the mobile phone.
value	The default value of the variable named in the name attribute.
format	Expected data format entered by the user. *M - Any character (default) *N - Any numeric character
emptyok	true - This input element accepts empty input. false – Empty input not allowed. If omitted, 'true' is assumed.
maxlength	Max number of characters that can be entered by the user
title	Prompting string
class	Optional type specification of the variable, used for conversion purposes in the WIG server. If omitted, "GSMDefault" is presumed in a Western character set page, and "UCS2" is presumed in a Unicode page.

If the attribute "title" is not included in the tag, the text immediately preceding the input tag is used as a prompting string for the input field. Omitting any prompting string will cause undefined results.

If the attribute “title” is included, the title will be used as the prompting string and if there is text immediately preceding the input tag this text will be displayed before the input title.

If the encoding=”utf-8” attribute is set on the WML page, the user of the mobile phone will be prompted for a Unicode string. In addition, the prompting string as well as the argument of the “value” attribute contains characters allowed by the Unicode character set.

The “class” attribute is used for conversion purposes when the variable is later on passed in a “go href” from the WIB client to the content provider. It may have the following values in the INPUT tag:

?? “GSMDefault”, the variable contains characters of the GSM default alphabet. When sent through the WIG server to the Content Provider, the data is converted to ASCII. This value is used by default in a page with the Western character set.

?? “UCS2”, the variable contains UCS2-encoded characters. When sent through the WIG server to the Content Provider, the data is converted to UTF-8. This value is used as default in a Unicode page.

On many mobile phones, passwords may only be entered as numbers, not as text. input type=password will fail on these phones without the format=”*N” attribute.

Example 1:

<input title=”Please enter your phone number” type="text" name="PHONE" format="*N" maxlength="20"/>

Example 2:

Assume that Unicode is used for encoding the page, but the mobile phone only supports the Western character set for input. The variable is manually set to be of type “GSMDefault”:

<input title=”Please enter your name” name="MYNAME" class=”GSMDefault"/>

3.3.5 Card

The card element defines a container of text and elements in a WML document. A document may contain multiple card elements but card elements may not be nested. The first card element in a document is the start card. The card element requires a start and an end tag.

<card> content </card>

Attribute	Argument
id	Advisory information about the element.
newcontext	Current browser context should be re-initialised, i.e. all variables are erased. True or false.

Example:

```
<card id="card1">
Please enter your first name
<input type="text" name="FIRSTNAME"/>
</card>
```

3.3.6 Select

The select element defines and displays a set of optional list items from which the user can select an item. An option element is required for each item in the list, see the option element section. The name of the menu, normally displayed by the telephone, is specified by the “title” attribute.

If the attribute “title” is not included the preceding text or text paragraph is automatically associated to the “select” tag and used as title text.

The select element requires a start and an end tag.

<select> content </select>

Attribute	Argument
title	Title of the menu.
name	Name of the variable to set.
iname	Name of the variable to set with the index result of the selection. See ref. [1]

Either the name or iname attribute can be used. If the iname attribute is used the option value attribute will be overridden with the calculated index.

If the encoding=”utf-8” attribute is set on the WML page, the value of the “title” attribute contains characters allowed by the Unicode character set.

3.3.7 Option

The option element represents a list item in a list defined by the select element. The option element requires a start and an end tag.

<option> content </option>

The content consists of text that is displayed as the option text.

Attribute	Argument
value	The select element name is set to this argument if this option element was selected.
onpick	A destination href (card name). The href can only be another card, not another WML page. See the syntax example below.

The content text to an option element is used as value argument if the value attribute is not present. Empty item text strings are not supported.If the encoding=”utf-8” attribute is set on the WML page, the content text as well as the argument of the “value” attribute contains characters allowed by the Unicode character set.

The option element can also specify a destination card in the same deck. The onpick event occurs when the user selects this option. In the example below, a jump will occur to “CARD2” if the user select the “BANKING” option, and to “CARD3” if the user selects the “GAMBLING” option.

Example:

<select title=”Please choose service“ name="SELECTION">
<option value="BANKING" onpick="#CARD2">Banking</option>
<option value="GAMBLING" onpick="#CARD3">Gambling</option>
</select>

3.3.8 Go

The go element declares a go task to a href (URI) or a specified card in the document.

The URL might contain variable references in the host name:

<go href=”www.$(HOSTNAME).com”/>

as well as in the parameters of the URL, following the host name and a question mark:

<go href=www.acrosswireless.com?name=$(NAME)/>

The following parameter names are reserved, i.e. they may not be used in the URL following a question mark:

Parameter Name	Description
WPLGN	Reserved for specifying name of server side plug-in to use when request is sent from mobile to content provider.
_PS_LI_	Reserved for Location Area Identification data when a positioning service shall be used.
_PS_NMR_	Reserved for network measurement data when a

	positioning service shall be used.
_PS_DT_	Reserved for date-time data when a positioning service shall be used.
_PS_IMEI_	Reserved for international mobile entity identification data when a positioning service shall be used.
_PS_TA_	Reserved for timing advance data when a positioning service shall be used.

The go element requires a start tag only.

<go/>

Attribute	**Argument**
href	A destination URI.

The URL may be specified as http://host/path or https://host/path, in that way selecting if SSL shall be used for the web server connection or not. Relative URL:s are not supported.

Only Western characters are allowed in the URL (including the query string), even if the encoding=”utf-8” attribute is set on the WML page. However, the URL may contain variable references where the variables contain characters encoded with UTF-8.

Note that after each “go href”, no more WML tags should be executed. Using text or WML tags after a GO HREF may cause problems for the application. Executing several GO HREF:s in a row may cause undefined behaviour.

Example:

```
<input title="Variable" type="text" name="VARIABLE"/>
<go
href="http://www.acrosswireless.com?f=$(VARIABLE)&amp;l=StaticText
"/>
```

Second example:

<input title="First name" type="text" name="FIRSTNAME"/>
<input title="Last name" type="text" name="LASTNAME"/>
<input title="Age" type="text" name="AGE"/>
<go
href="http://www.acrosswireless.com**?f=$(FIRSTNAME)&l=$(LASTNAME)&a=$(AGE)&x=You can write whatever you want here"/>**

Third example:

<go href="#CARD1"/>

3.3.9 Setvar

The setvar element allows the author to set the value of a variable without performing any side effects. The var element requires a start tag only.

<setvar/>

Attribute	Argument
name	Name of the variable to be set
value	The variable is set to this argument. VALUE may only contain fixed text. Variables are not allowed.
class	Optional type specification of the variable, used for conversion purposes in the WIG server. If omitted, "Binary" is presumed. See below.

If the encoding="utf-8" attribute is set on the WML page, the argument of the "value" attribute contains characters allowed by the Unicode character set.

The "class" attribute is used for conversion purposes when the variable is passed in a "go href" from the WIB client to the content provider. It may have the following values:

?? "GSMDefault", the variable contains characters of the GSM default alphabet. When sent through the WIG server to the Content Provider, the data is converted to ASCII.

?? "UCS2", the variable contains UCS2-encoded characters. When sent through the WIG server to the Content Provider, the data is converted to UTF-8. Note that UTF-8 should normally not be stored directly in a variable on the card, as the encoding used by the card is UCS2.

?? “Binary”, the variable contains binary data that is not to be converted. This is the default value if the “class” attribute is omitted. The “binary” class is used for instance when encrypted data is sent to the content provider.

The “setvar” element supports hexadecimal numeric character entities in the value attribute argument, e.g. "]".

Example 1:

<setvar name="Country" value="Sweden"/>

Example 2:

<setvar name="BYTES" value="&#x02;&#e3;?"/>

Example 3:

<setvar name="BYTES" value="&#x02;&#e3;?" class=”Binary”/>

3.3.10 Noop

The noop element specifies that nothing should be done, no operation. The noop element requires a start tag only.

<**noop**/>

3.3.11 Do

The “do” element is a general mechansim for the user to act upon the current card. The only supported type in the WIG Server at the moment is accept, and that gives that the task following the "do" element is always executed.

Because of this the execution of the script does not stop at the "do" command. If a stop before the "do" command is wanted a construction of the page like in the example below can accomplish that. The execution will continue after the name has been input.

<**do**> **content** </**do**>

Attribute	**Argument**
type	accept

Example:

<card>
<input type="text" name="NAME"/>
<do type="accept">
< go href="http://www.acrosswireless.com**?f=$(NAME)& "/>**
</do>
</card>

3.3.12 Plug-ins

Plug-ins are called with WML Script syntax in a "GO HREF", i.e. the call looks like a call to a WML Script subroutine. Although the "GO" tag is used, no message is sent to the server, as the routine is stored locally on the SIM. For more detailed information of the syntax see chapter 3.2.

The general syntax for calling a plug-in is as follows:

```
<wml>
<go href=http://www.across.se#wigObject
('myplugin', 'inputvalue', 'outputvalue', encryptkey, 'class')"/>
</wml>
```

Attribute	Argument	
name	A string representing the name of the plug-in.	M
inputvalue	Data to be processed including variable references.	M
outputvalue	A string naming the variable that will contain the output data from the plug-in.	M
encryptionkey	Decimal string for referencing the key to be used.	O
class	class identifier, used for conversion purposes in the WIG server. See appendix. If omitted, BINARY is assumed.	O

3.3.12.1 Standard plug-in SIGN

The standard SIGN plug-in computes a MAC according to ISO9797 by encrypting the input data with 3DES in outer CBC mode and 2 keys, and picking out the first half of the last 8-byte output block as signature.

Example:

```
<wml>
<go href="http://www.across.se#wigObject('sign','data to be signed', 'signeddata',
```

```
2)"/>
</wml>
```

3.3.12.2 Standard plug-in ENCR

The standard ENCR plug-in encrypts the input data with 3DES in outer CBC mode using two different keys.

The first byte of the output of the ENCR plug-in contains how many padding bytes that were added to the end of the data before encrypting it.

Example:

```
<wml>
<go href="http://www.across.se#wigObject('encr','data to be encrypted','encrd',
1)"/> (performs encryption)

<go href="http://www.mysite.com?DATA=$(encrd)"/>   (sends it to the content
provider)
</wml>
```

3.3.12.3 Standard plug-in DECR

The standard DECR plug-in decrypts data with 3DES in outer CBC mode using two different keys.

The input to the DECR plug-in is always an even number of eight-byte blocks, preceded by one byte telling how many padding bytes that shall be removed from the end of the result of the decryption. This is done automatically by the DECR plug-in.

Example:

In the example, one eight-byte block is decrypted. From the result of the decryption, the last two padding bytes is removed by the plug-in on the card, as the first byte of the encrypted data is '02h'.

```
<wml>

<setvar name="in1"
value="&#x02;&#x94;&#x23;&#xA4;&#xC6;&#x7D;&#x79;&#x88;&#x78;"/>
<go href="http://www.across.se#wigObject('decr','$(in1)','out1',1)"/> (performs
decryption)

Hi $(out1)!  (displays the data without padding bytes)
</wml>
```

3.3.13 SIM Application Toolkit Commands

The WIG provides access to some SIM Application Toolkit (SAT) commands. SAT commands are called with WML Script syntax in a "GO HREF", i.e. the call looks like a call to a WML Script subroutine. For detailed information on the

parameters and data format, see GSM 11.14, ref [3]. Although the “GO” tag is used, no message is sent to the server, as the routine is stored locally on the SIM. For detailed information on the WML Script Syntax see chapter 3.2.

Different SAT commands require a different function name with different numbers of argument.

Example:

<go href ="http://www.acrosswireless.com#functionname(arg1,,arg2,arg3)"/>

3.3.13.1 Provide Local Information

This command is used to get location information from the mobile station. Different location parameters can be fetched from the mobile phone and put into a variable.

Function name: **wigProvideLocalInfo**.

Argument	**Argument value**	
commandqualifier	00: location information (7 bytes) 01: IMEI of ME (8 bytes) 02: Network measurement results and BCCH list (16 bytes) 03: Date, time and time zone (7 bytes) 04: Language setting (2 bytes) 05: Timing advance (2 bytes)	M
outputname	Variable to contain output data.	M

In this example, the IMEI is fetched and put in the variable POSITION. On the next line, the IMEI is sent to a content provider.

```
<wml>
<go href="http://www.across.se#wigProvideLocalInfo(01,'imei')"/>
<go href=http://www.arne.se?IMEI=$(imei)/>
</wml>
```

3.3.13.2 Play tone

This command makes the mobile station play a tone.

Function name: **wigPlayTone**.

Argument	**Argument value**	
toneid	01: Dial tone 02: Called subscriber busy 03: Congestion 04: Radio path acknowledge 05: Radio path not available 06: Error / special information 07: Call waiting time 08: Ringing tone	M
durationtimeunit	00: minutes 01: seconds 02: tenths of seconds	M
durationtime	Coded as integer multiples of the time unit used, 01-FF.	M
text	Text to display.	O

In this example, the mobile phone is requested to play a CONGESTION tone of 10 seconds. The text is omitted, so no text is displayed.

```
<wml>
<go href="http://www.across.se#wigPlayTone(03,01,10,)"/>
</wml>
```

3.3.13.3 Set Idle Mode Text

This command sets a text on the idle screen of the mobile station.

If no text attribute is included or the text attribute consists of an empty string, the idle text will be removed.

Function name: **wigSetIdleModeText**.

Argument	Argument value	
text	Text to display.	O

Example:

```
<wml>
<go href="http://www.across.se#wigSetIdleModeText('Welcome')"/>
</wml>
```

3.3.13.4 Refresh

This command makes the SIM notify the mobile phone of changes in the SIM configuration as the result of SIM application activity. Depending on the command qualifier, different tasks will be performed. For more information see GSM 11.14, ref. [3].

Function name: **wigRefresh**.

Argument	Argument value	
commandqualifier	00: SIM Initialization and Full File Change Notification 01: File Change Notification (requires file list) 02: SIM Initialization and File Change Notification (requires file list) 03: SIM Initialization 04: SIM Reset	M
numberoffiles	Number of files included in the filelist.	O
filelist	List of files.	O

In the example, a SIM initialization is requested, and in addition, the mobile phone in notified that two files on the SIM have been updated, 7F10/6F3A (the ADN list) and 7F20/6F30 (the PLMN selector file)

```
<wml>
<go href ="http://www.across.se#wigRefresh(02,02,'3F007F106F3A3F007F206F30')"/>
<wml/>
```

Full paths are given to files. Each of these shall be at least 4 octets in length. An entry in the file description begins with 3FXX and there shall be no delimiters between files.

3.3.13.5 Set up call

This command requests the mobile phone to initiate a call.

Function name: **wigSetupCall**.

Argument	**Argument value**	
commandqualifier	00: only if not currently busy 01: only if not currently busy, with redial 02: putting all other calls on hold 03: putting all other calls on hold, with redial 04: disconnecting all other calls 05: disconnecting all other calls, with redial	M
text	Text to display.	O
capability	Capability Configuration Parameters. For coding, see GSM 04.08.	O
durationtimeunit	00: minutes 01: seconds 02: tenths of seconds	O
durationtime	Coded as integer multiples of the time unit used, 01-FF.	O
destinationaddress	The called party number.	M

Note: “Durationtime” defines the duration of time that automatic retries to set up the call will be made.

In the example, the SIM requests the mobile phone to, if not currently busy with another call, set up a call to "0707789613". No text is displayed, no Capabillity Configuration Parameters are attached, and no automatic retries to set up the call will be made.

```
<wml>
<go href="http://www.across.se#wigSetupCall(00,,,,,'0707789613')"/>
</wml>
```

3.3.13.6 Set version information to variable

This command reads the version information of the WIB client from the SIM and assigns it to the specified variable.

Function name: **wigSetBrowserVersion**.

Argument	**Argument value**	
outputname	Variable to contain output data.	M

In the example, the WIB version information on the SIM is copied from 2700/6F07 and put in the variable VERSION. On the next line, the version information is sent back to the Content Provider.

```
<wml>
<go href="http://www.across.se#wigSetBrowserVersion('version')"/>
<go href=”http://www.myserv.com?VERSION=$(version)/>
</wml>
```

3.3.13.7 Set script buffer size to variable

This command reads the current script buffer size and assigns it to the specified variable. The variable will not be sent back to the server automatically.

Function name: **wigSetScriptBufferSize**

Argument	**Argument value**	
outputname	Variable to contain output data.	M

In the example, the size of the internal WIB script buffer on the SIM copied put in the variable SZ. On the next line, it sent back to the Content Provider.

```
<wml>
<go href="http://www.across.se#wigSetScriptBufferSize('sz')"/>
<go href="http://sz)"/>
</wml>
```

3.3.13.8 Set return tar value

This command makes sure that the next submit from the browser to the server has destination TAR address as in the specified record id of the WIG setup file on the SIM.

Function name: **wigSetReturnTarValue**

Argument	**Argument value**	
recordid	Record of elementary file. 01 and 02 supported.	M

In the example, the WML page "index.wml" will be fetched from the WIG server with the TAR value specified in record 2 of the WIG setup file 2700/6F01 on the SIM.

```
<wml>
<go href="http://www.across.se#wigSetReturnTarValue(02)"/>
<go href=”http://www.myserv.com/index.wml/>
</wml>
```

3.3.13.9 Send USSD

This command sends a byte string by the Unstructured Supplementary Service.

Function name: **wigSendUSSD**.

Argument	**Argument value**	
text	Text to display.	O
ussd	According to GSM 02.30.	M

In the example, a USSD message with the contents "*21*1222#" is sent to the network. No text is displayed.

```
<wml>
<go href="http://www.across.se#wigSendUSSD(,'*21*1222#')"/>
</wml>
```

3.3.13.10 Send SM

This command sends a plain text SM from the WIG browser to a particular destination.

Function name: **wigSendSM**.

Argument	Argument value	
userdata	Text in the SM. Might contain variable references.	O
pid	Protocol identifier	O
dcs	Data Coding Scheme, according to GSM 03.38.	O
destinationaddress	The called party number.	M
servicecentreaddress	The number of the service center.	O

In the example, a text SM with the contents “Hello!” is sent to MSISDN “0706754321”. As “PID” and “DCS” are omitted, the default values “0” and “242” decimally are used. It is made sure that the specified Service Center “+46705008999” is used, regardless of the default value in the mobile phone.

```
<wml>
<input title=”Please enter his name” name="myname"/>
<go href="http://www.across.se#wigSendSM('Hello $(myname)!',,, '0706754321',
'+46705008999')"/>
</wml>
```

3.3.14 Server side plug-ins

This chapter describes server side plug-ins. Server side plug-ins are executed on the WIG server. Server side can be used both when the request travels from mobile phone to the content provider and when the request travels from the content provider to the mobile phone.

When the request is travelling from the mobile, the parameter *WPLGN* (note, case sensitive) must be included in the URL and the value is set to the name of the plug-in, e.g.

```
<go href="http://www.across.se?WPLGN=PlugInName"/>
```

As for now, no generic plug-in exists for requests travelling from the mobile.

When the request is travelling to the mobile, the call for a server side plug-in is inside the WML page fetched from the content provider. The command "go href" is used and the URL hostname will be www.across.se. A hash sign and the plug-in name will follow the hostname, e.g.

```
<go href="http://www.across.se#wigNoResponse()"/>
```

3.3.14.1 wigSendServerSM

This plug-in sends a Short Message directly from the WIG server via the Transport Server in the Delivery Platform to a particular destination.

Function name: **wigSendServerSM**.

Argument	**Argument value**	
userdata	Text in the SM.	O
pid	Protocol identifier	O
dcs	Data Coding Scheme, according to GSM 03.38.	O
destinationaddress	The called party number.	M

In the example, a text SM with the contents "Hello!" is sent to MSISDN "0706754321". As "PID" and "DCS" are omitted, the default values "0" and "242" decimally are used.

```
<wml>
<go href="http://www.across.se#wigSendServerSM('Hello! ',,, '0706754321')"/>
</wml>
```

3.3.14.2 wigNoResponse

This plug-in is used when no response from the request should be sent back to the ME. E.g, when a push request generates a response and that response should not generate any byte code to be sent to the ME. If the page includes other commands that normally results in generated byte code, this byte code will NOT be sent to the ME due to the "NoResponse" plug-in.

Function name: **wigNoResponse**.

No arguments are included in the call.

Argument	Argument value	

Example:

```
<wml>
<go href="http://www.across.se#wigNoResponse()"/>
</wml>
```

3.4 Examples of WML documents

3.4.1 Example 1

This example illustrates the use of WML elements.

```
<wml>

<card id="START">

<input title=”Please enter your first name” type="text" name="FIRSTNAME"
maxlength="10"/>

<select title=”Do you want to subscribe to The WIG journal?”>
<option onpick="#FILLOUTFORM ">Yes</option>
<option>No</option>
</select>

<select title=”Do you want to subscribe to The WIG journal for free?”>
<option onpick="#FREE">Yes</option>
<option>No</option>
</select>
</card>

<card id="FILLOUTFORM">

<input title=”Please enter your last name” type="text" name="LASTNAME"
maxlength="20"/>

<input title=”Please enter your password” type="password" name=" PWD"
maxlength="8"/>

<select title=”Select your favourite drink” name="DRINK">
<option value="VR">Vodka Russian</option>
<option value="GT">Gin & Tonic</option>
</select>
<go href="http://www.acrosswireless.com?f=$(FIRSTNAME)&amp; l=$(LASTNAME)&amp;
p=$(PWD)&amp; d=$(DRINK)"/>
Thank you $(FIRSTNAME) $(LASTNAME) for choosing our journal!
</card>

<card id="FREE">
$(FIRSTNAME), did you really believe that you could get it for free?
<go href="http://www.across.se#wigPlayTone(03,01,10,)"/>
</card>

</wml>
```

3.4.2 Example 2

This example illustrates how a user is requested to enter some data, the sign plug-in is activated and the data is submitted to an URL.

```
<wml>

<card id="START">
<input title=”Please enter your user name” type="text" name="USERNAME"
maxlength="10"/>
outputname="SIGNEDDATA"/>
```

```
<go href="http://www.across.se#wigObject('sign','$(USERNAME)','$(SIGNEDDATA)',
2)"/>
Thank you! This is now passed to the server.

<go href="http://www.acrosswireless.com?u=$(USERNAME)&amp; s=$(SIGNEDDATA)"/>
</card>

</wml>
```

Appendix: Character encodings and conversions made by the WIG server

In some cases, the character encoding that is used on the card differs from the one that is used on the Content Provider side and in the WML pages:

?? On the card, the SMS default alphabet is normally used for text strings, while ASCII is used for encoding strings in WML pages,

?? When using Unicode, the WML pages are encoded with UTF-8, while UCS2 is used on the card.

?? At the same time, encrypted data is sometimes passed through the WIG server to security plug-ins, which shall not be converted at all.

Converting of client-bound messages

In a WML page, all text strings are normally automatically converted to the GSM default alphabet when sent to the WIB browser client on the SIM.

If the WML page is a Unicode page, i.e. contains the

```
encoding="utf-8"
```

attribute in the prologue, all text strings are instead converted from UTF-8 to UCS2 before they are passed to the WIG browser.

However, data that is described on hexadecimal format is never converted, like in the following case:

?? **<setvar name="BYTES" value="Bytes:&#x02;&#e3;?"/>**

This means that SETVAR can be used for entering data on hexadecimal format to e.g. a security plug-in for decryption.

Converting of server-bound messages

When data is passed in a "go href" instruction from the WIB client to the content provider, the whole URL, including the query string, and including contents of variables, is first converted from the GSM default alphabet to ASCII, and then URL encoded, making it understandable for the web server and the content provider.

If the page is a Unicode page, the contents of variables passed in the Query String are converted back from UCS2 to UTF-8. This means that the Content Provider will receive contents of variables on the same UTF-8 format that he used for encoding the WML page.

If a variable has been used in a SETVAR instruction, or if it has been used as output from a plug-in, it is not converted to UTF-8 unless the CLASS attribute has been set on the variable. The reason is that the WIG server guesses as default that the variable contains data instead of a string in these cases.

Thus, the default behaviour of the WIG server is the following:

?? If the web page is a normal ASCII page, server-bound URLs, query strings and variable contents are first converted from Default SMS alphabet to ASCII, then URL encoded

?? If the web page is a Unicode page, variable contents in Query Strings of server-bound URLs, are first converted from UCS2 to UTF-8, then URL encoded.

?? If a variable has been used in a SETVAR or OBJECT instruction, its contents is not converted.

The default conversion of variables can be overridden by setting the CLASS attribute in a SETVAR or OBJECT instruction.

Example 1:

In this case, the contents of the variable is not converted by the WIG server, (but it is always URL encoded). This is the same as if no “class” attribute is used at all:

```
<wml>
   <setvar name=”VAR” value=”&#x00;&#x31;” class=”Binary”/>
   <go href=www.acrosswireless.com?VAR=$(VAR)/>
</wml>
```

Example 2:

In this case, the contents of the variable is interpreted as UCS2 and is converted to UTF-8 by the server, i.e. “0x00, 0x31” will become the single byte “0x31”:

```
<wml>
   <setvar name=”VAR” value=”&#x00;&#x31;” class=”UCS2”/>
   <go href=www.acrosswireless.com?VAR=$(VAR)/>
</wml>
```