S@T 01.10 v3.0.0 (Release 2006)

S@T Markup Language

S@TML

Published by simalliance now Trusted Connectivity Alliance

1 TABLE OF CONTENT

2 TERMINOLOGY

2.1 Notation

Lexical and syntactical specifications are given in EBNF (extended Backus Naur Form), with literals enclosed in single quotes `'xyz'` or given in a single character set like `[0-9]` for a digit, and using the operators `(...)` (precedence), `?` (optional), `*` (zero or more times), `+` (one or more times), `|` (alternative), and "... `::=` ... `.`" for rules. They are written in `typewriter` font, and will be used to explain the structure concisely without mentioning XML attributes.

The format of XML document type definitions (DTDs) is explained in /XML1.0/, they are written in `typewriter` font.

`typewriter` font is used for S@TML language examples.

2.2 Abbreviations

DE	S@TML/SBC Decoder / Encoder
DTD	Document Type Definition
GSM	Global System for Mobile Communication
HTTP	Hyper Text Transfer Protocol
S@T	SIM Alliance Toolbox
SB	S@T Browser
SBC	S@T Byte Code
SSP	S@T Session Protocol
SIM	Subscriber Identity Module
S@TML	S@T Markup Language
STK	SIM Application Toolkit
TLV	Tag Length Value encoding
URI	Unified Resource Identifier
URL	Unified Resource Locator
WAP	Wireless Application Protocol
WTAI	Wireless Telephony Application Interface
XML	Extensible Markup Language

3 LIST OF DOCUMENTS

/GSM02.30/ Digital cellular telecommunications system (Phase 2+); Man-Machine Interface (MMI) of the Mobile Station (MS) (GSM 02.30)

/GSM02.90/ Digital cellular telecommunications system (Phase 2+); Unstructured Supplementary Service Data (USSD) - Stage 1 (GSM 02.90)

/GSM03.38/ Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information (GSM 03.38)

/GSM03.48/ Digital cellular telecommunications system (Phase 2+); Security mechanisms for the SIM application toolkit; Stage 2 (GSM 03.48)

/GSM03.40/ Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS) (GSM 03.40)

/GSM04.07/ Digital cellular telecommunications system (Phase 2+); Mobile radio interface signalling layer 3; General aspects (GSM 04.07)

/GSM04.08/ Digital cellular telecommunications system (Phase 2+); Mobile radio interface; Layer 3 specification (GSM 04.08)

/GSM11.11/ Digital cellular telecommunications system (Phase 2+); Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.11)

/GSM11.14/ Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.14)

/RFC2396/ Uniform Resource Identifiers (URI): Generic Syntax

/SBC/ SBC, S@T Byte Code (Technical Specification S@T 01.00)

/WML1.1/ Wireless Application Protocol - Wireless Markup Language (WML) 1.1

/WML1.2Prop/ Wireless Application Protocol - Wireless Markup Language Specification - Proposed Version 1.2

/WTAI1.1/ Wireless Application Protocol - Wireless Telephony Application Interface Specification (WTAI) 1.1

/WTAIGSM1.1/ Wireless Application Protocol - Wireless Telephony Application Interface Specification - GSM Specific Addendum (WTAI GSM) 1.1

/XML1.0/ Extensible Markup Language (XML) 1.0

4 OVERVIEW

This is the released version of the S@T Markup Language 1.0 (S@TML) specification. It describes the format of S@TML „content“ stored on a HTTP server. The S@TML/SBC Decoder / Encoder (DE) on the S@T Gateway is a dynamic compiler translating S@TML content into S@T Byte Code (SBC) which can be displayed by the S@T Browser (SB) on SIM cards inside mobile phones supporting GSM Phase 2+ SIM Toolkit operations.

In the S@TML there is a *S@TML Core Language* which is a subset of the Wireless Application Protocol's (WAP's) Wireless Markup Language (WML) (see /WML1.1/).This eases the definition of services that are usable by mobile users with S@T Browser on their SIM card or with a handset with WAP functionality. A subset of the Wireless Telephony Application Interface (WTAI) for GSM (see /WTAI1.1/, /WTAIGSM1.1/) is also part of the S@TML Core Language. The *S@TML STK Extensions* allow for the migration of existing SIM Toolkit (STK) based services to S@TML, as well as for more sophisticated services with special requirements (e.g. additional security). They are tagged using the "`sat-`" prefix. In the *Future S@TML* section those WML features are described which in the S@TML Document Type Definition, but outside S@TML Core Language and beyond this specification's version (and may still be subject to change). For the relation between the mentioned language subsets see Figure 1.

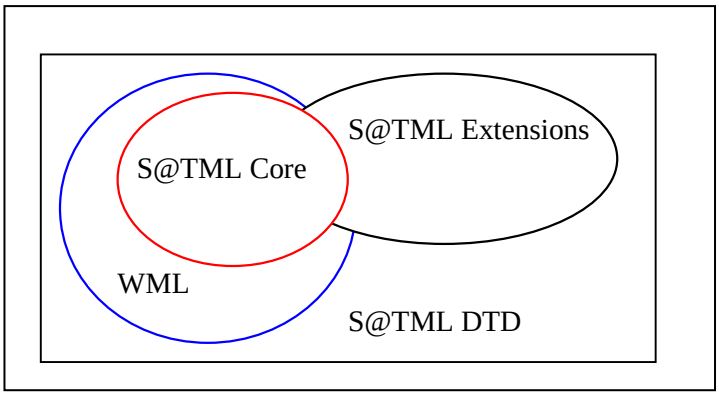

Figure 1

The WML elements and attributes being part of the S@TML Core Language must be translated into SBC byte code by the DE, just as the S@TML STK Extensions. WML elements and attributes which are not supported may either be ignored or an error may be indicated. The expected reaction of the DE is stated in the specification. Those elements or attributes which can yield an error are not recommended to be used in content meant for S@T browsers. It is allowed that a DE implementation translates elements or attributes into SBC which are classified as being ignored. This is, however, out of the scope of the specification, and may not interfere with the translation of supported elements and attributes.

Note, that Chapter 6 specifies the S@TML Core Language elements and attributes, as well as attributes belonging to the S@TML STK Extensions. The extensions are given in **`bold font`** and tagged with "`sat-`". Chapter 7 defines the subset of the Wireless Telephony Application Interface features and notations (/WTAI1.1/, /WTAIGSM1.1/) which are also part of the S@TML Core Language. Chapter 8 defines the S@TML STK Extensions, and Chapter 9 deals with Future S@TML which is not supported in this version and still subject to change.

The S@TML lexical and syntactical conventions follow the Extensible Markup Language (XML) 1.0 recommendations /XML1.0/. A complete document type definition (DTD) is given in Chapter 10. This document type definition is a superset of the WML 1.1 DTD. I.e. it includes S@TML Core Language, S@TML STK Extensions, and the not supported part of WML.

5 LEXICAL DEFINITIONS

The following lexical entities are used in S@TML (see also /XML1.0/ and /WML1.1/):

5.1.1 Named Character Entities

As defined in /XML1.0/ the following named character entities can be used in SATML:

```
<!ENTITY quot “&#34;”      <!-- quotation mark -->
<!ENTITY amp  “&#38;#38;”  <!-- ampersand -->
<!ENTITY apos “&#39;”      <!-- apostrophe -->
<!ENTITY lt   “&#38;#60;”  <!-- less than -->
<!ENTITY gt   “&#62;”      <!-- greater than -->
```

For compatibility with /WML1.1/ also the following named character entities are defined in SATML:

```
<!ENTITY nbsp “ ”<!-- non-breaking space -->
<!ENTITY shy  “&#173;”<!-- soft hyphen (discretionary hyphen) -->
```

5.1.2 Numeric Character Entities

As defined in /XML1.0/ unicode characters can be specified by giving their numeric value either in decimal (`&#`*num*`;`) or in hexadecimal (`&#x`*hexnum*`;`) notation.

5.1.3 CDATA

`CDATA ::= (`any text character or numeric or named character entity`)*`

5.1.4 PCDATA

`PCDATA ::= (`any text character or numeric or named character entity and element tags`)*`

5.1.5 NMTOKEN

`NMTOKEN ::= (`any name character`)+`

5.1.6 Basic Attribute Types

The following entities are used as basic attribute type definitions for SATML:

```
<!ENTITY % bool-t     “(true|false)”>
<!ENTITY % num-t      “NMTOKEN”>  <!-- number  -->
<!ENTITY % hex-t      “NMTOKEN”>  <!-- hex str  -->
```

Attributes of type `%num-t` may contain a decimal number (i.e. `[0-9]+`); those of type `%hex-t` may contain an even number of hexadecimal digits (i.e. an even-length string matching `[A-Fa-f0-9]+`) after having all white space characters removed. In the attributes declared with the following entities variables may be used as well as literal values (see Section 6.4 for the notation):

```
<!ENTITY % vdata-t “CDATA”>       <!-- string with variables -->
<!ENTITY % vnum-t  “%vdata-t;”>   <!-- number with variables -->
<!ENTITY % vhex-t  “%vdata-t;”>   <!-- hex str with variables -->
<!ENTITY % HREF-t  “%vdata-t;”>   <!-- URI, URL or URN with variables -->
```

In attributes of type %HREF-t hyperlink references will be given (see Section 8.6.1 for the notation).

6 S@TML CORE LANGUAGE

The S@TML Core Language elements and attributes form a subset of the Wireless Application Protocol’s Wireless Markup Language WML Version 1.1 (/WML1.1/). This eases the definition of services that are usable by mobile users with S@T Browser on their SIM card as well as those with a handset with WAP functionality. For information on future development only, see also the Proposed Version 1.2 document /WML1.2Prop/. The subset of Wireless

Telephony Application Interface features and notations (/WTAI1.1/, /WTAIGSM1.1/) described in Chapter 7 are also part of the S@TML Core Language.

Note, that attributes etc. given in **`bold font`** and tagged with "`sat-`" belong to the S@TML STK Extensions (Chapter 8), and are given in this S@TML Core Language Chapter 6 only in order to eliminate the need of duplicating text. The security rules given in Section 8.1 apply to these attributes equally as to those defined in Chapter 8.

6.1 Decks

6.1.1 deck

```
deck ::= prolog (satml|wml)
```

A `deck` defines a set of cards logically belonging to each other and will usually be the smallest unit of data transported to the SIM at one time.

6.1.2 %id-attrs

```
<!ENTITY % id-attrs
 “id        ID        #IMPLIED
  class     CDATA     #IMPLIED“>
```

attribute	type	default	%id-attrs
`id`	string	null	Provides a unique identification of an element in a deck. For `<card>` this defines the relative URI address. For other elements it is ignored by DE. May be used for tasks like server-side transformations and for compatibility with WML.
`class`	string	null	Provides a means to organise elements in classes by giving a space-separated list of case sensitive class names. To be ignored by DE.

6.1.3 prolog

```
prolog ::= xml-decl?
   ( ’<!DOCTYPE satml SYSTEM “http://www.simalliance.org/DTD/satml106.dtd“>’
   | ’<!DOCTYPE wml   SYSTEM “http://www.simalliance.org/DTD/satml106.dtd“>’)
```

The prolog contains the XML version and DOCTYPE information. The choice of the DOCTYPE declaration for satml or wml must fit to the root element being used. If the document encoding is neither UTF-8, nor UTF-16, an XML encoding declaration must be given at the beginning as specified in /XML1.0/. Some examples are:

```
xml-decl ::= ’<?xml version=“1.0“ encoding=“ISO-8859-1“ ?>’
           | ’<?xml version=“1.0“ encoding=“UTF-8“ ?>’
           | ’<?xml version=“1.0“ encoding=“EUC-JP“ ?>’
```

If the encoding is UTF-16 the document must start with the byte order mark.

6.1.4 satml

```
satml ::= ’<satml>’ head? %decls card+ ’</satml>’

<!ELEMENT satml (head?, %decls;, card+) >
<!ATTLIST satml
  xml:lang       NMTOKEN              #IMPLIED
  sat-storage    (dynamic|static)     “static“
  sat-serv-id    %hex-t;              #IMPLIED
  sat-help       %vdata-t;            #IMPLIED
  sat-dcs        (sms|ucs2|auto)      “auto“
```

```
  %id-attrs;
>
```

The satml element contains the complete deck of cards, declarations, and deck-level attributes.

attribute	type	default	<satml>
xml:lang	string	null	Specifies the natural language in which the document is written. Ignored by DE.
sat-storage	enum	static	Specifies the storage type of this deck. The content is either dynamic (should not be re-read from cache), or static.
sat-serv-id	hex	null	Specifies the Toolkit Application Reference for the service to which this deck belongs in 1-8 Bytes given in a hexadecimal string. Note, that this is an optional feature of SBC, and that the values have to be defined by the card issuer.
sat-help	string	null	Specifies a help text to be displayed by the SB when the help key is pressed.
sat-dcs	enum	auto	Specifies the default DCS of the deck. If 'auto' then usual content analysis procedures apply.

6.1.5 wml

```
wml ::= '<wml>' head? template? card+ '</wml>'

<!ELEMENT wml (head?, template?, card+) >
<!ATTLIST wml
  xml:lang       NMTOKEN                 #IMPLIED
  sat-storage    (dynamic|static)        "static"
  sat-serv-id    %hex-t;                 #IMPLIED
  sat-help       %vdata-t;               #IMPLIED
  sat-dcs        (sms|ucs2|auto)         "auto"
  %id-attrs;
>
```

The wml element can be used to enclose a complete deck of cards, declarations, and deck-level attributes. Note, that if the additional non-WML attributes or the S@TML STK Extensions are used, and not the S@TML DTD but the original WML DTD, a *validating* parser for WML will normally produce errors (compare /XML1.0/).

attribute	type	default	<wml>
xml:lang	string	null	Specifies the natural language in which the document is written. Ignored by DE.
sat-storage	enum	static	Specifies the storage type of this deck. The content is either dynamic (should not be re-read from cache), or static.
sat-serv-id	hex	null	Specifies the Toolkit Application Reference for the service to which this deck belongs in 1-8 Bytes given in a hexadecimal string. Note, that this is an optional feature of SBC, and that the values have to be defined by the card issuer.
sat-help	string	null	Specifies a help text to be displayed by the SB when the user requests help.
sat-dcs	enum	auto	Specifies the default DCS of the deck. If 'auto' then usual content analysis procedures apply.

6.1.6 head

```
head ::= '<head>' (access|meta)+ '</head>'

<!ELEMENT head (access|meta)+ >
<!ATTLIST head
  %id-attrs;
>
```

The head element contains deck-level administrative information.

6.1.7 access

```
access ::= '<access/>'

<!ELEMENT access EMPTY>
<!ATTLIST access
  domain      CDATA       #IMPLIED
  path        CDATA       #IMPLIED
  %id-attrs;
>
```

The access element specifies access control information. There may be at most one access element. If it is missing access control is disabled and any deck can access this deck. Else it depends on the DE gateway implementation whether an access check will be performed. A check may also be performed on the content server.

attribute	type	default	<access>
domain	string	null	Each suffix sub-domain element of the URI of the accessing deck must exactly match the given domain.
path	string	null	Each prefix path element of the URI of the accessing deck must exactly match the given path.

6.1.8 meta data

```
meta ::= '<meta/>'

<!ELEMENT meta EMPTY>
<!ATTLIST meta
  http-equiv  CDATA       #IMPLIED
  name        CDATA       #IMPLIED
  forua       %bool-t;    "false"
  content     CDATA       #IMPLIED
  scheme      CDATA       #IMPLIED
  %id-attrs;
>
```

The meta element contains generic meta-information about the S@TML deck given by property names and values.

attribute	type	default	<meta>
http-equiv	string	null	For compatibility with WML, ignored by DE.
name	string	null	For compatibility with WML, ignored by DE.
forua	bool	false	For compatibility with WML, ignored by DE.
content	string	null	For compatibility with WML, ignored by DE.
scheme	string	null	For compatibility with WML, ignored by DE.

6.2 Cards

6.2.1 %cardev-attrs

```
<!ENTITY % cardev-attrs
 "onenterforward     %HREF-t;        #IMPLIED
  onenterbackward    %HREF-t;        #IMPLIED
  ontimer            %HREF-t;        #IMPLIED"
>
```

The card event attributes define navigational tasks to be performed when the current card is entered via a go (onenterforward) resp. a prev (onenterbackward), or when a timer has expired (ontimer). In WML they can be used as attributes for the elements <template> and <card>.

attribute	type	default	%cardev-attrs

onenterforward	string URI	null	WML attribute, results in an error or will be ignored if found by DE.
onenterbackward	string URI	null	WML attribute, results in an error or will be ignored if found by DE.
ontimer	string URI	null	WML attribute, results in an error or will be ignored if found by DE.

6.2.2 card

```
card ::= '<card>' onevent* timer? (do|p|setvar|go|prev|refresh|%satstmt)*
'</card>'

<!ELEMENT card (onevent*, timer?, (do|p|setvar|go|prev|refresh|%satstmt;)*) >
<!ATTLIST card
  title            %vdata-t;      #IMPLIED
  newcontext       %bool-t;       "false"
  ordered          %bool-t;       "true"
  xml:lang         NMTOKEN        #IMPLIED
  sat-help         %vdata-t;      #IMPLIED
  sat-history      %bool-t;       "true"
  sat-chain-next   %bool-t;       "false"
  %cardev-attrs;
  %id-attrs;
>
```

The `card` element contains information to be displayed to the user by the SB.

attribute	type	default	<card>
title	string	null	For compatibility with WML, to be ignored by DE.
newcontext	bool	false	The browser context is re-set when this card is entered.
ordered	bool	true	For compatibility with WML, to be ignored by DE.
sat-help	string	null	Specifies a help text to be displayed by the SB when the user requests help. Overrides the `sat-help` attribute in an enclosing deck element.
sat-history	bool	true	Specifies that the card's address will be added to the history.
sat-chain-next	bool	false	Specifies the action to take when the browser reaches the end of this card. If this attribute is set to true, the browser starts executing the next card in the deck. Otherwise it finishes processing this deck.
xml:lang	string	null	Specifies the natural language in which the card content is written. Ignored by DE.

6.3 Declarations

6.3.1 %decls

```
%decls ::= (var|const|sps)* template?

<!ENTITY % decls "(sat-var|sat-const|sat-sps)*, template?">
```

For S@TML with STK Extensions not only a `template` element but also a list of temporary variable declarations, constant declarations, and service specific permanent variable references can be given for a deck of cards. Variables not declared will be assumed to be temporary variables.

6.3.2 template

```
template ::= '<template>' (do|onevent)* '</template>'

<!ELEMENT template (do|onevent)* >
```

```
<!ATTLIST template
  %cardev-attrs;
  %id-attrs;
>
```

The `template` element contains navigation event specifications which will be inherited to any card inside the current deck (unless overwritten by a card-level navigation event specification).

6.3.3 timer

```
timer ::= '<timer/>'

<!ELEMENT timer EMPTY>
<!ATTLIST timer
  name      NMTOKEN       #IMPLIED
  value     %vdata-t;     #REQUIRED
  %id-attrs;
>
```

The `timer` element is used in WML to declare a card timer. Its use results in an error or is ignored by the DE in this version of S@TML.

attribute	type	default	<timer>
`name`	string	null	Name of the variable to be set to the timer value. WML attribute, results in an error or is ignored if found by DE.
`value`	int		Default value of the timeout in 1/10 seconds. WML attribute, results in an error or is ignored if found by DE.

6.4 Variables

6.4.1 Overview

In S@TML variables can be used in attribute values identified by `vdata-t` and `PCDATA` content. They can not be used in attributes of other types like `CDATA`, `ID`, or `NMTOKEN`. To substitute a variable into a card or deck the following syntax is used:

$identifier
$(identifier)
$(identifier:conversion)

Parentheses are required if white space does not indicate the end of a variable. Just as in WML variable syntax has highest priority, i.e., anywhere the variable syntax is legal, an unescaped '$' character indicates a variable substitution. A sequence of two dollar signs '$$' represents a single dollar sign character in all `CDATA` attribute values and in `PCDATA` text (see also /WML1.1/ Section 10.3 "Variables"). In this version of S@TML the conversion part inside the variable syntax will be ignored by DE.

6.4.2 Lifetime

The lifetime of temporary variables is managed by the `newcontext` attribute of the `card` element. By default temporary variables are accessible in the current and other decks belonging to the same context.

SAT browsers may provide users means to reference and navigate to resources independent of the current content. For example, SAT browsers may provide bookmarks and a URL input dialog. Whenever a user navigates to a resource that was not the result of an interaction with the content in the current context, the SAT browser must establish another context for that navigation. The SAT browser may terminate the current context before establishing another one for the new navigation attempt.

6.5 Fields

6.5.1 %fields, %flow, %text, %layout, %emph, %TAlign, %WrapMode

```
%fields ::= %flow | input | select | fieldset | %satfld
%flow   ::= %txt | %layout | img | anchor | a | table
%txt    ::= #PCDATA | %emph
%layout ::= br
%emph   ::= em | strong | b | i | u | big | small

<!ENTITY % emph      "em | strong | b | i | u | big | small">
<!ENTITY % layout    "br">
<!ENTITY % txt       "#PCDATA | %emph;">
<!ENTITY % flow      "%txt; | %layout; | img | anchor | a | table">
<!ENTITY % fields    "%flow; | input | select | fieldset | %satfld; ">

<!ENTITY % TAlign    "(left|right|center)">
<!ENTITY % WrapMode  "(wrap|nowrap)">
```

6.5.2 p – paragraph

```
paragraph ::= '<p>' (%fields|do)* '</p>'

<!ELEMENT p (%fields; | do)*>
<!ATTLIST p
  align          %TAlign;           "left"
  mode           %WrapMode;         #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  sat-auto-clr   %bool-t;           "false"
  sat-prio       (normal|high)      "normal"
  %id-attrs;
>
```

The paragraph element p encloses text or fields to be displayed by the SB.

attribute	type	default	<p>
align	enum	left	For compatibility with WML, to be ignored by DE.
mode	enum	wrap	For compatibility with WML, to be ignored by DE.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.
sat-auto-clr	bool	false	If set to true text strings in this paragraph will be cleared automatically after a delay.
sat-prio	enum	normal	Specifies the priority to be used to display text strings in this paragraph.

6.5.3 input

```
input ::= '<input/>'

<!ELEMENT input EMPTY>
<!ATTLIST input
  title          %vdata-t;          #IMPLIED
  name           NMTOKEN            #REQUIRED
  type           (text|password)    "text"
  value          %vdata-t;          #IMPLIED
  format         CDATA              #IMPLIED
  emptyok        %bool-t;           "false"
  size           %num-t;            #IMPLIED
  maxlength      %num-t;            #IMPLIED
  sat-minlength  %num-t;            #IMPLIED
  sat-help       %vdata-t;          #IMPLIED
  tabindex       %num-t;            #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
```

>

The input element specifies an input field to be displayed by the SB.

attribute	type	default	<input>
title	string	null	Title to be displayed while the command is executed. If not present DE may use part of the preceding text up to the next tag (except the ignored %emph tags).
name	string		Specifies the variable to be assigned. It is an error to use read-only variables here.
type	enum	text	Specifies whether the input is visible while entered or not. If type equals "password", only characters '0'-'9','*','#' can be entered.
value	string	null	Specifies a default value to be displayed when the user is asked for input.
format	string	*M	Specifies the set of characters accepted as input: *M or *n*M = any character *N or *n*N = digit characters ('0'-'9', '+', '*', '#') If *n*M or *n*N is given minimal and maximal length of input will be set to *n*, and the attributes emptyok, maxlength, sat-minlength are ignored. Other formats that can be defined in WML will be interpreted just like *M. Note, that in WML the digit characters do not include '+', '*', and '#'.
emptyok	bool	false	If set to true the minimal length of input to be accepted is 0 else 1.
size	num		For compatibility with WML, to be ignored by DE.
maxlength	num		Specifies the maximal number of input characters accepted. Default is unlimited resp. user-agent-dependant.
sat-minlength	num		Specifies the minimal length of input to be accepted. If given the attribute emptyok is ignored.
sat-help	string	null	Specifies a help string which will be displayed when the user requests help for this input element. Overrides the sat-help attribute in an enclosing deck or card.
tabindex	num	0	For compatibility with WML, to be ignored by DE.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.5.4 select

```
select ::= '<select>' (optgroup|option)+ '</select>'

<!ELEMENT select (optgroup|option)+>
<!ATTLIST select
  title          %vdata-t;          #IMPLIED
  name           NMTOKEN            #IMPLIED
  value          %vdata-t;          #IMPLIED
  iname          NMTOKEN            #IMPLIED
  ivalue         %vdata-t;          #IMPLIED
  multiple       %bool-t;           "false"
  sat-help       %vdata-t;          #IMPLIED
  tabindex       %num-t;            #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>
```

The select element will be displayed as a field of options by the SB among which the user may choose. In this version of S@TML a select may either only contain navigation operations or only operations to set variable values. More complicated select elements are only optionally supported. Attributes for indication of defaults are ignored by DE.

attribute	type	default	<select>
title	string	null	Title to be displayed while the command is executed. If not present DE may use part of the preceding text up to the next tag (except the ignored %emph tags).
name	string	null	Specifies the variable to be assigned the selected option's value. It is an error to use read-only variables here.
value	string	null	Specifies default value to be assigned if nothing is selected. For compatibility with WML, ignored by DE.
iname	string	null	Specifies the variable to be assigned the selected option's index (starting with string "1" for the first index). Index numbers will be converted to strings before the assigment, represented in decimal system. Index numbering increases monotonically. It is an error to use read-only variables here.
ivalue	string	null	Specifies default item number to be chosen if nothing is selected. For compatibility with WML, ignored by DE.
multiple	bool	false	WML attribute not supported in this version of S@TML, the select element may be ignored or result in an error if it is found by DE.
sat-help	string	null	Specifies a help string which will be displayed when the user requests help for this select element. Overrides the sat-help attribute in an enclosing deck or card.
tabindex	string	null	For compatibility with WML, to be ignored by DE.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.5.5 option

```
option ::= '<option>' (#PCDATA|onevent)* '</option>'

<!ELEMENT option (#PCDATA|onevent)*>
<!ATTLIST option
  title          %vdata-t;          #IMPLIED
  value          %vdata-t;          #IMPLIED
  onpick         %HREF-t;           #IMPLIED
  sat-help       %vdata-t;          #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>
```

The option element contains one option field of a select or optgroup element.

attribute	type	default	<option>
title	string	null	Can be used to specify the option's title instead of doing so in the element's body. If both this attribute and the body are specified, then the body text will be used.
value	string	null	Value to be assigned to the variable given by the name attribute of the enclosing select.
onpick	string URI	null	Perform a go operation to this URI address when this option has been selected.
sat-help	string	null	Specifies a help string which will be displayed when the user requests help for this select option. Overrides the sat-help attribute in an enclosing deck, card or select.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.5.6 optgroup

```
optgroup ::= '<optgroup>' (optgroup|option)+ '</optgroup>'

<!ELEMENT optgroup (optgroup|option)+>
<!ATTLIST optgroup
```

```
  title          %vdata-t;          #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>
```

The `optgroup` element can be used to indicate a preferred grouping of selection options to the DE.

attribute	type	default	<optgroup>
`title`	string	null	Title which may be used in the presentation of this group of options by DE.
`xml:lang`	string	null	For compatibility with WML, to be ignored by DE.

6.5.7 fieldset

```
fieldset ::= '<fieldset>' (%fields|do)* '</fieldset>'

<!ELEMENT fieldset (%fields;|do)*>
<!ATTLIST fieldset
  title          %vdata-t;          #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  sat-auto-clr   %bool-t;           "false"
  sat-prio       (normal|high)      "normal"
  %id-attrs;
>
```

The `fieldset` element can be used to indicate a preferred grouping of card fields to the DE.

attribute	type	default	<fieldset>
`title`	string	null	For compatibility with WML, to be ignored by DE.
`xml:lang`	string	null	For compatibility with WML, to be ignored by DE.
`sat-auto-clr`	bool	`false`	If set to `true` text strings in this field set will be cleared automatically after a delay. This hides an `sat-auto-clr` attribute set in an enclosing paragraph tag `<p>`.
`sat-prio`	enum	`normal`	Specifies the priority to be used to display text strings in this field set. This hides an `sat-prio` attribute set in an enclosing paragraph tag `<p>`.

6.6 Content

6.6.1 br - break line

```
br ::= '<br/>'

<!ELEMENT br EMPTY>
<!ATTLIST br
  %id-attrs;
>
```

The `br` element indicates a line break to be displayed by the SB.

6.6.2 em - emphasis

```
em ::= '<em>' %flow* '</em>'

<!ELEMENT em (%flow;)*>
<!ATTLIST em
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like `<em>` for emphasis are ignored by DE.

attribute	type	default	
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.3 strong

```
strong ::= '<strong>' %flow* '</strong>'

<!ELEMENT strong (%flow;)*>
<!ATTLIST strong
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like for strong emphasis are ignored by DE.

attribute	type	default	
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.4 i - italics

```
i ::= '<i>' %flow* '</i>'

<!ELEMENT i (%flow;)*>
<!ATTLIST i
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like <i> for italics are ignored by DE.

attribute	type	default	<i>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.5 b - bold

```
b ::= '<b>' %flow* '</b>'

<!ELEMENT b (%flow;)*>
<!ATTLIST b
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like <b> for bold are ignored by DE.

attribute	type	default	<b>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.6 u - underline

```
u ::= '<u>' %flow* '</u>'

<!ELEMENT u (%flow;)*>
<!ATTLIST u
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like <u> for underline are ignored by DE.

attribute	type	default	<u>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.7 big

```
big ::= '<big>' %flow* '</big>'

<!ELEMENT big (%flow;)*>
<!ATTLIST big
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like `<big>` for big font are ignored by DE.

attribute	type	default	<big>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.6.8 small

```
small ::= '<small>' %flow* '</small>'

<!ELEMENT small (%flow;)*>
<!ATTLIST small
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

Text formatting tags like `<small>` for small font are ignored by DE.

attribute	type	default	<small>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.7 Navigation

6.7.1 %task

```
%task ::= go | prev | noop | refresh | sat-switch

<!ENTITY % task "go | prev | noop | refresh | sat-switch ">
```

6.7.2 anchor

```
anchor ::= '<anchor>'(#PCDATA|br|img|go|prev|refresh)* '</anchor>'

<!ELEMENT anchor (#PCDATA|br|img|go|prev|refresh)*>
<!ATTLIST anchor
  title       %vdata-t;        #IMPLIED
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>
```

The `anchor` element specifies a hyper-link to be displayed by the SB.

attribute	type	default	<anchor>
title	string	null	Title which may be used in the presentation of this hyper-link by DE.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

6.7.3 a – abbreviated anchor

```
a ::= '<a>' (#PCDATA|br|img)* '</a>'

<!ELEMENT a (#PCDATA|br|img)*>
<!ATTLIST a
  href        %HREF-t;         #REQUIRED
```

```
  title       %vdata-t;       #IMPLIED
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>
```

The abbreviated anchor element `a` specifies a hyper-link to be displayed by the SB. E.g. a sequence like "`<anchor>link-text<go href="dest"/></anchor>`" can be abbreviated by the following: "`<a href="dest">link-text</a>`".

attribute	type	default	<a>
`href`	string URI		Specifies the destination URI address of the `go` operation to be performed when the link has been selected.
`title`	string	null	Title which may be used in the presentation of this hyper-link by DE.
`xml:lang`	string	null	For compatibility with WML, to be ignored by DE.

6.7.4 do

```
do ::= '<do>' %task '</do>'

<!ELEMENT do (%task;)>
<!ATTLIST do
  type        CDATA           #REQUIRED
  label       %vdata-t;       #IMPLIED
  name        NMTOKEN         #IMPLIED
  optional    %bool-t;        "false"
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>
```

Inside a `do` element tasks are specified which can be activated by the user and normally will be shown using the contextual menu implemented by the SB.

attribute	type	default	<do>
`type`	string		This may be used by the DE to determine the category for this menu item. The following values are supported: • `accept` - positive acknowledgement • `prev` - backward history navigation • `help` - request for help • `reset` - clear or reset state or stop browser • `vnd.sat-process` - process the following task without user interaction. It is recommended to use this only for SATML, but not WML services, because a WML user agent may treat this type just as `unknown`. • `unknown` - generic, equal to empty string
`label`	string	null	This may be used by the DE as a label for this menu item.
`name`	string	null	Specifies the name of this task. If missing `type` is used as name. A `do` in a card will shadow a `do` in a deck if they have the same name.
`optional`	bool	`false`	If set to `true` this task may be ignored by DE.
`xml:lang`	string	null	For compatibility with WML, to be ignored by DE.

6.7.5 onevent

```
onevent ::= '<onevent>' %task '</onevent>'

<!ELEMENT onevent (%task;)>
<!ATTLIST onevent
  type        CDATA           #REQUIRED
  %id-attrs;
>
```

The onevent element specifies what is to be done for certain types of events. For type="onpick" the onevent tag can only be present inside an option element; if it is found outside of option in a card or template tag, an error is generated by the DE.

attribute	type	default	<onevent>
type	string		Specifies the name of the intrinsic event. Must be set to onpick for this version of S@TML. If type is ontimer, onenterforward or onenterbackward the DE will produce an error or ignore the onevent element.

6.7.6 go

```
go ::= '<go>' (postfield|setvar)* '</go>'

<!ELEMENT go (postfield|setvar)*>
<!ATTLIST go
  href             %HREF-t;            #REQUIRED
  no-wait          (no|yes|continue)   "no"
  sendreferer      %bool-t;            "false"
  method           (post|get)          "get"
  accept-charset   CDATA               #IMPLIED
  %id-attrs;
>
```

The go element specifies to which URI the SB should branch when the enclosing task has been selected.

attribute	type	default	<go>
href	string URI		Specifies the destination URI address of the go operation to be performed when the link has been selected.
no-wait	string	no	no: the SB waits for a response from the DE yes: the SB exits after this macro continue: the SB calls the next macro
sendreferer	bool	false	Specifies if the referring URI address will be sent to the server, e.g. for checking of access rights.
method	enum	get	Specifies the method to be used for fetching the contents.
accept-charset	string	null	For compatibility with WML, to be ignored by DE.

6.7.7 prev

```
prev ::= '<prev>' setvar* '</prev>'

<!ELEMENT prev (setvar)*>
<!ATTLIST prev
  %id-attrs;
>
```

The prev element specifies a how a backward navigation should be performed by the SB.

6.7.8 refresh

```
refresh ::= '<refresh>' setvar* '</refresh>'

<!ELEMENT refresh (setvar)*>
<!ATTLIST refresh
  %id-attrs;
>
```

When `refresh` is executed the current card is re-displayed.

6.7.9 noop

```
noop ::= '<noop/>'

<!ELEMENT noop EMPTY>
<!ATTLIST noop
  %id-attrs;
>
```

The `noop` element allows to specifies that in this task nothing has to be done.

6.8 Statements

6.8.1 setvar

```
setvar ::= '<setvar/>'

<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
  name              NMTOKEN       #REQUIRED
  value             %vdata-t;     #REQUIRED
  %id-attrs;
>
```

When the `setvar` statement is executed the SB sets a variable to the given value.

attribute	type	default	<setvar>
`name`	string		Specifies the name of the variable to be set. It is an error to use a read-only variable here. Note, that indirect references, i.e. giving the name using another variable, are not allowed (they are in WML).
`value`	string		Specifies the value to be assigned to this variable.

6.8.2 postfield

```
postfield ::= '<postfield/>

<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
  name              NMTOKEN;      #REQUIRED
  value             %vdata-t;     #REQUIRED
  %id-attrs;
>
```

The `postfield` element specifies values which will be posted to the origin server.

attribute	type	default	<postfield>
`name`	string		Specifies the field name to be sent to the server for an HTTP request. Note, that indirect references, i.e. giving the name using a variable, are not allowed (they are in WML).
`value`	string		Specifies the value to be sent to the server in this field.

7 S@TML TELEPHONY

The use of WTAI is not yet well supported by WAP handset providers and the execution model of WTAI URI notation seems not yet to be fully specified. For the time being it is recommended to use the alternative implementation in the S@TML STK Extensions chapter.

In the S@TML Core Language the following features from the Wireless Telephony Application Interface can be used in the URI notation (see also /WTAI1.1/, /WTAIGSM1.1/). The specially formatted WTAI URIs can be used for in the `href` attributes of the `go` or `option` elements. Eventually parameter values or a result variable name will be added:
`wtai://`<library>`/`<function> (`;`<parameter>)* [`!`<result>]
The WTAI function result will be assigned to the given variable after execution of the function.

7.1 Security

The DE will check the URL from which the current deck has been received to decide whether this is a trusted or not-trusted source. While WTAI functions of the public library can be used by any S@TML deck, the WTAI functions from other libraries can only be used when the deck is a trusted one. It is up to the DE implementation to provide further refinements of this security scheme, e.g. configurable levels of trust for the non-public libraries or functions.

7.2 Telephony

7.2.1 set-up call (voice control library)

```
setup-call ::= 'wtai://vc/sc' ';' telnumber ';' telmode ['!' result ]
telnumber  ::= CDATA
telmode    ::= num-t
result     ::= NMTOKEN
```

Set-up a mobile originated voice call to the specified number. The `mode` parameter indicates how the call should be handled if the context in the WTA user-agent terminates. There are two modes, “drop” and “keep”. “Drop” means that the OS will release the call if the context should be restarted. “Keep” makes it possible to maintain the call even after the current context has terminated.

parameter	type	default	wtai://vc/sc (set-up call)
`telnumber`	string		Destination number to call. All valid telephony number digits may be used (‘`0`’-‘`9`’), as well as ‘`+`’ for the international number prefix.
`telmode`	num		0=drop call when context is removed 1=keep call after context is removed For compatibility with WML, to be ignored by DE.
`result`	string		Name of variable to which a string is assigned that returns the identity of the created call or a negative WTAI error code in case of failure. For compatibility with WML, to be ignored by DE.

7.2.2 make call (public library)

```
make-call ::= 'wtai://wp/mc' ';' telnumber
telnumber  ::= CDATA
```

This function is used to initiate a mobile originated call using the specified number. The user must explicitly acknowledge the operation.

The *Make Call* function can be used from within any application, not only WTA, to present the user with a number that can be dialled.

Note, that the call must be terminated using standard MMI.

parameter	type	default	wtai://wp/mc (make call)
`telnumber`	string		Destination number to call. All valid telephony number digits may be used (‘`0`’-‘`9`’), as well as ‘`+`’ for the international number prefix.

7.2.3 send DTMF (voice control library)

```
send-DTMF  ::= 'wtai://vc/sd' ';' dtmfstring ['!' result ]
dtmfstring ::= CDATA
result     ::= NMTOKEN
```

Send DTMF tone sequence through an active voice connection. If the call succeeds the integer value zero is returned. In case of unsuccessful outcome an error code will be returned.

parameter	type	default	wtai://vc/sd (send DTMF)
dtmfstring	string		Any valid sequence of standard DTMF characters ('*', '#', ',', 'A'-'D', see /GSM 11.11/).
result	string		Name of variable to which a string is assigned that returns a negative WTAI error code in case of failure. For compatibility with WML, to be ignored by DE.

7.2.4 send DTMF (public library)

```
send-DTMF  ::= 'wtai://wp/sd' ';' dtmfstring ['!' result ]
dtmfstring ::= CDATA
result     ::= NMTOKEN
```

Send DTMF tone sequence through an active voice connection. The user must explicitly or implicitly acknowledge the operation. For instance, an acknowledgement made once for the public Make Call function could also be valid for all calls of the function Send DTMF Tones during that call. Or, acknowledgement can be made for each call of the Send DTMF function. This is implementation dependant.

Note, that like for the Make Call public function, the call must be terminated using standard MMI.

parameter	type	default	wtai://wp/sd (send DTMF)
dtmfstring	string		Any valid sequence of standard DTMF characters ('*', '#', ',', 'A'-'D', see /GSM 11.11/).
result	string		Name of variable to which a string is assigned that returns a negative WTAI error code in case of failure. For compatibility with WML, to be ignored by DE.

7.2.5 send USSD (GSM specific library)

```
send-ussd  ::= 'wtai://gsm/su' ';' ussdstring ';' ussddcs ';'
               ussdtype ';' ussdid ['!' result ]
ussdstring ::= CDATA
ussddcs    ::= CDATA
ussdtype   ::= num-t
ussdid     ::= num-t
result     ::= NMTOKEN
```

This function is used to make the handset send a USSD message. The assumption of the WTA user agent is that the SendUSSD command always succeeds. However, in case of failure an error code according to Section 9.2.2 is returned.

parameter	type	default	wtai://gsm/su (send USSD)
ussdstring	string		Contents of the USSD string to send. This may include any of the USSD characters permitted by /GSM 02.90/.
ussddcs	string		Permitted values specified in /GSM 02.90/.
ussdtype	string		0=ProcessUnstructuredSS-Request operation 1=result to an UnstructuredSS-Request operation 2=result to an UnstructuredSS-Request operation to be followed by release of the USSD transaction (i.e. after sending the FACILITY message, the mobile must send a RELEASE COMPLETE message for the transaction id associated with the USSD message)
ussdid	string		In the case where the sent USSD message is in response to

			a network initiated USSD message, i.e. a type 1 or 2 message, then this parameter takes the value of the transaction id of the corresponding network initiated USSD message. In case the sent USSD message is not in response to a network initiated USSD message, i.e. a type 0 message, then this parameter shall take the value –1.
`result`	string		Name of variable to which a string is assigned that returns the transaction id (see /GSM 04.07/ § 11) of the USSD message sent or a negative WTAI error code in case of failure. For compatibility with WML, to be ignored by DE.

8 S@TML STK EXTENSIONS

8.1 Security

The DE will check the URL from which the current deck has been received to decide whether this is a trusted or not-trusted source. The use of all S@TML STK Extensions is in general restricted to trusted decks. It is up to the DE implementation to provide further refinements of this security scheme, e.g. configurable levels of trust for different attributes and elements defined to be S@TML STK Extensions.

8.2 Declarations

8.2.1 sat-var

```
var ::= '<sat-var/>'

<!ELEMENT sat-var EMPTY>
<!ATTLIST sat-var
  sat-name          NMTOKEN        #REQUIRED
  sat-do-clr        %bool-t;       "false"
  %id-attrs;
>
```

The `sat-var` element specifies a temporary variable. There may be up to 127 temporary variables per deck. Names of variables must be unique within one deck. Note, that for compatibility with WML variables need not to be declared. In this case they are assumed to be temporary variables with the properties given by the default values of the attributes of `sat-var`.

attribute	type	default	<sat-var>
sat-name	string		Specifies the variable's identifier.
sat-do-clr	bool	`false`	If set to true the variable contents will be cleared when the current deck is left. Otherwise the variable contents is accessible in the current and other decks.

8.2.2 sat-const

```
const ::= '<sat-const/>'

<!ELEMENT sat-const EMPTY>
<!ATTLIST sat-const
  sat-name          NMTOKEN        #REQUIRED
  sat-value         CDATA          #REQUIRED
  %id-attrs;
```

```
>
```

The `sat-const` element specifies a constant text string accessible in all cards of a deck. There may be up to 64 constant text strings per deck. Names of constants must be unique within one deck, and all constants referred to inside a deck must have been declared in this deck.

attribute	type	default	<sat-const>
sat-name	string		Specifies the constant's identifier.
sat-value	string		Specifies the constant's value.

8.2.3 sat-sps

```
sps ::= '<sat-sps/>'

<!ELEMENT sat-sps EMPTY>
<!ATTLIST sps
  sat-name          NMTOKEN        #REQUIRED
  sat-var-id        %hex-t;        #REQUIRED
  sat-do-clr        %bool-t;       "false"
  %id-attrs;
>
```

The `sat-sps` element specifies a reference to a permanent variable hold in the service permanent store. There may be up to 64 entries per service in the SPS. Names of service permanent store references must be unique within one deck. Service permanent store variables are read-only in S@TML.

attribute	type	default	<sat-sps>
sat-name	string		Specifies the SPS variable's identifier used in S@TML. The "`sat-sps:`" prefix will be added automatically if missing.
sat-var-id	string		Specifies the SPS variable id used by SB and stored in the SIM. It is given in hexadecimal representation and must fit in the range of 80...BF. Note that this is an optional feature of SBC, and that the values have to be defined by the card issuer.
sat-do-clr	bool	`false`	If set to true the permanent variable's contents will be cleared when the current deck is left.

8.3 Variables and Constant References

If not stated otherwise for S@TML variables and constant references the rules given in the S@TML Core Language definition, Section 6.4, apply just as well. Note, that the use of variables in decks with different character encodings (i.e. different data coding schemes) may lead to unexpected results.

8.3.1 Environment variables

S@TML environment variables the identifiers have a prefix "**`sat-env:`**" to indicate the use of an S@TML STK Extension, e.g. "`sat-env:BrowserVersion`" for the version information of the SB. They can not be changed (e.g. used in a `setvar` or `input` element), but are read-only. The existing environment variables and their formats are defined in /SBC/, see also the table provided in Section 12. Note, that this format is often a binary TLV structured sequence of bytes.

8.3.2 Service permanent store (SPS) variables

S@TML variables can reside in the service permanent store (SPS) . Their identifiers have a prefix "**`sat-sps:`**" to indicate the use of an S@TML STK Extension. Note, that this is an optional feature of SBC, and that the variable identifiers have to be defined by the card issuer.

8.3.3 Constant references

For text constant references the same notation is used as for variable substitution. Their identifiers have a prefix "**`sat-const:`**", e.g. "`sat-const:x`". Text constants are of course read-only. As explained in Section 8.2.2 all constants referred to inside a deck must have been declared in this deck.

8.4 Fields

8.4.1 %satfld, %tone-t

```
%satfld ::= play-tone | inkey

<!ENTITY % satfld  “sat-play-tone | sat-inkey“>

<!ENTITY % tone-t  “(dial|busy|congestion|radio-ack|
                radio-gone|error|call-wait|ring|
                beep|ack|nack)“>
```

8.4.2 sat-play-tone

```
play-tone ::= ’<sat-play-tone/>’

<!ELEMENT sat-play-tone EMPTY>
<!ATTLIST sat-play-tone
  sat-title        %vdata-t;        #IMPLIED
  sat-tone         %tone-t;         “beep“
  sat-duration     %num-t;          #IMPLIED
  %id-attrs;
>
```

The `sat-play-tone` element specifies a tone which will be played on the handset.

attribute	type	default	<sat-play-tone>
`sat-title`	string	null	Title which may be displayed while the command is executed.
`sat-tone`	enum	`beep`	Specifies the tone to be played.
`sat-duration`	num		Specifies the length of the tone in $1/10^{th}$ of seconds if applicable. If not given a handset manufacturer default applies.

8.4.3 sat-inkey

```
sat-inkey ::= ’<sat-inkey/>’

<!ELEMENT sat-inkey EMPTY>
<!ATTLIST sat-inkey
  sat-title        %vdata-t;        #IMPLIED
  sat-name         NMTOKEN          #REQUIRED
  sat-format       CDATA            #REQUIRED
  sat-help         %vdata-t;        #IMPLIED
  xml:lang         NMTOKEN          #IMPLIED
  %id-attrs;
>
```

The `sat-inkey` element specifies a key input field to be executed by the SB.

attribute	type	default	<sat-inkey>
`sat-title`	string	null	Title to be displayed while the command is executed. If not present DE may use part of the preceding text up to the

			next tag (except the ignored %emph tags).
sat-name	string		Specifies the variable to be assigned.
sat-format	string	M	Specifies the set of characters accepted as input: M or 1M = any character Y or 1Y = "yes or no" choice N or 1N = digit character ('0'-'9', '+', '*', '#') Note, that in WML the digit characters do not include '+', '*', and '#', and the 'Y' or '1Y' formats do not exist.
sat-help	string	null	Specifies a help string which will be displayed when the user requests help for this input element. Overrides the sat-help attribute in an enclosing deck or card.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

8.5 Statements

The elements for statements from the S@TML STK Extensions Statements are executed in the order given by the p and %satstmt elements inside a card. This means that e.g. a sequence of a first p element, a sat-send-sms element, and a second p element will be executed by displaying the text of the first paragraph, then sending the short message, and then displaying the text of the second paragraph.

8.5.1 %satstmt, %check-t, %dcs-t

```
%satstmt ::=  send-sms | refresh | gen-stk | exit |
              setup-call | send-ussd | local-info |
              encrypt | decrypt | plug-in | extract

<!ENTITY % satstmt "sat-send-sms | sat-refresh | sat-gen-stk | sat-exit |
           sat-setup-call | sat-send-ussd | sat-local-info |
           sat-encrypt | sat-decrypt | sat-plug-in | sat-extract">

<!ENTITY % check-t "(none|mac)">

<!ENTITY % dcs-t "(sms|ucs2|binary)">
```

8.5.2 sat-send-sms

```
send-sms ::= '<sat-send-sms>' #PCDATA '</sat-send-sms>'

<!ELEMENT sat-send-sms (#PCDATA)>
<!ATTLIST sat-send-sms
  sat-title        %vdata-t;        #IMPLIED
  sat-dest         CDATA            #REQUIRED
  sat-smsc         CDATA            #IMPLIED
  sat-period       %num-t;          #IMPLIED
  sat-pid          %hex-t;          "00"
  sat-dcs          %dcs-t;          "sms"
  %id-attrs;
>
```

The sat-send-sms element specifies that a short message containing the enclosed text will be sent to the given destination. Note, that the use of variables from decks with different character encodings in the text part (i.e. different data coding schemes) may lead to unexpected results. If sat-dcs is binary the enclosed text will denote the short message user data by a sequence of hexadecimal digits just as defined for attributes of type hex-t in Section 5.1.6.

attribute	type	default	<sat-send-sms>
sat-title	string	null	Title which may be displayed as alpha identifier while the command is executed.
sat-dest	string		Specifies the destination address telephone number. All valid telephony number characters and digits may be used

			(‘0’-‘9’, ‘+’, ‘*’, ‘#’).
sat-smsc	string	null	Specifies the Short Message Service Centre (SMSC) address. All valid telephony number characters and digits may be used (‘0’-‘9’, ‘+’, ‘*’, ‘#’). If not present the current value of the ME is used.
sat-period	num	null	Specifies the validity period for the short message in a number denoting days (D), hours (h), and minutes (m): DDDhhmm with 7 digits, DDhhmm with 6 digits, Dhhmm with 5 digits, hhmm with 4 digits, hmm with 3 digits, mm with 2 digits, or m with 1 digit. It is rounded up by the DE to the next value that can be represented as relative validity period conforming to /GSM 11.14/, /GSM 03.40/. If not present the current value of the ME is used.
sat-pid	hex	00	Specifies the protocol identifier to be used for the short message as specified in /GSM 11.14/, /GSM 03.40/.
sat-dcs	enum	sms	Specifies the data coding scheme to be used for the short message to be SMS default charset, UCS2, or binary (see /GSM 03.38/). The GSM 11.14 command qualifier will be set accordingly by the DE.

8.5.3 sat-setup-call

```
setup-call ::= '<sat-setup-call/>'

<!ELEMENT sat-setup-call EMPTY>
<!ATTLIST sat-setup-call
  sat-confirm      %vdata-t;        #IMPLIED
  sat-title        %vdata-t;        #IMPLIED
  sat-dest         CDATA            #REQUIRED
  sat-cmdqual      %hex-t;          "00"
  %id-attrs;
>
```

The sat-setup-call element specifies that a call setup will be done to the given telephone number.

attribute	type	default	<sat-setup-call>
sat-confirm	string	null	A string to be displayed for user confirmation.
sat-title	string	null	The string which is displayed in the call set-up phase.
sat-dest	string		Specifies the destination address telephone number. All valid telephony number characters and digits may be used (‘0’-‘9’, ‘+’, ‘*’, ‘#’). A DTMF sequence can be added at the end of the string using a “,“ separator. Example: “0660773534,1234”.
sat-cmdqual	hex	00	Specifies the GSM 11.14 command qualifier to be used in two hex digits (see § 12.6 in /GSM11.14/): 00=set up call if not busy on another call 01=set up call if not busy on another call, with redial 02=set up call putting other calls on hold 03=set up call putting other calls on hold, with redial 04=set up call disconnecting other calls 05=set up call disconnecting other calls, with redial

8.5.4 sat-send-ussd

```
send-ussd ::= '<sat-send-ussd/>'

<!ELEMENT sat-send-ussd EMPTY>
<!ATTLIST sat-send-ussd
```

```
  sat-title         %vdata-t;        #IMPLIED
  sat-ussddcs       %hex-t;          #REQUIRED
  sat-data          %hex-t;          #REQUIRED
  %id-attrs;
>
```

The `sat-send-ussd` element specifies that unstructured supplementary services data will be sent to the given destination.

attribute	type	default	<sat-send-ussd>
sat-title	string	null	Title which may be displayed as alpha identifier while the command is executed.
sat-ussddcs	hex		Specifies the data coding scheme as defined in /GSM 03.38/
sat-data	string	null	Specifies the USSD string to be sent as defined in /GSM 02.30/, /GSM 02.90/ encoded in hex characters.

8.5.5 sat-local-info

```
local-info ::= '<sat-local-info/>'

<!ELEMENT sat-local-info EMPTY>
<!ATTLIST sat-local-info
  sat-name          CDATA;           #REQUIRED
  sat-href          %HREF-t;         #IMPLIED
  sat-method        (post|get)       "get"
  sat-cmdqual       %hex-t;          "00"
  %id-attrs;
>
```

The `sat-local-info` element specifies provides the local information data specified by the command qualifier. The result is stored in the output variable and optionally posted to the URI given.

attribute	type	default	<sat-local-info>
sat-name	string	null	The output variable containing the result of the local info operation element. It is TLV-formatted as described in the /GSM 11.14/ location information object, and returned in a binary string, (see /SBC/, /GSM 04.08/).
sat-href	string URI		Specifies the URI where to post the local information string. The parameter gets the same name as given for the output variable.
sat-method	enum	`get`	Specifies the method for the operation.
sat-cmdqual	hex	`00`	Specifies the /GSM 11.14/ command qualifier to be used: 00=Location Information (MCC, MNC, LAC and Cell Identity) 01=IMEI of the ME 02=Network Measurement results 03=Date, time and time zone $(DTTinPLI)$

location information	[this table is only informational] GSM 11.14
byte 1	Location information tag
byte 2	Length = '07'
bytes 3 - 5	Mobile Country & Network Codes (MCC & MNC)
bytes 6 - 7	Location Area Code (LAC)
bytes 8 - 9	Cell Identity Value (Cell ID)

8.5.6 sat-refresh

```
refresh ::= '<sat-refresh/>'

<!ELEMENT sat-refresh EMPTY>
<!ATTLIST sat-refresh
  sat-files        %hex-t;          #IMPLIED
  sat-cmdqual      %hex-t;          "01"
  %id-attrs;
>
```

The `sat-refresh` element specifies that a file change notification will be sent to the ME.

attribute	type	default	<sat-refresh>
sat-files	hex	null	Specifies the list of file change notifications as specified in /GSM 11.14/ encoded in hex characters, the number of files and Files.
sat-cmdqual	hex	`01`	Specifies the /GSM 11.14/ command qualifier to be used: `00`=SIM Initialisation and Full File Change Notification `01`=File Change Notification `02`=SIM Initialisation and File Change Notification `03`=SIM Initialisation `04`=SIM Reset

8.5.7 sat-gen-stk

```
gen-stk ::= '<sat-gen-stk/>'

<!ELEMENT sat-gen-stk EMPTY>
<!ATTLIST sat-gen-stk
  sat-cmdtype      %hex-t;        #REQUIRED
  sat-cmdqual      %hex-t;        #REQUIRED
  sat-destdev      %hex-t;        #REQUIRED
  sat-data         %hex-t;        #IMPLIED
  sat-output       %vdata-t;      #IMPLIED
  sat-encap        %bool-t;       "false"
  %id-attrs;
>
```

The `sat-gen-stk` element allows to specify a generic /GSM 11.14/ SIM Toolkit command to be sent to the ME.

attribute	type	default	<sat-gen-stk>
sat-cmdtype	hex		Specifies the type of command value as defined in § 12.6 of /GSM 11.14/ and given in 2 hexadecimal characters. This determines the meaning and range of legal values for the following attributes.
sat-cmdqual	hex		Specifies the command qualifier to be used as defined in § 12.6 of /GSM 11.14/ and given in 2 hexadecimal characters.
sat-destdev	hex		Specifies the destination device identity to be used as defined in § 12.7 of /GSM 11.14/ and given in 2 hexadecimal characters.
sat-data	hex		Specifies all further TLV objects for the SIM Toolkit command as defined in § 6.6 and Chapter 12 of /GSM 11.14/ and given in hexadecimal characters. Command details, device identities, and alpha identifier objects must not be included.
sat-output	string		Specifies the variable where to store the proactive command specific response data object, as described in

			/SBC/.
sat-encap	bool		Specifies whether the response data object must be encapsulated as described in /SBC/.

8.5.8 sat-exit

```
exit ::= '<sat-exit/>'

<!ELEMENT sat-exit EMPTY>
<!ATTLIST sat-exit
  sat-cleanbuf     %bool-t;       "false"
  sat-outvarlist   CDATA          #IMPLIED
  %id-attrs;
>
```

The `sat-exit` element allows to terminate the execution of the browsing session.

attribute	type	default	<sat-exit>
sat-outvarlist	string	null	A comma separated list of variable names.
sat-cleanbuf	bool	`false`	Specifies if cards stored in the SB's execution buffer should be cleared before exit (see also /SBC/).

8.5.9 sat-encrypt

```
encrypt ::= '<sat-encrypt/>'

<!ELEMENT sat-encrypt EMPTY>
<!ATTLIST sat-encrypt
  sat-check        %check-t;      "none"
  sat-crypt        %bool-t;       "false"
  sat-kic          %hex-t;        #IMPLIED
  sat-kid          %hex-t;        #IMPLIED
  sat-inlist       CDATA          #REQUIRED
  sat-out          CDATA          #REQUIRED
  %id-attrs;
>
```

The sat-encrypt element allows to encrypt data or calculate cryptographic checksums. The sat-out variable will contain a secure message built as follows:

1) The data from all input parameters (sat-inlist) will be concatenated to a data block (series of LVs) before processing.

2) A padding with 00 characters will be added if the total length of the data block is not a multiple of 8 bytes.

3) If requested, MAC calculation is done on the data block value (i.e. the LV serie) and padding. The MAC is inserted at the beginning of the data block

4) If requested the MAC value, data block value and padding will be encrypted by SB.

5) The result is stored in the secure message format defined in /SBC/. The SPI, kic/kid and PCntr of this structure are used to find the original data block value (serie of LVs) on the content provider side.

The maximum supported length for the clear data block (serie of LVs described in sat-inlist) is restricted (249 or 240 bytes depending on MAC usage).

At least triple DES CBC and cryptographic checksum must be supported by SB.

attribute	type	default	<sat-encrypt>

sat-check	enum	none	Denotes the checking algorithm to use: none=no checksum mac=cryptographic checksum
sat-crypt	bool	false	Set to true if the input data shall be encrypted.
sat-kic	hex	00	Gives the key index and algorithm used for encryption as specified in /GSM 03.48/ and written in 2 hex chars.
sat-kid	hex	00	Gives the key index and algorithm used for the checking as specified in /GSM 03.48/ and written in 2 hex chars.
sat-inlist	string		A comma separated list of input values given by variable references or constant strings. Typical usage: sat-inlist = "hello,$(var1),$(var2), const text,$(var3)"
sat-out	string		Gives the name of the output variable. It is an error to specify a read-only variable here. Typical usage: sat-out = "my_out_var"

KIc or KID	[this table is only informational] GSM 03.48
XXXXYYZZ	binary value of KIc or KID with the following fields:
XXXX	index of key to be used is XXXX
YY	00=DES CBC mode 01=Triple DES, outer CBC mode, 2 keys 10=Triple DES, outer CBC mode, 3 keys 11=DES ECB mode (not allowed for KID)
ZZ	00=algorithm known implicitly 01=DES 10=reserved 11=proprietary

8.5.10 sat-decrypt

```
decrypt ::= '<sat-decrypt/>'

<!ELEMENT sat-decrypt EMPTY>
<!ATTLIST sat-decrypt
  sat-check      %check-t;    "none"
  sat-crypt      %bool-t;     "false"
  sat-kic        %hex-t;      #IMPLIED
  sat-kid        %hex-t;      #IMPLIED
  sat-mac        %hex-t;      #IMPLIED
  sat-padding    %hex-t       "00"
  sat-in         CDATA        #IMPLIED
  sat-sec-msg    CDATA        #IMPLIED
  sat-outlist    CDATA        #REQUIRED
  %id-attrs;
>
```

The sat-decrypt element allows decrypting data or verifying cryptographic checksums. When verification of a cryptographic checksum fails, the SB processing stops.

Sat-outlist must contain the same number of variables as the number of clear values (LVs) inside the data block.
The maximum supported length for the clear data block (serie of LVs) is restricted (249 or 240 bytes depending on MAC usage).

At least triple DES CBC and cryptographic checksum must be supported by SB.

NOTE: if 'sat-sec-msg' attribute is present, it will be used to define the data to be decrypted, even if legacy attributes are also present.

Attribute	type	default	<sat-decrypt>
sat-check	enum	none	Denotes the checking algorithm to use: none=no checksum mac=cryptographic checksum
sat-crypt	bool	false	Set to true if the input data shall be decrypted.
sat-kic	hex	00	Gives the key index and algorithm used for encryption as specified in /GSM 03.48/ and written in 2 hex chars.
sat-kid	hex	00	Gives the key index and algorithm used for the checking as specified in /GSM 03.48/ and written in 2 hex chars.
sat-mac	hex		Gives the MAC value. Required if a cryptographic checksum is to be used.
sat-padding	hex	00	Padding counter.
sat-in	string		Datas are in the Enciphered or cleartext datablock format defined in /SBC/ specifications. They can be given by a variable reference or a constant string. Typical usage: sat-in = "$(sat-const:my_input)" or sat-in = "45 56 AA BB CC DD EE FF"
sat-sec-msg	string		Input data in SecMsg format. The data can either be given explicitly or implicitly. In the former case the data given must be in a hex-t format. No variable substitutions allowed and basic sanity checks are performed during translation time to ensure the data specified actually represents a SecMsg structure. In the latter case the data is specifed via a variable reference (sat-sec-msg="$var"). No additional checks are performed by the Decoder/Encoder, but the browser might still issue an error if the input data from the specified variable do not conform to SecMsg format.
sat-outlist	string		A comma separated list of output variable names. It is an error to specify a read-only variable here. Typical usage: sat-outlist = "var1,var2,var3"

For coding of sat-kic and sat-kid see 8.5.9.

8.5.11 sat-extract

```
extract ::= <sat-extract/>

<!ELEMENT sat-extract EMPTY>
<!ATTLIST sat-extract
  sat-destvar      NMTOKEN     #REQUIRED
  sat-srcvar       NMTOKEN     #REQUIRED
  sat-startindex   %num-t;     #IMPLIED
  sat-length       %num-t;     #REQUIRED
  xml:lang         NMTOKEN     #IMPLIED
  %id-attrs;
>
```

The sat-extract element get the substring from the source variable, and put the substring into the destination variable. If sat-srcvar refers to a variable coded in UCS2, the value for sat-startindex and sat-length (of substring) gets multiplied by 2 internally.

attribute	type	default	<select>
sat-destvar	string		Specifies the name of the destination variable
sat-srcvar	string		Specifies the name of the source variable
sat-startindex	num	0	The start index of the substring
sat-length	num		The length of the substring
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

8.6 Navigation

8.6.1 Use of URLs

In attributes of type `HREF-t` the URI of some S@TML (or WML) page is given. The notation of URIs is as defined in /WML1.1/, /WTAI1.1/, /WTAIGSM1.1/, and /RFC2396/. To access resident decks stored in the SIM card URIs are used that begin with the prefix string "**sim:**". All URIs starting with "**sim:/s/**" are reserved for S@T. The following reserved URIs are pre-defined:

name	pre-defined URIs
sim:/s/home	Access the starting deck (home page) of the S@T browser.

8.6.2 Contextual Menu

The S@T browser manages a contextual menu which can be called by the user for navigation. The contents can be modified by use of the `do` element of S@TML.

8.6.3 sat-switch

```
sat-switch ::= '<sat-switch>' (sat-case)+ '<sat-/switch>'

<!ELEMENT sat-switch (sat-case)+>
<!ATTLIST sat-switch
  sat-name            NMTOKEN     #REQUIRED
  sat-defaulturl      %HREF-t;    #IMPLIED
  sat-casesensitive   %bool-t;    "false"
  xml:lang            NMTOKEN     #IMPLIED
  %id-attrs;
>
```

The sat-switch element will compare its variable value with the sat-case value. If sat-switch variable value is same as a sat-case value, the SB will branch to the URI specified by that sat-case element.

attribute	type	default	<select>
sat-name	string		Specifies the sat-switch variable name
sat-defaulturl	string	Null	If all sat-case values don't match the sat-switch variable value, the SB will branch to this URL
sat-casesensitive	bool	False	Sepcifies if the case is sensitive when compare sat-switch variable value with sat-case value
xml:lang	string	Null	For compatibility with WML, to be ignored by DE.

8.6.4 sat-case

```
sat-case ::= <sat-case/>

<!ELEMENT sat-case EMPTY>
<!ATTLIST sat-case
  sat-value      %vdata-t;   #REQUIRED
  sat-href       %HREF-t;    #REQUIRED
  xml:lang       NMTOKEN     #IMPLIED
  %id-attrs;
>
```

The sat-case element specifies the value to be compared and the URL to which the SB will branch if the comparison is success.

attribute	type	default	<select>
sat-value	string		Specifies the value to be compared by the sat-switch element
sat-href	string		Specifies the URI of this case
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

8.7 Extension plug-ins

8.7.1 sat-plug-in

```
plug-in ::= '<sat-plug-in/>'

<!ELEMENT sat-plug-in EMPTY>
<!ATTLIST sat-plug-in
  sat-uid          %hex-t;       #REQUIRED
  sat-inlist       CDATA         #IMPLIED
  sat-outlist      CDATA         #IMPLIED
  sat-return       %bool-t;      "true"
  %id-attrs;
>
```

The `plug-in` element can be used to execute additional functions which are beyond the specification of the S@T browser.

attribute	type	default	<sat-plug-in>
`sat-uid`	hex	null	A unique ID that identifies the plug-in. The first byte is the manufacturer or S@T agreed ID and the second one is the execute element reference. See /SBC/ for details.
`sat-inlist`	string	null	A comma separated list of input values given by variable references or constant strings. Typical usage: `sat-inlist = "hello,$(var1),$(var2), const text,$(var3)"`
`sat-outlist`	string	null	A comma separated list of output variable names. It is an error to use read-only variables here. Typical usage: `sat-outlist = "var1,var2,var3"`
`sat-return`	bool	`true`	Indicates if the browser will retrieve control again after plug-in execution.

9 FUTURE S@TML

9.1 Content

The following elements for images and tables are taken from WML. They will not be supported by the present S@TML Core Language. They shall be ignored. They may be included in future S@TML versions. Nevertheless it is allowed that a S@TML DE implementation supports translation of these elements into SBC byte code that will be displayed by the SB.

9.1.1 %len, %IAlign

```
<!ENTITY % len   “CDATA“ >      <!--[0-9]+ for pixels,
                                    [0-9]+ “%“ for percentage length -->
<!ENTITY % IAlign “(top|middle|bottom)“ >
```

9.1.2 img

```
img ::= ’<img/>’

<!ELEMENT img EMPTY>
<!ATTLIST img
  alt          %vdata-t;        #REQUIRED
  src          %HREF-t;         #REQUIRED
  localsrc     %vdata-t;        #IMPLIED
  vspace       %len;            “0“
  hspace       %len;            “0“
  align        %IAlign;         “bottom“
  height       %len;            #IMPLIED
  width        %len;            #IMPLIED
  xml:lang     NMTOKEN          #IMPLIED
  %id-attrs;
>
```

attribute	type	default	<img>
alt	string		Will be used by DE as text to be displayed for the image by SB.
src	string URI		For compatibility with WML, to be ignored by DE.
localsrc	string		For compatibility with WML, to be ignored by DE.
vspace	num	0	For compatibility with WML, to be ignored by DE.
hspace	num	0	For compatibility with WML, to be ignored by DE.
align	enum	bottom	For compatibility with WML, to be ignored by DE.
height	num		For compatibility with WML, to be ignored by DE.
width	num		For compatibility with WML, to be ignored by DE.
xml:lang	string		For compatibility with WML, to be ignored by DE.

9.1.3 table

```
table ::= ’<table>’ tr+ ’</table>’

<!ELEMENT table (tr)+>
<!ATTLIST table
  title        %vdata-t;        #IMPLIED
  align        CDATA            #IMPLIED
  columns      %num-t;          #REQUIRED
  xml:lang     NMTOKEN          #IMPLIED
  %id-attrs;
>
```

attribute	type	default	<table>
title	string	null	Will be used by DE as title to be displayed for the table by SB.
align	string	null	For compatibility with WML, to be ignored by DE.
columns	num		Number of columns inside this table.
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

9.1.4 tr – table row

```
tr ::= '<tr>' td+ '</tr>'

<!ELEMENT tr (td)+>
<!ATTLIST tr
  %id-attrs;
>
```

9.1.5 td – table data

```
td ::= '<td>' %flow '<td>'

<!ELEMENT td (%flow;)*>
<!ATTLIST td
  %xml:lang   NMTOKEN          #IMPLIED
  %id-attrs;
>
```

As tables may not be nested `%flow` may not contain other tables in this case.

attribute	type	default	<td>
xml:lang	string	null	For compatibility with WML, to be ignored by DE.

9.2 WTAI

The following command is taken from WTAI. It will not be supported by the S@TML 1.0 WTAI subset, however, may be included in future S@TML versions.

9.2.1 provide local information (GSM specific library)

```
provide-local-info ::= 'wtai://gsm/li' ['!' result ]
result             ::= NMTOKEN
```

This function is used to provide the current location information of the GSM terminal. This information uniquely identifies the GSM cell in which the user is located at invocation time. The user must explicitly acknowledge the operation.

parameter	type	default	wtai://gsm/li (provide local information)
result	string		The return value is a string including the 8 octets of the GSM location information in hexadecimal representation as follows: Octets 1 – 3 Mobile Country and Network Codes (MCC & MNC) Octets 4 – 5 Location Area Code (LAC) Octets 6 – 7 Cell Identity Value (Cell ID) Octet 8 Timing Advance For this octet a default value is returned (TBD). The mobile country code (MCC), the mobile network code (MNC), the location area code (LAC), the Cell ID and the Timing Advance are coded as in GSM 04.08.

			In case of failure the return value is a negative number and the WTAI error code.

9.2.2 WTAI Error Codes

The following error code results are used in case of failures for WTAI functions (see /WTAI1.1/, /WTAIGSM1.1/):

value	WTAI error code results
-1	Id not found. Function could not be completed.
-2	Illegal number of parameters, function could not be resolved due to missing parameters.
-3	Service not available or non-existent function.
-4	Service temporarily unavailable.
-5	Called party is busy.
-6	Network is busy.
-7	No answer, i.e. call set-up timed out.
-8	Unknown.
-9 ... -63	Reserved for future use by WTA standard library functions.
-64	USSD dialogue in progress
-65	Illegal characters
-66 ... -127	Reserved for other network specific error codes.

10 COMPLETE DTD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--
S@T Markup Language (S@TML) Document Type Definition.
S@TML is an XML language. Typical usage:
  <?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE satml SYSTEM "http://www.simalliance.org/DTD/satml106.dtd">

  <satml>
  ...
  </satml>
For compatibility with the Wireless Markup Language 1.1 (WML)
also the respective DOCTYPE and <wml> ... </wml> may be used.

Change History (versions equal to S@T 01.10 S@TML specification):
satml100.dtd V1.0.0 2000/03 - First approved release.
satml101.dtd V1.0.1 2000/04 - Added missing sat-plug-in in DTD.
satml102.dtd V1.0.2 2000/05 - Lexical entities added / explained,
   new sat-do-clr in sat-const, sat-value in sat-const required.
satml103.dtd V1.0.3 2000/06 - Type dcs-t for sat-dcs, no sat-cmdqual
   in sat-send-sms, sat-ussddcs in sat-send-ussd, sat-in/out(list)
   in sat-encrypt/decrypt and sat-plug-in, sat-mac for sat-decrypt.
satml104.dtd V1.0.4 2000/07 - No changes to DTD, change history added
satml105.dtd V1.0.5 2001/05 - extension of sat-gen-stk
satml106.dtd V1.0.6 2001/06 – minor fix, string replaced by %vdata-t for sat-
output attribute in sat-gen-stk.
-->

<!-- ............... Entities ............................. -->
<!ENTITY quot "&#34;">      <!-- quotation mark -->
<!ENTITY amp  "&#38;#38;">  <!-- ampersand -->
<!ENTITY apos "&#39;">      <!-- apostrophe -->
<!ENTITY lt   "&#38;#60;">  <!-- less than -->
<!ENTITY gt   "&#62;">      <!-- greater than -->

<!ENTITY nbsp " "><!-- non-breaking space -->
<!ENTITY shy  "&#173;"><!-- soft hyphen (discretionary hyphen) -->

<!ENTITY % bool-t    "(true|false)">
<!ENTITY % num-t     "NMTOKEN">  <!-- number -->
<!ENTITY % hex-t     "NMTOKEN">  <!-- hex str -->

<!ENTITY % vdata-t   "CDATA">    <!-- string with variables -->
<!ENTITY % vnum-t  "%vdata-t;">  <!-- number with variables -->
<!ENTITY % vhex-t  "%vdata-t;">  <!-- hex str with variables -->
<!ENTITY % HREF-t  "%vdata-t;">  <!-- URI, URL or URN with variables -->

<!ENTITY % id-attrs   "id      ID        #IMPLIED
                       classCDATA     #IMPLIED">

<!-- ............... Decks ............................... -->
<!ENTITY % decls "(sat-var|sat-const|sat-sps)*, template?">

<!ELEMENT satml (head?, %decls;, card+) >
<!ATTLIST satml
  xml:lang       NMTOKEN              #IMPLIED
  sat-storage    (dynamic|static)     "static"
  sat-serv-id    %hex-t;              #IMPLIED
  sat-help       %vdata-t;            #IMPLIED
  %id-attrs;
>
```

```
<!ELEMENT wml (head?, template?, card+) >
<!ATTLIST wml
  xml:lang       NMTOKEN              #IMPLIED
  sat-storage    (dynamic|static)     “static“
  sat-serv-id    %hex-t;              #IMPLIED
  sat-help       %vdata-t;            #IMPLIED
  %id-attrs;
>

<!ELEMENT head (access|meta)+ >
<!ATTLIST head
  %id-attrs;
>

<!ELEMENT access EMPTY>
<!ATTLIST access
  domain      CDATA       #IMPLIED
  path        CDATA       #IMPLIED
  %id-attrs;
>

<!ELEMENT meta EMPTY>
<!ATTLIST meta
  http-equiv  CDATA       #IMPLIED
  name        CDATA       #IMPLIED
  forua       %bool-t;    “false“
  content     CDATA       #IMPLIED
  scheme      CDATA       #IMPLIED
  %id-attrs;
>

<!-- ............... Cards ................................ -->
<!ENTITY % cardev-attrs
 “onenterforward     %HREF-t;         #IMPLIED
  onenterbackward    %HREF-t;         #IMPLIED
  ontimer            %HREF-t;         #IMPLIED“
>

<!ENTITY % satstmt “sat-send-sms | sat-refresh | sat-gen-stk | sat-exit |
           sat-setup-call | sat-send-ussd | sat-local-info |
           sat-encrypt | sat-decrypt | sat-plug-in | sat-extract“>

<!ELEMENT card (onevent*, timer?, (do|p|setvar|go|prev|refresh|%satstmt;)*) >
<!ATTLIST card
  title           %vdata-t;     #IMPLIED
  newcontext      %bool-t;      “false“
  ordered         %bool-t;      “true“
  xml:lang        NMTOKEN       #IMPLIED
  sat-help        %vdata-t;     #IMPLIED
  sat-history     %bool-t;      “true“
  sat-chain-next  %bool-t;      “false“
  %cardev-attrs;
  %id-attrs;
>

<!ELEMENT template (do|onevent)* >
<!ATTLIST template
  %cardev-attrs;
  %id-attrs;
>

<!ELEMENT timer EMPTY>
<!ATTLIST timer
  name      NMTOKEN       #IMPLIED
```

```
  value     %vdata-t;     #REQUIRED
  %id-attrs;
>

<!-- ............... Fields ............................... -->
<!ENTITY % emph       "em | strong | b | i | u | big | small">
<!ENTITY % layout     "br">
<!ENTITY % txt        "#PCDATA | %emph;">
<!ENTITY % flow       "%txt; | %layout; | img | anchor | a | table">

<!ENTITY % satfld     "sat-play-tone | sat-inkey">

<!ENTITY % fields     "%flow; | input | select | fieldset | %satfld; ">

<!ENTITY % TAlign     "(left|right|center)">
<!ENTITY % WrapMode   "(wrap|nowrap)">

<!ELEMENT p (%fields; | do)*>
<!ATTLIST p
  align          %TAlign;           "left"
  mode           %WrapMode;         #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  sat-auto-clr   %bool-t;           "false"
  sat-prio       (normal|high)      "normal"
  %id-attrs;
>

<!ELEMENT input EMPTY>
<!ATTLIST input
  title          %vdata-t;          #IMPLIED
  name           NMTOKEN            #REQUIRED
  type           (text|password)    "text"
  value          %vdata-t;          #IMPLIED
  format         CDATA              #IMPLIED
  emptyok        %bool-t;           "false"
  size           %num-t;            #IMPLIED
  maxlength      %num-t;            #IMPLIED
  sat-minlength  %num-t;            #IMPLIED
  sat-help       %vdata-t;          #IMPLIED
  tabindex       %num-t;            #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>

<!ELEMENT select (optgroup|option)+>
<!ATTLIST select
  title          %vdata-t;          #IMPLIED
  name           NMTOKEN            #IMPLIED
  value          %vdata-t;          #IMPLIED
  iname          NMTOKEN            #IMPLIED
  ivalue         %vdata-t;          #IMPLIED
  multiple       %bool-t;           "false"
  sat-help       %vdata-t;          #IMPLIED
  tabindex       %num-t;            #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>

<!ELEMENT option (#PCDATA|onevent)*>
<!ATTLIST option
  title          %vdata-t;          #IMPLIED
  value          %vdata-t;          #IMPLIED
  onpick         %HREF-t;           #IMPLIED
  no-wait        (no|yes|continue)  "no"
  sat-help       %vdata-t;          #IMPLIED
```

```
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>

<!ELEMENT optgroup (optgroup|option)+>
<!ATTLIST optgroup
  title          %vdata-t;          #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  %id-attrs;
>

<!ELEMENT fieldset (%fields;|do)*>
<!ATTLIST fieldset
  title          %vdata-t;          #IMPLIED
  xml:lang       NMTOKEN            #IMPLIED
  sat-auto-clr   %bool-t;           "false"
  sat-prio       (normal|high)      "normal"
  %id-attrs;
>

<!-- ............... Content ............................. -->
<!ELEMENT br EMPTY>
<!ATTLIST br
  %id-attrs;
>

<!ELEMENT em (%flow;)*>
<!ATTLIST em
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT strong (%flow;)*>
<!ATTLIST strong
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT i (%flow;)*>
<!ATTLIST i
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT b (%flow;)*>
<!ATTLIST b
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT u (%flow;)*>
<!ATTLIST u
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT big (%flow;)*>
<!ATTLIST big
  xml:lang    NMTOKEN          #IMPLIED
  %id-attrs;
>

<!ELEMENT small (%flow;)*>
<!ATTLIST small
  xml:lang    NMTOKEN          #IMPLIED
```

```
  %id-attrs;
>

<!ELEMENT sat-extract EMPTY>
<!ATTLIST sat-extract
  sat-destvar      NMTOKEN     #REQUIRED
  sat-srcvar       NMTOKEN     #REQUIRED
  sat-startindex   %num-t;     #IMPLIED
  sat-length       %num-t;     #REQUIRED
  xml:lang         NMTOKEN     #IMPLIED
  %id-attrs;
>


<!-- ............... Navigation ........................... -->
<!ENTITY % task “go | prev | noop | refresh | sat-switch“>

<!ELEMENT anchor (#PCDATA|br|img|go|prev|refresh)*>
<!ATTLIST anchor
  title       %vdata-t;       #IMPLIED
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>

<!ELEMENT a (#PCDATA|br|img)*>
<!ATTLIST a
  href        %HREF-t;          #REQUIRED
  no-wait     (no|yes|continue) “no”
  title       %vdata-t;         #IMPLIED
  xml:lang    NMTOKEN           #IMPLIED
  %id-attrs;
>

<!ELEMENT do (%task;)>
<!ATTLIST do
  type        CDATA           #REQUIRED
  label       %vdata-t;       #IMPLIED
  name        NMTOKEN         #IMPLIED
  optional    %bool-t;        “false“
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>

<!ELEMENT onevent (%task;)>
<!ATTLIST onevent
  type        CDATA           #REQUIRED
  %id-attrs;
>

<!ELEMENT go (postfield|setvar)*>
<!ATTLIST go
  href            %HREF-t;          #REQUIRED
  no-wait         (no|yes|continue) “no”
  sendreferer     %bool-t;          “false“
  method          (post|get)        “get“
  accept-charset  CDATA             #IMPLIED
  %id-attrs;
>

<!ELEMENT prev (setvar)*>
<!ATTLIST prev
  %id-attrs;
>

<!ELEMENT refresh (setvar)*>
```

```
<!ATTLIST refresh
  %id-attrs;
>

<!ELEMENT noop EMPTY>
<!ATTLIST noop
  %id-attrs;
>

<!ELEMENT sat-switch (sat-case)+>
<!ATTLIST sat-switch
  sat-name           NMTOKEN     #REQUIRED
  sat-defaulturl     %HREF-t;    #IMPLIED
  sat-casesensitive  %bool-t;    "false"
  xml:lang           NMTOKEN     #IMPLIED
  %id-attrs;
>

<!ELEMENT sat-case EMPTY>
<!ATTLIST sat-case
  sat-value      %vdata-t;   #REQUIRED
  sat-href       %HREF-t;    #REQUIRED
  xml:lang       NMTOKEN     #IMPLIED
  %id-attrs;
>


<!-- ............... Statements ........................... -->
<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
  name             %vdata-t;     #REQUIRED
  value            %vdata-t;     #REQUIRED
  %id-attrs;
>

<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
  name             %vdata-t;     #REQUIRED
  value            %vdata-t;     #REQUIRED
  %id-attrs;
>

<!-- ............... S@TML STK Extensions .................. -->
<!-- ............... Declarations ......................... -->
<!ELEMENT sat-var EMPTY>
<!ATTLIST sat-var
  sat-name         NMTOKEN       #REQUIRED
  sat-do-clr       %bool-t;      "false"
  %id-attrs;
>

<!ELEMENT sat-const EMPTY>
<!ATTLIST sat-const
  sat-name         NMTOKEN       #REQUIRED
  sat-value        CDATA         #REQUIRED
  %id-attrs;
>

<!ELEMENT sat-sps EMPTY>
<!ATTLIST sat-sps
  sat-name         NMTOKEN       #REQUIRED
  sat-var-id       %hex-t;       #REQUIRED
  sat-do-clr       %bool-t;      "false"
```

```
  %id-attrs;
>

<!-- ............... Fields ............................... -->
<!ENTITY % tone-t  "(dial|busy|congestion|radio-ack|
              radio-gone|error|call-wait|ring|
              beep|ack|nack)">

<!ELEMENT sat-play-tone EMPTY>
<!ATTLIST sat-play-tone
  sat-title        %vdata-t;        #IMPLIED
  sat-tone         %tone-t;         "beep"
  sat-duration     %num-t;          #IMPLIED
  %id-attrs;
>

<!ELEMENT sat-inkey EMPTY>
<!ATTLIST sat-inkey
  sat-title        %vdata-t;        #IMPLIED
  sat-name         NMTOKEN          #REQUIRED
  sat-format       CDATA            #REQUIRED
  sat-help         %vdata-t;        #IMPLIED
  xml:lang         NMTOKEN          #IMPLIED
  %id-attrs;
>

<!-- ............... Statements ........................... -->
<!ENTITY % check-t "(none|mac)">
<!ENTITY % dcs-t "(sms|ucs2|binary)">

<!ELEMENT sat-send-sms (#PCDATA)>
<!ATTLIST sat-send-sms
  sat-title        %vdata-t;        #IMPLIED
  sat-dest         CDATA            #REQUIRED
  sat-smsc         CDATA            #IMPLIED
  sat-period       %num-t;          #IMPLIED
  sat-pid          %hex-t;          "00"
  sat-dcs          %dcs-t;          "sms"
  %id-attrs;
>

<!ELEMENT sat-setup-call EMPTY>
<!ATTLIST sat-setup-call
  sat-confirm      %vdata-t;        #IMPLIED
  sat-title        %vdata-t;        #IMPLIED
  sat-dest         CDATA            #REQUIRED
  sat-cmdqual      %hex-t;          "00"
  %id-attrs;
>

<!ELEMENT sat-send-ussd EMPTY>
<!ATTLIST sat-send-ussd
  sat-title        %vdata-t;        #IMPLIED
  sat-ussddcs      %hex-t;          #REQUIRED
  sat-data         %hex-t;          #REQUIRED
  %id-attrs;
>

<!ELEMENT sat-local-info EMPTY>
<!ATTLIST sat-local-info
  sat-name         CDATA            #REQUIRED
  sat-href         %HREF-t;         #IMPLIED
```

```
  sat-method       (post|get)       “get“
  sat-cmdqual      %hex-t;          “00“
  %id-attrs;
>

<!ELEMENT sat-refresh EMPTY>
<!ATTLIST sat-refresh
  sat-files        %hex-t;          #IMPLIED
  sat-cmdqual      %hex-t;          “01“
  %id-attrs;
>

<!ELEMENT sat-gen-stk EMPTY>
<!ATTLIST sat-gen-stk
  sat-cmdtype      %hex-t;       #REQUIRED
  sat-cmdqual      %hex-t;       #REQUIRED
  sat-destdev      %hex-t;       #REQUIRED
  sat-data         %hex-t;       #IMPLIED
  sat-output       %vdata-t;     #IMPLIED
  sat-encap        %bool;        “false“
  %id-attrs;
>

<!ELEMENT sat-exit EMPTY>
<!ATTLIST sat-exit
sat-cleanbuf       %bool-t;      “false“
sat-outvarlist     CDATA         #IMPLIED
%id-attrs;
>

<!ELEMENT sat-encrypt EMPTY>
<!ATTLIST sat-encrypt
  sat-check        %check-t;     “none”
  sat-crypt        %bool-t;      “false”
  sat-kic          %hex-t;       #IMPLIED
  sat-kid          %hex-t;       #IMPLIED
  sat-inlist       CDATA         #REQUIRED
  sat-out          CDATA         #REQUIRED
  %id-attrs;
>

<!ELEMENT sat-decrypt EMPTY>
<!ATTLIST sat-decrypt
  sat-check        %check-t;     “none”
  sat-crypt        %bool-t;      “false”
  sat-kic          %hex-t;       #IMPLIED
  sat-kid          %hex-t;       #IMPLIED
  sat-mac          %hex-t;       #IMPLIED
  sat-padding      %hex-t        “00”
  sat-in           CDATA         #REQUIRED
  sat-sec-msg      CDATA         #IMPLIED
  sat-outlist      CDATA         #REQUIRED
  %id-attrs;
>

<!ELEMENT sat-plug-in EMPTY>
<!ATTLIST sat-plug-in
  sat-uid          %hex-t;       #REQUIRED
  sat-inlist       CDATA         #IMPLIED
  sat-outlist      CDATA         #IMPLIED
  sat-return       %bool-t;      “true“
```

```
  %id-attrs;
>

<!-- ............... Future S@TML ......................... -->
<!-- ............... Content ............................... -->
<!ENTITY % len   “CDATA“ >   <!-- [0-9]+ for pixels,
                                 [0-9]+“%“ for percentage length -->

<!ENTITY % IAlign “(top|middle|bottom)“ >

<!ELEMENT img EMPTY>
<!ATTLIST img
  alt         %vdata-t;       #REQUIRED
  src         %HREF-t;        #REQUIRED
  localsrc    %vdata-t;       #IMPLIED
  vspace      %len;           “0“
  hspace      %len;           “0“
  align       %IAlign;        “bottom“
  height      %len;           #IMPLIED
  width       %len;           #IMPLIED
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>

<!ELEMENT table (tr)+>
<!ATTLIST table
  title       %vdata-t;       #IMPLIED
  align       CDATA           #IMPLIED
  columns     %num-t;         #REQUIRED
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>

<!ELEMENT tr (td)+>
<!ATTLIST tr
  %id-attrs;
>

<!ELEMENT td (%flow;)*>
<!ATTLIST td
  xml:lang    NMTOKEN         #IMPLIED
  %id-attrs;
>

<!-- ............... The End ............................... -->
```

11 Annex: WML 1.1 FEATURES NOT SUPPORTED [informative]

This Chapter contains an informative list of ignored and unsupported WML 1.1 features.

11.1 Ignored Element Tags and Attributes

The WML 1.1 element tags and attributes given here will be ignored by the DE.

- `<... class="...">`
- `<... xml:lang="...">`
- `<meta>`
- `<card title="...">`
- `<card ordered=...>`
- `<p align=...>`
- `<p mode=...>`
- `<input size=...>`
- `<input tabindex=...>`
- `<select value="...">`
- `<select ivalue="...">`
- `<fieldset title="...">`
- `<em>`
- `<strong>`
- `<i>`
- `<b>`
- `<u>`
- `<big>`
- `<small>`
- `<do type="options">`
- `<do type="delete">`
- `<do type="x-*">`
- `<do type="vnd-*">`
- `<go accept-charset="...">`
- `<img>`
- `<table>`
- `<tr>`
- `<td>`

11.2 Unsupported Element Tags and Attributes

The WML 1.1 element tags and attributes given here may result in an *error* or will be ignored if found by the DE:

- `<card onenterforward="...">`
- `<card onenterbackward="...">`
- `<card ontimer="...">`
- `<template onenterforward="...">`
- `<template onenterbackward="...">`
- `<template ontimer="...">`
- `<timer>`
- `<timer name="...">`
- `<timer value="...">`
- `<select multiple="...">`
- `<onevent type="ontimer">`
- `<onevent type="onenterforward">`
- `<onevent type="onenterbackward">`
- `<setvar name="$(x)">`

- `<postfield name="$(x)">`

12 Annex: ENVIRONMENT VARIABLES IN SBC

All SB environment variables are defined in /SBC/ - this chapter gives their names and a short description of the format. Note, that currently all environment variables are read-only, and most of them have a binary TLV-structure. Some environment variables are optional for the SB implementation (marked with "O" instead of "M" for mandatory).

name	M/O	type	environment variables
sat-env:ICCID	M	binary	ICCID of the SIM Coding as in /GSM11.11/ for EF ICCID.
sat-env:SIMBrowserSupplier	M	binary	Identifier of the SIM browser supplier. Coding as in /SBC/.
sat-env:BrowserVersion	M	binary	Version of the SIM browser Coding: 1 Byte with S@T version number in higher nibble, and the manufacturer release number in lower nibble
sat-env:BrowserProfile	M	binary	List of supported facilities of the browser Coding as in /SBC/ .
sat-env:OperatorName	O	string	Name of the network operator (SIM card issuer) Coding as in /GSM03.38/.
sat-env:TerminalProfile	M	binary	Terminal Profile of the currently used ME Coding as in /GSM11.14/.
sat-env:StatusWord	M	binary	Result of the previous macro Coding in 2 Bytes: `6F`xx=error (see /SBC/), `0000`=no error
sat-env:LocationInformation	O	binary	Location information (see /GSM11.14/, /GSM04.08/) Coding in 7 Bytes without tag and length (see /SBC/ or table in Section 8.5.5).
sat-env:UserType	O	binary	Mode of the user interface Coding: `00`=Beginner, `01`=Advanced, `02`=Expert
sat-env:IMEI	O	binary	IMEI of the ME (see /GSM 11.14/)
sat-env:NMR	O	binary	Network Measurement Results (see /GSM11.14/, /GSM04.08/)

13 History

Document history		
Release	Approved by	Comment
V1.0.0	S@T-TDG #15	First approved release.
V1.0.1	S@T-TDG #16	CR 20001 add sat-plug-in in DTD, new version numbering, CR 20003 list of ignored attributes, minor editorial changes.
V1.0.2	S@T-TDG #17	CR 20004 lexical entities, CR 20005/20007 sat-value in sat-const required but no sat-do-clr attribute, CR 20006 explain access of temporary variables
V1.0.3 (Rel. 2000)	S@T-TDG #18	CR 20002 make document more self-contained, CR 20010 dcs-t for sat-send-sms, CR 20011 constants for encrypt/decrypt/plug-in, CR 20012/20015/20018 adding clarifications, CR 20013 sat-ussddcs attribute, CR 20014 vnd.sat-process type for do, CR 20016 life time of variables, CR 20020 pre-defined URIs, CR 20021 sat-mac attribute for sat-decrypt
V1.0.4	S@T-TDG #19	CR 20022 sat-const identifiers, CR 20023 onpick, CR 20024 input, select, sat-inkey alpha identifer clarifications, history comment in DTD
V1.0.5	S@T-TDG #21-#25	CR 20025 Clarification of <sat-refresh> tag, CR 20026 Clarification of <sat-gen-stk> tag CR 20028 Inconsistency between spec and dtd CR 20029 Binary SMS / Encrypt / Decrypt CR 20030 Format attribute of input CR 20031 Clarification of the input type CR 20033 Clarification of the sat-uid attribute of the <sat-plug-in> tag. CR 20034 Extension of <sat-gen-stk> tag CR 20037 Clarification of <sat-decrypt> and <satencrypt> tags
V.1.0.6	S@T-TDG #30	Editorial changes, minor fixes for publication
V.2.0.0	SIM Alliance S@T Group	Editorial changes and modification on CR for publication June 2004 CR 2004-037 CR 2004-037
V.3.0.0	S@T-TDG	CR Axalto December 2003 #007 Outgoing SMS management CR Axalto December 2004 #014 TAG Service definition for variable in S@TML CR Axalto December 2004 #015 Administrative Plug-in Management cmd CR ORGA-January-2005 #005 Simplification of <sat-decrypt> element CR ORGA-January-2005 #006 Using 'chain next card' attribute from S@TML CR ORGA-January-2005 #007 Explicit specification of the default DCS CR ORGA-January-2005 #012 Globalize usage of <setvar> element CR ORGA-January-2005 #013 Using 'title' attribute of <option> element CR ORGA-January-2005 #014 Allow navigation elements without enclosing <do> CR GEMPLUS-December-2004 #004 IMEI environment variable CR GEMPLUS-December-2004 #005 No wait attributes CR GEMPLUS-December-2004 #006 Plug-ins Variable Types

13.1 Annex: LIST OF CHANGE REQUESTS [informative]

CR Number	CR Identifier	Subject	Document Reference	Status / Meeting No.
20001	Orga-SML-1-04-APR-2000	sat-plug-in missing in DTD	S@T 1.10 V1.0.1	Accepted #16
20002	Orga-SML-2-04-APR-2000	Make SATML document more self-contained	S@T 1.10 V1.0.3	Accepted #18
20003	Orga-WG2-April-2000#3	Correct list of ignored WML attributes	S@T 1.10 V1.0.1	Accepted #17
20004	Orga-WG2-April-2000#4	Add lexical definition for special characters	S@T 1.10 V1.0.2	Accepted #17
20005	Schlumberger-Workgroup2-04-2000#1	Value of a constant should be mandatory	S@T 1.10 V1.0.2	Accepted #17
20006	Schlumberger-Workgroup2-04-2000#2	Scope of a temporary variable	S@T 1.10 V1.0.2	Accepted #17
20007	Schlumberger-Workgroup2-04-2000#3	Scope of a constant	S@T 1.10 V1.0.2	Accepted #17
20010	Giesecke&Devrient-WG2-May-2000#1	Remove sat-dcs and sat-cmdqual from <sat-send-sms>	S@T 1.10 V1.0.3	Accepted #18
20011	Giesecke&Devrient-WG2-May-2000#2	Allow constants for crypto elements and Plug-in	S@T 1.10 V1.0.3	Accepted #18
20012	Giesecke&Devrient-WG2-May-2000#3	Default value of sat-minlength	S@T 1.10 V1.0.3	Accepted #18
20013	Giesecke&Devrient-WG2-May-2000#4	Missing DCS attribute in sat-send-ussd	S@T 1.10 V1.0.3	Accepted #18
20014	Giesecke&Devrient-WG2-May-2000#5	Automatic navigation	S@T 1.10 V1.0.3	Accepted #18
20015	Giesecke&Devrient-WG2-May-2000#6	Clarification for iname attribute of select	S@T 1.10 V1.0.3	Accepted #18
20016	Giesecke&Devrient-WG2-May-2000#7	Context restrictions	S@T 1.10 V1.0.3	Accepted #18
20018	Giesecke&Devrient-WG2-May-2000#8	Clarification for sat-period attribute	S@T 1.10 V1.0.3	Accepted #18
20020	Schlumberger-Workgroup2-31-2000#1	Reservation of predefined names for URLs	S@T 1.10 V1.0.3	Accepted #18
20021	Giesecke&Devrient-WG2-June-2000#1	sat-mac attribute for sat-decrypt	S@T 1.10 V1.0.3	Accepted #18
20022	Schlumberger-Workgroup2-06-2000#1	Declaration of SATML constants	S@T 1.10 V1.0.4	Accepted #19
20023	Schlumberger-Workgroup2-06-2000#2	Place of “onpick” attribute	S@T 1.10 V1.0.4	Accepted #19
20024	Schlumberger-Workgroup2-06-2000#3	Use of alpha-identifiers for “input”, “select” and “sat-inkey”	S@T 1.10 V1.0.4	Accepted #19
20025	Schlumberger-WG2 - September-2000#1	Clarification of <sat-refresh> tag	S@T 1.10 V1.0.5	Accepted #21
20026	Schlumberger-WG2 - September-2000#2	Clarification of <sat-gen-stk> tag	S@T 1.10 V1.0.5	Accepted #21
20028	Schlumberger-WG2 –	Inconsistency between spec and dtd	S@T 1.10	Accepted #21

	September-2000#4		V1.0.5	
20029	Orga-WG2-September-2000#1	Binary SMS / Encrypt / Decrypt	S@T 1.10 V1.0.5	Accepted #22
20030	Schlumberger-WG2 – October-2000#1	Format attribute of input	S@T 1.10 V1.0.5	Accepted #22
20031	Gemplus-WG2-October-2000#1	Clarification of the input type	S@T 1.10 V1.0.5	Accepted #25
20033	Gemplus-WG2-January-2001#2	Clarification of the sat-uid attribute of the <sat-plug-in> tag.	S@T 1.10 V1.0.5	Accepted #25
20034	Schlumberger-WG2-January-2001#1	Extension of <sat-gen-stk> tag.	S@T 1.10 V1.0.5	Accepted #25
20037	Gemplus-WG2-May-2001#1	Clarification of <sat-decrypt> and <satencrypt> tags	S@T 1.10 V1.0.5	Accepted #29
2004-037	XponCard A/S – 02 – 2004, #3	Clarification of "sat-gen-stk"	S@T 1.10 V2.0.0	Accepted (email vote 9th March 2004)
2004-038	XPONCARD A/S – 02 – 2004, #3	Clarification on Statements	S@T 1.10 V2.0.0	Accepted (email vote 9th March 2004)
2006-11	AXALTO DECEMBER 2003 #007	TAG Service definition for variable in S@T ML	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-17	ORGA-JANUARY-2005 #005	Simplification of <sat-decrypt> element	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-18	ORGA-JANUARY-2005 #006	Using 'chain next card' attribute from S@TML	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-20	ORGA-JANUARY-2005 #007	Explicit specification of the default DCS	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-25	ORGA-JANUARY-2005 #012	Globalize usage of <setvar> element	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-26	ORGA-JANUARY-2005 #013	Using 'title' attribute of <option> element	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-27	ORGA-JANUARY-2005 #014	Allow navigation elements without enclosing <do>	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-32	GEMPLUS-DECEMBER-2004 #004	IMEI environment variable	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
2006-33	GEMPLUS-DECEMBER-2004	No wait attributes	S@T 01.10 V3.0.0	Accepted by email voting

	#005			Feb 2006