S@T 01.10 v4.0.0 (Release 2009)

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
TL[A]V	TLV with optional attributes
TLV	Tag Length Value encoding
URI	Unified Resource Identifier
URL	Unified Resource Locator
WAP	Wireless Application Protocol
XML	Extensible Markup Language

3 LIST OF DOCUMENTS

/22.030/	3GPP TS 22.030: "Man-Machine Interface (MMI) of the User Equipment (UE)".
/23.090/	3GPP TS 23.090: "Unstructured Supplementary Service Data (USSD); Stage 2".
/23.038/	3GPP TS 23.038: "Alphabets and language-specific information".
/23.048/	3GPP TS 23.048: "Security mechanisms for the (U)SIM application toolkit; Stage 2".
/23.040/	3GPP TS 23.040: "Technical realization of Short Message Service (SMS) ".
/51.011/	3GPP TS 51.011: "Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface".
/51.014/	3GPP TS 51.014: "Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface".
/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
/XML1.0/	Extensible Markup Language (XML) 1.0

4 OVERVIEW

This is the released version of the S@T Markup Language 4.0 (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 Bytecode (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. 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). *S@TML STK Extensions* include `satml` element and all elements and attributes tagged using the "`sat-`" prefix. For the relation between the mentioned language subsets see Figure 1.

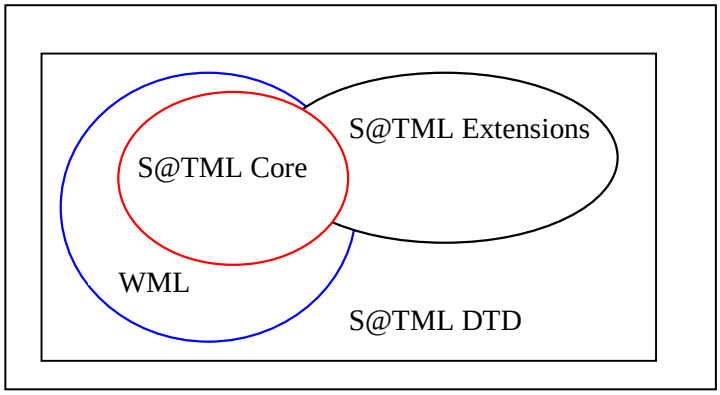

Figure 1

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. 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 element and attributes belonging to the S@TML STK Extensions. The extensions are tagged with "`sat-`".

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 7. This document type definition is a superset of the WML 1.1 DTD. I.e. it includes S@TML Core Language and the not supported part of WML.

5 Basic Attribute Types

The following basic types are used for S@TML attributes and element content specified in Chapter 6.

type	Description
bool	Boolean: `true | false`
string	String consisting of any legal XML character. It may contain hexadecimal strings that will be used by DE "as is"(i.e. without character encoding). The special syntax (similar to variable substitution) is used for this: `$(sat-hex:hex)` (e.g. "`hello$(sat-hex:303030)`"). A sequence of two dollar signs '`$$`' represents a single dollar sign character.
string with vars	Extension of the "string" type. Variable references can be used (as defined in 5.1.1)
hex	Even number of hexadecimal digits (i.e. an even-length string matching `[A-Fa-f0-9]+`) after having all white space characters removed.
hex with vars	Extension of the "hex" type. Variable references can be used (as defined in 5.1.1)
num	Decimal number (i.e. `[0-9]+`)
var name	Variable name (identifier). Variable name shall not be started with "`sat-hex:`" prefix.
var list	Comma separated list of variable names (e.g. `"var1,var2,var3"`)
input list	Comma separated list of input values given by variable references or strings (e.g. `"const text,$(var1),$(var2)"`)
dial number	Telephone number. All valid telephony number characters and digits may be used ('`0`'-'`9`', '`+`', '`*`', '`#`').
string URI	URI of some S@TML (or WML) page as defined in /WML1.1/ and /RFC2396/. URIs starting with prefix "sim:" are used to access resident decks stored in the SIM card All URIs starting with "sim:/s/" are reserved for S@T.

name	pre-defined URIs
`sim:/s/home`	Access the starting deck (home page) of the SB
`sim:/s/bookmarks`	Go to the bookmark list

5.1 Variables and Text Constants

5.1.1 Variable Substitution

To substitute a variable or text constant 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.

5.1.2 Temporary Variables

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.

The SAT browser terminates the current context on exit.

5.1.3 Environment Variables

Environment variable identifies have a prefix "`sat-env:`" (e.g. "`sat-env:BrowserVersion`"). Environment variables are read-only. The existing environment variables and their formats are defined in /SBC/, see also the table provided in Section 8.

5.1.4 Text Constants

Text constant identifiers have a prefix "sat-const:" (e.g. "`sat-const:x`"). Text constants are read-only. All constants referred to inside a deck must have been declared in this deck.

6 S@TML

The following section describes only elements (including child elements) and attributes relevant to S@T DE. Elements and attributes ignored during converting S@TML to S@T bytecode are not included into this section (see complete DTD in Chapter 7).

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 prolog

```
prolog ::= xml-decl?

   ( '<!DOCTYPE satml SYSTEM "http://www.simalliance.org/DTD/satml400.dtd">'

   | '<!DOCTYPE wml   SYSTEM "http://www.simalliance.org/DTD/satml400.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.

6.1.3 satml

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

Possible child elements:

`head, sat-const, template, card`

Attributes:

attribute	type	default	<satml>
`sat-storage`	`dynamic | static`	`static`	Specifies the storage type of this deck. The content is either dynamic (should not be re-read from cache), or static.
`sat-dcs`	`sms| ucs2 | auto`	`auto`	Specifies the default DCS of the deck. If '`auto`' then usual content analysis procedures apply.

6.1.4 wml

The `wml` element can be used to enclose a complete deck of cards, declarations, and deck-level attributes. The `wml` element is a root element.

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/).

Possible child elements:

`head, template, card`

Attributes:

attribute	type	default	<wml>
`sat-storage`	`dynamic | static`	`static`	Specifies the storage type of this deck. The content is either dynamic (should not be re-read from cache), or static.
`sat-dcs`	`sms| ucs2 | auto`	`auto`	Specifies the default DCS of the deck. If '`auto`' then usual content analysis procedures apply.

6.1.5 head

The `head` element contains deck-level administrative information.

Possible child elements:

`access, meta`

6.1.6 access

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.

Possible child elements:

None

Attributes:

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`.

6.1.7 meta

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

Possible child elements:

None

6.2 Declarations

6.2.1 template

The `template` element contains set of elements which will be inherited to any card inside the current deck (unless `sat-use-template` `card` attribute is `false`). Usually this element is used for contextual menu management in the deck context.

Possible child elements:

`do, setvar`

6.2.2 sat-const

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.

Possible child elements:

None

Attributes:

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

6.3 Cards

6.3.1 card

The `card` element defines the smallest navigation unit for SB.

Possible child elements:

`do, p, setvar, go, prev, refresh, 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, sat-switch`

Attributes:

attribute	type	default	<card>
`id`	string	`null`	Card Id for local navigation within the deck.
`newcontext`	bool	`false`	The browser context is re-set when this card is entered i.e. temporal variables will be cleared.
`sat-use-template`	bool	`true`	Specifies if the card inherits template: `false` – do not use template; `true` – use template.
`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.

6.4 Fields

6.4.1 p

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

Possible child elements:

string with vars, `br`, `input`, `select`, `sat-play-tone`, `sat-inkey`, `do`, `go`, `anchor`, `a`, `fieldset`

Attributes:

attribute	type	default	<p>
`sat-auto-clr`	bool	`false`	If set to `true` text strings in this paragraph will be cleared automatically after a delay.
`sat-prio`	`normal | high`	`normal`	Specifies the priority to be used to display text strings in this paragraph.

6.4.2 input

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

Possible child elements:

None

Attributes:

attribute	type	default	<input>
`title`	string with vars	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`	var name		Specifies the variable to be assigned. It is an error to use read-only variables here.
`type`	`text | password`	`text`	Specifies whether the input is visible while entered or not. If type equals "`password`", only characters '0'-'9','*','#' can be entered.
`value`	string with vars	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.
`maxlength`	num	255	Specifies the maximal number of input characters accepted. Default is 255.
`sat-minlength`	num		Specifies the minimal length of input to be accepted. If given the attribute `emptyok` is ignored.
`sat-dcs`	`sms | ucs2 | auto`	`auto`	Specifies expected DCS for the input. If attribute is '`auto`', then DSC will be inherited from the deck DCS.

6.4.3 select

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.

Possible child elements:

`option, optgroup`

Attributes:

attribute	type	default	<select>
`title`	string with vars	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`	var name	null	Specifies the variable to be assigned the selected option's `value`. It is an error to use read-only variables here.
`iname`	var name	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 assignment, represented in decimal system. Index numbering increases monotonically. It is an error to use read-only variables here.

6.4.4 option

The `option` element contains one option field of a `select` or `optgroup` element. Option title is taken either from the element enclosed text or `title` attribute.

Possible child elements:

string with vars

Attributes:

attribute	type	default	<option>
`title`	string with vars	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 with vars	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-no-wait`	bool	`false`	Specifies if SB shall enter waiting for response state: `false` – enter waiting for response state; `true` – do not enter waiting for response state.
`sat-dcs`	`sms | ucs2 | binary | auto`	`auto`	Specifies DCS for value text. If attribute is '`auto`', then DSC will be inherited from the deck DCS. If attribute is '`binary`', then value shall be coded as a hex string.

6.4.5 optgroup

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

Possible child elements:

`optgroup, option`

Attributes:

attribute	type	default	<optgroup>
`title`	string with vars	null	Title which may be used in the presentation of this group of options by DE.

6.4.6 fieldset

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

Possible child elements:

string with vars, `br`, `input`, `select`, `sat-play-tone`, `sat-inkey`, `do`, `go`, `anchor`, `a`, `fieldset`

Attributes:

attribute	type	default	<fieldset>
`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`	`normal | high`	`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>`.

6.4.7 sat-play-tone

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-play-tone>
`sat-title`	string with vars	null	Title which may be displayed while the command is executed.
`sat-tone`	`dial | busy | congestion | radio-ack | radio-gone | error | call-wait | ring | beep | ack | nack`	`beep`	Specifies the tone to be played.
`sat-duration`	num		Specifies the length of the tone in $1/10^{th}$ of seconds. If not given a handset manufacturer default applies.

6.4.8 sat-inkey

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-inkey>
`sat-title`	string with vars	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`	var name		Specifies the variable to be assigned.
`sat-format`	`M | 1M | Y | 1Y | N | 1N`	`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.

6.5 Content

6.5.1 br

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

Possible child elements:

None

6.6 Navigation

6.6.1 anchor

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

Possible child elements:

string with vars, `br`, `go`, `prev`, `refresh`

Attributes:

attribute	type	default	<anchor>
`title`	string with vars	null	Title which may be used in the presentation of this hyper-link by DE.

6.6.2 a

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>".`

Possible child elements:

string with vars, `br`

Attributes:

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 with vars	null	Title which may be used in the presentation of this hyper-link by DE.
`sat-no-wait`	bool	`false`	Specifies if SB shall enter waiting for response state: `false` – enter waiting for response state; `true` – do not enter waiting for response state.

6.6.3 do

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. 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.

Possible child elements:

`go, prev, noop, refresh, sat-switch`

Attributes:

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 `reset` - exit 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 with vars	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.

6.6.4 go

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

Possible child elements:

`setvar, postfield`

Attributes:

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.
`sat-no-wait`	bool	`false`	Specifies if SB shall enter waiting for response state: `false` – enter waiting for response state; `true` – do not enter waiting for response state.
`sendreferer`	bool	`false`	Specifies if the referring URI address will be sent to the server, e.g. for checking of access rights.
`method`	`post | get`	`get`	Specifies the method to be used for fetching the contents.

6.6.5 prev

The `prev` element specifies that backward history navigation should be performed by the SB.

Possible child elements:

`setvar`

6.6.6 refresh

When `refresh` specifies that the current card should be re-displayed by the SB.

Possible child elements:

`setvar`

6.6.7 noop

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

Possible child elements:

None

6.6.8 sat-switch

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.

Possible child elements:

`sat-case`

Attributes:

attribute	type	default	<sat-switch>
`sat-name`	var name		Specifies the sat-switch variable name
`sat-defaulturl`	string URI	null	If all sat-case values don't match the sat-switch variable value, the SB will branch to this URL or continue deck execution if this attribute is not set.
`sat-casesensitive`	bool	`false`	Specifies if the case is sensitive when compare `sat-switch` variable value with `sat-case` value

6.6.9 sat-case

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-case>
`sat-value`	string with vars		Specifies the value to be compared by the sat-switch element
`sat-href`	string URI		Specifies the URI of this case
`sat-dcs`	`sms | ucs2 | binary | auto`	`auto`	Specifies DCS for `sat-value` text. If attribute is '`auto`', then DSC will be inherited from the deck DCS. If attribute is '`binary`', then `sat-value` shall be coded as a hex string.

6.6.10 sat-exit

The `sat-exit` element specifies the SB should terminate the execution of the browsing session.

Possible child elements:

None

6.7 Statements

6.7.1 setvar

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

This element can be used to assign special variable Id with the given variable name. In this case `value` attribute can be omitted. For example: `<setvar name="var1" sat-var-id="F1"/>`.

Possible child elements:

None

Attributes:

attribute	type	default	<setvar>
`name`	var name		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 with vars		Specifies the value to be assigned to this variable.
`sat-var-id`	hex	null	Specifies the variable Id. If this attribute is not used, variable Id will be calculated by DE. Variable Id coding: '00' – 'FF'
`sat-dcs`	`sms | ucs2 | binary | auto`	`auto`	Specifies DCS for `value` text. If attribute is '`auto`', then DSC will be inherited from the deck DCS. If attribute is '`binary`', then `value` shall be coded as a hex string.

6.7.2 postfield

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

Possible child elements:

None

Attributes:

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 with vars		Specifies the value to be sent to the server in this field.
`sat-dcs`	`sms | ucs2 | binary | auto`	`auto`	Specifies DCS for `value` text. If attribute is '`auto`', then DSC will be inherited from the deck DCS. If attribute is '`binary`', then `value` shall be coded as a hex string.

6.7.3 sat-extract

The `sat-extract` element gets 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.

Possible child elements:

None

Attributes:

attribute	type	default	<sat-extract>
`sat-destvar`	var name		Specifies the name of the destination variable. It is an error to use read-only variable here.
`sat-srcvar`	var name		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

6.7.4 sat-encrypt

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 0x00 characters will be added if the total length of the data block is not a multiple of 8 bytes.
3. If requested, MAC calculation will be done on the data block value (i.e. the LV series) and padding. The MAC will be inserted at the beginning of the data block
4. If requested the MAC value, data block value and padding will be encrypted by SB. If encryption is not applied, padding will be removed after MAC calculation.
5. The result will be stored in the secure message format defined in /SBC/. The SPI, KIc/KID and padding counter of this structure are used to find the original data block value (series of LVs) on the content provider side.

The maximum supported length for the clear data block (series 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 as defined in /SBC/.

Possible child elements:

None

Attributes:

attribute	type	default	<sat-encrypt>
`sat-check`	`none | mac`	`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 /23.048/ and written in 2 hex chars.
`sat-kid`	hex	`00`	Gives the key index and algorithm used for the checking as specified in /23.048/ and written in 2 hex chars.
`sat-inlist`	input list		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`	var name		Gives the name of the output variable. It is an error to specify a read-only variable here.

6.7.5 sat-decrypt

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

Attribute `sat-outlist` should 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 (series 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 as defined in /SBC/.

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.

Possible child elements:

None

Attributes:

Attribute	type	default	<sat-decrypt>
`sat-check`	`none | mac`	`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 /23.048/ and written in 2 hex chars.
`sat-kid`	hex	`00`	Gives the key index and algorithm used for the checking as specified in /23.048/ 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 with vars		Data block (LV sequence optionally padded with 0x00) in enciphered or cleartext format as defined in /SBC/. Data block can be given by a variable reference or a constant string. Typical usage:
			`sat-in = "$(sat-const:my_input)"`
			`sat-in = "45 56 AA BB CC DD EE FF"`

`sat-sec-msg`	string with vars		Input data in Secure Message format (Secure Message TL[A]V as defined in /SBC/). Secure Message can be given by a variable reference or a constant string.
`sat-outlist`	var list		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"`
`sat-dcs`	`sms | ucs2 | binary`	`binary`	Specifies DCS of the encrypted values. This DCS will be inherited by the output variables.

6.8 STK Commands

6.8.1 sat-send-sms

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.

Possible child elements:

string with vars

Attributes:

attribute	type	default	<sat-send-sms>
`sat-title`	string with vars	null	Title which may be displayed as alpha identifier while the command is executed.
`sat-dest`	dial number		Specifies the destination address telephone number.
`sat-smsc`	dial number	null	Specifies the Short Message Service Centre (SMSC) address. 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 /51.014/, /23.040/. 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 /51.014/, /23.040/.
`sat-dcs`	`sms | ucs2 | binary`	`sms`	Specifies the data coding scheme to be used for the short message to be SMS default charset, UCS2, or binary (see /23.038/). The 51.014 command qualifier will be set accordingly by the DE. If `sat-dcs` is `binary` the enclosed text will denote the short message user data by a sequence of hexadecimal digits.

6.8.2 sat-setup-call

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-setup-call>
`sat-confirm`	string with vars	null	A string to be displayed for user confirmation.
`sat-title`	string with vars	null	The string which is displayed in the call set-up phase.
`sat-dest`	dial number		Specifies the destination address telephone number. A DTMF sequence can be added at the end of the string using a ",“ separator. Example: “`0660773534,1234`”.
`sat-cmdqual`	hex	`00`	Specifies the 51.014 command qualifier to be used in two hex digits (see § 12.6 in /51.014/): `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

6.8.3 sat-send-ussd

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-send-ussd>
`sat-title`	string with vars	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 /23.038/
`sat-data`	hex	null	Specifies the USSD string to be sent as defined in /22.030/, /23.090/ encoded in hex characters.
`sat-name`	var name	null	Gives the name of the output variable. The text retuned on SEND USSD command will be written to the output variable. It is an error to use read-only variable here.

6.8.4 sat-local-info

The `sat-local-info` element provides the local information data specified by the command qualifier. The result is stored in the output variable.

Possible child elements:

None

Attributes:

attribute	type	default	<sat-local-info>
`sat-name`	var name	null	The output variable containing the result of the local info operation element. It is TLV-formatted as described in the /51.014/ location information object, and returned in a binary string. It is an error to use read-only variable here.
`sat-cmdqual`	hex	`00`	Specifies the /51.014/ 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

6.8.5 sat-refresh

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

Possible child elements:

None

Attributes:

attribute	type	default	<sat-refresh>
`sat-files`	hex	null	Specifies the list of file change notifications as specified in /51.014/ encoded in hex characters, the number of files and Files.
`sat-cmdqual`	hex	`01`	Specifies the /51.014/ 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

6.8.6 sat-gen-stk

The `sat-gen-stk` element allows to specify a generic /51.014/ SIM Toolkit command to be sent to the ME.

Possible child elements:

None

Attributes:

attribute	type	default	<sat-gen-stk>
`sat-cmdtype`	hex		Specifies the type of command value as defined in § 12.6 of /51.014/ 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 /51.014/ and given in 2 hexadecimal characters.
`sat-destdev`	hex		Specifies the destination device identity to be used as defined in § 12.7 of /51.014/ 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 /51.014/ and given in hexadecimal characters. Command details and device identities objects must not be included.
`sat-output`	var name		Specifies the variable where to store the proactive command specific response data object, as described in /SBC/. It is an error to use read-only variable here.
`sat-encap`	bool	`false`	Specifies whether the response data object must be encapsulated as described in /SBC/.

6.9 Plug-ins

6.9.1 sat-plug-in

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

Possible child elements:

None

Attributes:

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`	input list	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`	var list	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"`

7 COMPLETE DTD

```
<!--
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/satml400.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.
satml400.dtd V4.0.0 2009    – Removed: sat-var, sat-sps, sat-help, sat-serv-id,
no-wait, sat-cleanbuf, sat-outvarlist, sat-return, sat-href (sat-local-info),
sat-method; For "sat-" tagged elements removed: xml:lang, id, class;
Added: sat-dcs (satml, wml, setvar, postfield, option, sat-case, input, sat-
decrypt), sat-no-wait (go, option, a), sat-var-id (sat-var-id), sat-name (sat-
send-ussd), sat-use-template (card); Added as a possible child: sat-switch in
card, go in p, setvar in template.
-->
<!-- ............... Entities ............................. -->
<!-- quotation mark -->
<!ENTITY quot "&#34;">
<!-- ampersand -->
<!ENTITY amp "&#38;#38;">
<!-- apostrophe -->
<!ENTITY apos "&#39;">
<!-- less than -->
<!ENTITY lt "&#38;#60;">
<!-- greater than -->
<!ENTITY gt "&#62;">
<!-- non-breaking space -->
<!ENTITY nbsp " ">
<!-- soft hyphen (discretionary hyphen) -->
<!ENTITY shy "&#173;">
<!-- boolean -->
<!ENTITY % bool-t "(true|false)">
<!-- number -->
<!ENTITY % num-t "NMTOKEN">
<!-- hex str -->
<!ENTITY % hex-t "NMTOKEN">
<!-- string withiout variables -->
<!ENTITY % data-t "CDATA">
<!-- string with variables -->
<!ENTITY % vdata-t "CDATA">
<!-- hex str with variables -->
<!ENTITY % vhex-t "%vdata-t;">
<!-- URI, URL or URN with variables -->
<!ENTITY % HREF-t "%vdata-t;">
<!-- variable name -->
```

```
<!ENTITY % varname-t "NMTOKEN">
<!-- comma separated list of variable names (e.g. "var1,var2,var3")-->
<!ENTITY % varlist-t "CDATA">
<!-- comma separated list of input values given by variable references or
constant strings (e.g. "const text,$(var1),$(var2)")-->
<!ENTITY % inputlist-t "CDATA">
<!-- Dial number (e.g. +12345)-->
<!ENTITY % dialnumber-t "CDATA">
<!ENTITY % id-attr "
    id ID #IMPLIED">
<!ENTITY % class-attr "
    class CDATA #IMPLIED">
<!ENTITY % id-attrs "
    %id-attr;
    %class-attr;">
<!-- ............... Decks ............................... -->
<!ENTITY % decls "sat-const*, template?">
<!ELEMENT satml (head?, %decls;, card+)>
<!ATTLIST satml
    sat-storage (dynamic | static) "static"
    sat-dcs (sms | ucs2 | auto) "auto"
    %id-attr;
>
<!ELEMENT wml (head?, template?, card+)>
<!ATTLIST wml
    sat-storage (dynamic | static) "static"
    sat-dcs (sms | ucs2 | auto) "auto"
    %id-attr;
>
<!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;
>
<!-- ............... Declarations ........................ -->
<!ELEMENT template (do | onevent | setvar)*>
<!ELEMENT sat-const EMPTY>
<!ATTLIST sat-const
    sat-name %varname-t; #REQUIRED
    sat-value %data-t; #REQUIRED
>
<!-- ............... Cards ............................... -->
<!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 | sat-switch">
<!ELEMENT card (onevent*, timer?, (do | p | setvar | go | prev | refresh |
%satstmt;)*)>
<!ATTLIST card
    newcontext %bool-t; "false"
```

```
    sat-history %bool-t; "true"
    sat-chain-next %bool-t; "false"
    sat-use-template %bool-t; "true"
    %id-attr;
>
<!-- ............... 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; ">
<!ELEMENT p (%fields; | do | go)*>
<!ATTLIST p
    sat-auto-clr %bool-t; "false"
    sat-prio (normal | high) "normal"
>
<!ELEMENT input EMPTY>
<!ATTLIST input
    title %vdata-t; #IMPLIED
    name %varname-t; #REQUIRED
    type (text | password) "text"
    value %vdata-t; #IMPLIED
    format CDATA "*M"
    emptyok %bool-t; "false"
    maxlength %num-t; "255"
    sat-minlength %num-t; #IMPLIED
    sat-dcs (sms | ucs2 | auto) "auto"
>
<!ELEMENT select (optgroup | option)+>
<!ATTLIST select
    title %vdata-t; #IMPLIED
    name %varname-t; #IMPLIED
    iname %varname-t; #IMPLIED
>
<!ELEMENT option (#PCDATA | onevent)*>
<!ATTLIST option
    title %vdata-t; #IMPLIED
    value %vdata-t; #IMPLIED
    onpick %HREF-t; #IMPLIED
    sat-no-wait %bool-t; "false"
    sat-dcs (sms | ucs2 | binary | auto) "auto"
>
<!ELEMENT optgroup (optgroup | option)+>
<!ATTLIST optgroup
    title %vdata-t; #IMPLIED
>
<!ELEMENT fieldset (%fields; | do | go)*>
<!ATTLIST fieldset
    sat-auto-clr %bool-t; "false"
    sat-prio (normal | high) "normal"
>
<!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
>
<!ELEMENT sat-inkey EMPTY>
<!ATTLIST sat-inkey
```

```
    sat-title %vdata-t; #IMPLIED
    sat-name %varname-t; #REQUIRED
    sat-format (M | 1M | Y | 1Y | N | 1N) "M"
>
<!-- ............... Content ............................. -->
<!ELEMENT br EMPTY>
<!-- ............... Navigation ........................... -->
<!ELEMENT anchor (#PCDATA | br | img | go | prev | refresh)*>
<!ATTLIST anchor
    title %vdata-t; #IMPLIED
>
<!ELEMENT a (#PCDATA | br | img)*>
<!ATTLIST a
    href %HREF-t; #REQUIRED
    sat-no-wait %bool-t; "false"
    title %vdata-t; #IMPLIED
>
<!ENTITY % task "go | prev | noop | refresh | sat-switch">
<!ELEMENT do (%task;)>
<!ATTLIST do
    type CDATA #REQUIRED
    label %vdata-t; #IMPLIED
    name NMTOKEN #IMPLIED
>
<!ELEMENT go (postfield | setvar)*>
<!ATTLIST go
    href %HREF-t; #REQUIRED
    sat-no-wait %bool-t; "false"
    sendreferer %bool-t; "false"
    method (post | get) "get"
>
<!ELEMENT prev (setvar)*>
<!ELEMENT refresh (setvar)*>
<!ELEMENT noop EMPTY>
<!ELEMENT sat-switch (sat-case)+>
<!ATTLIST sat-switch
    sat-name %varname-t; #REQUIRED
    sat-defaulturl %HREF-t; #IMPLIED
    sat-casesensitive %bool-t; "false"
>
<!ELEMENT sat-case EMPTY>
<!ATTLIST sat-case
    sat-value %vdata-t; #REQUIRED
    sat-href %HREF-t; #REQUIRED
    sat-dcs (sms | ucs2 | binary | auto) "auto"
>
<!ELEMENT sat-exit EMPTY>
<!-- ............... Statements ........................... -->
<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
    name %varname-t; #REQUIRED
    value %vdata-t; #IMPLIED
    sat-var-id %hex-t; #IMPLIED
    sat-dcs (sms | ucs2 | binary | auto) "auto"
>
<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
    name NMTOKEN #REQUIRED
    value %vdata-t; #REQUIRED
    sat-dcs (sms | ucs2 | binary | auto) "auto"
>
<!ELEMENT sat-extract EMPTY>
<!ATTLIST sat-extract
```

```
    sat-destvar %varname-t; #REQUIRED
    sat-srcvar %varname-t; #REQUIRED
    sat-startindex %num-t; "0"
    sat-length %num-t; #REQUIRED
>
<!ENTITY % crypto-attrs "
    sat-check (none | mac) 'none'
    sat-crypt %bool-t; 'false'
    sat-kic %hex-t; '00'
    sat-kid %hex-t; '00'">
<!ELEMENT sat-encrypt EMPTY>
<!ATTLIST sat-encrypt
    %crypto-attrs;
    sat-inlist %inputlist-t; #REQUIRED
    sat-out %varname-t; #REQUIRED
>
<!ELEMENT sat-decrypt EMPTY>
<!ATTLIST sat-decrypt
    %crypto-attrs;
    sat-mac %hex-t; #IMPLIED
    sat-padding %hex-t;  "00"
    sat-in CDATA #REQUIRED
    sat-sec-msg CDATA #IMPLIED
    sat-outlist %varlist-t; #REQUIRED
    sat-dcs (sms | ucs2 | binary)  "binary"
>
<!-- ............... STK Commands ........................... -->
<!ELEMENT sat-send-sms (#PCDATA)>
<!ATTLIST sat-send-sms
    sat-title %vdata-t; #IMPLIED
    sat-dest %dialnumber-t; #REQUIRED
    sat-smsc %dialnumber-t; #IMPLIED
    sat-period %num-t; #IMPLIED
    sat-pid %hex-t; "00"
    sat-dcs (sms | ucs2 | binary) "sms"
>
<!ELEMENT sat-setup-call EMPTY>
<!ATTLIST sat-setup-call
    sat-confirm %vdata-t; #IMPLIED
    sat-title %vdata-t; #IMPLIED
    sat-dest %dialnumber-t; #REQUIRED
    sat-cmdqual %hex-t; "00"
>
<!ELEMENT sat-send-ussd EMPTY>
<!ATTLIST sat-send-ussd
    sat-title %vdata-t; #IMPLIED
    sat-ussddcs %hex-t; #REQUIRED
    sat-data %hex-t; #REQUIRED
    sat-name %varname-t; #IMPLIED
>
<!ELEMENT sat-local-info EMPTY>
<!ATTLIST sat-local-info
    sat-name %varname-t; #REQUIRED
    sat-cmdqual %hex-t; "00"
>
<!ELEMENT sat-refresh EMPTY>
<!ATTLIST sat-refresh
    sat-files %hex-t; #IMPLIED
    sat-cmdqual %hex-t; "01"
>
<!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 %varname-t; #IMPLIED
    sat-encap %bool-t; "false"
>
<!-- ............... Plug-ins ............................ -->
<!ELEMENT sat-plug-in EMPTY>
<!ATTLIST sat-plug-in
    sat-uid %hex-t; #REQUIRED
    sat-inlist %inputlist-t; #IMPLIED
    sat-outlist %varlist-t; #IMPLIED
>
<!-- ....................................................
    The following section contains attributes and elements
    that are ignored during converting S@TML to S@T bytecode-->
<!-- ....................................................
     ............... Ignored attributes ...................
     .................................................... -->
<!-- ................ Decks ............................... -->
<!ATTLIST satml
    xml:lang NMTOKEN #IMPLIED
    %class-attr;
>
<!ATTLIST wml
    xml:lang NMTOKEN #IMPLIED
    %class-attr;
>
<!-- ................ Cards ............................... -->
<!ENTITY % cardev-attrs "
    onenterforward %HREF-t; #IMPLIED
    onenterbackward %HREF-t; #IMPLIED
    ontimer %HREF-t; #IMPLIED">
<!ATTLIST card
    title %vdata-t; #IMPLIED
    ordered %bool-t; "true"
    xml:lang NMTOKEN #IMPLIED
    %cardev-attrs;
    %class-attr;
>
<!ATTLIST template
    %cardev-attrs;
    %id-attrs;
>
<!-- ................ Fields .............................. -->
<!ENTITY % TAlign "(left|right|center)">
<!ENTITY % WrapMode "(wrap|nowrap)">
<!ATTLIST p
    align %TAlign; "left"
    mode %WrapMode; #IMPLIED
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST input
    size %num-t; #IMPLIED
    tabindex %num-t; #IMPLIED
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST select
    value %vdata-t; #IMPLIED
    ivalue %vdata-t; #IMPLIED
    multiple %bool-t; "false"
```

```
    tabindex %num-t; #IMPLIED
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST option
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST optgroup
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST fieldset
    title %vdata-t; #IMPLIED
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!-- ............... Content ............................. -->
<!ATTLIST br
    %id-attrs;
>
<!-- ............... Navigation ........................... -->
<!ATTLIST anchor
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST a
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST do
    optional %bool-t; "false"
    xml:lang NMTOKEN #IMPLIED
    %id-attrs;
>
<!ATTLIST go
    accept-charset CDATA #IMPLIED
    %id-attrs;
>
<!ATTLIST prev
    %id-attrs;
>
<!ATTLIST refresh
    %id-attrs;
>
<!ATTLIST noop
    %id-attrs;
>
<!-- ............... Statements ........................... -->
<!ATTLIST setvar
    %id-attrs;
>
<!ATTLIST postfield
    %id-attrs;
>
<!-- ....................................................
     ............... Ignored elements ....................
     .................................................... -->
<!ELEMENT timer EMPTY>
<!ATTLIST timer
    name NMTOKEN #IMPLIED
    value %vdata-t; #REQUIRED
    %id-attrs;
```

```
>
<!ELEMENT onevent (%task;)>
<!ATTLIST onevent
    type CDATA #REQUIRED
    %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;
>
<!ENTITY % len "CDATA">
<!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 ............................... -->
```

8 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 /SBC/
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 as in /SBC/
sat-env:BrowserProfile	M	binary	List of supported facilities of the browser Coding as in /SBC/
sat-env:TerminalProfile	M	binary	Terminal Profile of the currently used ME Coding as in /SBC/
sat-env:StatusWord	M	binary	Result of the previous S@T bytecode command Coding as in /SBC/
sat-env:LocationInformation	O	binary	Location information Coding as in /SBC/
sat-env:IMEI	O	binary	IMEI of the ME Coding as in /SBC/
sat-env:NMR	O	binary	Network Measurement Results Coding as in /SBC/
sat-env:ResidentDeckBufferSize	M	binary	Resident Decks Buffer Size Coding as in /SBC/
sat-env:ResidentDeckList	M	binary	Resident Deck List Coding as in /SBC/
sat-env:PluginList	M	binary	Plug-in List Coding as in /SBC/
sat-env:ReceptionBufferSize	M	binary	Reception Buffer Size Coding as in /SBC/

9 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
V.3.0.1	S@T-TDG	CR PRISM-Feb-2008 #002 integrated.

4.0.0	SIM Alliance TDG	S@T Task Force: Review of Spec
		
		- Sections “S@TML TELEPHONY”, “WTAI” removed
		- Environmental variables added: `ResidentDeckBufferSize, ResidentDeckList, PluginList, ReceptionBufferSize`
		- Added `$(sat-hex:hex)` syntax for the “string” type
		- DTD change:
		  - Version changed
		  - Ignored attributes and elements moved to the last section of DTD
		  - Element `sat-var` removed
		  - Element `sat-sps` removed
		  - Attribute `sat-help` removed from all possible elements
		  - Attributes `xml:lang`, `id` and `class` removed from all element tagged with “`sat-`” prefix
		  - Element `satml/wml`:
		  - Attribute `sat-dcs` added
		  - Attribute `sat-serv-id` removed
		  - Element `input`:
		  - Attribute `sat-dcs` added
		  - Elements `go`, `option`, `a`:
		  - Attribute `no-wait` removed
		  - Attribute `sat-no-wait` added
		  - Element `setvar`:
		  - Attribute `sat-dcs` added
		  - Attribute `sat-var-id` added
		  - Element `postfield`:
		  - Attribute `sat-dcs` added
		  - Element `sat-exit`:
		  - Attribute `sat-cleanbuf` removed
		  - Attribute `sat-outvarlist` removed
		  - Element `sat-plug-in`:
		  - Attribute `sat-return` removed
		  - Element `sat-send-ussd`:
		  - Attribute `sat-name` added
		  - Element `sat-local-info`:
		  - Attribute `sat-href` removed
		  - Attribute `sat-method` removed
		  - Elements `option`, `sat-case`:
		  - Attribute `sat-dcs` added
		  - Element `sat-decrypt`:
		  - Attribute `sat-dcs` added
		  - Element `card`:
		  - Attribute `sat-use-template` added
		  - Element `sat-switch` added as possible child element of the `card` element
		  - Element `go` added as a possible child element of the `p` element
		  - Element `setvar` added as a possible child element of the `template` element

9.1 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 -	**Clarification of <sat-gen-stk> tag**	S@T 1.10	Accepted #21

	September-2000#2		V1.0.5	
20028	Schlumberger-WG2 – September-2000#4	**Inconsistency between spec and dtd**	S@T 1.10 V1.0.5	Accepted #21
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 <sat-encrypt> 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 #005	**No wait attributes**	S@T 01.10 V3.0.0	Accepted by email voting Feb 2006
	PRISM-Feb-2008 #002	Clarifications for certification	S@T 1.10 V3.0.1	Accepted at WG meeting Feb 2008