Guidelines

WIG Application Guidelines

Document number: ST17455120

Revision: B

Contents

1 Introduction 3

2 References 4

3 Communication 5

4 Browser request 6

5 Push request 7

6 The Plug-in architecture 8

7 WML and HTML tags 9
7.1 Text 9
7.2 Input 10
7.3 Card 10
7.4 Select 11
7.5 Option 12
7.6 Go 14

8 WML application examples 16
8.1 Example 1 16
8.2 Example 2 17
8.3 Example 3 17
8.4 Example 4 18

9 Browser configuration 19
9.1 EF TAR (6F01) 19
9.2 EF Error Text (6F02) 20
9.3 EF Byte Code (6F03) 23
9.4 EF SMS Header (6F04) 24
9.5 EF 03.48 Header (6F05) 25
9.6 EF 03.48 Counter (6F06) 26
9.7 EF Version Information (6F07) 27
9.8 EF WIB Configuration (6F08) 28

10 Personalisation issues 29
10.1 File sizes 29
10.2 OTA data download 29
10.3 Browser application size 29

11 Browser SM vs Server SM 30

12 WIG Browser 1.0 vs WIG Browser 1.1 31

Document number: ST17455120
Revision: B

13 Unicode _____ 32

14 Supported WML elements and attributes _____ 33

Document number: ST17455120
Revision: B

1 Introduction

This document aims to help application developers develop applications on top of the Wireless Internet Gateway. It highlights the differences between writing a standard web application and an application that should work well on a mobile phone.

Even though the application is written using standard WML or HTML, there are certain requirements on the application that are different from when you write a web page that should be accessible through a normal web browser. The GSM network and the communication media used are different from the Internet. The mobile phone and the SIM card used put other constraints on what is possible to do or not.

In addition to this, SIM aspects of the WIG are discussed. Personalisation issues and the browser configuration data on the SIM are briefly described.

Document number: ST17455120
Revision: B

2 References

Ref.	Title
[1]	WML Specification, ST17455119
[2]	“GSM 03.48, SIM Toolkit Secure Messaging”, version 7.0.1, Release 1998
[3]	“GSM 11.14, SIM Application Toolkit Interface”, version 7.3.1, Release 1998
[4]	“GSM 03.38, Alphabet and language-specific information”, version 7.2.0, Release 1998
[5]	"GSM 03.40, Technical Realization of the SMS-PP", version 7.2.0, Release 1998
[6]	“GSM 04.11, Point-to-Point short message support on mobile radio interface", version 7.0.0, Release 1998
[7]	Across DP5 Product Specification, 17409077.
[8]	Browser Request Protocol Specification, ST17455121
[9]	Push Request Protocol Specification , ST17455122

3 Communication

When sending URL requests from the phone to the web server and WML/HTML pages back to the phone, SMS is used as the communication channel in the GSM network.

Notable is that SMS is a fairly narrow communication channel where the delay has to be taken into account and where the size of the page is of importance. Each SMS can contain approximately 120 bytes of data. If the WML/HTML page is larger than that, the page is sent over several SMS's. However, the delay in the communication increases with each SMS so it is recommended that the number of SMS's sent are kept to a minimum. If a web page is larger that 5 SMS's, the time the end user has to wait for a response may exceed 25 seconds. At present, the browser is able to handle 5 SMS's from the server to the browser and 2 SMS's from the browser to the server.

To calculate the actual number of bytes sent to the mobile phone, use the tool WIG Transformer (supplied by Across).

Document number: ST17455120
Revision: B

4 Browser request

The browser request is initiated from the Mobile Station. Included in the request is a mandatory URL and in some cases an optional query string. The WIG Server receives the request and translates it into an HTTP Get request. The MSISDN of the mobile phone is attached to the HTTP Get request at the end of the form data set (query string).

Example of a HTTP Get message:

www.across.se?first+name=Jim&last+name=Brown&MSISDN=0701234567

A more detailed description of the protocol between the Wig Server and the content provider can be found in ref. [8], Browser Request Protocol Specification.

5 Push request

The push request is initiated from the content provider. The push request contains three parts, an XML document (control entity), a content entity and a capabilities entity. In the first part, the control entity the WAP Push Access Protocol (PAP) is used to describe all control information needed to deliver the message. In the second part the information intended for the client is carried, the WML page. The third part is not used by the Wig Server at the moment.

The PAP protocol and its usage in the WIG Server are described in detail in ref. [9], Push Request Protocol Specification. The important elements and its attributes are though described here.

The attribute address-value included in the address element is used to communicate the MSISDN of the wireless client.

The attribute ppg-notify-requested-to included in the push-message element is used to communicate the host and path to send the confirm message to.

The attribute delivery-method included in the element quality-of-service is used to indicate if a confirmed message should be sent to the initiator of the request. The attribute should then be set to confirmed.

Example of a control entity with the above elements used.

```
<?xml version='1.0'?>
<!DOCTYPE pap PUBLIC \"-//WAPFORUM//DTD PAP
1.0//EN\\"http://www.wapforum.org/DTD/pap_1.0.dtd\">
<pap>
<push-message ppg-notify-requested-to=”http://www.across.se/test2/test”>
<quality-of-service delivery-method= “confirmed”>
</quality-of-service>
<address address-value="+46070123456">
</address>
</push-message>
</pap>
```

6 The Plug-in architecture

Commonly there are things an application would like to do that are not automatically supported by HTML/WML. In the Internet world that is solved by the notion of plug-in, that is an external application that ads new functionality to the browser. In the same way the WIG browser can be extended with plug-ins to perform actions that are not standard.

One example of a plug-in is if the application wishes to have a certain security scheme implemented. That could be solved using a plug-in that performs some specific security operation on some data before it is sent back to the content provider.

Other examples would be if the application requires other SAT commands than the ones supported by the browser.

The syntax for calling a plug-in is specified in WML specification – Wireless Internet Gateway [1].

New plug-ins may be specified and added to the browser. For the details on how to do that, contact the SIM vendor.

7 WML and HTML tags

This chapter describes each WML and HTML tag that needs special attention and points out the constraints applicable for each one.

It is generally wise to shorten the tags as much as possible in the WML used in the applications, e.g., shorten variable and parameter names, etc. This is most often possible without loosing the name space consistency in your WML pages.

Some web page development environments and web servers have the option of adding default parameters in your HTML/WML tags as soon as a web page is saved. Make sure that this feature generates syntactically correct WIG/WML parameters [1].

7.1 Text

The important thing to know about text is that there is a limit for how many characters the mobile phone can display at the same time. The limit is 110 characters.

In order to be able to display more than these 110 characters, text has to be separated by two carriage returns. This will force the mobile phone to display the first paragraph, implying that the user must press OK before proceeding to see the next paragraph.

Example:

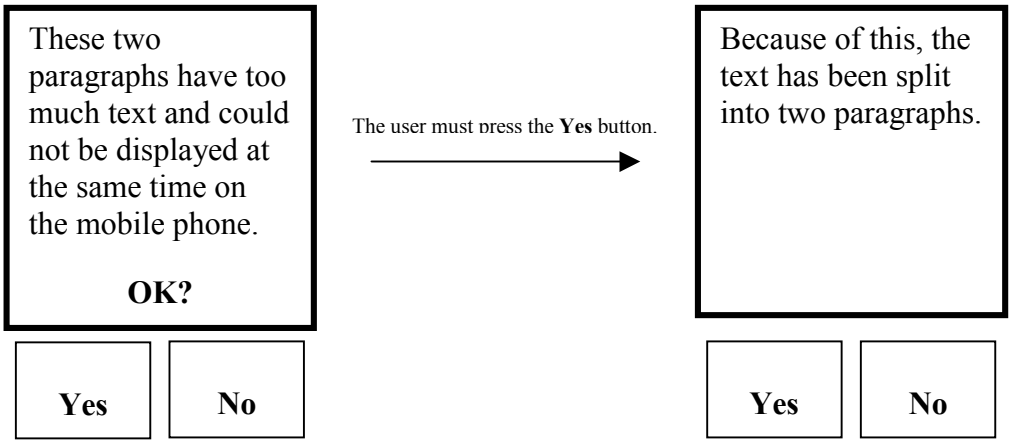
<WML>
<CARD ID=”Main”>

These two paragraphs have too much text and could not be displayed at the same time on the mobile phone.

Because of this, the text has been split into two paragraphs.

</CARD>
</WML>

Result:



7.2 Input

Some mobile phones does not allow the use of TYPE=”PASSWORD” together with alphanumeric input. Thus, for the TYPE=”PASSWORD” feature to work on all mobiles it has to be used together with FORMAT=”*N”.

7.3 Card

In version 1.0 and 1.1 of the WIG server the CARD tag has one feature worth mentioning. The browser does not stop executing when a `</CARD>` is encountered. This limitation has been removed in WIG server 2.0.

Example 1

```
<WML>
<CARD ID=”Main”>
Hi!
</CARD>

<CARD ID=”Next”>
Hi again!
</CARD>
</WML>
```

When this deck is executed the user will first see the text “Hi!” then the browser will fall through to the next card to display the text “Hi again!”

To solve this one rewrites the WML like example 2 below.

Example 2:

```
<WML>
<CARD ID=”Main”>
Hi!
<GO HREF=”#End”/>
</CARD>

<CARD ID=”Next”>
Hi again!
</CARD>

<CARD ID=”End”>
</CARD>
</WML>
```

In this case the browser will jump to the end of the deck and thus stop the execution after having displayed the text “Hi!”

7.4 Select

As with text, there are a limited number of characters that can be used in a SELECT tag. The sum of all character displayed on the phone can not exceed 110, that is the heading plus the text in each OPTION.

Example:

```
<SELECT TITLE=”Colour” NAME=”C”>
<OPTION VALUE=”1”>Blue</OPTION>
<OPTION VALUE=”2”>Red</OPTION>
<OPTION VALUE=”3”>Yellow</OPTION>
</SELECT>
```

In this example the number of characters is “Colour” = 6 + “Blue” = 4 + “Red” = 3 + “Yellow” = 6 = **19**.

The attributes INAME and NAME can not be used at the same time. Also, if INAME is used the VALUE attribute in the OPTION tag may not be used.

7.5 Option

The ONPICK attribute may only contain references to a CARD and not a full URL.

Example 1:

<SELECT TITLE=”Colour” NAME=”C”>
<OPTION ONPICK=”#Main”>Blue</OPTION>
</SELECT>

.

.

.

<CARD ID=”Main”>

</CARD>

The example is correct.

Example 2:

<SELECT TITLE=”Colour” NAME=”C”>
<OPTION ONPICK=”http://server/path/file.wml”>Blue</OPTION>
</SELECT>

The example is not correct.

In the browser version 1.0 there is a limitation on to where you can jump using ONPICK. It is not possible to jump “backwards” in a deck. This limitation has been removed in browser version 1.1.

The following example does works because in the card “Next” in the ONPICK tag jumps to a card “after” the current card.

Example 3:

```
<WML>
<CARD ID=”Main”>
<SELECT TITLE=”Colour” NAME=”C”>
<OPTION ONPICK=”#Next”>Blue</OPTION>
</SELECT>
</CARD>

<CARD ID=”Next”>
Hi!
</CARD>

</WML>
```

However, the following example does not work since the ONPICK jumps to a card that is “before” the current card.

Example 4:

```
<WML>
<CARD ID=”Main”>
Hi!
<GO HREF=”#Next”/>
</CARD>

<CARD ID=”Next”>
<SELECT TITLE=”Colour” NAME=”C”>
<OPTION ONPICK=”#Main”>Blue</OPTION>
</SELECT>
</CARD>

</WML>
```

7.6 Go

In the browser version 1.0 there is a limitation on to where you can jump using GO. It is not possible to jump “backwards” in a deck. This limitation has also been removed in browser version 1.1.

The following example does work because in the card “Main” the GO jumps to a card that is “after” the current card.

Example 1:

```
<WML>
<CARD ID=”Main”>
<GO HREF=”#Next”/>

</CARD>

<CARD ID=”Next”>
Hi!
</CARD>

</WML>
```

However, this example does not work since the GO in card “Next” jumps to a card that is “before” the current card.

Example 2:

```
<WML>
<CARD ID=”Main”>
Hi!
<GO HREF=”#Next”/>
</CARD>
<CARD ID=”Next”>
Hi again!
<GO HREF=”#Main”/>
</CARD>

</WML>
```

When a GO tag is executed the browser does not stop the execution of the rest of the card. In the example below the browser would first send a new HTTP request and then display the text “Hi!”

Example 3:

```
<WML>
<CARD>
<GO HREF=http://server/path/file.wml/>
Hi!
</CARD>
</WML>
```

However, it is strongly recommended that nothing further shall be executed in the CARD after a GO tag, except a jump to an 'End' CARD. It may cause undefined behaviour. Thus WML pages like the example above shall be avoided.

8 WML application examples

This chapter contains a number of WML example applications to get an idea of how to write a WML application.

Please note that all URL's in the example code are invalid.

8.1 Example 1

Navigation between cards within a WML deck.

```
<wml>

<card id="main">
Card selection example.
<select title="Select card">
<option onpick="#CARD1">Card1</option>
<option onpick="#CARD2">Card2</option>
</select >
</card>

<card id="CARD1">
Now you're in CARD1.
</card>

<card id="CARD2">
Now you're in CARD2.
</card>

</wml>
```

8.2 Example 2

Navigation between WML decks.

```
<wml>
<card id=”main”>
WML deck navigation example.
<select title=”Select Link“ name=”PATH”>
<option value=”banking/index.wml”>Banking</option>
<option value =”info/index.wml “>Info</option>
</select >

<go href=”https://hostname.se/$(PATH)”/>
</card>
</wml>
```

8.3 Example 3

Play tone example. A dial tone will be played during 3 second and the text “Tone!” will be displayed at the same time.

```
<wml>
<card id=”main”>
This example will play a tone!
Press ok.
<go href=”http://www.across.se#wigPLAYTONE(0.1,0.2,30,Tone!)”/>
</card>
</wml>
```

8.4 Example 4

Example of navigation between WML decks. Notice that all decks in the example below are different physical files.

<wml>
<card id=”main”>
Deck selection example.
<select title=”Select card“ name=”service”>
<option value="deck1.wml">Link1</option>
<option value="deck2.wml">Link2</option>
</select>
<go href=”http://acrosswireless.se/$(service)”/>
</card>
</wml>

<wml>
<card id=”Service1”>
Welcome to service 1!
<input title=”Please enter your firstname.” type=”text” name=”firstname”/>
You entered $(firstname).
</card>
</wml>

<wml>
<card id=”Service2”>
Welcome to service 2!
Location data will be sent to location service!
Press ok!
<go href=http://www.across.se#wigProvideLocalInfo(00,Info)**”/>**
<go
href=”http://www.acrosswireless.com/locationservice/position.asp?info=$(Info)”>
</card>
</wml>

9 Browser configuration

This chapter requires some knowledge about the WIG system and how it uses the GSM network. For an overview of the product, read the Product Specification – Wireless Internet Gateway [7].

On the SIM card where the browser resides there are a number of files used to define the behaviour of the browser. All files used by the browser are listed below. The files are stored in the directory 2700 directly under the master file.

This section describes each file and how they are used by the browser.

Note that for browser version 1.0 only the files 6F01 to 6F04 are applicable. Browser version 1.1 and up use the rest of the files.

9.1 EF TAR (6F01)

The file contains the Toolkit Application Reference values that the browser listens to. The file might contain several records. Incoming 03.48 messages that do not contain a TAR value listed in any of the records of this file are discarded.

When the browser is sending a request to the server as a result of the user selecting a menu item on the phone, the TAR value in record number one is used. When the browser sends a request as a result of an incoming message from the server, the TAR value in the received message is used.

Identifier: '6F01'	Structure: linear fixed	Optional
Record length : 3 bytes	Update activity: low	
Access Conditions: READ ADM UPDATE ADM INVALIDATE ADM REHABILITATE ADM		

Bytes	Description	M/O	Length
1-3	TAR	M	3 byte

Details

Field	Contents	Coding
TAR	Toolkit Application Reference	As defined in [2]

9.2 EF Error Text (6F02)

If an error occurs when the browser is executing a byte code string, the error is presented to the user in the format “ErrorMessage: ErrorCode AdditionalInfo”.

The text “ErrorMessage” is fetched from the file 6F02. If the file is not present or if the particular error is not included in the file, the default text “Error” is used.

The text “ErrorCode” is one of the following:

File Access Errors

Error code (range 1-1F)	**Description**
0x1	Failed to find/read byte code file
0x2	Failed to find/read TAR file
0x3	Failed to access/read EF (Error Text)
0x4	Failed to find/read EF (SMS header)
0x5	Failed to read EF (Key File)

Byte Code Errors

Error code (range 20-3F)	**Description**
0x20	Unknown command found
0x21	Variable substitution failed
0x22	Too many variables used
0x23	Out of variable memory
0x24	Byte code too large to handle
0x25	SMS TPDU Tag in incoming SMS not found
0x26	Creation of SELECT ITEM failed
0x27	Encryption/decryption failed

Error code (range 20-3F)	**Description**
0x28	Out of buffer space
0x29	Plug-in not found

Mobile Phone Error

Error code (range 40-5F)	**Description**
0x40	Proactive command rejected by ME
0x41	Wrong type of command returned by ME in terminal response
0x42	Get Input did not return a string
0x43	No item identifier was returned by ME in the terminal response to a Select Item
0x44	ME returned a temporary error
0x45	Error in format of received SM
0x46	Command not supported according to TERMINAL_PROFILE

The text “AdditionalInfo” contains data that is only useful for support matters.

Identifier: '6F02'	Structure: transparent	Mandatory
File size: X bytes	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1	Length of entry #1 (1+X)	M	1
2	Error identifier #1	M	1
3	Data Coding Scheme	M	1
X	Alpha Identifier	M	X
3+X	Length of entry #2 (1+Y)		
3+X+1	Error identifier #2	...	1
3+X+2	Data Coding Scheme	...	1
...	...	...	...
Z	Table is ended when length = 0	M	1

Details

Field	Contents	Coding
Length	Length of the alpha identifier.	-
Data Coding Scheme	The Data Coding Scheme to be used for the Alpha Identifier	According to GSM 03.38
Alpha Identifier	Error message to be displayed to the user in case of an error.	According to the Data Coding Scheme.

9.3 EF Byte Code (6F03)

This file contains all byte codes that may be run by the browser when the user selects an item in the menu. It thus contains the start WML of all the web applications implemented using the WIB and the WIG.

If you have three applications, the submenu file will have three records representing the applications. Each record will have a text showing the application name to the user. Each record will also have a pointer to a WML deck in this file (6F03). Each WML deck is encoded into a byte code string.

The byte code strings are put one after another in this file. The first string is executed when the user selects the first menu item on the phone. The second string is used if the second menu item is selected and so on.

Identifier: '6F03'	Structure: transparent	Mandatory
File size: X bytes	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1-2	Length	M	2
X	Byte code	M	X bytes
	…		

Details

Field	Contents	Coding
2-byte length	Length of the following byte code.	-
Byte code	Browser byte code	a series of commands coded as defined in [1]

9.4 EF SMS Header (6F04)

When the browser is sending a request to the server there are some SMS parameters needed for the message to reach the server. Those are defined in this file.

Identifier: '6F04'	Structure: linear fixed	Optional
Record length: 27 bytes	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1 to 12	TP-Destination Address	M	12 bytes
13 to 24	TS-Service Centre Address	M	12 bytes
25	TP-Protocol Identifier	M	1 byte
26	TP-Data Coding Scheme	M	1 byte
27	TP-Validity Period	M	1 byte

Details

Field	Contents	Coding
TP-Destination Address	As defined for SM-TL address fields in GSM 03.40 ref. [5].	Unused bytes are set to FF.
TS-Server Centre Address	As defined for RP-Destination address Centre Address in GSM 04.11 [6]	Unused bytes are set to FF.
TP-Protocol Identifier	As defined in GSM 03.40 [5].	
TP-Data Coding Scheme	As defined in GSM 03.38 [4]	
TP-Validity Period	As defined in GSM 03.40 [5] for the relative time format	

9.5 EF 03.48 Header (6F05)

The communication between the browser and the WIG server relies on the standard GSM 03.48. The browser uses the parameters in this file when sending requests to the server but also to check messages received from the server. For the details, please read ref. [2].

Identifier: '6F05'	Structure: linear fixed	Optional
Record length: 4 bytes	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1 to 2	Security Parameter Indicator (SPI)	M	2 bytes
3	Ciphering Key Identifier (KIc)	M	1 byte
4	Key Identifier (KID)	M	1 byte

Details

Field	Contents	Coding
SPI	Security Parameter Indicator	As defined in [2]
Kic	Ciphering Key Identifier	As defined in [2]
KID	Key Identifier	As defined in [2]

9.6 EF 03.48 Counter (6F06)

As with the file 6F05, this file is used by GSM 03.48.

The file shall contain two records. The first record contains the counter for outgoing messages and the second record hold the counter for incoming messages.

Identifier: '6F06'	Structure: linear fixed	Optional
Record length: 5 bytes	Update activity: High	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1 to 5	Counter (CNTR)	M	5 bytes

Details

Field	Contents	Coding
CNTR	Counter	As defined in [2]

9.7 EF Version Information (6F07)

The version information file contains the version of the browser as well as information on the plug-ins loaded on the card.

Identifier: '6F07'	Structure: transparent	Optional
Record length: X bytes, X>=5	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1	Manufacturer	M	1 byte
2-4	WIB version number	M	3 bytes
5	Number of following plug-in entries	M	1 byte
6	Length of plug-in name	O	1 byte
7-(7+x)	plug-in name	O	x bytes
(8+x)-(11+x)	Manufacturer-specific plug-in version	O	3 bytes
...	...	O	...

- Version number coding:

First and second byte: System version id of the WIG: First byte 1 and second 1 means WIG release1.1. For this version of the browser the first byte shall be set to 1 and the second byte shall be set to 1.

Third byte: manufacturer-specific WIB version.

9.8 EF WIB Configuration (6F08)

Identifier: '6F08'	Structure: Transparent	Optional
Record length: X bytes, X>=1	Update activity: low	

Access Conditions:

READ	ADM
UPDATE	ADM
INVALIDATE	ADM
REHABILITATE	ADM

Bytes	Description	M/O	Length
1	Persevere counter initial value	M	1
2..X	RFU	O	X-1

- Persevere counter initial value:

When the WIB is waiting for a response from the WIG server and the user starts the browser, a PLAY TONE is issued. If the user retries starting the browser a few times, the selection is finally accepted. This counter indicates how many times a tone is played before the browser starts. If '0' the function is disabled, if '1', the browser is started on the second attempt etc.

- RFU:

Set to 'FF'

10 Personalisation issues

When an SIM card is personalised in order to be used as a browser card there are some issues to consider. This chapter contains the things that are important to think about when personalising the card.

10.1 File sizes

When a service is first launched the initial menu and byte code files are put on the card. However, when a second service is launched, it might be necessary to add more entries in the menu file and to put some more byte code in the byte code file. To accommodate this, the menu file and the byte code file should be big enough to be able to handle additional data at a later stage.

10.2 OTA data download

There are situations in the life cycle of the SIM card where there is a need of updating the card using OTA technology. The menu file and the byte code files could be updated using OTA when a second service is introduced. This should be taken into consideration when personalising the card and if desirable, some or all of the browser files should be updateable OTA.

10.3 Browser application size

The size of the browser differ depending on card supplier and to get an exact figure, please contact the SIM card supplier of your choice. In addition to that the browser configuration files will take some space. Notable is the byte code file.

In view of the above, it is recommended that the browser is run on a 32 Kb SIM card. It is possible to use a 16 Kb card but there will not be much space left for other GSM files.

11 Browser SM vs Server SM

The Wig server supports two different ways of sending text SM. One that is called “server SM” is sent as an ordinary SM directly to the mobile phone. The other, here called “browser SM” is sent as byte code to the WIG browser where the browser generates a SM and sends to the mobile phone. To generate either of the different SM’s from the wml page, the WML Script syntax described in detail in ref. [1] is used. The function call for “server SM” is wigSendServerSM and the function call for “browser SM” is wigSendSM.

12 WIG Browser 1.0 vs WIG Browser 1.1

The WIG Browser 1.1 has the same functionality as the WIG Browser 1.0 and it also supports the following new functionality.

Support for new SIM Application Toolkit Commands.

- Provide Local Information
- Play Tone
- Set Up Idle Mode Text
- Refresh
- Setup Call
- Assign Version Information to Variable
- Assign Script Buffer Size To Variable
- Newcontext
- Set Return Tar
- Send USSD
- Send Text SM

The Newcontext command is initiated by the card tag as an attribute named newcontext.

The rest of the new SIM Application Toolkit commands are initiated with WML script syntax described in ref. [1].

The GetInput command that corresponds to the input tag has support for two new attributes, emptyok and default value.

Unicode is now supported.

The possibility to jump between cards has been extended to also support backward jump.

03.48 security is implemented.

13 Unicode

The Wig server supports unicode but for the unicode to work the prologue element has to be included first in the page and the encoding should be set to utf-8.

<?xml version=”1.0”? encoding=”utf-8”>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

……..

</wml>

14 Supported WML elements and attributes

Element	Attribute
wml	xml:lang
card	id
	newcontext
input	name
	type
	value
	format
	emptyok
	maxlength
	title
select	title
	name
	iname
option	value
	onpick
setvar	name
	value
	class
noop	
do	type