



Specification  
**WIG WML v. 5**

Document number: SALC/UG/Spec/09:877776

Revision: C. 2010-03-09



© 2006, 2008, 2009 SmartTrust AB. All rights reserved.

SmartTrust endeavors to ensure that the information in this document is correct and fairly stated, but does not accept liability for any error or omission. The development of SmartTrust products and services is continuous and published information may not be up to date. It is important to check the current position with SmartTrust. This document is not part of a contract or license save insofar as may be expressly agreed.

Unless otherwise noted, all names of companies, products, street addresses and persons contained herein are part of a completely fictitious scenario and are designed solely to document the use of the described product or service.

SmartTrust and SmartTrust Wib are trademarks of SmartTrust AB.

All the other trademarks are the property of their respective owners.



## Contents

1 Introduction	6
1.1 Delivery Platform compatibility	6
1.2 Backward compatibility	6
2 References	7
3 Definitions and abbreviations	9
4 Document structure	10
5 WIG WML syntax	11
5.1 The WIG WML document	11
5.2 Prologue	11
5.3 Comments	11
5.4 Elements	12
5.5 Attributes	12
5.6 White space	12
5.7 Variables	12
5.8 Icons	15
6 Character set	17
6.1 Encoding	17
6.2 Character entities	18
6.3 Numeric character references	18
6.4 Hexadecimal escape characters	18
7 Data types	20
7.1 float(x-y)	20
7.2 hex-bin	20
7.3 ID	20
7.4 integer(x-y)	20
7.5 string	20
7.6 variable-name	20
7.7 WED-string	20
7.8 Wib-URI	21
8 Presentation elements	23
8.1 wml Element	23
8.2 card Element	24
8.3 p Element	25
8.4 br Element	27
8.5 input Element	28
8.6 select Element	30
8.7 option Element	32
8.8 setvar Element	34
8.9 go Element	36
8.10 postfield Element	40



8.11 progressinfo Element .....	40
8.12 bookmarkinfo Element .....	42
9 STK command oriented elements .....	44
9.1 launchbrowser Element .....	44
9.2 bearer Element .....	46
9.3 playtone Element .....	46
9.4 providelocalinfo Element.....	48
9.5 refresh Element .....	49
9.6 file Element.....	50
9.7 network Element.....	51
9.8 sendsm Element.....	52
9.9 destaddress Element.....	54
9.10 servicecentreataddress Element .....	55
9.11 userdata Element.....	55
9.12 sendussd Element.....	58
9.13 setupcall Element.....	60
9.14 confirminfo Element.....	61
9.15 setupinfo Element .....	62
9.16 setupidlemodetext Element.....	63
9.17 executestk Element .....	64
9.18 tlv Element.....	66
9.19 sendss Element.....	66
9.20 catcherror Element.....	68
10 Wib specific elements .....	70
10.1 add Element .....	70
10.2 sub Element .....	72
10.3 checkterminalprofile Element.....	73
10.4 check Element.....	75
10.5 conditionaljump Element.....	77
10.6 test Element .....	79
10.7 convert Element .....	80
10.8 getbuffersize Element .....	84
10.9 getbrowserinfo Element.....	84
10.10 plugin element .....	86
10.11 setreturntarvalue Element .....	87
10.12 substring Element .....	88
10.13 swapnibbles Element .....	90
10.14 timer Element.....	90
10.15 transcode Element.....	92
10.16 groupvar Element.....	93
10.17 ungroupvar Element.....	94
10.18 var Element.....	95
10.19 terminalprofile Element .....	96
10.20 executewiblet Element.....	97



10.21 handleexit Element .....	97
10.22 createtlv Element .....	99
10.23 extracttlv Element .....	101
10.24 geticcid Element .....	102
11 Server Side elements .....	104
11.1 wigplugin Element .....	104
11.2 param Element .....	104
12 Other elements .....	106
12.1 head Element .....	106
12.2 meta Element .....	106
Appendix A Examples of WIG WML documents .....	108
Appendix B Character encoding and conversions made by the UG .....	110
B.1 Conversion of server-bound messages .....	110
Appendix C UG Side plug-ins .....	112
C.1 sendserverism .....	112
C.2 sendserverdatasm .....	113
C.3 noresponse .....	115
C.4 applevelcharginginfo .....	116



## 1 Introduction

This document specifies version 5 (v5) of the *WIG Wireless Markup Language (WIG WML)*. The purpose is to provide a formally correct definition as well as to provide some examples of how language features are used. The intended audience is primarily application developers that develop or are about to develop Wib services.

The prime intention of the WIG WML language is to serve as a page description language for pages that may be rendered/executed using *SmartTrust Wib™* in combination with the *SmartTrust Delivery Platform (DP)*.

### 1.1 Delivery Platform compatibility

The WIG WML specified in this document is supported by the Universal Gateway (UG) Server from version 6.0. This corresponds to Delivery Platform version 8.x and newer.

### 1.2 Backward compatibility

The WIG WML v5 syntax is fully backward compatible with versions 4.x.

Refer to *UG & WSM – Application Developers Guide – Development of Wib Services* [10] for further information.



## 2 References

- [1] 3GPP. *TS 23.040. Technical realization of the Short Message Service (SMS)*. Version 5.1.0 (2001-09). Available: <http://www.3gpp.org/>
- [2] Ericsson Mobile Communications AB. *Enhanced Messaging Service – White Paper*. April 2001. Publication nr. LZT 108 4854 R1C. Available: <http://www.ericsson.com>
- [3] ETSI. *GSM 02.30. Man-Machine Interface (MMI) of the Mobile Station (MS)*. Version 7.1.0. Release 1998.
- [4] ETSI. *GSM 03.38. Alphabets and language specific information*. Version 7.2.0. Release 1998.
- [5] ETSI. *GSM 04.08. Mobile radio interface layer 3 specification*. Version 7.10.0. Release 1996.
- [6] ETSI. *GSM 07.05. Equipment (DTE - DCE) interface for SMS and CBS*. Version 5.5.0. Release 1998.
- [7] ETSI. *GSM 11.11. Specification of the Subscriber Identity Module – Mobile Equipment (SIM - ME) interface*. Version 8.8.0. Release 1999.
- [8] 3GPP *TS 11.14 V8.18.0 (2007-06) 3rd Generation Partnership Project; Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface* (Release 1999).
- [9] Nokia. *Smart Messaging Specification*. Rev. 3.0.0. 2000-12-18. Available: <https://secure.forum.nokia.com/>
- [10] *UG & WSM – Application Developers Guide – Development of Wib Services*, SmartTrust.
- [11] *Wib™ Plug-in Specification*, SmartTrust.
- [12] *WML Specification – Wireless Internet Gateway – Delivery Platform 6*, doc. no. 60084047, SmartTrust.
- [13] Sony Ericsson, et al. *How to Create EMS Services*. Version 1.2 September 2002. Available: <http://www.ericsson.com>
- [14] Sony Ericsson. *Enhanced Messaging Service (EMS) – Developers Guidelines*. September 2002. Publication nr. EN/LZT 108 5256 R2A. Available: <http://www.ericsson.com>
- [15] W3C. *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation 6 October 2000. Available: <http://www.w3.org/>
- [16] -void
- [17] Collective reference to STK see [8], [18] and [19]



- [18] 3GPP TS 31.111 V8.1.0 (2008-04) 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) (Release 8)
- [19] ETSI TS 102 223 V7.10.0 (2008-02) Smart Cards; Card Application Toolkit (CAT) (Release 7)
- [20] W3. XML Schema. Available: <http://www.w3.org/2001/XMLSchema>
- [21] 3GPP TS 31.102 V8.1.0 (2008-03) 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Characteristics of the
- [22] Universal Subscriber Identity Module (USIM) application (Release 8)



### 3 Definitions and abbreviations

<b>Acronym</b>	<b>Definition</b>
BCD	Binary Coded Decimal
DCS	Data Coding Scheme
DP	SmartTrust Delivery Platform
DTD	Document Type Definition
EMS	Enhanced Messaging Service
MAC	Message Authentication Code
ME	Mobile Equipment
MCC	Mobile Country Code
MNC	Mobile Network Code
NSM	Nokia Smart Messaging
SIM	Subscriber Identity Module
SMS	Short Message Service
STK	SIM Application Toolkit
TLV	Tag Length Value
UCS	Universal Character Set
WAP	Wireless Application Protocol
Wib	SmartTrust Wib™
wiblet	A program that can be executed in Wib runtime platform.
WIG	Wireless Internet Gateway
WML	Wireless Markup Language
XML	eXtensible Markup Language



## 4 Document structure

This document has two major parts, one introductory part consisting of chapter 5 – 7 where topics that are general or common to WIG WML are described. In the syntactical part, covered by chapters 8 – 12, the language and its elements are described in detail.

In order to make the syntactical part more manageable, the presentation has been divided into 5 separate chapters, where each chapter covers a subset of the elements of WIG WML.

- Chapter 8 – Presentation elements. This chapter covers the most frequently used elements in WIG WML, namely those that deal with presentation, navigation and input/output. These are also the elements that are most strongly influenced by (WAP) WML.
- Chapter 9 – STK command oriented elements. This chapter covers elements that have a strong relationship with STK commands offered by the SIM/ME.
- Chapter 10 – Wib specific elements. This chapter covers elements that are specific to Wib in the sense that they are not tightly linked to a specific STK command. The elements cover various aspects of Wib such as doing arithmetic, text conversion and calling extended functionality.
- Chapter 11 – Server Side elements. This chapter covers elements that are exclusively “executed” in the UG.
- Chapter 12 – Other elements. This chapter covers elements that do not belong in any of the previously mentioned groups.



## 5 WIG WML syntax

An XML Schema Definition (XSD)[20] is used to formally describe the syntax of WIG WML. The XSD is published at <http://www.smarttrust.com/xsd/wigwml-5.0.xsd>.

**Note:** The authoritative reference to WIG WML is represented by the syntactical description in the following chapters, and not by the XSD, which may be considered as supplementary information.

### 5.1 The WIG WML document

A WIG WML document is typically stored on a web server. It may also be compiled into executable form, called a wiblet, and stored in the Wib client.

The WIG WML document can be written in ISO-8859-1 (Western), ISO-8859-7 (Greek), or in UTF-8 format (Unicode), which is the default. The character encoding should be stated in the XML Declaration. See Chapter 6 for more information.

### 5.2 Prologue

The WIG WML document shall start with an XML declaration and a reference to a WIG WML XML schema. It is an error to omit the prologue. Several examples in this specification will omit the prologue and other entities for purposes of compactness.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
. . .
</wml>
```

### 5.3 Comments

Comments have the following syntax:

```
<!-- a comment -->
```

Comments are filtered out and do not affect the size of the wiblet.



## 5.4 Elements

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

```
<tag/>
```

or

```
<tag>content</tag>
```

Note that the order of the elements in a WIG WML document is significant, where nothing else is stated, since Wib will interpret the elements in sequence.

## 5.5 Attributes

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

```
<tag attr="abcd"/>
```

or

```
<tag attr="abcd">content</tag>
```

All attribute values must be quoted using single (') or double (") quotation marks.

## 5.6 White space

For content within an element (not attribute values), the following rule applies: White space immediately before and after an element is ignored. In addition, all other sequences of white spaces are compressed into a single inter-word space.

The numeric character reference `&#xA0;` (non-breaking space) can be used for indicating white space that shall be preserved.

## 5.7 Variables

### 5.7.1 Using 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, a `$` character followed by `(VARIABLENAME)` indicates a variable substitution. Variable names are case sensitive. Note that in some attributes (e.g. in `setvar`) only the variable names (without `$`) should be used. See Example [1]. When variables otherwise are referred, this is the only syntax allowed:

```
$(VARIABLENAME)
```

Variables have to be named with characters supported by ISO-8859-1. The length of the variable name does not affect the size of the wiblet.



A variable can have content equal to the empty string (""). The maximum length of the content of a variable is 8191 bytes.

Different variables may contain characters from different character sets. The encoding of a variable is set the first time the variable is defined in the WIG WML document (for instance in a `setvar`, `input` or `select` element or in a Wib plug-in call). See also Appendix B.

A sequence of two dollar signs (\$\$) represents a single dollar sign, where variable syntax is legal.

### Example [1]

This example will display the text "Item: CD Price: \$10".

```
<card>
  <p>
    <setvar name="ITEM" value="CD"/>
    <setvar name="PRICE" value="10"/>
    Item: $(ITEM) Price: $$$ (PRICE)
  </p>
</card>
```

## 5.7.2 Variable persistence

Variables come in two different flavors linked to their maximum persistence in Wib:

*Local variables* – Variables that are created during the execution of a wiblet and deleted automatically when the (same) wiblet stops executing. Local variables occupy variable IDs in the range 0x00 to 0xDF, except 0x80, 0x81 and 0x82 which are reserved for Wib internal purposes. This is the most common variable type supported by all Wib versions.

*Global variables* – Variables that are persistent throughout the execution of multiple wiblets. Global variables occupy variable IDs in the range 0xE0 to 0xFF. Global variables are cleared by SIM reset, normally caused by a ME power-off. Global variables may also be cleared in other ways. See *Guidelines – Development of Wib Services* [10] for details about Wib variable handling. Global variable support is intended primarily for passing data to and from wiblets.

*Compatibility note: Global variables are only applicable for Wib 1.3 and later.*

## 5.7.3 Specifying Wib variable ID in variable names

Wib keeps track of local variables with an identifier value in the range 0x01–0xDF except for the values 0x80, 0x81 and 0x82 which are reserved. It is possible to specify in the WIG WML document what identifier value Wib should use for a variable. This is done by suffixing the variable name with the identifier value it is supposed to take.



NAME:IDxHH

HH is in hexadecimal format, and it should be replaced by the desired value. It is also legal to omit the variable name completely but not the colon.

:IDxHH

This syntax is useful when creating WIG WML documents for event triggered wiblets. See *Guidelines – Development of Wib Services* [10] for further information.

Note that variables that do not have a specified identifier will automatically be assigned identifiers from 1 and upwards. This means that it may not always be possible to assign a certain identifier to a variable if the identifier already is in use.

### Example [2]

This example uses the variable identified by the hexadecimal value "91" in Wib.

```
<card>
  <p>
    <go href="http://webserver/update.py?loc=$(LOC:IDx91)"/>
  </p>
</card>
```

### 5.7.4 Locally stored wiblet variable passing

*Compatibility note: This section is only applicable for Wib 1.3 and later.*

8 variables are reserved for parameter passing to and between locally stored wiblets. To indicate that a variable is intended for parameter passing, the variable name must be suffixed with `stack01 – stack08`. E.g.:

NAME:stackDD

DD is in decimal format, and it should be replaced by the desired value. It is also legal to omit the variable name completely but not the colon.

:stackDD

20 variable names are reserved for global variables. In the same way as with parameter variables, global variables must be suffixed with `global01 – global20`.

NAME:globalDD

or

:globalDD

Wib treats global variables and stack variables in exactly the same way, and that is according to the global variables in Section 5.7.2 . The recommendation for the application developer is to use `stack` variables for parameter passing and to use `global` variables only when needed, for example when calling another wiblet which is expected to return to the calling wiblet.



### Example [3]

This example would display "Veni vidi vici". Since the default behavior is to clear variables on wiblet entry, the called wiblet must override this using the `clearonentry` attribute.

Calling wiblet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <card>
    <p>
      <setvar name="veni:global01" value="Veni"/>
      <setvar name="vici:stack01" value="vici"/>
      <go href="wiblet://smarttrust.com/sample/caesar.wml"/>
        $(veni:global01) $(vici:stack01) $(:stack02)
    </p>
  </card>
</wml>
```

Called wiblet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd"
      clearonentry="false">
  <head>
    <meta name="wiblet-uri"
          content="wiblet://smarttrust.com/sample/caesar.wml"/>
  </head>
  <card>
    <p>
      <setvar name=":stack02" value="$(:stack01)"/>
      <setvar name=":stack01" value="vidi"/>
      <go href="wiblet://return"/>
    </p>
  </card>
</wml>
```

## 5.8 Icons

*Compatibility note: This section is only applicable for Wib 1.3 and later.*

A number of the WIG WML elements specified in this document result in textual information being displayed to the user. In most of these cases, it is also possible to request that an icon should be displayed together with the text or possibly also instead of the text, to improve the user experience.

Two attributes, `iconid` and `iconusage`, are provided with all elements that support the icon feature, to represent the graphical appearance of the icon and the behavior demonstrated when the icon is displayed.



`iconid` – the icon identifier.

`iconusage` – specifies whether the icon should be displayed together with or instead of the accompanying text.

Since icons are an optional feature of the ME, an accompanying text must always be specified together with an icon, even if the intention is to display the icon without text. This is to cover the situation where icons are not supported by the ME, in which case the ME shall display the text as a fallback solution.

Exactly which text that is accompanying an icon is described whenever an `iconid` attribute is included by an element.

**Note:** The following preconditions need to be fulfilled before icons can be used in a WIG WML document:

- 1) The SIM card must be configured with a set of appropriate icons.
- 2) The `iconid` as described above of each icon must somehow be published so that the information is accessible to the application developer.



## 6 Character set

### 6.1 Encoding

The following character encodings are supported in a WIG WML document:

- ISO-8859-1 (Western)
- ISO-8859-7 (Greek)
- UTF-8 (Unicode) (default)

The document shall begin with an XML declaration optionally containing the encoding attribute. For example:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

The encoding of the WIG WML document does not affect the encoding of text strings in Wib. Instead this is determined by the `wml` element attribute `wibletenc`.

```
<wml wibletenc="UCS2">  
...  
</wml>
```

**Note:** Throughout this document, the value of the `wibletenc` attribute in the `wml` element is referred to as the *wiblet encoding*. The term *Wib encoding* refers to the encoding used in Wib for specific data.

If the wiblet encoding is SMS-DEFAULT, text strings in Wib are encoded with the SMS Default Alphabet [4]. Conversions from the document encoding are made according to GSM 07.05 [6]. Any character that can not be represented by the SMS Default Alphabet is replaced by a blank space.

If the wiblet encoding is UCS2, text strings in Wib are encoded with UCS2. UCS2 uses two bytes for every character, i.e. double the space that is needed when using the SMS Default Alphabet.

For certain elements and attributes, for example the `value` attribute in the `input` element, it is possible to override the wiblet encoding. For example, it is possible to specify that the Wib encoding shall be SMS Default Alphabet for the `value` attribute, even if the wiblet encoding is UCS2.

See also Appendix B for details regarding conversions made by the UG.

#### Example [4]

This example shows how Unicode can be used for text that is to be input and output on the mobile station, and for the content of variables. It also shows that the Unicode variable content can be passed to the Content Provider as a parameter value to the "go href" attribute. The URL within "go href", including the query string, must contain valid URL characters. However, the content of the variables that are passed in the query string can be Unicode. In the example, the content of the Wib variable `SELECTION` is in Unicode.



```
<?xml version="1.0" encoding="UTF-8" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd"
      wibletenc="UCS2">
<card id="start">
  <p>
    <select name="SELECTION" title="حساب">
      <option>الرميد او الفاتوره</option>
      <option>شحن الرصيد</option>
      <option>تحويل الرصيد</option>
    </select>
    <go href="http://webserver/path/page.jsp?selection=$(SELECTION)"/>
  </p>
</card>
</wml>
```

## 6.2 Character entities

Character entities are used to specify characters in the document character set which must be escaped in WIG WML. All entities begin with an ampersand and end with a semicolon.

WIG WML allows the following named character entities:

&quot;	quotation mark
&amp;	ampersand
&apos;	apostrophe
&lt;	less than
&gt;	greater than

## 6.3 Numeric character references

WIG WML allows numeric character references according to XML [15].

Examples of numerical character references are "&#x40;" and "&#64;". Both can be used instead of the @ character.

## 6.4 Hexadecimal escape characters

WIG WML supports a specific syntax for entering binary data. A single byte can be entered in the following format:

\xHH

HH is in hexadecimal format, and it should be replaced by the desired value. A byte specified like this will not go through any character encoding. The '\' character is escaped by double '\'. Hexadecimal escape characters are only supported for certain elements and attributes according to this specification.



### Example [5]

The string “Hello World” will be displayed on the screen of the mobile station.

```
<card>
  <p>
    <setvar name="GREETING"
value="\x48\x65\x6c\x6c\x6f\x20\x57\x6f\x72\x6c\x64" class="binary"/>
    $(GREETING)
  </p>
</card>
```



## 7 Data types

This chapter describes the different data types used in this specification for attribute values and element content.

### 7.1 float(x-y)

The *float(x-y)* type represents a decimal float value within the range (x-y).

### 7.2 hex-bin

The *hex-bin* type is used to represent arbitrary hexadecimal-encoded binary data. It has a lexical representation where each binary byte is encoded as a character tuple, consisting of two hexadecimal digits "0" – "F" representing the byte code.

### 7.3 ID

The *ID* type is used with attributes to indicate a unique name not shared by any other *ID* type attribute in the document. An *ID* must start with a letter or underscore.

### 7.4 integer(x-y)

The *integer(x-y)* type represents a decimal integer value within the range (x-y).

### 7.5 string

The *string* type may consist of any legal XML characters where nothing else is stated. The Wib encoding is specified separately where this type occurs.

### 7.6 variable-name

The *variable-name* type represents a variable name. Variables have to be named with characters supported by ISO-8859-1.

### 7.7 WED-string

The *WED-string* type may consist of any legal XML characters and is Wib encoded according to the wiblet encoding or as potentially overridden in a command. (WED = Wiblet Encoding Dependent.)



## 7.8 Wib-URI

The *Wib-URI* type represents one of the following types:

1. *External-URL* - An external URL referring to a wiblet loaded over the-the-air.
2. *Card-reference* - A card in the current WIG WML document.
3. *Wiblet-URI* - A locally stored wiblet.

The following sub chapters describe the types more closely.

### 7.8.1 External-URL

An *external-URL* is an absolute URL referring to a WIG WML document located on the Internet. Relative URLs are not supported.

The URL may contain variable references for instance in the domain name:

```
<go href="http://www. $(HOSTNAME) .com/page.wml"/>
```

or in the host and path:

```
<go href=" $(HOST) $(PATH) ?id=43"/>
```

or in parameter values of the URL:

```
<go href="http://webserver/page.jsp?name=$(NAME) &id=$(ID)"/>
```

When a URL is sent through the UG to a Content Provider, static text is never URL encoded. Variable content, on the other hand, is always URL encoded, with the exception of the characters '/' and ':' if they occur before the query string. This means that the characters '?', '=' and '&' (which has to be written as '&amp;') has to be in static text, to preserve their reserved meaning.

Note: There is one exception to this rule. If one (1) variable is used for the complete URL, no URL encoding at all will be performed in the UG.

The URL normally starts with "http://" or "https://". The latter must be used if SSL shall be used for connecting to the Web server.

The URL is always Wib encoded with the SMS Default Alphabet [4], even if the wiblet encoding is UCS2. However, the URL may contain variable references where the variables contain characters supported by UCS2.

If an *External-URL* is used outside a `go` element (e.g. in the `onpick` attribute of the `option` element), the Wib execution will always terminate after Wib has submitted the request, and the rest of the behavior will be the same as for a plain `go` element without special settings (i.e. `<go href="URL"/>`). See section 8.9 .

The following parameter names are reserved, i.e. they must not be used as parameter names in the query string.



Parameter Name	Description
WPLGN	Reserved for specifying name of server side plug-in. Used when a request is sent from Wib to Content Provider.
MSISDN	Reserved for the MSISDN parameter added to the requests by the UG.
_PS_DT_ _PS_ERROR_ _PS_ERRORRANGE_ _PS_IMEI_ _PS_LATITUDE_ _PS_LI_ _PS_LONGITUDE_ _PS_NMR_ _PS_RADIUS_ _PS_SEMIMAJOR_ _PS_SEMIMAJORDEV_ _PS_SEMIMINOR_ _PS_STATUSCODE_ _PS_STATUSTEXT_ PS_TA	All these parameters are used for positioning services.

### 7.8.2 Card-reference

For referencing a card, a hash sign ('#') is used:

```
<go href="#CARD"/>
```

Variables are not allowed in a *card-reference*.

There is a limitation regarding how many elements there may be between the card-reference and the referenced card.

*Compatibility note: If a card is referenced from a go element and the Wib version is 1.3 or later there is no limitation regarding the number of elements.*

### 7.8.3 Wiblet-URI

*Compatibility note: Wiblet-URI is only supported by Wib 1.3 and later.*

A reference to a locally stored wiblet uses the URL scheme "wiblet":

```
<go href="wiblet://smarttrust.com/foo/bar.wml"/>
```

The Wiblet-URI shall be a valid URL. For returning to a calling wiblet, the following syntax is used:

```
<go href="wiblet://return"/>
```

Only one level of wiblet call/return is guaranteed to work.

Variables are not allowed in a WIBLET-URI.



## 8 Presentation elements

This chapter, as well as subsequent chapters dealing with WIG WML, utilizes the same structure for the presentation of elements. Each element is described from the following perspectives:

<b>Description</b>	Overall description of the purpose of the element as well as general considerations how the element may be used.
<b>Content</b>	Elements that may or must be contained.
<b>Position</b>	Elements that may or must contain the element.
<b>Attributes</b>	<p>A table showing the attributes offered by the element. The table contains the following columns:</p> <p><i>Name</i>; the name of the attribute as well as mandatory/optional characteristics.</p> <p><i>Value</i>; the type of the attribute value. Either a single string value or one of the types described in chapter 7</p> <p><i>Explanation</i>; information regarding the purpose/functionality offered by the attribute. The column may also describe the default value for the attribute (for optional attributes), as well as Wib compatibility information.</p> <p><i>Var</i>; determines if the attribute value may contain variable references. This may depend on the Wib version, in which case it is noted in the <i>Explanation</i> column or in the <b>Compatibility</b> section.</p>
<b>Compatibility</b>	Wib version compatibility information.

### 8.1 wml Element

#### Description

The `wml` element defines a WIG WML document and encloses all information in the document.

#### Content

One or more of the following elements:

`card`, `head`, `wigplugin`

#### Position

Root element.



## Attributes

Name	Value	Explanation	Var
<code>xmlns</code> mandatory	As defined in 5.2	Part of prologue	
<code>xmlns:xsi</code> mandatory	As defined in 5.2	Part of prologue	
<code>xsi:schemaLocation</code> mandatory	As defined in 5.2	Part of prologue	
<code>wibletenc</code> optional	SMS-DEFAULT   UCS2	Determines the default character encoding for text displayed on the screen of the mobile station during the execution of a wiblet. Default value is SMS-DEFAULT.  Throughout this document, the value of this attribute is commonly referred to as the "wiblet encoding".	No
<code>clearonentry</code> optional	true   false	Determines if global variables are cleared when Wib starts executing the wiblet. Default value is true. Compatibility: Wib 1.3 and later.	No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## 8.2 card Element

### Description

The `card` element defines a container of text and elements in a WIG 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.

### Content

Zero or more occurrences of each of the following elements:

`p`, `go`, `plugin`, `providelocalinfo`, `playtone`, `setupidlemodetext`, `refresh`, `setupcall`, `getbrowserinfo`, `getbuffersize`, `setreturntarvalue`, `sendussd`, `sendsm`, `conditionaljump`, `launchbrowser`, `checkterminalprofile`, `substring`, `add`, `sub`, `convert`, `groupvar`, `ungroupvar`, `swapnibbles`, `transcode`, `timer`, `executestk`, `sendss`, `terminalprofile`, `executewiblet`, `handleexit`, `createtlv`, `extracttlv`, `geticcid`

### Position

`card` may only occur as a child of the `wml` element.



## Attributes

Name	Value	Explanation	Var
<b>clear</b> optional	none   local   global   all	This attribute allows for fine grained control of which variables that should be cleared by Wib. This is especially useful when calling locally stored wiblets. Default value is none. Compatibility: local and global is supported by Wib 1.3 and later.	No
<b>id</b> optional	ID	The unique id of the card within the document.	No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## Example [6]

```
<card id="card1" clear="local">
  <p>
    <input type="text" name="FIRSTNAME" title="Please enter your first
name."/>
  </p>
</card>
```

## 8.3 p Element

### Description

The `p` element may contain text to be displayed to the user. Also other elements may be contained within the `p` element. Text contained in a `p` element will be displayed separately to the user. This means that text can be split in separate parts by using more than one `p` element.

Text that exceed a certain length (typically 160 bytes, but the exact figure is ME dependent) within one `p` element will be divided into more than one segment. Each segment will be displayed separately to the user, but will inherit all attribute values (e.g. `iconid`) from the `p` element.

### Content

Any combination of *WED-string* including variable references, and the following elements:

`br`, `input`, `select`, `setvar`, `go`, `plugin`, `providelocalinfo`, `playtone`,  
`setupidlemodetext`, `refresh`, `setupcall`, `getbrowserinfo`,  
`getbuffersize`, `setreturntarvalue`, `sendusdsd`, `sendsm`,  
`conditionaljump`, `launchbrowser`, `checkterminalprofile`, `substring`,  
`add`, `sub`, `convert`, `groupvar`, `ungroupvar`, `swapnibbles`, `transcode`,



timer, executestk, sendss, terminalprofile, executewiblet, handleexit, createtlv, extractlv, geticcid

## Position

p may only occur as a child of the card element.

## Attributes

Name	Value	Explanation	Var
<b>class</b> optional	user   delay	Specifies if the text displayed to the end user should be cleared automatically after a delay (which is ME dependent) or if the end user needs to clear the text himself. Default value is user. Compatibility: delay is only supported by Wib 1.2 and later.	No
<b>continue</b> optional	true   false	Controls whether the text display should halt the wiblet execution until the text is cleared, or if the wiblet is allowed to continue execution immediately. Use true with caution. If the ME does not support it, the text will not be displayed at all. Default value is false. Compatibility: true is only supported by Wib 1.3 and later.	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the text contained by the p element. If an icon ID is specified, a text string must be present within the element. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the contained text. May only be used when the iconid attribute is specified. Default value is adjacent. Compatibility: Wib 1.3 and later.	No
<b>priority</b> optional	high   normal	Priority used when the mobile station displays the text to the end user. normal means that the text will only be shown if the mobile entity is not busy with other tasks. Default value is high. Compatibility: normal is only supported by Wib 1.3 and later.	No



## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

### Example [7]

```
<card>
  <p>
    This text has to be acknowledged by the user before...
  </p>
  <p>
    ...this text will be displayed!
  </p>
</card>
```

### Example [8]

```
<card>
  <p class="delay">
    All text contained in this p element will be cleared after a delay.
    <input title="Number?" name="NUMBER"/>
    Even this text...
  </p>
  <p>
    ...but not this one.
  </p>
</card>
```

### Example [9]

```
<card>
  <p continue="true" iconid="3">
    This text with icon will linger on the screen, but Wib will
    continue execution immediately.
    <go href="http://webserver/page.jsp"/>
  </p>
</card>
```

### Example [10]

If an icon ID is specified, a text string must be present within the p element. The minimum text string is a space, which can be entered as shown by this example.

```
<card>
  <p iconid="3">&#xA0;</p>
</card>
```

## 8.4 br Element

### Description

The <br/> tag inserts a "CRLF" (Carriage Return - Line Feed) sequence into the text.

### Content

None; the element is always empty.



## Position

`br` may only occur as a child of the `p` element.

## Attributes

None.

## Compatibility

Wib 1.1 and later.

## Example [11]

```
<card>
  <p>
    Line 1<br/>
    Line 2<br/>Line 3
  </p>
</card>
```

## 8.5 input Element

### Description

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

The `class` attribute is used for character conversion purposes. It may have the following values:

- `SMS-DEFAULT` - The user of the mobile station will be prompted for an SMS Default encoded string, and the content of the variable will be stored in Wib with characters of the SMS Default Alphabet. If sent through the UG to the Content Provider, the data is converted to ISO-8859-1. The text given by the `value` attribute will be Wib encoded with SMS Default Alphabet.
- `UCS2` - *This value is only possible if the wiblet encoding is UCS2.* The user of the mobile station will be prompted for a Unicode string, and the content of the variable will be stored in Wib with UCS2 encoded characters. If sent through the UG to the Content Provider, the data is converted to UTF-8. Also the text given by the `value` attribute will be Wib encoded with UCS2.

The encoding of the input will per default follow the wiblet encoding. If the wiblet encoding is UCS2, the input may be changed to SMS-DEFAULT. If the wiblet encoding is SMS-DEFAULT, the input can only be given in SMS-DEFAULT.

See Appendix B for more detailed information regarding character conversions performed by the UG.

### Content

None; the element is always empty.



## Position

`input` may only occur as a child of the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	variable-name	The name of the variable to be set.	No
<b>class</b> optional	SMS-DEFAULT   UCS2	Used for conversion purposes. UCS2 can not be used in a SMS-DEFAULT encoded wiblet. Default value is determined by the wiblet encoding.	No
<b>emptyok</b> optional	true   false	This specifies whether or not empty input should be accepted. Default value is true.	No
<b>format</b> optional	*M   *N	The expected format of the data entered by the end user. *N indicates numeric characters and *M any characters. Default value is *M.	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is adjacent. Compatibility: Wib 1.3 and later.	No
<b>minlength</b> optional	integer (0–255)	The minimum number of bytes that has to be entered by the user.	No
<b>maxlength</b> optional	integer (0–255)	The max number of bytes that can be entered by the user.	No
<b>title</b> optional	WED-string	The prompting string.	Yes
<b>type</b> optional	text   password	The type of the input dialogue. <code>text</code> is used for regular input and <code>password</code> whenever the entered text should not be echoed on the screen of the mobile station. Default value is <code>text</code> . On many mobile stations, passwords may only be entered as numbers, not as text. Therefore <code>type="password"</code> will fail on these phones without the <code>format="*N"</code> attribute.	No
<b>value</b> optional	string	The default value of the variable named in the <code>name</code> attribute.	Yes



## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

For Wib 1.1.0 and Wib 1.2.0, the `class` attribute determines the Wib encoding of the `title` text.

### Example [12]

```
<input title="Please enter your phone number" type="text" name="PHONE"
format="*N" maxlength="20" iconid="3" iconusage="replace"/>
```

### Example [13]

Assume that the wiblet encoding is UCS2, but the mobile station only supports the SMS Default Alphabet for input. The variable is manually set to be of type

"SMS-DEFAULT":

```
<input title="Please enter your name" name="MYNAME" class="SMS-DEFAULT"/>
```

## 8.6 select Element

### Description

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 section 8.7. The name of the menu, normally displayed by the mobile station, is specified by the `title` attribute.

Either the `name` or `iname` attribute can be used. If the `iname` attribute is used, the `value` attribute in the contained `option` elements will be overridden with the calculated index.

The `class` attribute is used for conversion purposes. It may have the following values:

- `SMS-DEFAULT` - The content of the variable given by the `iname` or `name` attribute will be stored in Wib with characters of the SMS Default Alphabet. If sent through the UG to the Content Provider, the data is converted to ISO-8859-1.
- `UCS2` - *This value is only possible if the wiblet encoding is UCS2.* The content of the variable given by the `iname` or `name` attribute will be stored in Wib with UCS2 encoded characters. If sent through the UG to the Content Provider, the data is converted to UTF-8.

The encoding of the input will per default follow the wiblet encoding. If the wiblet encoding is UCS2, the input may be changed to SMS-DEFAULT. If the wiblet encoding is SMS-DEFAULT, the input can only be given in SMS-DEFAULT.



See Appendix B for more detailed information regarding character conversions performed by the UG.

### Content

One or more `option` elements.

Zero or one occurrence of the `catcherror` element if the `onpick` attribute of the sub element `option` contains an `external-URL`. The `catcherror` element has to be the last of all other sub elements.

### Position

`select` may only occur as a child of the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>class</b> optional	SMS-DEFAULT   UCS2	Used for conversion purposes. UCS2 can not be used in an SMS-DEFAULT encoded wiblet. Default value is determined by the wiblet encoding.	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No
<b>iname</b> optional	variable-name	The name of the variable to be set with the index result of the selection. If the first option is selected the variable will be set to “1”, if the second variable is selected the variable will be set to “2”, etc.	No
<b>name</b> optional	variable-name	The name of the variable to be set.	No
<b>title</b> optional	WED-string	The title of the menu.	Yes
<b>locinfo</b> optional	force config	Forces inclusion of location info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No



<b>langinfo</b> optional	force config	Forces inclusion of language info in the uplink message. Default is config, which means follow configuration.	No
<b>imeiinfo</b> optional	force config	Forces inclusion of IMEI/IMEISV info in the uplink message. Default is config, which means follow configuration. Compatibility: Wib 2.0 and later	No

---

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## 8.7 option Element

### Description

The `option` element represents a list item in a list defined by the `select` element. The content consists of text that is displayed as the option text. This text is used in the same way as the `value` attribute if that attribute is not present. Empty `option` text strings are not supported.

When an `option` is selected, the variable named in the enclosing `select` element is set to the value given by the `value` attribute. Then Wib navigates to the URI specified by the `onpick` attribute if present.

### Content

*WED-string* including variable references.

### Position

`option` may only occur as a child of the `select` element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the option text. If any option within a select element specifies an iconid, the default value for the rest of the option elements (within the same select element) is the same as the first option that specifies an iconid. iconid specified by the first option. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the option text. All iconusage attributes in a list with option elements should have the same value. replace will be used for all option elements within a select element if any option indicates it. Default value is adjacent. Compatibility: Wib 1.3 and later.	No
<b>onpick</b> optional	WIB-URI	Destination URI to go to if this option element is selected. Compatibility and variable references: See section 7.8 .	Yes/ No
<b>value</b> optional	String	String to be copied to the variable named in select attribute name, if this option element is selected. The class attribute of the parent select element determines the Wib encoding. value will be ignored if it is used in combination with the iname attribute of the select element. Compatibility: Variable references are supported by Wib 1.3 and later.	Yes/ No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.



### Example [14]

This example illustrates the use of `select` and `option`. A jump will occur to "CARD2" if the user selects the "Banking" option, and to "CARD3" if the user selects the "Gambling" option. If "[Home]" is selected a GET request will be sent for the "home.wml" document. Note that the `value` attribute in the `option` element cannot be used for anything if the corresponding `onpick` attribute refers to an external URL.

```
<card>
  <p>
    <select title="Please choose service" name="SELECTION">
      <option value="Banking" iconid="1" iconusage="replace"
onpick="#CARD2">Banking</option>
      <option value="Gambling" onpick="#CARD3">Gambling</option>
      <option value="Not used."
onpick="http://webserver/home.wml">[Home]</option>
    </select>
  </p>
</card>
```

## 8.8 setvar Element

### Description

The `setvar` element sets the value of a variable. Variable references used in the value of the `value` attribute are replaced on a byte-per-byte basis in Wib. See Example [17].

The `class` attribute is used for conversion purposes. It may have the following values:

- `SMS-DEFAULT` - The variable will be encoded with the SMS Default Alphabet in Wib. If sent through the UG to the Content Provider, the content of the variable is converted to ISO-8859-1.

This value is most likely if the wiblet encoding is `SMS-DEFAULT` and the variable is intended to be displayed on the screen at some stage.

- `UCS2` - The variable will be encoded with UCS2 in Wib. If sent through the UG to the Content Provider, the content of the variable is converted to UTF-8. Note that UTF-8 encoding is not used directly in a variable in Wib, as the encoding used by the SIM is UCS2.

This value is most likely if the wiblet encoding is `UCS2` and the variable is intended to be displayed on the screen at some stage.

- `binary` - The variable contains data that is not to be converted when sent through the UG. This is the default value if the `class` attribute is omitted. This value can for instance be used for encrypted data.



- `hex-binary` - The value of the `value` attribute is in hexadecimal format. The data is binary encoded in Wib. If sent through the UG to the Content Provider, the content is not further converted and it will remain in binary data. Variable references are only allowed between bytes, i.e. two characters.
- `base64-binary` - The value of the `value` attribute is in base64 format. The data is binary encoded in Wib. If sent through the UG to the Content Provider, the content is not further converted and it will remain in binary data. Variable references are only allowed between base64 blocks of four characters.

If the class is `SMS-DEFAULT`, `UCS2` or `binary`, hexadecimal escape characters can be used for specifying binary data. See section 6.4 .

See also Appendix B.

### Content

None; the element is always empty.

### Position

`setvar` may only occur as a child of the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	variable-name	The name of the variable to be set.	No
<b>value</b> mandatory	string	The value the variable is set to. Compatibility: Variable references in the value are only supported by Wib 1.2 and later.	Yes/ No
<b>class</b> optional	<code>SMS-DEFAULT</code>   <code>UCS2</code>   <code>binary</code>   <code>hex-binary</code>   <code>base64-binary</code>	Used for conversion purposes. Default value is <code>binary</code> . See also description above for more details.	No

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

### Example [15]

The variable `COUNTRY` is set to "Sweden". The variable may later be used by referring to `$(COUNTRY)` .

```
<setvar name="COUNTRY" value="Sweden"/>
```



### Example [16]

The variable `BYTES1`, `BYTES2` and `BYTES3` are all set to the hexadecimal value "67E34F".

```
<card>
  <p>
    <setvar name="BYTES1" value="\x67\xe3\x4f" class="binary"/>
    <setvar name="BYTES2" value="67E34F" class="hex-binary"/>
    <setvar name="BYTES3" value="Z+NP" class="base64-binary"/>
  </p>
</card>
```

### Example [17]

First, the variable `VAR1` will be set to the hexadecimal value "0031". Then, the variable `VAR2` will be set to the hexadecimal value "310031". If sent through the UG to the Content Provider, `VAR2` will be interpreted as the text string "1@1", since the value "00" corresponds to the character '@' in the SMS Default Alphabet.

```
<setvar name="VAR1" value="1" class="UCS2"/>
<setvar name="VAR2" value="1$(VAR1)" class="SMS-DEFAULT"/>
```

## 8.9 go Element

### Description

The `go` element is typically used to do one of the following:

- Load and execute a new wiblet. The new wiblet may be stored locally in the SIM (possible with Wib version 1.3 and later) or loaded through the UG.
- Divert execution of the currently executing card to another card in the same wiblet.
- Send arbitrary information to the Content Provider.

*Compatibility note: The following paragraphs of this section are only applicable with Wib 1.3 and later versions.*

Please note that if the `go` element does not contain a `progressinfo` element of a certain type, the missing `progressinfo` element will be automatically inserted with the `onempty` attribute set to `fallback`. This is to ensure that progress information is presented to the end user as the default behavior, rather than just an optional feature. This improves the usability and end user experience of Wib.

A side effect of this behavior is that if progress information is not desired at all (which is normally not the case), the `go` element must contain three `progressinfo` elements, one of each type, with the `onempty` attribute set to `suppress` and no text content.

For further information regarding the `progressinfo` element, see section 8.11 .



## Content

Zero or more occurrences of each of the following elements:

`progressinfo`, `postfield`

Zero or one occurrence of the `bookmarkinfo` element.

Zero or one occurrence of the element `catcherror`. The `catcherror` element has to be the last of all other sub elements.

The `go` element may only contain other elements if the `WIB-URI` refers to an `external-URL`.

## Position

`go` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>href</b> mandatory	WIB-URI	The destination. Compatibility and variable references: See section 7.8 .	Yes/ No
<b>enterwait</b> optional	<code>true</code>   <code>false</code>   <code>mode-</code> <code>dependent</code>   <code>no-reply</code>	Controls whether Wib shall terminate the execution of the current wiblet and wait for a response as the result of the <code>go</code> element ( <code>true</code> ), or continue the wiblet execution immediately ( <code>false</code> ). Attribute value <code>mode-dependent</code> indicates that the wait-state will be entered depending on the operational mode of Wib. The operational mode of Wib can be changed through the <code>setreturntarvalue</code> element. See section 10.11 . Attribute value <code>no-reply</code> indicates that the wait-state will not be entered and UG will block any response sent to ME. Default value for Wib 1.3 and later is <code>true</code> . Default value for Wib version prior to Wib 1.3 is <code>mode-dependent</code> . This attribute is only valid if the <code>WIB-URI</code> is an <code>External-URL</code> . Compatibility: <code>true</code> and <code>false</code> are only supported by Wib 1.3 and later.	No



<b>method</b> optional	get   post	Controls the HTTP submission method to be used by the UG. The default is <i>get</i> . This attribute is only valid if the WIB-URI is an <i>External-URL</i> .	No
<b>locinfo</b> optional	force   config	Forces inclusion of location info in the uplink message. Default is <i>config</i> , which means follow configuration.	
<b>langinfo</b> optional	force   config	Forces inclusion of language info in the uplink message. Default is <i>config</i> , which means follow configuration.	
<b>imeiinfo</b> optional	force   config	Forces inclusion of IMEI/IMEISV info in the uplink message. Default is <i>config</i> , which means follow configuration.	

---

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above and progress information default behavior described in the Description section.

Wib 2.0 and later for the `catcherror` element described in the Content section.

### Example [18]

If an error occur when sending the input to the content provider the text "Error caught!" together with the error code will be displayed on screen.

```
<card>
  <p>
    <input title="Variable" type="text" name="VARIABLE"/>
    <go
href="http://webserver/page.jsp?f=$(VARIABLE) &amp;l=StaticText">
      <catcherror result="RESULT" resultdetails="DETAILS" />
    </go>
    <conditionaljump compare="$(RESULT)">
      <test href="#handleerror" value="\x01" />
    </conditionaljump>
  </p>
</card>

<card id="handleerror">
  <p>
    Error caught! Result Details: $(DETAILS)
  </p>
</card>
```



### Example [19]

```
<card>
  <p>
    <input title="First name" type="text" name="FN"/>
    <input title="Last name" type="text" name="LN"/>
    <go method="post"
href="http://webserver/page.jsp?f=$ (FN) &amp;l=$ (LN) "/>
  </p>
</card>
```

### Example [20]

Note that a card reference starts with a hash sign ('#').

```
<card id="CARD1">
  <p>
    <go href="#CARD2"/>
  </p>
</card>
<card id="CARD2">
  <p>
    You have jumped to CARD2.
  </p>
</card>
```

### Example [21]

Note that a reference to a locally stored wiblet uses the URL scheme 'wiblet'. The variable global01 will be accessible from within the locally stored wiblet due to the special naming of the variable while the FORGOTTEN variable will be cleared.

```
<card>
  <p>
    <setvar name="FORGOTTEN" value="This text will be cleared"/>
    <setvar name=":global01" value="112233"/>
    <go href="wiblet://smarttrust.com/foo/bar.wml"/>
  </p>
</card>
```

### Example [22]

Location information is fetched from the ME and sent back to the Content Provider without entering the wait state.

```
<card>
  <p>
    <providelocalinfo cmdqualifier="location" destvar="LOC"/>
    <go enterwait="false" href="http://webserver/update.py?loc=$(LOC) "/>
  </p>
</card>
```



## 8.10 postfield Element

### Description

The `postfield` element is used for specifying a field name and a value for transmission to an origin server during a URL request.

### Content

None; the element is always empty.

### Position

`postfield` may only occur as a child of the `go` element.

### Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	string	The field name. The Wib encoding is SMS-DEFAULT.	No
<b>value</b> mandatory	string	The field value. The Wib encoding is SMS-DEFAULT.	Yes

### Compatibility

Wib 1.1 and later.

### Example [23]

The following markup:

```
<go href="http://webserver/page.jsp?no=10&firstname=$(FIRSTNAME)" />
```

is identical in behavior to:

```
<go href="http://webserver/page.jsp">
  <postfield name="no" value="10" />
  <postfield name="firstname" value="$(FIRSTNAME)" />
</go>
```

Both will generate a GET request to the Content Provider.

## 8.11 progressinfo Element

### Description

The `progressinfo` element is used to display progress information on the screen of the mobile station while Wib is sending data to, or receiving data from, the UG.

If the progress information type defined by the `type` attribute is `receiving`, the text may also include the following escape sequences:

- "%C" – replaced with the ordinal of the SM being received.



- "%T" – replaced with the total number of SMs expected.
- "%P" – replaced with the percentage of SMs received (integer 0-100).

To include a percentage sign in the text, it must be escaped with another percentage sign. So "%%" in the text will be displayed as "%" on the screen of the mobile station.

Since progress information is generally considered desirable from a usability point of view, the `progressinfo` element is part of a default behavior scheme explained in section 8.9 , aiming to maximize the use of progress information.

If Wib does not enter the wait-state, progress information of type receiving and intermediate will have no effect. See the `enterwait` attribute in the `go` element (section 8.9 ) for further information.

### Content

*WED-string* including variable references.

### Position

`progressinfo` may only occur as a child of the `go` element.

### Attributes

Name	Value	Explanation	Var
<b>type</b> mandatory	sending   intermediate   receiving	The type of the progress information. sending is used when Wib is sending data to the UG, intermediate in the intermediate state thereafter, and finally receiving when the next wiblet is being received by Wib.	No
<b>onempty</b> optional	fallback   none   suppress	Determines the behavior when the progress information text or iconid is omitted. fallback – Use default text/icon configured in Wib. none – Let the ME decide the behavior. suppress – Suppress display of progress information. Value suppress is only applicable when both progress information text and icon are missing. Default value is fallback.	
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the progress information text. See section 5.8 for further information.	No



<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the progress information text. May only be used when the <code>iconid</code> attribute is specified. Default value is adjacent.	No
------------------------------	-----------------------	---	----

## Compatibility

Wib 1.3 and later.

## Example [24]

In the following example, various text strings along with icons will be displayed on the mobile station in the different phases while waiting for the next wiblet. The reception phase will also display the progress in percent.

```
<card>
  <go href="http://webserver/page.jsp">
    <progressinfo type="sending" iconid="20">
      Sending...
    </progressinfo>
    <progressinfo type="intermediate" iconid="21">
      Waiting...
    </progressinfo>
    <progressinfo type="receiving" iconid="22">
      Receiving %P%
    </progressinfo>
  </go>
</card>
```

## 8.12 bookmarkinfo Element

### Description

The `bookmarkinfo` element is used to give the end user a possibility to bookmark a URI as part of a `go` element. The text specified as content defines the default bookmark name that will be displayed to the end user.

Bookmarked URIs may be resubmitted at a later stage using a bookmark menu.

### Content

*WED-string* including variable references.

### Position

`bookmarkinfo` may only occur as a child of the `go` element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when Wib presents the bookmark in the list of bookmarks. See section 5.8 for further information.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the bookmark name. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> .	No

## Compatibility

Bookmarks are an optional feature of Wib 1.3. This means that Wib may silently ignore this element.

### Example [25]

In the following example, the end user will be given the option of storing the URI indicated in the `href` attribute, as a bookmark. The default bookmark name presented on the screen will be "News Headlines".

```
<card>
  <go href="http://webserver/news/headlines.wml">
    <bookmarkinfo>
      News Headlines
    </bookmarkinfo>
  </go>
</card>
```

### Example [26]

In this example, the end user is requested to select a service which will be bookmarked before it is loaded if the end user so wants. The APP variable in the URI and the bookmark text will be expanded before the bookmark is stored.

```
<p>
  <select title="Please select service" name="APP">
    <option value="News">News</option>
    <option value="Fun">Fun</option>
    <option value="Info">Info</option>
  </select>
  <go href="http://webserver/$(APP)/Main.wml">
    <bookmarkinfo>
      $(APP)
    </bookmarkinfo>
  </go>
</p>
```



## 9 STK command oriented elements

### 9.1 launchbrowser Element

#### Description

The `launchbrowser` element launches a browser (e.g. a WAP browser) in the mobile station.

#### Content

Zero or more `bearer` elements listed in decreasing priority order.

Zero or one occurrence of the element `catcherror`. The `catcherror` element has to be the last of all other sub elements.

#### Position

`launchbrowser` may occur as a child of the `card` or the `p` element.

#### Attributes

Name	Value	Explanation	Var
<b>browserid</b> optional	default or integer (0–255)	The browser identity. See STK [17] for details. Default value is <code>default</code> (which corresponds to integer value 0).	No
<b>cmdqualifier</b> optional	<code>launch-if-not-launched</code>   <code>use-existing</code>   <code>relaunch</code> or integer (0–255)	Defines conditions for launching the browser. See STK [17] for details. Default value is <code>launch-if-not-launched</code> (which corresponds to integer value 0).	No
<b>gatewayid</b> optional	string	The <code>gatewayid</code> is used to specify a Gateway/Proxy Identity. The value must contain characters supported by the SMS Default Alphabet. See STK [17] for details.  If the element contains one or more <code>bearer</code> elements, this attribute must be present.	Yes/ No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No



<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No
<b>provfileref</b> optional	hex-bin	Provisioning file reference according to STK [17]. If present, this attribute takes precedence over <code>gatewayid</code> and <code>bearer</code> .	Yes/ No
<b>title</b> optional	WED-string	Text displayed prior to the launch of the browser. Corresponds to the alpha identifier according to STK [17].	Yes/ No
<b>url</b> optional	URI	The URI from which the launched browser should request content. Default value is defined by the mobile station.	Yes/ No

### Compatibility

Wib 1.2 and later except for attributes as noted in the table above. Variable references in the attributes (as listed in the table above) are only supported by Wib 1.3 and later.

Wib 2.0 and later for the `catcherror` element described in the Content section.

### Example [27]

This example launches a browser in the mobile station, and the document "home.wml" is requested. The value `launch-if-not-launched` will be used as command qualifier.

If an error occurs when launching the browser the text "Error caught!" And the error code is displayed on screen.

```
<card>
  <launchbrowser url="http://webserver/home.wml">
    <catcherror result="RESULT" resultdetails="DETAILS" />
  </launchbrowser>
  <conditionaljump compare="$(RESULT)">
    <test href="#handleerror" value="\x01"/>
  </conditionaljump>
</card>

<card id="handleerror">
  <p>
    Error caught! Result Details: $(DETAILS)
  </p>
</card>
```



## 9.2 bearer Element

### Description

The `bearer` element defines a specific bearer to be used when launching a browser in the mobile station.

### Content

One of the following text strings: `sms`, `csd`, `gprs`, `ussd`

### Position

`bearer` may only occur as a child of the `launchbrowser` element.

### Attributes

None.

### Compatibility

Wib 1.2 and later.

### Example [28]

This example launches a browser in the mobile station, and the document "home.wml" is requested. The value `launch-if-not-launched` will be used as command qualifier. CSD shall be used as bearer if possible, otherwise SMS. The Gateway/Proxy Identity and title are also given in the example.

```
<card>
  <launchbrowser url=http://webserver/home.wml title="Launching browser"
gatewayid="192.168.0.50">
    <bearer>csd</bearer>
    <bearer>sms</bearer>
  </launchbrowser>
</card>
```

## 9.3 playtone Element

### Description

The `playtone` element makes the mobile station play a tone. *Standard supervisory tones* are normally generated in the internal earphone of the mobile station. The general-beep tone is normally generated in the external ringer of the mobile station as a beep. See the table below.

### Content

None; the element is always empty.

### Position

`playtone` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>duration</b> mandatory	float (0.1 – 15300.0)	The duration time in seconds.	No
<b>toneid</b> mandatory	<i>Standard</i> <i>supervisory tones</i> <i>(generally internal):</i> dial   busy   congestion   radio-path-ack   radio- path-nack   error   waiting   ringing <i>ME proprietary</i> <i>tones:</i> general-beep   ack-tone   nack-tone   user-ringing   user-sms   critical-alert   vibrate-only  <i>Themed tones</i> happy   sad   urgent-action   question   message- received <i>Melody tones</i> melody1   melody2   melody3   melody4   melody5   melody6   melody7   melody8   or integer (0–255)	The tone identifier. For details regarding the nature of these values, see STK [17].	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No



<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No
<b>title</b> optional	WED-string	Text to display while playing the tone.	Yes

---

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

### Example [29]

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

```
<card>
  <p>
    <playtone toneid="congestion" duration="10.0"/>
  </p>
</card>
```

### Example [30]

In this example, the mobile station is requested to play a general-beep tone of 2.5 seconds. Icon and title is also included.

```
<card>
  <p>
    <playtone toneid="general-beep" duration="2.5" iconid="1"
    iconusage="replace" title="Play it again, Sam!"/>
  </p>
</card>
```

## 9.4 providelocalinfo Element

### Description

The `providelocalinfo` element is used to get local information from the mobile station.

### Content

None; the element is always empty.

### Position

`providelocalinfo` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The variable to receive the provided information. The Wib encoding of the variable will always be binary.	No
<b>cmdqualifier</b> mandatory	location   imei   nmr   dtz   language   timing   accesstech   esn   imeisv   searchmode   battery   meid   wsid or integer (0-255)	The type of information that should be provided. For details regarding the nature of these values, see STK [17]. Compatibility: accesstech, esn, imeisv, searchmode, battery, meid and wsid supported by Wib 2.0 and later	No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## Example [31]

In this example, the Location and the Network Measurement Results are fetched and put into separate variables. Thereafter, both values are sent to a Content Provider.

```
<card>
  <p>
    <providelocalinfo cmdqualifier="location" destvar="LOC"/>
    <providelocalinfo cmdqualifier="nmr" destvar="NMR"/>
    <go href="http://webserver/page.jsp?loc=${LOC} & nmr=${NMR}" />
  </p>
</card>
```

## 9.5 refresh Element

### Description

The `refresh` element makes the SIM notify the mobile station of changes in the SIM configuration as the result of SIM application activity. Depending on the command qualifier, different tasks will be performed. If the value given by the `cmdqualifier` attribute is `file-change` (integer value 1) or `sim-init-file-change` (integer value 2) the element must contain at least one `file` element. For more information see STK [17].

If the value given by `cmdqualifier` attribute is `roaming` the element must contain at least one `network` element. When the refresh command is run with the `roaming` attribute the phone is told to perform a roaming steering refresh as



defined in STK [17]. If the phone does not support that operation, Wib will perform a best effort attempt at achieving the same end result. The order in which the network elements appear in the refresh element controls the priority of the networks. The first network listed has the highest priority, the last one the lowest priority.

## Content

Zero or more `file` elements or one or more `network` elements.

## Position

`refresh` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>cmdqualifier</b> optional	sim-init-full -file-change   file-change   sim-init- file-change   sim-init   sim-reset   roaming  or  integer (0–255)	Command qualifier. Default value is <code>sim-init-full-file-change</code> (which corresponds to integer value 0). For details regarding the nature of these values, see STK [17]. Compatibility: <code>roaming</code> supported by Wib 2.0 and later	No

## Compatibility

Wib 1.1 and later, except for attributes as noted above.

## Example [32]

In this example, the SIM is reinitialized with full file change notification.

```
<card>
  <p>
    <refresh cmdqualifier="sim-init-full-file-change"/>
  </p>
</card>
```

## 9.6 file Element

### Description

The `file` element specifies a SIM file that should be refreshed.



## Content

Text representing a fully qualified file path, including file identifier, formatted as hex-bin. All texts must consist of at least 8 hexadecimal digits. Variables are not allowed.

## Position

`file` may only occur as a child of the `refresh` element.

## Attributes

None.

## Compatibility

Wib 1.1 and later.

## Example [33]

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

```
<card>
  <p>
    <refresh cmdqualifier="sim-init-file-change">
      <file>3F007F106F3A</file>
      <file>3F007F206F30</file>
    </refresh>
  </p>
</card>
```

## 9.7 network Element

### Description

The `network` element specifies a network and the technology it uses.

### Content

None.

### Position

`network` may only occur as a child of the `refresh` element.

### Attributes

Name	Value	Explanation	Var
<code>mcc</code> mandatory	string of decimal digits	The Mobile Country Code	No



<b>mnc</b> mandatory	string of decimal digits	The Mobile Network Code	No
<b>technology</b> mandatory	gsm   gsm-compact   utran or hex-bin	The technology used by the network. More than one entry may be entered separated by white space. Coding is the same as for Access Technology Identifier in EF <sub>PLMNwAcT</sub> , see 31.102 [21]	No

## Compatibility

Wib 2.0 and later.

## Example [34]

The example below attempts to influence the phone to select the three listed networks in the priority order listed and make this preference permanent.

```
<card>
  <p>
    <refresh cmdqualifier="roaming">
      <network mcc="240" mnc="01" technology="gsm gsm-compact"/>
      <network mcc="240" mnc="02" technology="utran"/>
      <network mcc="240" mnc="06" technology="8000"/> <!-- utran -->
    </refresh>
  </p>
</card>
```

## 9.8 sendsm Element

### Description

The `sendsm` element sends a binary or plain text SM from Wib to a particular destination.

### Content

Exactly one occurrence of each of the following elements:

`userdata`, `destaddress`

Zero or one occurrence of the element `servicecentreaddress`.

Zero or one occurrence of the element `catcherror`. The `catcherror` element has to be the last of all other sub elements.

### Position

`sendsm` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No
<b>pid</b> optional	integer (0–255)	Protocol identifier. Default value is 0.	No
<b>title</b> optional	WED-string	Text displayed on the screen while sending the SM. Corresponds to the alpha identifier according to STK [17]. Compatibility: Wib 1.3 and later.	Yes

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

Wib 2.0 and later for the `catcherror` element described in the Content section

## Example [35]

In the example, a text SM with the content "Hello!" is sent to MSISDN "+15185551234". It is made sure that the specified Service Center "+15185559999" is used, regardless of the default value in the mobile station.

If an error occur when sending the text sm, the message "Error caught!" together with the error code is displayed.

```
<card>
  <p>
    <sendsm>
      <destaddress value="+15185551234"/>
      <userdata>Hello!</userdata>
      <servicecentreadress value="+15185559999"/>
      <catcherror result="RESULT" resultdetails="DETAILS"/>
    </sendsm>
    <conditionaljump compare="$(RESULT)">
      <test href="#handleerror" value="\x01" />
    </conditionaljump>
  </p>
</card>
<card id="handleerror">
  <p>
    Error caught! Result Details: $(DETAILS)
  </p>
</card>
```



## 9.9 destaddress Element

### Description

The `destaddress` element defines the called party number.

### Content

None; the element is always empty.

### Position

`destaddress` may occur as a child of the `sendsm` or `setupcall` element.

### Attributes

Name	Value	Explanation	Var
<b>value</b> mandatory	A valid dial string, i.e. a string of decimal digits and extended BCD characters as listed in the two tables in section 10.7 . Please note that the extended BCD characters listed are case-sensitive, i.e. character "a" is not the same as character "A".	The called party number. If the <code>wibenc</code> attribute is set to <code>ADN</code> , two rules apply. 1) Static text and variable references can not be mixed. 2) The variables shall be formatted according to <code>EF<sub>ADN</sub></code> [7]. Compatibility: Variable references are supported by Wib 1.3 and later.	Yes/ No
<b>wibenc</b> optional	<code>SMS-DEFAULT</code>   <code>UCS2</code>   <code>ADN</code>	The Wib encoding of the <code>value</code> attribute. Default value is <code>ADN</code> . Compatibility: <code>SMS-DEFAULT</code> and <code>UCS2</code> are only supported by Wib 1.3 and later.	No

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

### Example [36]

In the example, the end user will be requested to enter an MSISDN and then some text. The text will be sent in a Short Message to the MSISDN entered by the end user.

```
<card>
  <p>
    <input title="Enter phone number" name="NO" class="SMS-DEFAULT"/>
```



```
<input title="Enter text" name="TXT" class="SMS-DEFAULT"/>
<sendsm>
  <destaddress value="$ (NO) " wibenc="SMS-DEFAULT"/>
  <userdata smtextenc="SMS-DEFAULT">$ (TXT) </userdata>
</sendsm>
</p>
</card>
```

## 9.10 servicecentreaddress Element

### Description

The `servicecentreaddress` element defines the service centre address to be used when sending an SM from Wib.

### Content

None; the element is always empty.

### Position

`servicecentreaddress` may only occur as a child of the `sendsm` element.

### Attributes

This element offers the same attributes as the `destaddress` element. See section 9.9 for details.

### Compatibility

See section 9.9 for details.

## 9.11 userdata Element

### Description

The `userdata` element defines text or binary data which should be included as user data in an SM sent from Wib.

The attribute `udh` is used to specify the User Data Header. For normal text SMs, no `udh` is needed. Note that UDHL (User Data Header Length) shall not be included in the `udh` attribute value. It is implicitly given by the length of the value. For further information regarding the technical realization, see [1].

This element can be used for sending Enhanced Message Service (EMS) or Nokia Smart Messaging (NSM). For details regarding the EMS format and the creation of EMS messages, see [2], [13], and [14]. For details regarding the NSM format, see [9].

Note that the maximum length for one SM can not be exceeded. If concatenated SMs are to be used, that must be handled on WIG WML document level by using many `sendsm` elements and adequate `udh` values. The maximum length (including



UDH) for the userdata after variable substitution is 160 characters for SMS Default Alphabet, 70 for UCS2 and 140 bytes if binary data is used.

## Content

*WED-string* including variable references. Encoded according to the `docudenc` attribute.

## Position

`userdata` may only occur as a child of the `sendsm` element.

## Attributes

Name	Value	Explanation	Var
<b>docudenc</b> optional	text   hex- binary   base64-binary	The document encoding of the text (i.e. the user data) contained in the element. Default value is <code>text</code> . The effect of using attribute values <code>hex-binary</code> and <code>base64-binary</code> are the same as described in section 8.8 .	No
<b>dcs</b> optional	SMS-DEFAULT   UCS2   SMS- DEFAULT-FLASH   UCS2-FLASH or integer (0-255)	Data Coding Scheme for the outgoing SM. If DCS indicates "Default Alphabet" according to GSM 03.38 [4], Wib will pack any data not included in the User Data Header. In that case, it does only make sense to set <code>docudenc</code> to <code>text</code> and <code>smttextenc</code> to <code>SMS-DEFAULT</code> . DCS indicating compression is not supported. The FLASH suffix indicates that the SM is flagged for immediate display on the ME. The attribute values correspond to the following DCS integer values: SMS-DEFAULT - 242 UCS2 - 26 SMS-DEFAULT-FLASH - 240 UCS2-FLASH - 24 Default value is determined by the <code>smttextenc</code> attribute if given, otherwise by the wiblet encoding.	No
<b>smttextenc</b> optional	SMS-DEFAULT   UCS2	The encoding of text in the SM. Only applicable if <code>docudenc</code> is set to <code>text</code> . Default value is determined by the wiblet encoding.	No



<b>udh</b> optional	hex-bin	The User Data Header. User Data Header Length (UDHL) shall not be included. Any variable reference included in this attribute is replaced byte-by-byte with its current value, i.e. the variable shall not contain a string with hexadecimal digits. Compatibility: Only supported by Wib 1.3 and later.	Yes
------------------------	---------	--	-----

### Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

### Example [37]

In this example, an EMS message containing a ring-tone is sent to MSISDN "+15185551234".

```
<card>
  <p>
    <sendsm pid="0">
      <destaddress value="+15185551234"/>
      <userdata
udh="0C8000424547494E3A494D454C4F44590D0A56455253494F4E3A312E300D0A464F52
4D41543A434C415353312E300D0A4D454C4F44593A2A33663366336633236331236433236
4332364336331723366336633663323633323663323663323663366336633236332A34236333236333
2363332A332361310D0A454E443A494D454C4F44590D0A"
dcs="245"/>
      <servicecentreadress value="+15185559999"/>
    </sendsm>
  </p>
</card>
```

### Example [38]

In this example, an NSM message containing a business card is sent to MSISDN "+15185551234".

```
<card>
  <p>
    <sendsm pid="0">
      <destaddress value="+15185551234"/>
      <userdata udh="050400E20000" docudenc="hex-binary" dcs="245">
424547494E3A56434152440D0A56455253494F4E3A322E310D0A4E3A536D6974683B4D696
B650D0A54454C3B505245463A2B35353531323334350D0A454E443A56434152440D0A
      </userdata>
      <servicecentreadress value="+15185559999"/>
    </sendsm>
  </p>
</card>
```



## 9.12 sendussd Element

### Description

The `sendussd` element sends a message by means of the Unstructured Supplementary Service.

### Content

Zero or one occurrence of the `catcherror` element.

### Position

`sendussd` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>ussd</b> mandatory	string	According to GSM 02.30 [3]. Compatibility: Variable references supported by Wib 1.3 and later.	Yes/ No
<b>destvar</b> optional	variable-name	Variable to contain the USSD return result message.	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No
<b>title</b> optional	WED-string	Text to display when sending. Corresponds to the alpha identifier according to STK [17]. Encoding follows wiblet encoding. Compatibility: Variable references supported by Wib 1.3 and later.	Yes/ No
<b>dcs</b> optional	SMS-DEFAULT   UCS2   binary or integer (0–255)	Data Coding Scheme for the received <code>ussd</code> message. Informs Wib how the data in the <code>destvar</code> shall be interpreted.	No
<b>class</b> optional	SMS-DEFAULT   UCS2   binary	Compatibility: Wib 2.0 and later The Data Coding Scheme of the sent <code>ussd</code> string. Default value is determined by the wiblet encoding. Compatibility: Wib 2.0 and later	No



## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

Wib 2.0 and later for the `catcherror` element described in the Content section

### Example [39]

In the example, a USSD message with the content `*120#` is sent to the network. The title "Sending USSD" is displayed. The USSD return result message is stored in the variable named `OUT` and then displayed to the user.

If an error occur when sending the ussd string, the text "Error caught!" together with the error code is displayed.

```
<card>
  <p>
    <sendussd ussd="*120#" destvar="OUT" title="Requesting data"
    dcs="binary">
      <catcherror result="RESULT" resultdetails="DETAILS" />
    </sendussd>
    <conditionaljump compare="$(RESULT)">
      <test href="#handleerror" value="\x01" />
    </conditionaljump>
    $(OUT)
  </p>
</card>

<card id="handleerror">
  <p>
    Error caught! Result Details: $(DETAILS)
  </p>
</card>
```

### Example [40]

In this example a USSD message with content `*166#` is sent to the network. The title "Requesting data" is displayed. The USSD return result is stored in the variable named `OUT` and then executed as a wiblet.

```
<card>
  <p>
    <sendussd ussd="*166#" destvar="OUT" title="Requesting data"
    dcs="binary"/>
    <executewiblet srcvar="OUT"/>
  </p>
</card>
```

### Example [41]

In this example a USSD message with content `*166#`+ the text specified by the value attribute of the selected option, is sent to the network as a string encoded as SMS-DEFAULT. The title "Requesting data" is displayed. The USSD return result is stored in the variable named `OUT`.



```
<?xml version="1.0" encoding="UTF-8" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd"
      wibletenc="UCS2">
<card id="start">
  <p>
    <select name="SELECTION" title="حساب">
      <option value="enquiry">الرصيد او الفاتوره</option>
      <option value="refill">شحن الرصيد</option>
      <option value="transfer">تحويل الرصيد</option>
    </select>
    <sendussd class="SMS-DEFAULT" ussd="*166#$(SELECTION) " destvar="OUT"
    title="Requesting data" dcs="UCS2"/>
  </p>
</card>
</wml>
```

## 9.13 setupcall Element

### Description

The `setupcall` element requests the mobile station to initiate a call.

### Content

Exactly one occurrence of element `destaddress`.

Zero or one of each of the following elements:

`setupinfo`, `confirminfo`, `catcherror`.

The `catcherror` element has to be the last of all other sub elements.

### Position

`setupcall` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<code>capability</code> optional	hex-bin	Capability Configuration Parameters. For coding, see GSM 04.08 [5]	No



<b>cmdqualifier</b> optional	if-not-busy   if-not-busy- with-redial   put-on-hold   put-on-hold- with-redial   disconnect- other   disconnect- other-with- redial or integer (0-255)	Defines conditions for setting up the call. See STK [17] for details. Default value is if-not-busy.	No
<b>duration</b> optional	float (0.1 – 15300.0)	Defines the duration in seconds that automatic retries to set up the call will be made. Default is dependent of the mobile station.	No

## Compatibility

Wib 1.1 and later.

Wib 2.0 and later for the catcherror element described in the Content section

## Example [42]

In the example, the SIM requests the mobile station to set up a call to "+15185551234". If the ME is already involved in another call, this call will be disconnected. No text is displayed, no Capability Configuration Parameters are attached, and no automatic retries to set up the call will be made.

```
<card>
  <p>
    <setupcall cmdqualifier="disconnect-other">
      <destaddress value="+15185551234"/>
    </setupcall>
  </p>
</card>
```

## 9.14 confirminfo Element

### Description

The confirminfo element is used to specify a text string that will be displayed to the user when the mobile station requests authorization to initiate a call setup.

### Content

*WED-string* including variable references.

### Position

confirminfo may only occur as a child of the setupcall element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the confirmation text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the confirmation text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## Example [43]

This is basically the same example as Example [42], but with user confirmation text added.

```
<card>
  <setupcall>
    <destaddress value="+15185551234"/>
    <confirminfo>
      Do you really want to do this?
    </confirminfo>
  </setupcall>
</card>
```

## 9.15 setupinfo Element

### Description

The `setupinfo` element is used to specify informational text and possibly also an icon that will be displayed to the user during the call setup phase.

### Content

*WED-string* including variable references.

### Position

`setupinfo` may only occur as a child of the `setupcall` element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the call setup text. See section 5.8 for further information.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the call setup text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> .	No

## Compatibility

Wib 1.3 and later.

## Example [44]

This is basically the same example as Example [42], but with the call setup text added.

```
<card>
  <setupcall cmdqualifier="if-not-busy">
    <destaddress value="+15185551234"/>
    <setupinfo>
      Setup in progress
    </setupinfo>
  </setupcall>
</card>
```

## 9.16 setupidlemodetext Element

### Description

The `setupidlemodetext` element sets a text on the idle screen of the mobile station. If the text is left unspecified or empty, the idle text will be removed.

### Content

One of the following:

- *WED-string* including variable references.
- Empty if the idle mode text should be removed. In this case, no icon can be specified.

### Position

`setupidlemodetext` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the idle mode text. See section 5.8 for further information. Compatibility: Wib 1.3 and later.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the idle mode text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> . Compatibility: Wib 1.3 and later.	No

## Compatibility

Wib 1.1 and later except for attributes as noted in the table above.

## Example [45]

```
<card>
  <p>
    <setupidlemodetext>Welcome!!</setupidlemodetext>
  </p>
</card>
```

## 9.17 executestk Element

### Description

The `executestk` element is used to allow execution of any STK command.

### Content

One or more `tlv` elements.

### Position

`executestk` may occur as a child of the `card` or a `p` element.

### Attributes

Name	Value	Explanation	Var
<b>typeofcmd</b> mandatory	See <code>typeofcmd</code> on page 65		No
<b>cmdqualifier</b> optional	hex-bin	Value depends on the command type. Default value is '00'. See STK [17] for details.	No



<b>generalresult</b> mandatory	variable-name	The name of the variable that will hold the general result of running the STK command.	No
<b>additionalinfo</b> mandatory	variable-name	The name of the variable that will hold the additional information that some of the STK commands returns.	No
<b>destvar</b> mandatory	variable-name	The name of the variable that will hold all additional output TLV, if any, as a result of running the STK command.	No

### typeofcmd

Defines the type of STK command to execute, See STK [17] for details.

Possible values:

```
refresh | more-time | poll-interval | polling-off | set-up-event-
list | set-up-call | send-ss | send-ussd | send-short-message |
send-dtmf | launch-browser | play-tone | display-text | get-
inkey | get-input | select-item | set-up-menu | provide-local-
information | timer-management | set-up-idle-mode-text | perform-
card-apdu | power-on-card | power-off-card | get-reader-status |
run-at-command | language-notification | open-channel | close-
channel | receive-data | send-data | get-channel-status | service-
search | get-service-information | declare-service | set-frames |
get-frames-status | retrieve-multimedia-message | submit-
multimedia-message | display-multimedia-message | activate
or
```

integer (0–255)

### Compatibility

Wib 2.0 and later.

### Example [46]

In this example an attribute tlv is created.

0F - item tag

01 – length

00 – Local bearer Technology identifier, in this case Technology independent

It is then used to execute the stk command `get-service-information`.

```
<card>
  <p>
    <createtlv destvar="ATTRIBUTE" tagliteral="0F" value="0100"/>
    <executestk typeofcmd="get-service-information" cmdqualifier="00"
generalresult="RESULT" additionalinfo="ADDITIONAL" destvar="OUT">
      <tlv>$(ATTRIBUTE)</tlv>
    </executestk>
  </p>
</card>
```



## 9.18 tlv Element

### Description

The `tlv` element is used to describe a TLV.

### Content

A `tlv`, formatted as hex-bin. Can include variable references.

### Position

`tlv` may only occur as a child of the `executestk` element.

### Attributes

None.

### Compatibility

Wib 2.0 and later.

### Example [47]

The following wml code will play a Call waiting tone with a duration of 42 seconds.

```
<card>
  <p>
    <createtlv destvar="DURATION" tagliteral="84" value="012A"/>
    <executestk typeofcmd="play-tone" cmdqualifier="00"
generalresult="RESULT" destvar="OUT" additionalinfo="ADDITIONAL">
      <tlv>8E0107</tlv>
      <tlv>$(DURATION)</tlv>
    </executestk>
  </p>
</card>
```

## 9.19 sendss Element

### Description

The `sendss` element sends a message by means of the Supplementary Service.

### Content

Zero or one occurrence of the `catcherror` element.

### Position

`sendss` may occur as a child of the `card` or a `p` element.



## Attributes

Name	Value	Explanation	Var
<b>ss</b> mandatory	string	According to GSM 02.30 [3].	Yes
<b>destvar</b> optional	variable-name	Variable to contain the SS return result message consisting of SS Return Result operation followed by SS Return Result parameters.	No
<b>iconid</b> optional	integer (1–254)	ID of the icon to be used when displaying the <code>title</code> text. See section 5.8 for further information.	No
<b>iconusage</b> optional	adjacent   replace	Controls whether the icon should be displayed together with or instead of the <code>title</code> text. May only be used when the <code>iconid</code> attribute is specified. Default value is <code>adjacent</code> .	No
<b>title</b> optional	WED-string	Text to display when sending. Corresponds to the alpha identifier according to STK [17].	Yes
<b>wibenc</b> optional	SMS-DEFAULT   UCS2   TN-BCD	The Wib encoding of the <code>ss</code> attribute. Default value is <code>SMS-DEFAULT</code> . TN-BCD: The content of the <code>ss</code> attribute is preformatted according to TON+NPI+SSC String as for EF <sub>ADN</sub> .	No

## Compatibility

Wib 2.0 and later.

### Example [48]

```
<card>
  <p>
    <sendss ss="*23*004686859300#" destvar="OUT" title="Forwarding to
SmartTrust"/>
    $(OUT)
  </p>
</card>
```

### Example [49]

In the following example the variable `Number` contains a number fetched from the phonebook on the sim (EF<sub>ADN</sub>). A call forwarding to that number is then performed sending a supplementary service (`*21*<number>#`)

```
<card id="Main">
  <p>
    <substring srcvar="Number" destvar="TONNPI" start="0" span="1"/>
    <substring srcvar="Number" destvar="BCDNumber" start="1" span="255"/>
    <setvar name="sms-pre" value="*21*"/>
  </p>
</card>
```



```

    <setvar name="sms-post" value="#" />
    <convert srcvar="sms-pre" destvar="bcd-pre" type="sms-to-bcd" />
    <convert srcvar="sms-post" destvar="bcd-post" type="sms-to-bcd" />
    <sendss ss="$ (TONNPI) $ (bcd-pre) $ (BCDNumber) $ (bcd-post) "
destvar="Result" wibenc="TN-BCD" />
  </p>
</card>

```

## 9.20 catcherror Element

### Description

The `catcherror` element is used to instruct Wib to continue execution even though an error has occurred. If the parent element can have other sub elements than `catcherror`, the `catcherror` element has to be placed after all other sub elements.

### Content

None.

### Position

`catcherror` may only occur as a child of the `sendusssd`, `launchbrowser`, `setupcall`, `sendss`, `sendsm`, `go`, `select`, `checkterminalprofile` and `conditionaljump` element.

### Attributes

Name	Value	Explanation	Var
result mandatory	variable-name	The name of the variable that will hold the interpreted result of the executed STK command.	No
resultdetails optional	variable-name	The name of the variable that will hold the detailed result of the executed STK command.	No

### Compatibility

Wib 2.0 and later.

### Example [50]

This is basically the same example as Example [42], but with user confirmation text added. Also if the `setupcall` command fails, the execution of the wiblet continues and the reason for the failure of the command can be found in the variable `DETAILS`.

```

<card>
  <setupcall>
    <destaddress value="+15185551234" />
    <confirminfo>

```



```
        Do you really want to do this?
    </confirminfo>
    <catcherror result="RESULT" resultdetails="DETAILS"/>
</setupcall>
<conditionaljump compare="$(RESULT)">
    <test href="#handleerroroncall" value="\x01"/>
</conditionaljump>
</card>
<card id="handleerroroncall">
    <p>
        Failed to setup call. Errorcode: $(DETAILS)
    </p>
</card>
```



## 10 Wib specific elements

### 10.1 add Element

#### Description

The `add` element is used to arithmetically add two integers. These can have three different encodings – SMS-Default, BER-length encoded or binary encoding. Both operands must have the same encoding.

For SMS-Default encoding, values in the range  $-2^{63}$  to  $2^{63}-1$  are supported. Both operands should be formatted as a sequence of digits represented by the characters "0" to "9", possibly preceded by a minus sign to indicate a negative value.

For binary representation values in the range  $-2^{63}$  to  $2^{63}-1$  are supported. The values are encoded using 8 byte two's complement.

For BER-length representation, values in the range 0-255 are supported.

The operation performed is `destvar = destvar + srcvar`.

#### Content

None; the element is always empty.

#### Position

`add` may occur as a child of the `card` or the `p` element.

#### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The name of the variable holding the second operand. The result of the addition will also be stored in this variable. The encoding must follow the <code>class</code> attribute.	No
<b>srcvar</b> optional	variable-name	The name of variable holding the first operand. If this attribute is omitted, the value 1 will be used.	No
<b>class</b> optional	SMS-DEFAULT   ber-length- encoded   hex- binary	The representation of the numbers. Default is SMS-DEFAULT SMS-DEFAULT: See 8.8 for information. ber-length-encoded: BER TLV encoded on 1-2 bytes as defined in ISO 7816-6. hex-binary: See 8.8 for information. Compatibility Wib 2.0 and later.	No



## Compatibility

Wib 1.3 and later except for attributes as noted in the table above. In Wib 1.3 the variables must be coded in SMS-DEFAULT.

### Example [51]

This example illustrates how to add two values and display the result.

```
<card>
  <p>
    <setvar name="OP1" value="100"/>
    <setvar name="OP2" value="200"/>
    <add srcvar="OP1" destvar="OP2"/>
    100+200 = $(OP2)
  </p>
</card>
```

### Example [52]

This example illustrates how to add two binary values and display the result.

```
<card>
  <p>
    <setvar name="OP1" value="31" class="hex-binary"/>
    <setvar name="OP2" value="31" class="hex-binary"/>
    <add srcvar="OP1" destvar="OP2" class="hex-binary"/>
    $(OP2) <!-- Will display 'b' on screen-->
    <convert destvar="RESULT" srcvar="OP2" type="bin-to-hexbin" />
    $(RESULT) <!-- Will display '62' on screen-->
  </p>
</card>
```

### Example [53]

This example illustrates how to add two binary values and display the result.

In this example the display shows 2, since the add element is told to interpret the content of the variables as SMS-DEFAULT.

```
<card id="Main">
  <p>
    <setvar name="SRC" value="1"/>
    <setvar name="DEST" value="31" class="hex-binary"/>
    <add srcvar="SRC" destvar="DEST" class="SMS-DEFAULT"/>
    $(DEST)
  </p>
</card>
```

### Example [54]

This example illustrates how to add two BER-length-encoded values and display the result.

In this example the display shows '8180', since the add element is told to interpret the content of the variables as BER-length-encoded.



```
<card id="Main">
  <p>
    <setvar name="OP1" value="7f" class="hex-binary"/>
    <setvar name="OP2" value="01" class="hex-binary"/>
    <add srcvar="OP1" destvar="OP2" class="ber-length-encoded"/>
    <convert destvar="RESULT" srcvar="OP2" type="bin-to-hexbin" />
    $(RESULT)
  </p>
</card>
```

## 10.2 sub Element

### Description

The `sub` element is used to arithmetically subtract two integers. These can have three different encodings – SMS-Default, BER-length encoded or binary encoding. Both operands must have the same encoding.

For SMS-Default encoding, values in the range  $-2^{63}$  to  $2^{63}-1$  are supported. Both operands should be formatted as a sequence of digits represented by the characters "0" to "9", possibly preceded by a minus sign to indicate a negative value.

For binary representation values in the range  $-2^{63}$  to  $2^{63}-1$  are supported. The values are encoded using 8 byte two's complement.

For BER-length representation, values in the range 0-255 are supported.

The operation performed is `destvar = destvar - srcvar`.

### Content

None; the element is always empty.

### Position

`sub` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The name of the variable holding the first operand. The result of the subtraction will also be stored in this variable. The encoding must follow the <code>class</code> attribute.	No
<b>srcvar</b> optional	variable-name	The name of variable holding the second operand. If this attribute is omitted, the value 1 will be used.	No



<b>class</b> optional	SMS-DEFAULT   ber-length- encoded   hex- binary	The representation of the number. Default is SMS-DEFAULT. SMS-DEFAULT: See 8.8 for information. BER-length-encoded: BER TLV encoded on 1-2 bytes as defined in ISO 7816-6. hex-binary: See 8.8 for information. Compatibility Wib 2.0 and later.	No
--------------------------	--	--	----

### Compatibility

Wib 1.3 and later except for attributes as noted in the table above. In Wib 1.3 the variables must be coded in SMS-DEFAULT.

### Example [55]

In the example, one integer value is subtracted from another and the result is displayed.

```
<card>
  <p>
    <setvar name="OP1" value="200"/>
    <setvar name="OP2" value="100"/>
    <sub srcvar="OP1" destvar="OP2"/>
      100-200=$(OP2)
    </p>
</card>
```

For more examples see add element 10.1

## 10.3 checkterminalprofile Element

### Description

When the mobile station is being initialized after a power-up or reset, the ME notifies the SIM of its capabilities by sending it the so called “Terminal Profile” [8]. The “Terminal Profile” is stored in the SIM and may be tested by Wib using the `checkterminalprofile` element.

The `checkterminalprofile` element allows multiple tests to be performed simultaneously, where each test is represented by a contained `check` element.

If one of the tests fails, the text given by the `text` attribute is displayed, and a jump occurs to the specified URI or alternatively the wiblet execution stops.

### Content

One or more `check` element.

Zero or one occurrence of the `catcherror` element if the `href` attribute of the sub element `check` contains an `external-URL`. The `catcherror` element has to be the last of all other sub elements.



## Position

`checkterminalprofile` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>text</b> optional	WED-string	The text to be displayed if a specified bit in the "Terminal profile" is not set. Default value is the empty string, and thus no text is displayed.	Yes
<b>locinfo</b> optional	force config	Forces inclusion of location info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No
<b>langinfo</b> optional	force config	Forces inclusion of language info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No
<b>imeiinfo</b> optional	force config	Forces inclusion of IMEI/IMEISV info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No

## Compatibility

Wib 1.2 and later.

## Example [56]

This example will check bit 1 in the first byte of the "Terminal profile". If the bit is not set, the text "Error" will be displayed and a jump to card "CARD1" will occur. If the bit is set, bit 3 in the second byte of the "Terminal profile" is checked. If the bit is not set, the text "Error" will be displayed, and then the execution will be stopped.

```
<card>
  <checkterminalprofile text="Error">
    <check index="1" bitmask="1" href="#CARD1"/>
    <check index="2" bitmask="4"/>
  </checkterminalprofile>
</card>
```



## 10.4 check Element

### Description

The `check` element is used to specify a bitmask that is to be checked against a byte or a range of bytes in the “Terminal Profile”. The bitmask can be defined in two different formats. If `bitmask` is used the bitmask is one byte long. If `bitmaskext` is used, the bitmask can be several bytes long. The `index` attribute defines the starting position of the bitmask in the terminal profile.

Note that if the `bitmaskext` attribute is present only one check element is allowed. This implies also that it is not possible to mix check element containing both `bitmask` and `bitmaskext` attributes.

### Content

None; the element is always empty.

### Position

`check` may only occur as a child of the `checkterminalprofile` element.

### Attributes

Name	Value	Explanation	Var
<b>index</b> mandatory	integer (1–255)	The starting position in the "Terminal Profile" where the bitmask shall be applied.	No
<b>bitmask</b> optional	integer (0–255)	The bitmask for specifying what bits to check in the "Terminal Profile". Default value is 0. If this attribute is present, <code>bitmaskext</code> attribute is not allowed.	No
<b>bitmaskext</b> optional	hex-bin	The bitmask for specifying a range of bytes to check in the "Terminal Profile". Default value is '00'h. If this attribute is present, a <code>bitmask</code> attribute is not allowed.	Yes
<b>href</b> optional	WIB-URI	Compatibility: Wib 2.0 and later The destination URI in case the check fails. If the attribute is missing or the value is empty, it indicates that the wiblet should stop the execution instead of jumping. Compatibility and variable references: See section 7.8 .	Yes/ No

### Compatibility

Wib 1.2 and later, except for attributes as noted in the table above



### Example [57]

This example will check if bits 1 and 2 of the thirteenth byte of the terminal profile are both set. If this is the case, the wiblet will terminate normally without any further action. Conversely, if any of the bits are not set, the string “GPRS and CSD not supported” will be displayed to the end-user and (after the message has been cleared by the end-user) a new wiblet identified by the WIB-URI in the check:href attribute will be loaded.

```
<card>
  <checkterminalprofile text="GPRS and CSD not supported">
    <check index="13" bitmask="3" href="!evc!"/>
  </checkterminalprofile>
</card>
```

### Example [58]

This example will check byte 5 and 6 (event driven information) in the terminal profile.

In byte 5 we check if the events MT call, Call connected and Call disconnected are supported. This mask has the value '0E'h ('0000 1110'b) and is stored in the variable MTCALL.

In byte 6 we check if the events Access Technology Change and Network Search Mode Change are supported. This mask is entered as hex-bin value '90'h ('1001 0000'b).

```
<card>
<setvar name="MTCALL" value="0E" class="hex-binary"/>
  <checkterminalprofile text="Required events not supported.">
    <check bitmaskext="$(MTCALL)90" href="!evc!" index="5"/>
  </checkterminalprofile>
</card>
```

### Example [59]

This example will check bit 1 and 2 in byte 8 in the terminal profile to see if timers are supported. If not supported a text will be displayed and the execution of the wiblet will stop.

```
<card id="main">
  <p>
    <checkterminalprofile text="Timers not supported">
      <check index="8" bitmask="3" />
    </checkterminalprofile>
    <input title="Enter timer value on format h:mm:ss" type="text"
name="INPUT" format="*N" maxlength="6" iconid="3" iconusage="replace"/>
    <timer operation="start" var="INPUT"
href="wiblet://smarttrust.com/wakeup.wml"/>
  </p>
</card>
```

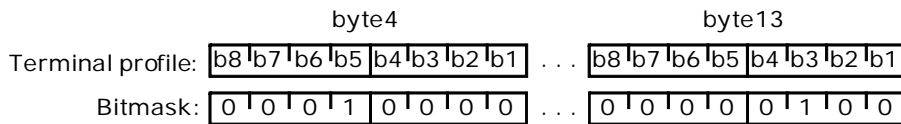


### Example [60]

This example will check if bit 5 of the fourth byte and bit 3 of the thirteenth byte of the terminal profile are set. If this is the case, the wiblet will display the string "ME supports Set up call and Bluetooth!" and terminate normally without further action.

Conversely, if any of the bits are not set, the ME will display the string "Set up call or Bluetooth NOT supported!". After this message has been cleared execution continues at "CARD1" causing the ME to display the string "Here we could execute a fallback routine."

The picture below illustrates how the bitmask is applied:



```

<card>
  <p>
    <setvar name="BITMASK" value="10000000000000000004" class="hex-
binary"/>
    <checkterminalprofile text="Set up call or Bluetooth NOT Supported!">
      <check index="4" bitmaskext="$(BITMASK)" href="#CARD1" />
    </checkterminalprofile>
    ME supports Set up call and Bluetooth!
  </p>
</card>

<card id="CARD1">
  <p>
    Here we could execute a fallback routine.
  </p>
</card>

```

## 10.5 conditionaljump Element

### Description

The conditionaljump element jumps to a URI depending on the value of a test string. Test strings are supplied as a list of contained test elements, and a jump will occur upon the first matching string. If nothing matches, the wiblet execution continues as normal with the next element.

### Content

One or more test elements.

Zero or one occurrence of the catcherror element if the href attribute of the sub element test contains an n. The catcherror element has to be the last of all other sub elements.



## Position

`conditionaljump` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>compare</b> mandatory	string	The value to compare against. The Wib encoding is determined by the <code>class</code> attribute. Hexadecimal escape characters are supported.	Yes
<b>class</b> optional	SMS-DEFAULT   UCS2   binary	Used for conversion purposes. Determines the Wib encoding of the <code>compare</code> attribute and the <code>value</code> attributes in any contained <code>test</code> elements. Value <code>binary</code> has the same effect as explained in section 8.8 . Default value is determined by the wiblet encoding.	No
<b>locinfo</b> optional	force config	Forces inclusion of location info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No
<b>langinfo</b> optional	force config	Forces inclusion of language info in the uplink message. Default is <code>config</code> , which means follow configuration.	No
<b>imeiinfo</b> optional	force config	Forces inclusion of IMEI/IMEISV info in the uplink message. Default is <code>config</code> , which means follow configuration. Compatibility: Wib 2.0 and later	No

## Compatibility

Wib 1.2 and later.

## Example [61]

In this example, a jump will occur to card "CARD2", since there will be a match in the second condition.

```
<card>
  <p>
    <setvar name="A" value="a" class="SMS-DEFAULT" />
    <setvar name="B" value="b" class="SMS-DEFAULT" />
    <conditionaljump compare="$ (A) $ (B) c">
      <test href="#CARD1" value="a$ (A) $ (B) "/">
      <test href="#CARD2" value="a$ (B) c"/>
  </p>
</card>
```



```

    </conditionaljump>
  </p>
</card>

```

### Example [62]

If the wiblet encoding is UCS2, a jump will occur to card "CARD2", since there will be a match in the second condition in the *first* conditionaljump, due to the default value of the class argument. If the wiblet encoding is SMS-DEFAULT, a jump will occur to card "CARD4", since there will be a match in the second condition in the *second* conditionaljump.

```

<card>
  <p>
    <setvar name="VAR1" value="abc" class="UCS2"/>
    <setvar name="VAR2" value="abc" class="SMS-DEFAULT"/>
    <setvar name="VAR3" value="ab" class="UCS2"/>
    <conditionaljump compare="$(VAR1)">
      <test href="#CARD1" value="$(VAR2)"/>
      <test href="#CARD2" value="$(VAR3) c"/>
    </conditionaljump>
    <conditionaljump compare="$(VAR1)" class="UCS2">
      <test href="#CARD3" value="$(VAR2)"/>
      <test href="#CARD4" value="$(VAR3) c"/>
    </conditionaljump>
  </p>
</card>

```

## 10.6 test Element

### Description

The test element is used to specify a test string for the conditionaljump element. The value attribute will be compared byte-by-byte to the value of the compare attribute in the conditionaljump element.

### Content

None; the element is always empty.

### Position

test may only occur as a child of the conditionaljump element.

### Attributes

Name	Value	Explanation	Var
<b>href</b> mandatory	WIB-URI	The destination URI in case the test string matches the condition. Compatibility and variable references: See section 7.8 .	Yes/ No



<b>value</b> mandatory	string	Defines the value that will be compared with the value of the <code>compare</code> attribute in the <code>conditionaljump</code> element. The Wib encoding is determined by the <code>class</code> attribute of the parent <code>conditionaljump</code> element.	Yes
---------------------------	--------	--	-----

## Compatibility

Wib 1.2 and later.

## Example [63]

In this example, a jump to `ERRORHANDLING` will occur if there is an error when executing the `go` element.

```
<card id="main">
  <p>
    <input title="First name" type="text" name="FN"/>
    <input title="Last name" type="text" name="LN"/>
    <go method="post"
href="http://webserver/page.jsp?f=${FN} & l=${LN}">
      <catcherror result="RESULT" resultdetails="DETAILS"/>
    </go>
    <conditionaljump compare="${RESULT}">
      <test href="#ERRORHANDLING" value="\x01"/>
    </conditionaljump>
  </p>
</card>

<card id="ERRORHANDLING">
  <p>
    Failed to send content to provider. Errorcode: ${DETAILS}
  </p>
</card>
```

## 10.7 convert Element

### Description

The `convert` element is used to convert various forms of binary data to a text representation, and vice versa, to create an alpha identifier from a text and vice versa and to convert between packed 7-bit SMS-DEFAULT text to unpacked 8-bit SMS-DEFAULT text and vice versa.

### Content

None; the element is always empty.

### Position

`convert` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The name of the variable where the converted data will be stored. The Wib encoding of the variable depends on the type of conversion. See below for details.	No
<b>srcvar</b> mandatory	variable-name	The name of the variable holding the data to convert.	No
<b>type</b> mandatory	bin-to-decimal-groups   bin-to-hexbin   bin-to-int   unpacked-to-alpha   packed-to-alpha   ucs2-to-alpha   alpha-to-text   packed-to-unpacked   unpacked-to-packed   bcd-to-sms   sms-to-bcd	The type of conversion. For details regarding the conversion types, see below.  Note: <code>unpacked-to-alpha</code>   <code>packed-to-alpha</code>   <code>ucs2-to-alpha</code>   <code>alpha-to-text</code>   <code>unpacked-to-packed</code>   <code>packed-to-unpacked</code> is only supported by Wib 2.0 or later.	No
<b>details</b> optional	ignore-ext   ext-type-1   ext-type-2	This attribute may be used when the conversion type is either <code>bcd-to-sms</code> or <code>sms-to-bcd</code> to provide further guidance for the conversion. Default value is <code>ext-type-1</code> . For more details, see below.	No
<b>outputdcs</b> optional	variable-name	The DCS of the data in <code>destvar</code> after conversion. Compatibility Wib 2.0 and later.	No

`bin-to-decimal-groups` converts each byte in the source variable to 1–3 decimal digits. Each digit is expressed using characters "0" – "9" in the SMS Default Alphabet. Converted numbers are separated by a space character. E.g. (hexadecimal) value "FF0A0330" is converted to "255 10 3 48". The Wib encoding of `destvar` will be SMS-DEFAULT.

`bin-to-hexbin` converts each byte in the source variable to two hexadecimal digits. Each digit is expressed using characters "0" – "F" in the SMS Default Alphabet. Converted numbers are separated by a space character. E.g. (hexadecimal) value "2F7A13BC" is converted to "2F 7A 13 BC". The Wib encoding of `destvar` will be SMS-DEFAULT.



`bin-to-int` converts an integer in binary format to an integer represented as a string of decimal digits. Each digit is expressed using characters "0" – "9" in the SMS Default Alphabet. E.g. (hexadecimal) value "FFFF" is converted to "65535". The Wib encoding of `destvar` will be SMS-DEFAULT.

`unpacked-to-alpha` converts a GSM 7 bit default alphabet in unpacked format into an alpha identifier in GSM 7 bit default alphabet format.

`packed-to-alpha` converts a text in GSM 7 bit default alphabet in packed format into an alpha identifier in GSM 7 bit default alphabet format.

`ucs2-to-alpha` converts an UCS2 formatted text into an alpha identifier in UCS2 format.

`alpha-to-text` converts an Alphaidentifer to text. If the Alphaidentifer is in SMS DEFAULT format the output in `destvar` will be in SMS DEFAULT unpacked format. If the Alphaidentifer is in UCS2 format the output in `destvar` will be in UCS2 format. The DCS of the converted text will be stored in `outputdcs` if that attribute is present.

`packed-to-unpacked` converts an packed 7-bit SMS-DEFAULT text into unpacked 8-bit SMS-DEFAULT text. The DCS of the converted text will be stored in `outputdcs` if that attribute is present.

`unpacked-to-packed` converts an unpacked 8-bit SMS-DEFAULT text into packed 7-bit SMS-DEFAULT text. The DCS of the converted text will be stored in `outputdcs` if that attribute is present.

`bcd-to-sms` converts each (BCD formatted) byte in the source variable to two characters in the SMS Default Alphabet. The Wib encoding of `destvar` will be SMS-DEFAULT.

`sms-to-bcd` converts a string of SMS Default Alphabet characters to BCD. The Wib encoding of `destvar` will be Binary.

When converting to/from BCD, mapping of extended BCD characters/values is controlled by the `details` attribute, according to the two tables below.



BCD	SMS
0xA	"A"
0xB	"B"
0xC	"C"
0xD	"D"
0xE	"E"
0xF	"F"

*Extended type 1*  
(ext-type-1)

BCD	SMS
0xA	"*"
0xB	"#"
0xC	"a"
0xD	"b"
0xE	"c"
0xF	""

*Extended type 2*  
(ext-type-2)

ignore-ext means that extended BCD characters will be ignored in the conversion.

### Compatibility

Wib 1.3 and later, except for attributes as noted in the table above.

### Example [64]

In the example 255 is displayed on screen.

```
<card>
  <p>
    <setvar name="BYTES" value="FF" class="hex-binary"/>
    <convert destvar="DEST" srcvar="BYTES" type="bin-to-int"/>
    $(DEST)
  </p>
</card>
```

### Example [65]

In the example an alpha identifier is created in card "MAIN" and stored in the variable "DEST". This alpha identifier is then used in CARD2 to create a tlv ALPHAID, that is later used in the executestk element.

```
<card id="MAIN">
  <p>
    <setvar name="TEXT" value="Playing tone" class="SMS-DEFAULT"/>
    <convert destvar="DEST" srcvar="TEXT" type="packed-to-alpha"/>
  </p>
  <go href="#CARD2"/>
</card>

<card id="CARD2">
  <p>
    <createtlv destvar="ALPHAID" tagliteral="05" value="$(DEST)"/>
    <createtlv destvar="DURATION" tagliteral="84" value="012A"/>
    <executestk typeofcmd="play-tone" cmdqualifier="00"
    generalresult="RESULT" destvar="OUT" additionalinfo="ADDITIONAL">
      <tlv>$(ALPHAID)</tlv>
      <tlv>8E0107</tlv>
```



```

        <tlv>$(DURATION)</tlv>
    </executestk>
</p>
</card>

```

## 10.8 getbuffersize Element

### Description

The `getbuffersize` element reads the current Wib receive buffer size in bytes and assigns it to the specified variable.

### Content

None; the element is always empty.

### Position

`getbuffersize` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The variable to receive the buffer size value. The Wib encoding of the variable will always be binary.	No

### Compatibility

Wib 1.1 and later.

### Example [66]

In the example, the size of the Wib receiving buffer on the SIM is put in the variable `SZ`. Then it is sent back to the Content Provider.

```

<card>
  <p>
    <getbuffersize destvar="SZ"/>
    <go href="http://webserver/page.jsp?sz=$(SZ)"/>
  </p>
</card>

```

## 10.9 getbrowserinfo Element

### Description

The `getbrowserinfo` element reads information associated with Wib from the SIM and assigns it to the specified variable.

The tables below describe the structure of the returned information.



<b>Contents</b>	<b>M/O</b>	<b>Length</b>
Total length of following data.	M	1
Manufacturer identifier. Integer (0 – 255).	M	1
First level version number. Integer (0 – 255).	M	1
Second level version number. Integer (0 – 255).	M	1
Manufacturer specific Wib version info. Integer (0 – 255).	M	1
List of plug-in entries. See table below.	O	A

As shown, plug-in entries are optional. If they are omitted, the total length if the output data is 4. If plug-ins are included, they are formatted according to the following two tables:

<b>Contents</b>	<b>M/O</b>	<b>Length</b>
Number of plug-in entries	M	1
Plug-in entry #1.	M	B
Plug-in entry #2.	O	C
...	...	...
Plug-in entry #N.	O	M

Each plug-in entry is formatted according to:

<b>Contents</b>	<b>M/O</b>	<b>Length</b>
Length of plug-in name	M	1
Plug-in name.	M	X
Plug-in version.	M	3

### Content

None; the element is always empty.

### Position

getbrowserinfo may occur as a child of the card or the p element.



## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The variable to receive the information. The Wib encoding of the variable will always be binary.	No

## Compatibility

Wib 1.1 and later.

## Example [67]

In the example, the information is put in the variable BINFO. On the next line, the version information is sent back to the Content Provider.

```
<card>
  <p>
    <getbrowserinfo destvar="BINFO"/>
    <go href="http://webserver/page.jsp?version=${(BINFO)}"/>
  </p>
</card>
```

## 10.10 plugin element

### Description

The `plugin` element is used to call plug-ins in Wib.

For more information about Wib plug-ins, please consult *SmartTrust Wib™ Plug-ins Specification* [11].

### Content

None; the element is always empty.

### Position

`plugin` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>class</b> optional	SMS-DEFAULT   UCS2   binary	The Wib encoding of the plug-in output. Default is binary.	No
<b>destvar</b> mandatory	variable-name	Name of a variable that will contain the output data from the plug-in.	No
<b>name</b> mandatory	string	The name of the plug-in to call. The Wib encoding is SMS-DEFAULT.	No



<b>params</b> mandatory	string	The input parameters to the plug-in. The Wib encoding is SMS-DEFAULT. Hexadecimal escape characters are allowed.	Yes
----------------------------	--------	--	-----

## Compatibility

Wib 1.1 and later.

## Example [68]

In the example, the P7 plug-in is called and the output is sent back to the content-provider.

```

<card>
  <p>
    <setvar value="S" name="CES"/>
    <setvar value="\x00" name="OPTS"/>
    <setvar value="O dear Ophelia, I am ill at these numbers"
name="TTBS"/>
    <plugin name="P7" destvar="SIG" params="$ (CES) $ (OPTS) $ (TTBS) "/>
    <go href="http://webserver/main?sig=$(SIG) "/>
  </p>
</card>

```

## 10.11 setreturntarvalue Element

### Description

The `setreturntarvalue` element makes sure that the next message submitted from Wib to the UG has destination TAR address as identified by `recordid`.

**Note:** As a side effect, changing the return TAR value may also change the operational mode of Wib. The operational mode affects the behavior upon submitting data from Wib to the UG, as explained for the (`go` element) attribute `enterwait` in section 8.9 .

### Content

None; the element is always empty.

### Position

`setreturntarvalue` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>recordid</b> mandatory	integer (1–254)	Identifies a new TAR value that will be used when submitting data from Wib. This also determines the Wib operational mode.	No



## Compatibility

Wib 1.1 and later.

## Example [69]

In the example, the WIG WML document "index.wml" will be fetched by the UG with the TAR value specified in record 2 of the Wib EF TAR file on the SIM.

```
<card>
  <p>
    <setreturntarvalue recordid="2"/>
    <go href="http://webserver/index.wml"/>
  </p>
</card>
```

## 10.12 substring Element

### Description

The `substring` element copies a substring of a variable to another variable. The copying is made on a byte-per-byte basis.

### Content

None; the element is always empty.

### Position

`substring` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The name of the variable to copy the substring to.	No
<b>srcvar</b> mandatory	variable-name	The name of the variable to copy the substring from.	No
<b>span</b> optional	integer (0-255) or integer (-8191-8191) excluding the value 0	Number of bytes to copy. If the value is larger than the number of bytes available, the rest of the variable is copied. Default value is 8191. If <code>span</code> is a negative number it defines that the number of bytes preceding, and including the position defined by <code>start</code> shall be copied Note: integer (-8191-8191) is only supported by Wib 2.0 and later.	Yes/ No



<b>start</b> optional	integer (0-254) or integer (0-8190)	Position where to start copying. First position in the source string is position 0. Default value is 0. Note: integer (0-8190) is only supported by Wib 2.0 and later.	Yes/ No
--------------------------	---	---	------------

## Compatibility

Wib 1.2 and later, except for attributes as noted in the table above.

### Example [70]

The variable `OUT` will be set to the string "abc".

```
<card>
  <p>
    <setvar name="IN" value="abcdef" class="SMS-DEFAULT" />
    <substring srcvar="IN" destvar="OUT" start="0" span="3"/>
  </p>
</card>
```

### Example [71]

The variable `OUT` will be set to a binary string with byte values 00 62 00 63, i.e. "bc" in UCS2 format.

```
<card>
  <p>
    <setvar name="IN" value="abcdef" class="UCS2"/>
    <substring srcvar="IN" destvar="OUT" start="2" span="4"/>
  </p>
</card>
```

### Example [72]

The value of variable `IN` will be copied to variable `OUT`.

```
<card>
  <p>
    <setvar name="IN" value="abcdef"/>
    <substring srcvar="IN" destvar="OUT"/>
  </p>
</card>
```

### Example [73]

The variable `OUT` will be set to the string "de".

```
<card>
  <p>
    <setvar name="IN" value="abcdef" class="SMS-DEFAULT" />
    <substring srcvar="IN" destvar="OUT" start="4" span="-2" />
  </p>
</card>
```



## 10.13 swapnibbles Element

### Description

The `swapnibbles` element is used to swap the nibbles of each byte of a value stored in a variable.

### Content

None; the element is always empty.

### Position

`swapnibbles` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	Name of the variable holding the value to be nibble-swapped.	No

### Compatibility

Wib 1.3 and later.

### Example [74]

This example illustrates how to swap the nibbles of all bytes in a binary string. The value contained in the variable after the nibble-swap will be the byte values ED AC BF DA.

```

<card>
  <p>
    <setvar name="OP" value="DECAFBAD" class="hex-binary"/>
    <swapnibbles destvar="OP" />
  </p>
</card>

```

## 10.14 timer Element

### Description

The `timer` element is used to specify a timer operation in Wib. This includes start of a timer, fetching the value of a timer and deactivating a timer.

When a timer "goes of", the wiblet specified when the timer was started, is executed.

**Note:** Time values used with the `timer` element are relative, i.e. they indicate a time difference. In other words, timers in Wib are actually countdowns.



## Content

None; the element is always empty.

## Position

`timer` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>href</b> mandatory	WIBlet-URI	URI to a locally stored wiblet that is the subject of the operation. The "wiblet://return" URI can not be used.	No
<b>operation</b> mandatory	<code>start</code>   <code>get</code>   <code>deactivate</code>	The operation to perform. <code>start</code> associates one of the available timers (the mobile station has 8) to a wiblet so that the wiblet will be executed after the amount of time indicated by the variable named by attribute <code>var</code> has elapsed. <code>get</code> reads the current timer value associated with the wiblet and stores it in the variable named by attribute <code>var</code> . <code>deactivate</code> disassociates the named wiblet with all timers. This operation also stores the timer value in the variable named by attribute <code>var</code> .	No
<b>var</b> mandatory	variable-name	The name of the variable holding the timer value before ( <code>start</code> ) or after ( <code>get</code> and <code>deactivate</code> ) the operation. The value of the variable is formatted "hhmmss" where <i>hh</i> is hours, <i>mm</i> minutes and <i>ss</i> seconds. If a returned value is empty after <code>get</code> or <code>deactivate</code> , this indicates that the wiblet specified in the <code>href</code> attribute is not associated with any timers.	No

## Compatibility

Wib 1.3 and later.

## Example [75]

In this example a timer is set to go off in one hour and consequently execute the named wiblet at that time. This assumes that the wiblet has been stored in Wib at



an earlier stage.

```

<card>
  <p>
    <setvar name="TIMER" value="010000"/>
    <timer operation="start" var="TIMER"
href="wiblet://smarttrust.com/check_imei.wml"/>
  </p>
</card>

```

### Example [76]

This example shows how to read and present the time left until the named wiblet should go off.

```

<card>
  <p>
    <timer operation="get" var="TIMER"
href="wiblet://smarttrust.com/wake/me/up/before/you/gogo.wml"/>
    I will wake you up in $(TIMER) hours.
  </p>
</card>

```

## 10.15 transcode Element

### Description

The `transcode` element is used to convert text represented in the SMS Default Alphabet encoding to a text in the UCS2 encoding, and vice versa.

### Content

None; the element is always empty.

### Position

`transcode` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The name of the variable where the converted text should be stored.	No
<b>dir</b> mandatory	sms-to-ucs2   ucs2-to-sms	The direction of the conversion. sms-to-ucs2 converts from the SMS Default Alphabet to UCS2. ucs2-to-sms converts from UCS2 to the SMS Default Alphabet.	No
<b>srcvar</b> mandatory	variable-name	The name of the variable holding the text to convert.	No



## Compatibility

Wib 1.3 and later.

### Example [77]

In the example, the SMS Default Alphabet encoded text "Smarttrust" is converted to the UCS2 encoding and stored in variable TO.

```
<card>
  <p>
    <setvar name="FROM" value="Smarttrust" class="SMS-DEFAULT" />
    <transcode dir="sms-to-ucs2" srcvar="FROM" destvar="TO"/>
  </p>
</card>
```

### Example [78]

In the example, the user inputs some text in SMS-DEFAULT which is converted to UCS2 and displayed.

```
<wml wibletenc="UCS2">
  <card>
    <p>
      <input title="Answer?" name="A" class="SMS-DEFAULT" />
      <transcode dir="sms-to-ucs2" srcvar="A" destvar="B"/>
      You answered: $(B)
    </p>
  </card>
</wml>
```

## 10.16 groupvar Element

### Description

The `groupvar` element is used to group the content of several variables into one. The variables to be grouped are represented as a list of contained `var` elements.

The resulting variable will be formatted LV (length-value) where the V part contains the LV's for the grouped variables. The length L can be coded on one or two bytes.

LV -- the resulting variable LV

LVLVLV -- LV's for the grouped variables

`groupvar` is typically used to group variables before they are utilized as parameters in a plug-in call.

### Content

One or more `var` elements.

### Position

`groupvar` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	Name of the variable where the grouped value will be stored. The Wib encoding of the variable will be Binary.	No
<b>lengthenc</b> optional	one-byte   two-byte	Length encoding of L in the LV-pairs. Compatibility Wib 2.0 and later.	

## Compatibility

Wib 1.3 and later, except for attributes as noted in the table above.

## Example [79]

In the example, two text string entered by the user is sent to a plug-in for further processing, and finally the plug-in output is sent back to the Content Provider.

```
<card>
  <p>
    <input title="Name?" name="NAME"/>
    <input title="Number?" name="NUMBER"/>
    <groupvar destvar="NN">
      <var name="NAME"/>
      <var name="NUMBER"/>
    </groupvar>
    <plugin name="DOIT" destvar="OUT" params="$ (NN)"/>
    <go href="http://webserver/main?OUT=$(OUT)"/>
  </p>
</card>
```

## 10.17 ungroupvar Element

### Description

The `ungroupvar` element is used to ungroup the content of one variable into several variables. The variables to be populated are represented as a list of contained `var` elements. The length L in the LV-pairs can be coded on one or two bytes.

`ungroupvar` is typically used to split the output of a plug-in into several separate variables.

### Content

One or more `var` elements.

### Position

`ungroupvar` may occur as a child of the `card` or the `p` element.



## Attributes

Name	Value	Explanation	Var
<b>srcvar</b> mandatory	variable-name	Name of the variable where the value to be ungrouped is stored.	No
<b>lengthenc</b> optional	one-byte   two-byte	Length encoding of L in the LV-pairs. Compatibility Wib 2.0 and later.	

## Compatibility

Wib 1.3 and later, except for attributes as noted in the table above.

## Example [80]

In the (fictive) example, a sentence with three words is tokenized by a plug-in and the output from the plug-in is ungrouped into three distinct variables holding one word each.

```
<card>
  <p>
    <setvar name="WORDS" value="Veni vidi vici"/>
    <plugin name="TOKENIZE" destvar="OUT" params="$ (WORDS)"/>
    <ungroupvar srcvar="OUT">
      <var name="VENI"/>
      <var name="VIDI"/>
      <var name="VICI"/>
    </ungroupvar>
    $ (VENI) $ (VIDI) $ (VICI)
  </p>
</card>
```

## 10.18 var Element

### Description

The `var` element is used to represent a variable name in conjunction with the `group` or `ungroup` element.

### Content

None; the element is always empty.

### Position

`var` may only occur as a child of the `group` or the `ungroup` element.



## Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	variable-name	Name of the variable. The Wib encoding of the variable will be Binary if the <code>var</code> element is used within <code>ungroupvar</code> .	No

## Compatibility

Wib 1.3 and later.

## 10.19 terminalprofile Element

### Description

When the mobile station is being initialized after a power-up or reset, the ME notifies the SIM of its capabilities by sending it the so called “Terminal Profile”, see STK [17] for more details. The “Terminal Profile” is stored in the SIM and may be retrieved by Wib using the `terminalprofile` element.

### Content

None.

### Position

`terminalprofile` may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The variable to receive the terminal profile. The Wib encoding of the variable will always be binary and are according to STK [17].	No

## Compatibility

Wib 2.0 and later.

## Example [81]

In the example, the information is put in the variable `TERMPROF`. On the next line, the version information is sent back to the Content Provider.

```
<card>
  <p>
    <terminalprofile destvar="TERMPROF"/>
    <go href="!tprof!?tp=${TERMPROF}"/>
  </p>
</card>
```



## 10.20 executewiblet Element

### Description

The `executewiblet` element is used to start execution of a wiblet stored in the `srcvar` variable. There is a possibility to verify the MAC before execution. In this case the key to be used can be specified.

### Content

None.

### Position

`executewiblet` may occur as a child of the `card` or the `p` element.

### Attributes

Name	Value	Explanation	Var
<b>srcvar</b> mandatory	variable-name	The variable that contain the bytecode to be executed. The content shall be coded in hex-binary.	No
<b>verifymac</b> optional	true   false	Indicator if a MAC shall be validated or not. Default value is <code>false</code> .	No
<b>mackey</b> optional	Integer (0-7)	The key record to use for MAC validation.	No

### Compatibility

Wib 2.0 and later.

### Example [82]

In this example an ussd message is sent to a content provider. The result is stored in the variable `ussdwiblet` and is then executed.

```
<card>
  <p>
    <sendussd destvar="ussdwiblet" ussd="*0100*#" />
    <executewiblet srcvar="ussdwiblet"/>
  </p>
</card>
```

## 10.21 handleexit Element

### Description

This `handleexit` element gives the executing wiblet the possibility to override the default behavior of Wib on normal wiblet termination as well as on error termination.

The latest call to `handleexit` replaces the effect of any earlier call.



## Content

None.

## Position

handleexit may occur as a child of the card or p element.

## Attributes

Name	Value	Explanation	Var
<b>normal</b> optional	wiblet-URI	Defines what shall happen when a wiblet has terminated normally. If the attribute is omitted Wib reverts to default behavior on normal exit. wiblet-URI: The reserved wiblet_URI wiblet://mainmenu causes Wib to show the main menu. Any other wiblet-URI will cause a navigation action. Compatibility and variable references: See section 7.8 .	Yes
<b>error</b> optional	wiblet-URI	Defines what shall happen when a wiblet has terminated with error. If the attribute is omitted Wib reverts to default behavior on error exit. wiblet-URI: The reserved wiblet-URI wiblet://mainmenu causes Wib to show the main menu. Any other wiblet-URI will cause a navigation action. A wiblet-URI ending with "?var" will cause variable values to be retained. wiblet://errorhandler?var Compatibility and variable references: See section 7.8 .	Yes

## Compatibility

Wib 2.0 and later.



### Example [83]

In the example, two text strings entered by the user is sent to a plug-in for further processing, and finally the plug-in output is sent back to the Content Provider. If an error occur while sending the data, an error handling wiblet is called. Specific information about the error is sent to the content provider. See *UG & WSM – Application Developers Guide[10]* for more information.

```
<card id="Main">
  <p>
    <handleexit error="wiblet://errorhandler.wml?var" />
    <input title="Name?" name="NAME:stack01"/>
    <input title="Number?" name="NUMBER:stack02"/>
    <groupvar destvar="NN">
      <var name="NAME:stack01"/>
      <var name="NUMBER:stack02"/>
    </groupvar>
    <plugin name="DOIT" destvar="OUT" params="$ (NN)"/>
    <go href="http://webserver/main?OUT=$(OUT)"/>
  </p>
</card>
```

This is the error handling wiblet stored on the SIM. The content of the variables with id 90 and 91 are sent to the content provider.

```
<head>
  <meta name="wiblet-uri"
content="wiblet://smarttrust.com/sample/errorhandler.wml"/>
</head>
<card id="errorhandler">
  <go href="http://webserver/main?Code=$( :IDx90) & amp; Tag=$( :IDx91)"/>
</card>
```

## 10.22 createtlv Element

### Description

The `createtlv` element is used to create a TLV given a value and a tag. The generated TLV is stored in the `destvar` variable. Variable references used in the value of the `value` attribute are replaced on a byte-per-byte basis in Wib. See Example [85].

### Content

None

### Position

`createtlv` may occur as a child of the `card` or a `p` element.



## Attributes

Name	Value	Explanation	Var
<b>tagliteral</b> conditional	hex-bin	The tag value of the TLV to be created expressed like a hex-bin. Either this attribute or the attribute tagvariable needs to be present.	No
<b>tagvariable</b> conditional	variable-name	The tag value of the TLV to be created expressed like a variable name. Either this attribute or the attribute tagliteral needs to be present.	No
<b>value</b> mandatory	string	The value of the TLV to be created,	Yes
<b>destvar</b> mandatory	variable-name	Variable that will contain the TLV after creation.	No
<b>class</b> optional	SMS-DEFAULT   UCS2   binary   hex-binary   base64-binary	Used for conversion purposes. Default value is binary. See 8.8 for more information on the coding.	No

## Compatibility

Wib 2.0 and later.

### Example [84]

The following example will create a TONE TLV object and a DURATION TLV object. It will then use the executestk command to play the tone “Called subscriber busy” for 10 seconds.

```
<card>
  <p>
    <setvar name="TONETAG" value="0E" class="hex-binary"/>
    <createtlv destvar="TONE" tagvariable="TONETAG" class="hex-binary"
value="02"/>
    <createtlv destvar="DURATION" tagliteral="04" class="hex-binary"
value="0110"/>
    <executestk typeofcmd="play-tone" cmdqualifier="00"
generalresult="RESULT" additionalinfo="addinfo" destvar="OUT">
      <tlv>$(TONE)</tlv>
      <tlv>$(DURATION)</tlv>
    </executestk>
  </p>
</card>
```

### Example [85]

First, the variable SWE will be set to the hexadecimal value "7376". Then we create a language tag ('2D'h)

After that the variable LANGUAGE will be set to the hexadecimal value "2D027376".



Last we do an execution of the STK command `language-notification` with the command qualifier set to 00, i.e. non-specific language notification.

```
<card>
  <p>
    <setvar name="SWE" value="7375" class="hex-binary"/>
    <setvar name="LANGUAGETAG" value="2D" class="hex-binary"/>
    <createtlv destvar="LANGUAGE" tagvariable="LANGUAGETAG"
value="$ (SWE) "/>
    <executestk typeofcmd="language-notification" cmdqualifier="00"
generalresult="RESULT" additionalinfo="addinfo" destvar="OUT">
      <tlv>$(LANGUAGE)</tlv>
    </executestk>
  </p>
</card>
```

### 10.23 extracttlv Element

#### Description

The `extracttlv` element is used to break apart a TLV into its components and store the tag, length and value in separate variables. The input to the command is a list of TLVs. After execution the TLV list without the extracted TLV is stored in the variable `outputlist`.

#### Content

None.

#### Position

`extracttlv` may occur as a child of the `card` or a `p` element.

#### Attributes

Name	Value	Explanation	Var
<b>tagliteral</b> optional	hex-bin	The tag value of the TLV to be extracted expressed like a hex-bin. If <code>tagliteral</code> and <code>tagvariable</code> is not present or if the value is '00'h, the first TLV in the list will be extracted. If non zero, the first TLV with that tag will be extracted.	No
<b>tagvariable</b> optional	variable-name	The tag value of the TLV to be extracted expressed like a variable name. If <code>tagliteral</code> and <code>tagvariable</code> is not present or if the value is '00'h, the first TLV in the list will be extracted. If non zero, the first TLV with that tag will be extracted.	No



<b>inputlist</b> mandatory	hex-bin	A TLV list that is an octet string that may contain variable references.	Yes
<b>outputtag</b> mandatory	variable-name	Output variable that will hold the tag of the requested TLV. Will be zero if the requested tag is not found or if the <code>inputlist</code> is empty.	No
<b>outputlength</b> mandatory	variable-name	Output variable that will hold the length of the requested TLV. Will be zero if the requested tag is not found or if the <code>inputlist</code> is empty.	No
<b>outputvalue</b> mandatory	variable-name	Output variable that will hold the value of the requested TLV. Will be empty if the requested tag is not found or if the <code>inputlist</code> is empty.	No
<b>remaininglist</b> mandatory	variable-name	Output variable that will hold the TLV list without the extracted TLV.	No

### Compatibility

Wib 2.0 and later.

### Example [86]

In the example the `inputlist` is searched for a TLV with the tag '04'h. The tag, length and value of that tlv is then written into the corresponding variables.

```
<card id="main">
  <p>
    <setvar name="inputlist" value="0E010204020110" class="hex-binary"/>
    <go href="#extractcard"/>
  </p>
</card>

<card id="extractcard">
  <p>
    <extracttlv tagliteral="04" inputlist="$(inputlist)"
    outputtag="TAGNAME" outputlength="LENGTH" outputvalue="VALUE"
    remaininglist="LIST"/>
  </p>
</card>
```

After execution the variables will hold the following values:

```
TAGNAME : 04
LENGTH : 02
VALUE : 0110
LIST : 0E0102
```

## 10.24 geticcid Element

### Description

The `geticcid` element is used to retrieve the ICCID that is stored on the SIM.



## Content

None.

## Position

geticcid may occur as a child of the `card` or the `p` element.

## Attributes

Name	Value	Explanation	Var
<b>destvar</b> mandatory	variable-name	The variable to receive the ICCID. The Wib encoding of the variable will be as it is stored in EF <sub>ICCID</sub> .	No

## Compatibility

Wib 2.0 and later.

## Example [87]

In the example, the ICCID is put in the variable ICCID. On the next line, the ICCID is displayed on screen and sent back to the Content Provider.

```
<card>
  <p>
    <geticcid destvar="ICCID"/>
    <swapnibbles destvar="ICCID"/>
    <convert type="bcd-to-sms" destvar="SMSICCID" srcvar="ICCID" />
    $(SMSICCID)
    <go href="http://webserver/page.jsp?iccid=$(SMSICCID)"/>
  </p>
</card>
```



## 11 Server Side elements

### 11.1 wigplugin Element

#### Description

The `wigplugin` element defines a call to a Server Side plug-in that is supposed to execute in the UG.

Refer to Appendix C for a listing of available Server Side plug-ins and examples.

#### Content

Zero or more `param` elements.

#### Position

`wigplugin` may only occur as a child of the `wml` element.

#### Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	string	Name of the UG Side plug-in.	No

#### Compatibility

Independent of Wib version.

### 11.2 param Element

#### Description

The `param` element defines an input parameter to the Server Side plug-in.

#### Content

None; the element is always empty.

#### Position

`param` may occur only as a child of the `wigplugin` element.

#### Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	string	Name of the parameter.	No
<b>value</b> mandatory	string	Value of the parameter.	No



## Compatibility

Independent of Wib version.



## 12 Other elements

### 12.1 head Element

#### Description

The `head` element contains information related to the WIG WML document as a whole (meta-data).

#### Content

Zero or more `meta` elements.

#### Position

`head` may only occur as a child of the `wml` element.

#### Attributes

None.

#### Compatibility

Independent of Wib version.

### 12.2 meta Element

#### Description

The `meta` element contains generic meta-information relating to the WIG WML document. Meta-information is defined by property names and contents.

The only property name handled is "wiblet-uri", which shall be used for indicating the wiblet-URI for a locally stored wiblet. The property content shall contain a *wiblet-URI*, but the value "wiblet://return" is not allowed.

The property names `version`, `description` and `author` can be used but has no special handling.

#### Content

None; the element is always empty.

#### Position

`meta` may only occur as a child of the `head` element.

#### Attributes

Name	Value	Explanation	Var
<b>name</b> mandatory	string	The property name.	No



<b>content</b> mandatory	string	The property content.	No
-----------------------------	--------	-----------------------	----

---

## Compatibility

Independent of Wib version.

## Example [88]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
                          http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <head>
    <meta name="wiblet-uri" content="wiblet://smarttrust.com/myApp"/>
  </head>
  <card>
    <p>
      This is a locally stored wiblet.
    </p>
  </card>
</wml>
```



## Appendix A Examples of WIG WML documents

### Example [A1]

This example illustrates the use of WIG WML elements.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
<card id="START">
  <p>
    <input title="Please enter your first name"
           type="text" name="FIRSTNAME" maxlength="10"/>
    <select title="Do you want to subscribe to The UG journal?">
      <option onpick="#FILLOUTFORM">Yes</option>
      <option>No</option>
    </select>
    <select
      title="Do you want to subscribe to The UG journal for free?">
      <option onpick="#FREE">Yes</option>
      <option>No</option>
    </select>
  </p>
</card>
<card id="FILLOUTFORM">
  <p>
    <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://webserver/page.jsp?f=${FIRSTNAME}&l=${LASTNAME}&p=${
      PWD}&d=${DRINK}"/>
  </p>
</card>
<card id="FREE">
  <p>
    ${FIRSTNAME}, did you really believe that you could get it for free?
  </p>
</card>
</wml>
```



## Example [A2]

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

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <card id="START">
    <p>
      <input title="Please enter your user name" type="text" name="UN"
maxlength="10"/>
      <plugin name="SIGN" params="\x02$(UN)" destvar="SD"/>
      Thank you! This is now passed to the server.
      <go href="http://webserver/page.jsp?u=$(UN) & s=$(SD)"/>
    </p>
  </card>
</wml>
```



## Appendix B Character encoding and conversions made by the UG

The character encoding that is used on the SIM (in Wib) differs from the one that is used on the Content Provider side. When writing WIG WML documents that use the UG for sending data to a Content Provider, it is necessary to understand the conversions made by the UG.

### B.1 Conversion of server-bound messages

When data is passed from Wib to the Content Provider (e.g. in a "go href"), the whole URL, except variables, is first converted from the SMS Default Alphabet to ISO-8859-1.

Variables are then handled in a separate way depending on the encoding of the variable:

- Variable content encoded with the SMS Default Alphabet in Wib is converted to ISO-8859-1 and then URL encoded before it is sent to the Content Provider.
- Variable content encoded with UCS2 in Wib is converted to UTF-8 and then URL encoded before it is sent to the Content Provider.
- Variable content encoded as "Binary" in Wib is not converted, but URL encoded before it is sent to the Content Provider.

The encoding of a variable is set the first time the variable is defined in the WIG WML document, e.g. in a `setvar` element or in a Wib plug-in call.

#### Example [B1]

In this case, the UG does not convert the content of the variable but it is URL encoded. This is the same as if no `class` attribute is used at all. Note that static text in the URL is not URL encoded. The resulting GET message sent to the Content Provider will ask for `"/page.jsp?var1=%23%45&var2=%00A%AE"`.

```
<card>
  <p>
    <setvar name="VAR" value="\x00\x41\xAE" class="Binary"/>
    <go href="http://webserver/page.jsp?var1=%23%45&var2=$(VAR)"/>
  </p>
</card>
```



### Example [B2]

In this case, the content of the variable is interpreted as UCS2 and it is converted to UTF-8 by the UG, i.e. "0x00, 0x31" will be converted to the single byte "0x31" before it is sent to the Content Provider. The resulting GET message sent to the Content Provider will ask for "/page.jsp?var=1".

```
<card>
  <p>
    <setvar name="VAR" value="\x00\x31" class="UCS2"/>
    <go href="http://webserver/page.jsp?VAR=$(VAR)"/>
  </p>
</card>
```

### Example [B3]

In this example, there are two variables involved. The TEXT variable is encoded with SMS-DEFAULT. The ENCRTEXT will be encoded as Binary. Assuming that the ENCR plug-in returns the hexadecimal value '02','74','26','FA','4A','44','05','31','FD', the resulting GET message sent to the Content Provider will ask for "/page.jsp?text=toencrypt&encr=%02t%26%FAJD%051%FD".

```
<card>
  <p>
    <setvar name="TEXT" value="toencrypt" class="SMS-DEFAULT"/>
    <plugin name="ENCR" params="\x01$(TEXT)" destvar="ENCRTEXT"/>
    <go
href="http://webserver/page.jsp?text=$(TEXT) &amp;encr=$(ENCRTEXT)"/>
  </p>
</card>
```



## Appendix C UG Side plug-ins

This Appendix defines the Server Side plug-ins currently available in the UG.

### C.1 sendserverism

#### Description

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

The MSISDN of the mobile station that initiated the UG request will be used as originating address for the SM. If push is used, the MSISDN of the push destination mobile station will be used as originating address.

#### Parameters

Name	Value	Explanation
<b>dcsc</b> optional	SMS-DEFAULT   UCS2   SMS- DEFAULT-FLASH   UCS2-FLASH or integer (0-255)	Data Coding Scheme for the outgoing SM. If DCS indicates "Default Alphabet" according to GSM 03.38 [4], the UG will pack the data. In that case, it does not make sense to set <code>smttextenc</code> to UCS2. The FLASH suffix indicates that the SM is flagged for immediate display on the ME. The attribute values correspond to the following DCS integer values: SMS-DEFAULT - 242 (Store on SIM) UCS2 - 26 (Store on SIM) SMS-DEFAULT-FLASH - 240 or 16 UCS2-FLASH - 24  The following custom DCS values are recommended to use since the SMS is stored on the ME and not on the SIM: 17 - SMS-DEFAULT 25 - UCS2
<b>destaddress</b> mandatory	string	Default value is SMS-DEFAULT. The called party number.
<b>pid</b> optional	integer (0-255)	Protocol identifier. Default value is 0.
<b>smttextenc</b>	SMS-DEFAULT   UCS2	The encoding of the text in the SM. Default value is SMS-DEFAULT.
<b>userdata</b> optional	string	Text in the SM.



### Example [89]

In the example, a text SM with the content "Hello!" is sent to MSISDN "+15185551234". The rest of the document (i.e. the text "SM Sent!") is delivered to Wib.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
                          http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <wigplugin name="sendserverasm">
    <param name="userdata" value="Hello!"/>
    <param name="destaddress" value="+15185551234"/>
  </wigplugin>
  <card>
    <p>
      SM Sent!
    </p>
  </card>
</wml>
```

## C.2 sendserverdatasm

### Description

This plug-in sends binary data in a Short Message directly from the UG, via the Transport Server in the Delivery Platform, to a particular destination.

The MSISDN of the mobile station that initiated the UG request will be used as originating address for the SM. If push is used, the MSISDN of the push destination mobile station will be used as originating address.

The plug-in can be used for sending of Enhanced Message Service (EMS) and Nokia Smart Messaging (NSM) messages from the UG. EMS and NSM are both message formats that allow inclusion of multi-media elements, like pictures and sounds, in a Short Message.

For details regarding the EMS format and the creation of EMS messages, see [2], [13], and [14].

For details regarding the NSM format, see [9].

### Parameters

Name	Value	Explanation
<b>dcs</b> optional	integer (0–255)	Data Coding Scheme for the outgoing SM. If DCS indicates "Default Alphabet" according to GSM 03.38 [4], the UG will pack the data. In that case, it does only make sense to set encoding to SMS-DEFAULT. Default value is 245.





### Example [91]

In this example, an NSM message containing a business card is sent to MSISDN "+15185551234". The rest of the document (i.e. the text "NSM Sent!") is delivered to Wib.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <wigplugin name="sendserverdatasm">
    <param name="udh" value="050400E20000"/>
    <param name="userdata"
value="424547494E3A56434152440D0A56455253494F4E3A322E310D0A4E3A536D697468
3B4D696B650D0A54454C3B505245463A2B35353531323334350D0A454E443A56434152440
D0A"/>
    <param name="destaddress" value="+15185551234" />
    <param name="encoding" value="hex-binary"/>
    <param name="pid" value="0"/>
  </wigplugin>
  <card>
    <p>
      NSM Sent!
    </p>
  </card>
</wml>
```

### C.3 noresponse

#### Description

The `noresponse` element is used when no response from the request should be sent back to Wib. E.g., when a push request generates a response and that response should not generate any byte code to be sent to Wib. If the document includes other elements that normally results in generated byte code, this byte code will NOT be sent to Wib due to this element.

Note that this element should be used with caution in a WIG WML document delivered as a response to a request by Wib, since Wib may be configured to block new requests while waiting for the response. See *UG & WSM Application Developers Guide – Development of Wib Services* [10].

#### Parameters

None.



### Example [92]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <wigplugin name="noresponse"/>
  <card>
    <p>
      This will never reach Wib
    </p>
  </card>
</wml>
```

## C.4 applelevelcharginginfo

### Description

This plug-in is used to attach arbitrary charging information to the WIG WML document. The charging information will be added verbatim to the charging/event-log records that the UG generates for each WIG WML document it handles.

### Parameters

Name	Value	Explanation
<b>info</b>	string	The charging information that should appear in the charging/event-log record.

### Example [93]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<wml xmlns="http://www.smarttrust.com/WIG-WML/5.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.smarttrust.com/WIG-WML/5.0
      http://www.smarttrust.com/xsd/wigwml-5.0.xsd">
  <wigplugin name="applelevelcharginginfo">
    <param name="info" value="transact-id=1234567-432-1"/>
  </wigplugin>
  <card>
    <p>
      Transaction succeeded
    </p>
  </card>
</wml>
```