S@T 01.00 v3.0.0 (Release 2007)

S@T Bytecode
SBC

Published by simalliance now Trusted Connectivity Alliance

TABLE OF CONTENTS

1 TERMINOLOGY

1.1 Abbreviations

HTTP	Hyper Text Transfer Protocol
S@T	SIM Alliance Toolbox
SBC	S@T Byte Code
SSP	S@T Session Protocol
S@TML	S@T Markup Language
STK	SIM Application Toolkit
STLS	S@T Transport Layer Security
TLV	Tag Length Value encoding
URL	Unified Resource Locator

2 LIST OF DOCUMENTS

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

/GSM 11.11/ GSM 11.11. "Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

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

/GSM 03.48/ GSM 03.48. "Digital cellular telecommunications system (Phase 2+); Security Mechanisms for the SIM Application Toolkit; Stage 2".

/SSP/ SSP, S@T Session Protocol (Technical Specification S@T 01.20)

/Admin/ S@T Administration Commands (Technical Specification S@T 01.21)

/Operational/ S@T Operational Commands (Technical Specification S@T 01.22)

/Push/ S@T Push Commands (Technical Specification S@T 01.23)

This document is part of a specification set, please refer to "S@T Release Note" for a comprehensive document list, including document versions.

3 OVERVIEW

This document describes the S@T byte code that is transported between the gateway and the browser. This byte code aims to describe applications running on a distributed system. It is executed on the client side (browser).

The byte code is focused on SIM Toolkit commands description used to perform user interaction with the Mobile Equipment.

It is containing also macros used to perform additional local processing or branching.

The objectives are :

- To have a compact byte code that will use efficiently the SMS (Short Messages Services) bandwidth.
- To keep the browser complexity as low as possible to minimise the code size in the SIM.

This byte code is inheriting some WML concepts like the deck and card organisation to match the WML mediation needs.

Some trade-offs have been made between the byte code efficiency and its ease of parsing. It has been taken into account the evolution too, in aim to allow an easy extension or an incomplete implementation for non mandatory elements.

4 GENERIC SIMPLE-TL[A]V FORMAT

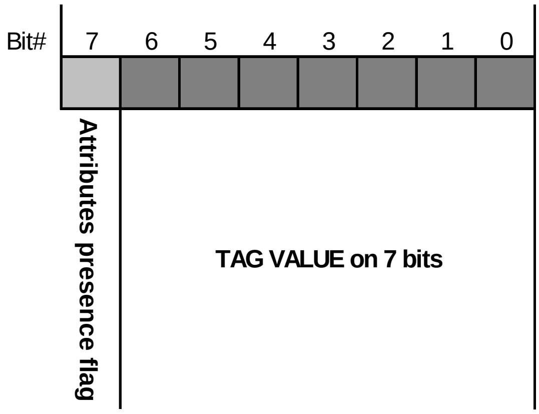
This notation is placed at the beginning of this document as it is used everywhere in the data structures to simplify the byte code parsing.

The TL[A]V (standing for Tag Length [Attributes] Value) is the basic structure element. Each of these TL[A]Vs can have optional attributes. The Value part of a TL[A]V can itself contain TL[A]V types of elements.

Advantage of the TL[A]V structure is that each element is self explaining and that default values are used when TL[A]Vs are not specified, this is reducing the need to transmit every tag value each time.

The TL[A]Vs are coded of the following manner :

- TL[A]V format

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	Tag value	Tag (+ 1st attribute byte presence flag)  Bit# 7 6 5 4 3 2 1 0 Attributes presence flag TAG VALUE on 7 bits **Attributes presence flag:** If Set : indicates that optional "Attribute bit field" is present after the "Length field". (see coding afterwards) If not Set : indicates that optional " Attribute bit field" is not present.	M
1-3	L (Length value)	Length of subsequent data in BER-TLV format including the attribute bytes : BER-TLV coding is specifying - 1 byte coding for L in [0-127] , - 2 bytes coding for L in [128-255] - 3 bytes coding for L in [256-65535] L = 0 is allowed	M
N(0+)	Attribute bit field	Attribute bytes (see coding afterwards)	O
L-N	“bytestream”	Value	O

Note: In any case the length is coded in BER-TLV format. Every element tag can have one or more attributes. These attributes are defined for each tag in the following chapters.

- Coding of the attributes bits :

Each tag has a predefined set of attributes. Each attribute bit is corresponding to a predefined value. The default value is always 0.

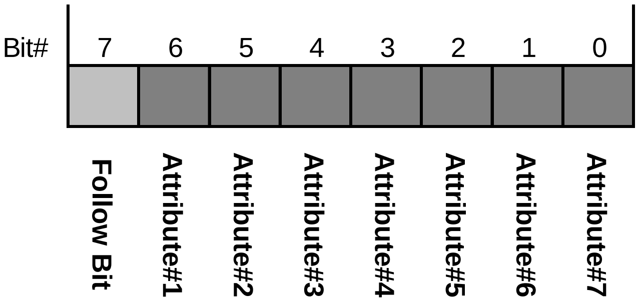
For each of these attributes, one bit is reserved in a bit stream.

The bit stream is coded on a byte array, each byte containing up to 7 attribute bits.

Whenever a tag requires more than 7 attributes, the number of attribute bytes will be extended.

The order of attributes in the bit stream is from left to the right (attribute 1 to 2…) corresponding to the MSB -> LSB bits in the structure.

- For tags requiring attributes , we use the following structure :



Each attribute byte is able to contain up to 7 predefined values of attributes. As the byte code is established today this is enough, but in aim to be more future proof a Follow Bit is present in this byte to allow one or more other byte(s) of attributes.

The previous coding is used for all the byte code.

Example: (the tag values are only defined for this example)

- Tag value 0x03 without specific attribute

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	0x03	Tag (optionally requiring attributes)	
1	0x0A	Length of subsequent data	
0x0A	“Teststring”	Value	

- Tag value 0x03 with 2 attributes selected, and a longer string.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	0x83	Tag (optionally requiring attributes) with attribute bit set	
2	0x81,0x91	Length of subsequent data (bigger than 127)	
1	0x50	Attribute indicator (here 0b01010000). Attribute#1 and #3 set.	
0x90	“Teststring……”	Value (shorten for the example)	

- Tag value 0x03 with 10 attributes defined, and a longer string.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	0x83	Tag (optionally requiring attributes) with attribute bit set	
3	0x82,0x0202	Length of subsequent data (bigger than 255)	
2	0xD0,0x70	- Attribute follow bit set (more than 7 attributes) attribute indicator (Here 0bx1010000) attribute#1 and #3 set. - Attribute indicator (Here 0b01110000) attribute#8, #9 and #10 set.	
0x0200	"Teststring……"	Value (shortened for the example)	

5 STRUCTURE OF SERVICES

5.1 INTRODUCTION

The purpose of the project is to offer a generic browsing tool embedded in a SIM card. The sources of information to browse are either WML or S@TML pages. They are located either on operator’s intranet or on the web.

A service can be seen as a set of pages that are browsed successively with interactions to the user. A service is expected to be provided by a Service Provider.

The service pages are converted to deck and card structures.

5.2 NAVIGATION CONCEPTS

5.2.1 Default Behaviour

The browser provides a way to navigate through cards belonging to one or more decks and services. The requirements of browsing include a way to go ahead to next card, to go back to previous card, to go to a new URL that can be resident or remote and to have contextual information.

By default, it is considered that the browser manages a stack of N last history cards. Each card successfully parsed is added to this history list, except if the flag DoNotHistorize is set in this card.

The Back operation is managed on logical back, not physical back, meaning that this is not dependent on the card location in the deck.

When reaching the last byte code of the last card of a deck, then the browser will switch to an implicit pause state. During the pause state, the user will be able to interact with the browser.

5.2.2 Navigation Elements

The service developer is able to specify his needs for the contextual menus.

The browser manages three logical menus for which it maps a physical key press.

The logical menus are contextual menus displayed on demand by the user pressing one of the mobile keys :

- Back Menu (associated by default to the Back key)
- Help Menu (associated by default to the Help key)
- Abort Menu (associated by default to the Abort key)

The content of each contextual menu is predefined with so called “system elements”.

Each of them can be set visible or hidden by a specific byte code (Manage Contextual Menu Item).

It is also possible to add new “application elements” that allow to go to a new URL.

The byte code provides a way to add or remove menu items to contextual menus.

Actions on these menus could be :

- Add an item (used to show a system item or create a new application item)
- Remove an item (used to hide a system item or delete an application item)

In aim to reference the items during these add/remove operations, it is necessary to have a coding for them.

The specified encoding is :

Menu Id	System/Application Item	Card/Operator Item	MenuItem Id

The Menu Id is defined in the following table (value xx on 2 bits)

The System / Application Item bit is 0 if application, 1 if system item.

The Card/Operator Item bit is 0, if the update of the contextual menu item is done using the Manage Contextual Menu Item byte code inside a card, 1 if the update is done using the byte code inside an administrative command.

The Menu Item Id is a number used to identify uniquely an item.

For the application elements of these menus, the gateway generates dynamically a Menu Item Id. The Menu Id put in the structure will indicate where the application element is to be displayed. The System/Application bit is not set.

	CODE	Default Visibility	Back Menu xx=00	Help Menu xx=01	Abort Menu xx=10
Back to previous card	**xx1 00001**	√	√		
Next (after a back)	**xx1 00010**	√	√		
Restart the current deck (Start at the 1st card)	**xx1 00011**		√		
Go to starting deck (resident deck home page)	**xx1 00100**	√	√		
Display bookmarks (The bookmark entry is calling a browser specific function to display current bookmarks to the user). Implementation of bookmarks is optional.	**xx1 00101**		√		

Display the current help string, if present	**xx1 00110**	√		√	
Stop browser (always visible)	**xx1 00111**	√			√
Pause the session. Implementation of the pause mechanism is optional.	**xx101000**		√		
Push Activation/Deactivation. This entry launches a Card Manufacturer specific menu to allow the user to activate/deactivate Low and/or High Priority Push management on the browser side. According to user choice, bits b3 (for High priority push) and/or b4 (for Low priority push) are set/unset in Browser Profile. Implementation of Push is optional.	**xx101001**		√		
Service specific items allowing to go to a new URL.	**xx0 yyyyy**		√	√	√

This is corresponding to the following default menu :

- BACK Menu :
  - Back to previous card.
  - Go next card (after a back)
- HELP Menu :
  - Display the current help string
- ABORT Menu :
  - Stop the browser

The contextual menus are reset to their default at the exit of each card. It is possible to use a card template object to have the same contextual menus defined for several cards of a deck.

5.3 DECKS & CARDS

5.3.1 Rules and Limits

The card is defined as being the elementary browsing element, meaning that the user can skip from one card to another back and forward.

A deck is a set of cards that is stored as a unit in the browser.

The deck is the smallest unit that the S@T gateway can send to the browser using the SSP.

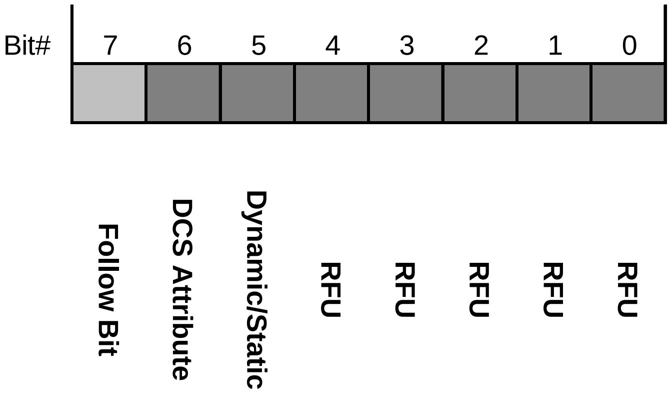
Decks and cards are stored together in the browser. The structure is described later in this document. These decks and cards are stored either permanently in the SIM card or received and interpreted on the fly.

They are referenced using URL as described in the next paragraphs.

To be able to create complex services, these URL references are used to fetch new decks or to link decks together.

In order to implement WML “templates”, a card template object type is available. It has the same structure as a card, but can only contain byte codes that do not require user interaction and have no branching. If a template is present in a deck, all cards will inherit this template except if the attribute “DoNotUseTemplate” is set for this card.

5.3.2 Decks

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	DeckTag	Deck tag (+ attribute byte presence flag)	M
1-3	L	Length of subsequent data (length coding in BER-TLV format)	M
		Optional attributes bytes	O
0+		Attributes byte  Bit# 7 6 5 4 3 2 1 0 Follow Bit DCS Attribute Dynamic/Static RFU RFU RFU RFU RFU **DCS Attribute :** Default Value : SMS Default Alphabet in unpacked format If Set : this attribute means UCS2. **Dynamic /Static** Default Value : Static If Set : Dynamic This attribute gives the browser an indication wether to cache the deck (static) or not (dynamic).	
		Deck identification element	M
1	DeckIdTag	Deck identification tag	
1-3	X	Deck identification value length	
X		Deck identification value (Unique identification of the deck) (Refer to Address Reference for coding)	
		Service permanent store reference	O
1	SPSTag	SPS tag	
1-3	A	Length (maximum length : 8)	

A		SPS value	
		Cleanup variable list element (when exiting the deck)	O
1	VarRefListTag	Variable reference list tag	
1-3	N	Length of the variables reference list (= number of variable references)	
N		Variable ID	
		Text element table	O
1	TextElementTableTag	Text element table tag	
1-3	N	Length of the text element table	
N	LV	LV value of each element	
		Card template	O
3-x	TLV	Card template TLV structure	
		Card TLV	M(1-n)
3-x	TLV	Card element TLV structure	

If the DCS is SMS default alphabet in unpacked format, the alphabet used is the SMS default 7-bit coded alphabet as defined in GSM 03.38 with bit 8 set to zero

5.3.3 SPS Reference

Any deck must refer to a SPS reference prior to be able to use its variables. For SPS refer to 5.4.4 'Service Permanent Store'.

5.3.4 Text Element Table

The text element table contains a list of text elements that can be used by reference during the deck execution. Text elements are constant texts that cannot be modified by the byte code. The reference to text elements and variables is unified to ease the browser work.

The text reference identifier is the index of the text element in the deck prefixed with the 0xC0 mask to indicate that it is a text element. Elements are numbered sequentially starting with 0.

Text reference index is in the range 0 to 0x3F (see 5.4.6).

5.3.5 Cleanup Variable List

This list is an enumerated list of variables references that need to be cleared at the exit of the deck. References can be either permanent or temporary variables.

This mechanism is in complement with the card attribute allowing the cleanup of all the temporary variables at the entry of a card (new context).

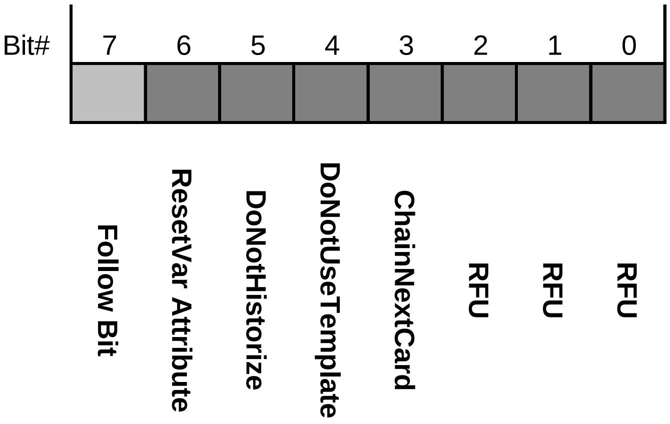
For coding see 5.5.2 'Variable Reference List'.

5.3.6 Cards

The card element specifies one unit of navigation. It can contain several interactions. Note that back operation will refer to the first action of the previous card not to the previous action of the current card.

While designing decks, this notion of card can help to manage the granularity of parsing.

A card can contain display elements, select items, input fields, ...

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	CardTag	Card tag (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional card attribute	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit ResetVar Attribute DoNotHistorize DoNotUseTemplate ChainNextCard RFU RFU RFU **ResetVar Attribute :** Default Value : Keep variables context If Set : Reset all the temporary variables when entering the card **DoNotHistorize :** Default Value : Historize the card. If Set : Do not insert this card reference in the history. **DoNotUseTemplate :** Default Value : Use the template when entering the card. If Set : Ignore the template. **ChainNextCard :** Default Value : Wait for user interaction at the end of the card (Browser idle mode), when the last byte code was executed and no branching was done . If Set : Automatically start the next card after execution of the last byte code of this card.	

		Card ID Element	O
1	CardIdTag	Card ID tag	
1-3	M	Card ID length	
M	Value	Card ID in the deck (same coding as <CardName> in AddressReference)	
		Byte code element(s)	O(0-n)
1	0xYY	Byte code tag (among predefined set) (+attribute presence flag)	M
1 – 3	D+X	Length	M
X (0+)	0xZZ	Attribute bytes	O
D	Value	Byte code structure (associated to the tag)	M

5.3.7 Card Template

„One card template element can be included in a deck. It will be stored at the beginning of a deck to simplify parsing.

When a deck contains a card template, all the cards of this deck will inherit from that template, meaning that the byte codes stored in the template are executed at each card start.

Optionally, one card can have an attribute to specify that it does not make use of the template.

User interaction and flow control byte codes are not allowed in the card template.

The not allowed macros are:

- Init Variable Selected
- Go Back
- Go Selected
- Switch Case On Variable
- Exit
- STK Generic Macro
- Execute Generic Macro

It is mainly used for the Contextual Menu management and Help String management.“

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	CardTemplate Tag	Card template tag (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Byte code element(s)	M(1-n)

1	0xYY	Byte code tag (among predefined set) (+attribute presence flag)	M
1 – 3	D+X	Length	M
X (0+)	0xZZ	Attribute bytes	O
D	Value	Byte code structure (associated to the tag)	M

5.3.8 Referencing Decks and Cards

There are different levels of object referencing in the deck :

- Card to card in the same deck
- Card to another deck.
- Card to card in a different deck.

An unified coding for deck and card name is used. It is called address reference.

5.3.8.1 Deck and Card Naming Convention : Address Reference

Some decks can be preinstalled in the SIM during personalisation time. In order to reference these decks, management of long deck name is mandatory.

The grammar of an address reference string is the following :

AddressReference :==[(<LongDeckName> | <CodedDeckName>)] [# <CardName>]

The CardName is recommended to be an up to 2 characters string, but longer card names are managed for WML mediation.

The LongDeckName and CardName are strings coded in GSM 03.38 default alphabet in unpacked format.

# is treated as a separator between the deck name and the card name.

The CodedDeckName is a byte array with the 1st most significant bit of the 1st byte set to 1. This value is computed by the gateway.
Constraint : the '#' character is forbidden in the CodedDeckName and LongDeckName byte array to avoid the misinterpretation of this value.

The deck and card identifier found in the deck and card structure use the same coding rules.

When resident decks are stored during personalisation time, the reference is instantiated in a long name format.

It is assumed that the gateway can transform the long names references to coded deck references by an efficient mean to earn bandwidth.

On the server side, the long name is replaced by a coded name in the downloaded byte code and the back translation reuses this unique identifier prior to access the WML server.

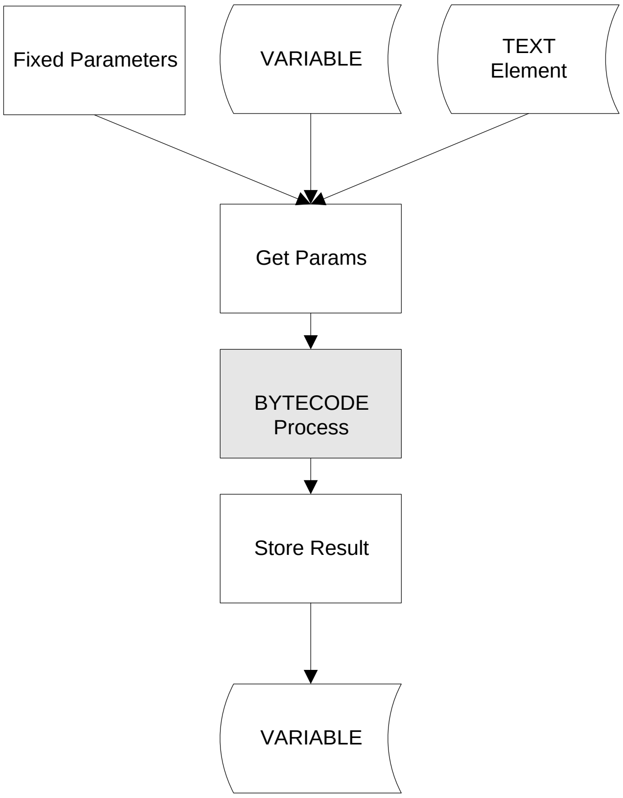
5.3.8.2 URL Reference

For URL reference, a TLV structure has been chosen to encapsulate the address reference element.

This is providing the way to specify how the access to the page is done, and complement this reference with optional parameters.

5.3.9 Execution of the Byte Code

The byte code behaviour can be synthesised as follows :



Each byte code takes parameters for input (fixed, variables or text elements) , processes them and provides an output that is stored in an output variable. Some byte codes may have no output (as switch case for example).

5.4 VARIABLES, TEXT ELEMENTS & SERVICE PERMANENT STORE

5.4.1 Introduction

Services are developed to display information, get inputs from the user, perform some computations or verifications. The need for variables is obvious.

The variable manipulation in the browser has been unified with the text element manipulation. Variable substitution mechanism is then unified and reused.

Text elements are part of the current deck. These elements are to be considered as constants and no update is allowed on them.

Two types of variables are defined : Service specific permanent variables, and temporary variables.

The variables can be used :

- To store the proactive command specific response data object of a byte code execution (Output value).
- To manipulate data in the browser.
- To replace a parameter in a byte code structure.
- To construct requests for the gateway

5.4.2 Temporary Variables

The temporary variables are defined as deck by deck by default, meaning that they are shared among the cards but can be reset at the entry of a card optionally.

Their scope can be enlarged to several decks if a card reset is not requested.

A set of variables (permanent or temporary) can be reset at the exit of a deck if specified in the deck structure.

5.4.3 Text Elements

These text elements are part of the deck structure. They need to be considered as constants and cannot be modified by the byte code.

The text element structure is LV (length L on 1 byte + value V).

The DCS of these elements is inherited from the deck level.

5.4.4 Service Permanent Store

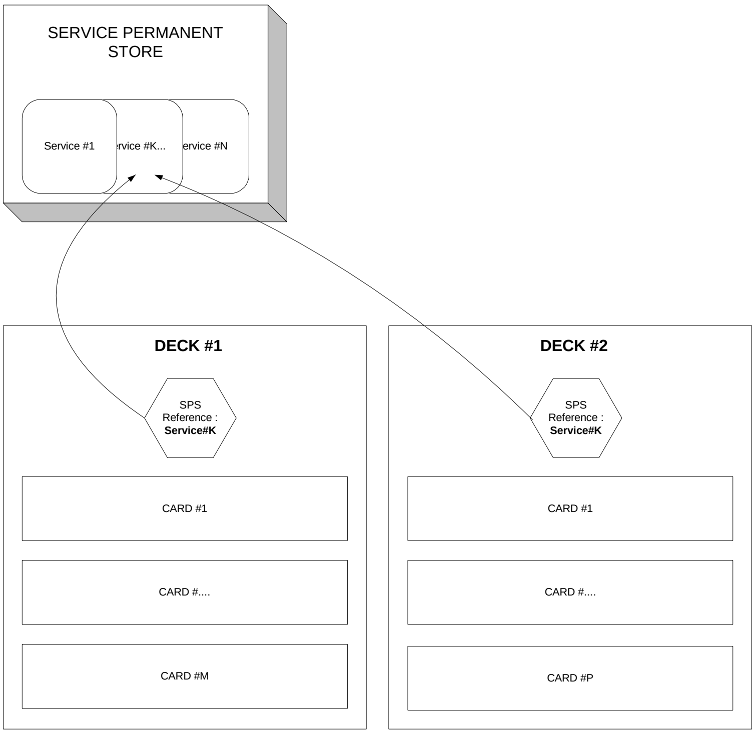
A service is represented by a set of decks (WML jargon). The decks are able to interact with the user using STK commands, store values in variables, send requests to the gateway.

A persistent storage area for variables, is optionally provided in the SIM card, and shared by all decks of a given service. It is called the Service Permanent Store.

The store size and id are allocated by a dedicated administrative command.

It is expected that the service area is allocated by the operator or under his control.

Any deck must refer to a SPS reference prior to be able to use its variables.



The permanent variables stored in service permanent store can be remotely managed .

5.4.5 Variable Types

The variables are all defined as :

- LV meaning length + value : L is coded on 1 byte (0-254)

The advantage of this representation is that it is flexible for substitution, comparison and concatenation.

The variables are used to fill placeholders (see references later) in all the STK byte code where parameters in TLV are required.

Explicit Variable References or Variable Reference List structures are used to refer the variable in all other byte codes.

A variable can contain an address (Deck#Card) and any string as well.

This representation can also represent short variables like bytes or short integer used to perform minimal computation (loops, comparisons…)

Note: The individual type of the variable has to be managed by the browser to handle for example the DCS of the strings and apply a minimum type checking on them.

5.4.6 Variable Identification

Variables and text elements (constants of the deck) are referred by a common mechanism. They are accessed by an identifier coded in 1 byte.

The byte value range is split as follows to identify the types of variables/elements :

Bit#	7	6	5	4	3	2	1	0	
	0	Index							Temporary variable reference
	1	0	Index						Permanent variable reference
	1	1	Index						Deck text element reference

This allows a split of the reference of :

• Permanent variables :	64	variables
• Temporary variables :	128	variables
• Deck text element :	64	variables

Variable reference coding rule :

The permanent variables are referred in the SPS declared by the deck.

If a permanent variable is referenced in a deck not declaring a service permanent store, the access should be refused.

5.4.7 Variable Reference

Depending on the byte code context, 2 variable references are used :

- The simple reference, that is just the identification code of the variable (1 byte)
- The encapsulated reference, that is the same value contained in a TLV.

The first one is used mainly during the substitution mechanism, while the last one is used when it is necessary to specify the type of the reference.

5.4.8 Substitution Mechanism

In the specific context of a STK generic byte code variable references are replaced by their values in the following way:

The reference are of the type :

- <TAG>
- <LEN=0xFF>
- <VAL = Variable Reference>

When the 1st byte of length is equal to 0xFF, it does mean that the value of the tag is containing a variable reference and that a substitution is required.

There is no conflict with the BER-TLV field as the 1st byte of a L is never 0xFF in this coding.

Note: Nested substitution will not be performed.

5.5 GENERAL STRUCTURES

5.5.1 Variable Reference

This tag is used to refer to a variable content when another parameter type is allowed. For example a SetHelp command can have both inline value and/or variable content.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Variable reference	
1	VarRefTag	Variable reference tag	M
1-3	1	Variable ID length	M
1		Variable ID	M

5.5.2 Variable Reference List

This tag is used to refer to a variable list content when another TLV structure is allowed.

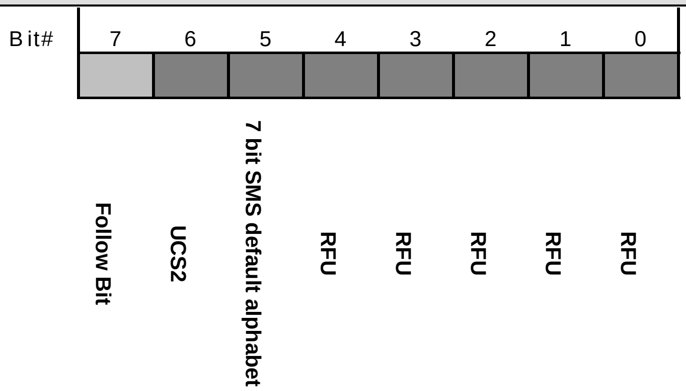
LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Variable reference list	
1	VarRefList Tag	Variable reference list tag	M
1-3	n	Length (= number of variable references)	M
N		Variable ID	M(n)

5.5.3 Inline Value

This tag is used to indicate value presence when another parameter type is allowed. For example a Set Help command can have both inline value and/or variable content.

Text can be embedded directly in the byte code when no reference to variable is necessary.

It is just a TLV with the V containing a text or binary string.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Inline value	
1	InlineValue Tag	Inline value tag	M
1-2	X	Inline value length (length <255)	M
		Optional attributes	O
0+		 Bit# 7 6 5 4 3 2 1 0 Follow Bit UCS2 7 bit SMS default alphabet RFU RFU RFU RFU RFU ➢ **UCS2**: If this attribute is set, then the variable is coded in UCS2 coding. ➢ **7 bit SMS default alphabet**: If this attribute is set, then the variable is coded in 7 bit SMS default alphabet in unpacked format.	O
X		Inline value content (text or binary string)	M

The attribute coding is to be used in the constant parameter structure (parameter value) if DCS information for inline values is available for the browser. If the DCS information is unknown, the browser shall not set any of the two attributes (Bit 6, Bit 5).

For Gateway originated inline values the attribute may be used to indicate a dedicated DCS information for the value. A missing attribute byte (or both bits mentioned above set to zero) in this direction forces the browser to inherit the DCS information from the related deck. In general, the Gateway shall not use the optional type information for inline values to be sent to the browser at all.

Type information coding summary:

Type indicated in attribute **(Browser originated /** **Gateway originated)**	**Bit 6** **(UCS2)**	**Bit 5** **(7-Bit)**
Unknown / inherited from deck	-	-
Unknown / inherited from deck	0	0
UCS2 / UCS2	1	0
7-bit SMS default / 7-bit SMS default	0	1
Unknown / inherited from deck	1	1

5.5.4 Input List

This object is containing a composite list of other elements.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Input list	
1	InpListTag	Input list tag	M
1-3	X	Length	M
X		Content (Variable Reference, Inline Value, ..). The content will be specified each time the input list is used.	M(1-n)

5.5.5 Parameter

This structure is used to refer to a variable content and an optional text association (named parameter).

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Parameter element	
1	ParTag	Parameter tag	M
1 or 2	X	Parameter length	M
1	0-255	Variable reference to get value from	M

X-1		Parameter name (text string)	O

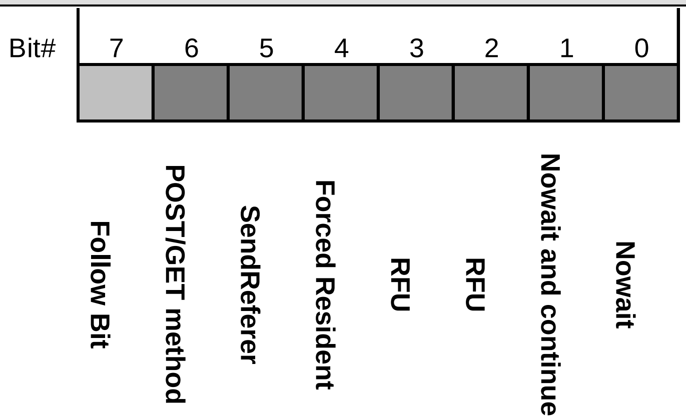
5.5.6 Constant Parameter

This structure is used to refer to a constant parameter in an URL reference.

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
		Constant parameter element	M
1	ConstParTag	Constant parameter tag	
1 or 2	X	Constant parameter length	
		Parameter value	M
1	InlineValueTag	Inline value tag	
1 or 2	Y	Length of parameter value	
Y		Parameter value	
		Parameter name	O
1	InlineValueTag	Inline value tag	
1 or 2	Z	Length of parameter name	
Z		Parameter name	

5.5.7 URL Reference

This tag is used everywhere an URL reference is needed. It can represent a card, a deck or card + deck reference.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	URLTag	URL tag (+ attribute presence flag)	M
1 or 2	X	URL length	M
		Optional URL attributes	O
0+		 Bit# 7 6 5 4 3 2 1 0 Follow Bit POST/GET method SendReferer Forced Resident RFU RFU Nowait and continue Nowait ➢ **Post/Get Method** : If this attribute is set, then Post Method, else Get Method used. This is useful for a temporary deck request for which the parameters have to be sent. ➢ **SendReferer** : By default, the SendReferer is not sent. The deck/card caller is sent in the GET/POST message when Set. ➢ **Forced Resident** : If set: the deck reference is resident and if not found, generate an error, but do not go online. If not set: the deck reference is not resident, meaning it must not be searched among the resident decks. ➢ **Nowait:** If set, the request is sent and the browser exits ➢ **Nowait and continue**: If set, the request is sent and the next macro is executed. If the 2 nowait bits are set, the Nowait and continue attribute has to be taken into account	O
		Address Reference or VarRef element	M
3+	AddrRefTLV	Address reference TLV	
3	VarRefTLV	Variable reference TLV (contains an address reference string)	
		Parameters or Constant parameters.	O(0-n)
3+	ParTLV	Parameter element (TLV) structure	
3+	ConstParTLV	Constant parameter element	

5.5.8 Couple

This tag is used to encapsulate two items, and generally appears when a list is needed.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
		Couple	
1	CoupleTag	Couple tag	M
1-3	X	Couple length	M
		TLV Content 1 (for example Variable Reference Tag)	
	TLV		M
		TLV Content 2 (for example URL Tag)	
	TLV		M

The Tag contents 1 & 2 are specified each time a couple is used (for example in Switch Case on Variable).

5.6 FLOW CONTROL OPERATIONS

5.6.1 Introduction

The flow control operations allow the deck to embed decision points to branch to one or another card depending on tests or comparison.

The generic branching byte code is the “Go Selected”. It is starting the execution of another card in a specified deck or the same deck.

5.6.2 Go Selected

The role of this macro is to display a list of items on the mobile and to branch to the card or deck linked to the selected item when the user presses OK. The same byte code has a deprecated behaviour allowing the go to a card without display .

5.6.3 Switch Case on a Variable

That is a well known structure in languages to branch to an address depending of the value of a variable.

5.7 Browser Link to Extensions (EXECUTE)

The execute byte code allows the execution of a piece of code that is not belonging to the browser, but as an external function (for example: cryptographic computation…)

It can also be used to launch another application after the exit of the browser.

5.8 Resident and Temporary Decks

As the browser engine is available in the SIM card, it can be used to execute online applications received over the air or start the execution of decks located permanently in the SIM memory (which means decks put in the SIM card during personalisation time).

The decks stored resident are managed by the gateway. They are callable by decks received on the fly.

The resident decks can be installed by a browser administrative command.

6 STRUCTURE OF THE BYTECODE

6.1 Introduction

The byte code structure is unified as TL[A]V format.

Each byte code will have its own tag value, used to interpret it.

The list of error codes appended to each byte code description is not complete. General errors may occur on every byte code. Only macro specific error codes are listed.

LENGTH	*VALUE*	*DESCRIPTION*	*M/O*
1	0x00 to 0xFF	Byte code tag (+ attribute presence flag)	M
1-3	0-65535	Length of subsequent data	M
		Optional attributes	O
X		Attributes byte	
		Byte code parameters	M
L – X		Byte code parameters list	

The byte code values are in range 0-127 as the bit 7 is used to indicate the attribute presence.

The byte code can be seen as 3 types :

- General Purpose Macros.
- STK Generic Macro : A single generic macro that allows the description of any STK command, as specified in the GSM 11.14 current release or future.
- Execute Generic Macro : A macro used to interface the browser with external functions like cryptographic functions, applets.

Each macro generates a general result code (used in the browser) and an output value that can be stored in a specified variable.

NOTE : This specification assumes that the parameters are always in the specified order, in order to simplify the browser implementation. Unknown byte code tags shall not force the browser to stop.

6.2 Macros

6.2.1 Variable Management

6.2.1.1 Init Variables

INIT VARIABLES			
Description	Initialise a list of variables(1-N) with a list of values or variable references.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Argument (VarId, Value)	M(1-N)
1	DestVarId	Id of the variable to initialise.	
1	Tag	Tag allowed : Variable Reference, Inline Value Element	
1-3	M	Length	
M	Value	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Syntax error		Syntax error (for example try to initialise a text element)	Stop
Reference to undefined		Reference to undefined variable	Stop
Problem in memory management		Memory allocation problem	Stop

6.2.1.2 Init Variable Selected

INIT VARIABLE SELECTED			
Description	Displays a menu on the ME (as in the STK command Select item). Depending of the user choice, a specific value is assigned to a variable.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Destination variable identifier	M
1	DestVarId	Variable where to store the selected value.	
		Select Item Title TLV	O
1	Tag	Tag allowed : Variable Reference, Inline Value Element	
1-3	M	Length	
M	Value	Value	
		Select item couple list	M (1-n)
1	CoupleTag	Tag Couple	
1-3	N	Length	
X	TLV	Item TLV (to be displayed for selection). Tag allowed: Variable Reference, Inline Value Element	
Y	TLV	Value TLV (to be assigned after selection). Tags allowed: Variable Reference, Inline Value Element	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Reference to undefined		Reference to undefined	Stop
Problem in memory management		Memory allocation problem	Stop
Syntax error		Syntax error (for example: try to initialise a text element)	Stop
STK use failed		STK command embedded failed.(STK command not being built)	Stop

6.2.2 Environment Variables

Environment variables define parameters of the browser client. These environment variables are used both by the gateway for administrative purposes and by the applications loaded online to the SIM to determine certain application related (standard) procedures. The settings defined by environment variables are shared among services.

There are **three types** of environment variables: System variables, runtime variables and user preferences.

System Variables define the system parameters of the SIM Browser client including the mobile phone characteristics.

Runtime Variables store information, that changes during runtime of an executed service.

User Preferences include all settings the user carries out to configure his SIM browser.

Environment variables can be read using the GET ENV macro described below.

For detailed information see Annex A.

6.2.2.1 Getenv

This macro will generate a string corresponding to the element required into a variable.

GETENV			
Description	Get the value of an environment variable Note: Access to any variable is allowed in this version.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Variable to initialise	M
1	DestVarId	Id of variable to set	
		Environment variable reference	M
1	EnvVarRef	Environment variable identifier to get	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined		Reference to undefined variable	NoStop
Syntax error		Syntax error (for example: try to initialise a Text Element)	Stop

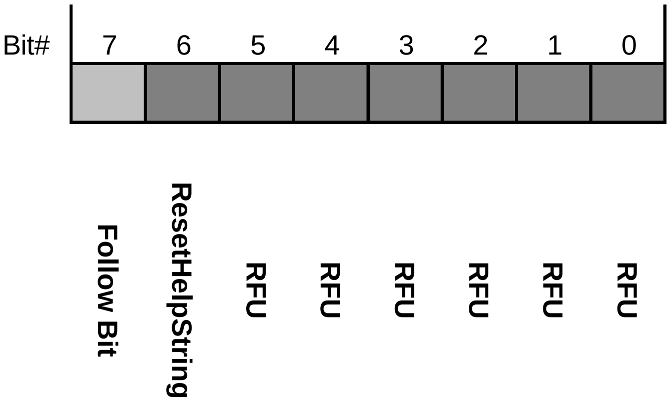
Error handling procedure :

If reference does not exist, return a NULL string.

6.2.3 Set Help

SET HELP	
Description	The SetHelp macro is used to set or reset a help reference variable. The help is defined as a list of strings that are context sensitive, meaning that for a select item based command, it is possible to associate a help string to each choice. Caution : At least one help string is required when the ResetHelpString attribute is not set. Use of ResetHelpString attribute: • If the flag “reset help string” is set to 0 and the parameter “help string list” is not empty : the help string list is append to the existing help string list in the browser. • If the flag “reset help string” is set to 0 and the parameter “help string list” is empty : an error occurs (Reference to undefined). • If the flag “reset help string” is set to 1 and the parameter “help string list” is not empty : the existing help string list in the browser (if one) is replaced by the one given in the macro. • If the flag “reset help string” is set to 1 and the parameter “help string list” is empty : all the help strings items will be deleted. Rules to apply: When the number of help strings is different than the number of items of the currently processed command, the first items is matched with the first help string, the second item with the second help string, etc Examples: - there is one item in the SBC command (item1) and 2 help strings (help1, help2): item1 is matched with help1 - a help string (help3) is added, and the help strings list becomes (help1, help2, help3) - there are 4 items in the next SBC command (item1, item2, item3, item4): item1 is matched with help1, item2 is matched with help2, item3 is matched with help3, item4 has no help

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M

		Optional byte code attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit ResetHelpString RFU RFU RFU RFU RFU RFU **ResetHelpString** : Default Value : 0 : Append to existing help strings If Set : 1 : Reset all help strings	
		Help strings list	O(0-n)
1	Tag	Tag allowed : Variable Reference, Inline Value Element	
1-3	L	Length	
L	Value	Value	
SCOPE OF VALIDITY		***DESCRIPTION***	
		The Help String is reset at the deck exit.	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined		This error applies if the flag "reset help string" is set to 0 and the parameter "help string list" is empty	NoStop

6.2.4 Concatenate

CONCATENATE			
Description	Concatenates two or more values and stores the result in a variable. The result is in LV form. Checks are performed on the DCS coding of all elements.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Destination variable identifier	M
1	DestVarId	Variable that will contain the result	
		Values to concatenate	M(2-n)
1	Tag	Tag allowed : Variable Reference, Inline Value Element	
1-3	L	Length	
L	V	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Syntax error		Syntax error (for example try to initialise a text element)	Stop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined		Reference to undefined	NoStop
Type mismatch		Incompatible DCS in variables	Stop

6.2.5 Extract

		EXTRACT	
Description	Extracts a byte array from a value and stores the result in a variable.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Destination variable identifier	M
1	DestVarId	Variable that will contain the result	
		Input variable identifier	M
1	InpVarId	Variable containing the source string to process.	
		Start indexes and length of the substring	M
1	StartIndex	Index in the string in the range : 0-[Len(InpVarId)-1]	
1	Length	Length of the string to extract. If Length + Startindex exceeds length of InpVarId, the length is internally reduced to length of InpVarId – Startindex and no error will be generated.	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Syntax error		Syntax error (for example try to initialise a text element)	Stop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined		Reference to undefined variable	Stop
Out of range		Index out of range.	Stop

If InpVarId refers to a variable coded in UCS2, the value for startindex and length (of substring) gets multiplied by 2 internally.

6.2.6 Crypto Macros

The crypto macros are used for end to end security.

The principle is that a structure is exchanged between an application server (this could be a Bank) and the browser.

The exchanged structure is defined as follow :

	SecMsg	Message identification element	

1	SecMsg Tag	Tag for Secure Message	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Key identification : Same coding as 3.48 (SPI Subset,Kic/Kid)	M
1	SPI	SPI value (00, 10, 100, 110 binary values allowed)	
1	Kic	Kic : Cryptographic key index	
1	Kid	Kid : Checksum key index	
1	Pcntr	Padding Counter	
		MAC element (8 bytes)	O
1	L	Length	
L	Value	Value	
		Enciphered or ClearText DataBlock (same length)	M
1	L	Length	
L	Value	Value structured as a series of LV,LV before ciphering	

In contrast to GSM 03.48 sequence counters are not supported in the SecMsg structure.

Kic and Kid are pointing to 16 bytes keys for Triple DES.

The MAC elements presence is linked to the SPI value. If SPI has the value b00 or b100, the checksum is not required and the whole field L + Value is not present in the SecMsg structure. If SPI has the value b10 or b110, the checksum is required and the whole field L + Value is present in the SecMsg structure.

The MAC has a fixed length of 8 bytes and is part of the encrypted message. For this reason the following part of the SecMsg will be ciphered: <MAC value><DataBlock value><Padding>. The MAC calculation is done over <DataBlock value><Padding> and before encryption and after decryption.

The data block is organised as a list of LV corresponding each to a logical value.

The cryptographic byte code is then able to find the structure of data, and offers the service of allocating the values to a set of browser variables.

For message creation on the browser, a list of variable is concatenated prior to be ciphered and sent to the application server.

It is assumed that the maximum length of the SecMsg is fitting in a max variable length (255) in mobile terminated direction, and in one SMS in mobile originated direction.

In Enciphered or ClearText DataBlock, each LV is formatted as length L on 1 byte + value V, as each LV is linked with a variable.

6.2.6.1 Encrypt

ENCRYPT			
Description	Encrypts a set of values to a data block If the length of the data to encrypt is not a multiple of 8 bytes(for DES), it is padded with NULL characters. In that case, the result variable is longer than the sum of the LVs of the plain text and rounded to the upper multiple of 8 (for DES)		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Key element reference for encrypt and sign	M
1	SPI	SPI	
1	Kic	Kic	
1	Kid	Kid	
		Output value element (Result of operations) in SecMsg format	M
1	DestVarId	Destination Variable Id	
		Variable list reference element to encrypt.	M
1	Tag	Variable Reference List tag	
1-3	L	Length	
L	Value	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Syntax error		Syntax error (for example try to initialise a text element)	Stop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined			Stop
Security problem			Stop

Kic and Kid are are pointing to 16 bytes keys for Triple DES.

If both signature and encryption are used, then calculation of the signature must be done on the uncrypted message.

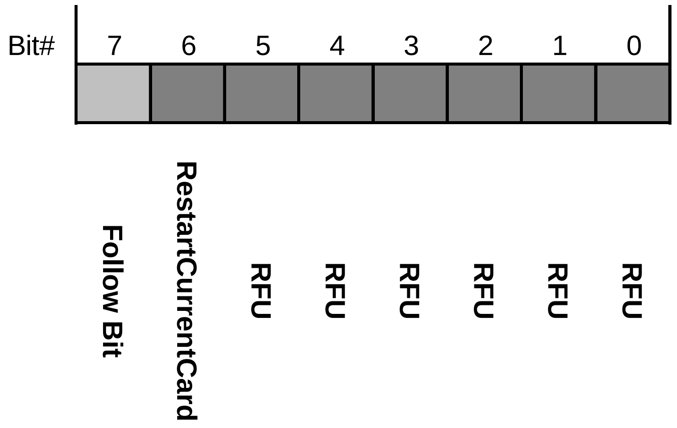
6.2.6.2 Decrypt

DECRYPT			
Description	Decrypt a datablock, verify MAC and assign results to a variable list		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Input datablock element in SecMsg format	M
1	Tag	Tag (Variable Reference or Inline Value Element)	
1-3	L	Length	
L	Value	Value	
		Output variable list reference	M
1	Tag	Variable Reference List tag	
1-3	L	Length	
L	Value	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Syntax error		Syntax error (for example try to initialise a text element)	Stop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined		Reference to undefined	Stop
Security problem		MAC not verified	Stop

If both signature and encryption are used, then calculation of the signature to be checked must be done on the decrypted message

6.2.7 Navigation & Branching Macros

6.2.7.1 Go Back

		GO BACK	
Description	This byte code is requesting the branching to the card before the last card memorised in the history buffer No impact on the history.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional go back attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit RestartCurrentCard RFU RFU RFU RFU RFU RFU **RestartCurrentCard** : Default Value : 0 : branch to the card before the last card in the history If Set : 1 : branch to the last card in the history	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	
Jump to undefined		Reference to undefined (case of history empty)	Stop

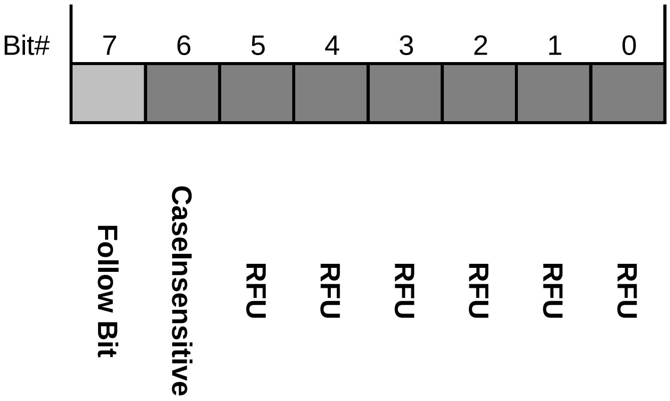
Note: In a Go Back the flag DoNotHistorize shall be set.

6.2.7.2 Go Selected

GO SELECTED			
Description	The GO SELECTED macro jumps to an URL (deck or card), depending on user input. The URL can be located resident or on the gateway. In the last case, the download of the requested deck is started. Depending on the URL referenced, this go is allowing a branch to a card or a deck. If no couple list is present a direct go will be performed.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Title element (Alpha identifier).	O
1	XxTag	Tag (Variable Reference or Inline Value Element tag)	
1-3	M	Length	
M	Value	Value	
		Select item element couple list or (exclusive) single URL reference to go	M (1-n)/M
N	CoupleTLV	Couple containing Variable Reference or Inline Value and URL reference to go on selection	M(1-n)
	or		
K	URL Reference TLV	URL reference to go	M
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Problem in memory management		Memory allocation problem	Stop
Reference to undefined			Stop
Jump to undefined		Reference card not found.	Stop
STK use failed		Case of command too big for the buffer	Stop
Out of range		Too long strings concatenated.	Stop

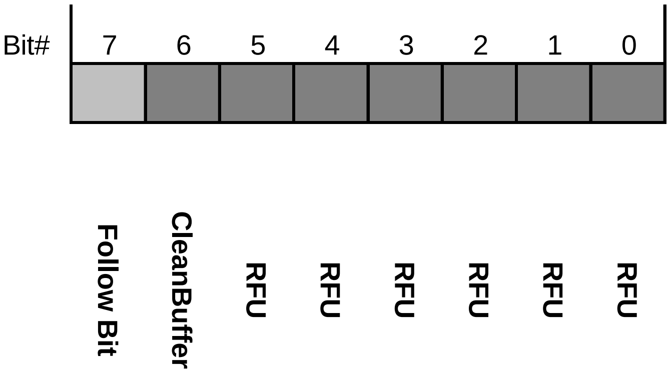
6.2.7.3 Switch Case On Variable

SWITCH CASE	
Description	Compares a variable to a list of values, which have an associated URL. When a match is found, the selected URL execution is started. Optionally, the match can be case insensitive.

LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional byte code attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit CaseInsensitive RFU RFU RFU RFU RFU RFU **CaseInsensitive Attribute :** Default Value : 0 : Case sensitive If Set : 1 : Not case sensitive (bit 7 ignored)	
		Variable ID to compare	M
1	0x00-0xFF	Variable identifier	
		Couples (Value to compare, URL to go on match)	M(1-n)
1	Couple tag	Tag for couple	
1-3	L	Length	
X	TLV	Value to compare tag (Variable Reference or Inline Value Element)	
Y	TLV	URL to go on match (URL reference)	
		URL to go if value not found in the list	O

1	XxTag	Tag (URL Reference)	
1-3	L	Length	
L	Value	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Reference to undefined			Stop
Jump to undefined		Reference card not found.	Stop

6.2.7.4 Exit

EXIT			
Description	The Exit macro stops the browser. In the case of browser launched by outside application, an output variable list can be provided to the caller.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional byte code attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit CleanBuffer RFU RFU RFU RFU RFU RFU **CleanBuffer Attribute :** Default Value : 0 : Do not clean the execution buffer If Set : 1 : CleanUp the execution buffer	

		Exit list element	O
1	VarRefListTag	Variable Reference List tag	
1-3	L	Length	
L	Value	Value	

ERROR CODES	***DESCRIPTION***	***Action***
No error	OK	Stop
Reference to undefined		Stop

6.2.8 Manage Contextual Menu Item

MANAGE CONTEXTUAL MENUITEM			
Description	The manage function is used to display/hide system contextual menu items, to add/update/remove application specific items. If a application menu item id is already existing, then it is updated with the new value.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional byte code attributes	O
1	0xXX	Attribute byte : Bit# 7 6 5 4 3 2 1 0 Follow Bit Remove/Hide RFU RFU RFU RFU RFU RFU **Remove/Hide Attribute :** Default Value : 0 : Add/Display the item specified If Set : 1 : Remove/Hide the item specified.	
		Contextual menu item identifier	M
1	Id	Either system or application depending on the coding.	
		Couple of contextual menu item text and URL to go, if item was selected	O
1	CoupleTag	Tag couple	
1-3	L	Length	
X	TLV	Inline Value or Variable Reference	
Y	TLV	URL reference	
ERROR CODES		***DESCRIPTION***	***Action***

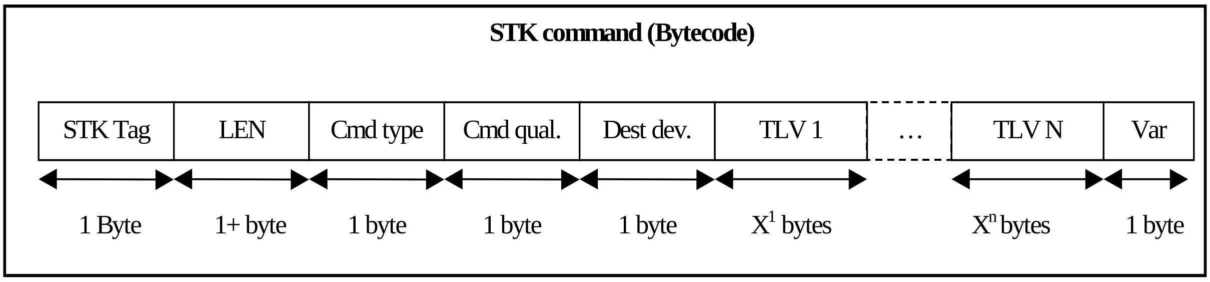
No error	OK	NoStop
Reference to undefined		Stop
Problem in memory management	Too many items allocated	Stop
Type mismatch	System or application menu item mismatch.	Stop

6.2.9 STK Generic Macro

The STK byte code, as defined in GSM 11.14, has various parameters. However, as we limit ourselves to the case of STK commands used by a SIM card, most of these parameters have a fixed value. The only parameters that can really change are the following:

- The command type (see GSM 11.14 V 7.2, section 13.4).
- The command qualifier (see GSM 11.14 V 7.2, section 12.6).
- The destination device identity (see GSM 11.14 V7.2, section 12.7).
- The various TLV parameters of the command.

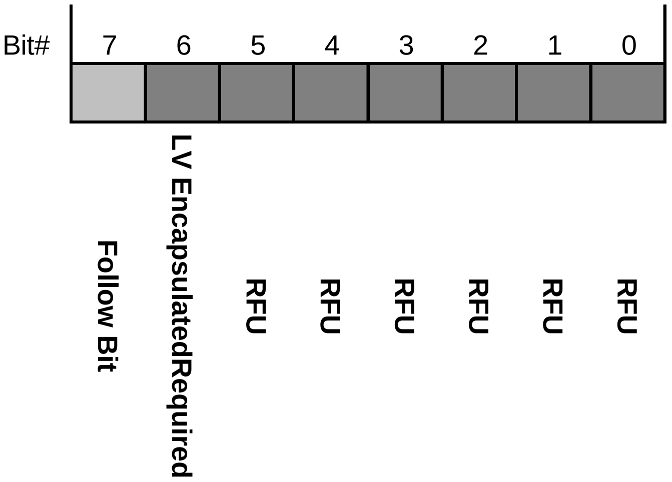
The value field of the STK command tag is described below:



STK command (Bytecode)

STK Tag	LEN	Cmd type	Cmd qual.	Dest dev.	TLV 1	…	TLV N	Var
1 Byte	1+ byte	1 byte	1 byte	1 byte	X^1 bytes		X^n bytes	1 byte

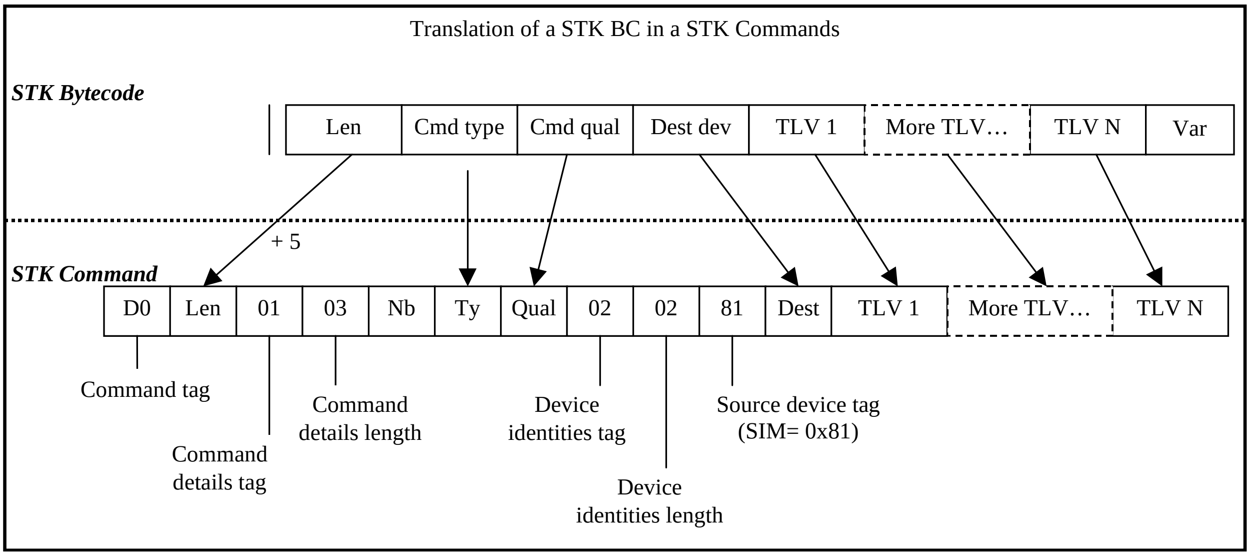
The structure of the byte code value field representing a simple STK command is the following:

STK_GENERIC			
Description	Generate a ***STK*** command based on given ***STK*** parameters		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional byte code attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit LV EncapsulatedRequired RFU RFU RFU RFU RFU RFU **LV Encapsulation Requested Attribute :** Default Value : 0 : TLV proactive command specific response data object is stored as LV in variable. (T is removed. When applicable, DCS is removed of the V field but used for variable management) If Set : 1 : Proactive command specific response data object of ME is encapsulated in LV (TLV/TLV). No processing is done as complex structure not known.	
		Command type value	M
1	Cmdtype	Section 12.6 of the GSM 11.14 (Version 7.2)	
		Command qualifier value	M
1	Cmdqual	Section 12.6 of the GSM 11.14 (Version 7.2)	
		Destination device	M
1	DestDev	Section 12.6 of the GSM 11.14 (Version 7.2)	
		TLV Parameters for the ***STK*** command	M(0-N)

1	XxTag	Tag of 11.14 for ***STK*** command	
1-3	L	Length	
L	Value	Value	
		Output variable id	O
1	DestVarId	Id of the variable where to store the proactive command specific response data object	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Reference to undefined			Stop
Problem in memory management		Memory problem in the preparation of the ***STK*** command	Stop
Syntax error		Try to initialise a text element	Stop
STK use failed			Stop

NOTE : If new TLVs are introduced in the evolution, then the DestVarId will stay at the last position.

When executing such an STK byte code, the browser issues a normal STK command to the mobile using the SIM Toolkit protocol. The translation procedure for STK command is described below:



All TLV parameters of this byte code are parsed for variable substitution as explained in next paragraph. Processing of output values is defined just after.

6.2.9.1 Byte Code and Variable Substitution

In the context of the STK generic byte code, variable substitution is possible on standard STK TLV parameters with the following technique :

The references are of the type :

```
<TAG>
        <LEN=0xFF>
                <VAL = Variable Id>
```

When a length is equal to 0xFF, the value field must contain a variable reference and the substitution is executed

6.2.9.2 General Result of a STK Byte Code

The general result is the value byte of the general result TLV (as defined in the GSM 11.14) produced by the execution of the command.

This general result is used by the browser internally to generate error codes or detect key presses.

6.2.9.3 Output Value of a STK Byte Code

The output value is the proactive command specific response data object (as defined in the GSM 11.14) produced by the execution of the command.

Note: Not all proactive commands produce a command specific response data object.

As some outputs from the proactive commands are organised as a suite of TLVs, when assigned to a variable, the output of this byte code is optionally encapsulated in a LV structure. (See attribute field)

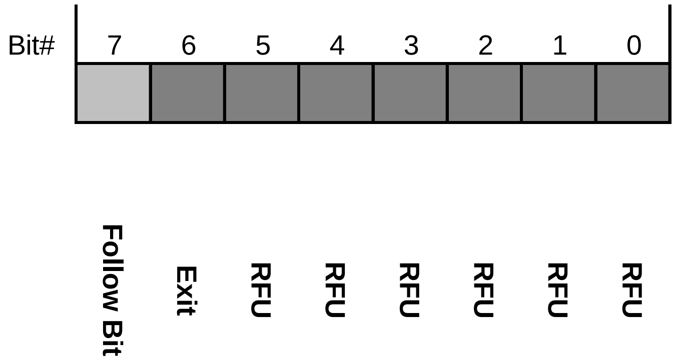
The manipulation of such a variable is possible with the extract byte code. It can also be sent to the server as is.

6.2.10 Execute Generic Macro

This macro is used to start the execution of a function or an application external to the browser.

The execute element references are coded on a 2 bytes structure :

- 1st byte : Manufacturer or S@T agreed
- 2nd byte : Execute element reference

EXECUTE			
Description	This macro starts the execution of an external function or application external to the browser. An attribute will indicate if the execution is returning to the browser after its execution or not. Parameters are passed for input [and output]. The output is stored in a list of variables.		
LENGTH	***VALUE***	***DESCRIPTION***	***M/O***
1	BC Tag	Byte code (+ attribute presence flag)	M
1-3	L	Length of subsequent data (length coded in BER-TLV)	M
		Optional byte code attributes	O
1	0xXX	Attribute byte :  Bit# 7 6 5 4 3 2 1 0 Follow Bit Exit RFU RFU RFU RFU RFU RFU **Exit Attribute :** Default Value : 0 : Behaves as a call to the execute element Explanation: The execute element is called, and after returning to the caller, the bytecode continues with the next macro. If Set : 1 : Process the exit event prior to launch the execute element. Explanation: The execute element is called, and instead of returning to the caller, the browser terminates the current session. The effect is the same as if the exit macro (6.2.7.4) is executed after calling the execute element.“	

		Execute element reference	M
2	ExId	Id of the execute element (manufacturer + reference)	
		Input list element	O
1	InputListTag	Tag input list	
1-3	M	Length	
M	Value	Value (made of a mixed list of Variable Reference, Variable Reference List or Inline Value Element)	
		Output List element	O
1	VarRefListTag	Variable Reference List tag	
1-3	N	Length	
N	Value	Value	
ERROR CODES		***DESCRIPTION***	***Action***
No error		OK	NoStop
Reference to undefined		Reference to undefined	Stop
Jump to undefined		Execute element does not exist	Stop
Problem in memory management		Memory problem in the preparation of the structure	Stop
User Abort		Execute was aborted by user	NoStop
Syntax Error		Bad number of parameters passed to the execute element.	Stop
Execution Failure		Execute element generated an internal error.	Stop

Note: The type of each output variable should have one of the 3 following DCS values set: 7 bit gsm default, UCS2, unknown

7 ERROR MANAGEMENT

7.1 Levels of Error Management

There are two predefined levels of error management:

Level 0: User notification and call of specific error handler as mandatory minimum level to be supported.

Level 1: Level 0 + store error code in a log file

7.2 Error indication

For the indication of errors occurring during byte code processing error codes listed in the following table are defined. This information can be accessed using the Status Word environment variable.

Table of predefined Error Codes

The coding of the error codes is in analogy to Execute command done in 2 bytes (selector and information byte)

Type of error	Coding
Communication problem	0x6F01
Syntax error	0x6F02
STK use failed	0x6F03
Jump to undefined	0x6F04
Problem in memory management	0x6F05
Security problem	0x6F06
Reference to undefined	0x6F07
Arithmetics fail	0x6F08
Type mismatch	0x6F09
Out of range	0x6F0A
Temporary problem	0x6F0B
User abort	0x6F0C
Unknown tag	0x6F0D
URL not found	0x6F0E
Execute failed	0x6F0F
General unspecific error	0x6FFF

In the case no error occurs the value 0x0000 is assigned to the Status Word

8 EXECUTE ELEMENTS CODING

The structure of the execute element coding is as follows :

	Values
1st Byte : Manufacturer ID :	0xFF : S@T agreed 0x01 : Giesecke&Devrient 0x02 : Gemplus / Gemalto 0x03 : Sagem Orga 0x04 : Axalto / Gemalto 0x05 : Oberthur Card Systems 0x06 : XPonCard 0x07 : Prism 0xFE : Others
2nd Byte : Execute Element Reference. For S@T agreed execute elements, the coding is specified here. For proprietary execute elements, the coding is maintained by the SIM card provider.	0x01 : ConvertTextPhoneNumberToGSMPhoneNumber 0x02: ComputeValueLength

Coding of the S@T agreed executable modules is given below. The availability of referenced modules in a browser is not mandatory.

8.1 ConvertTextPhoneNumberToGSMPhoneNumber

Description : Allows to convert a text string containing a phone number to a GSM 11.11 dialing number string, before using it in a Setup Call command (for example).

Attribute byte : Exit Attribute not set (call)

Input parameters :

Variable containing a text phone number (result of a user input for example)

Output parameters :

Variable to contain the new phone number format : Length + TON/NPI + swapped nibbles coded according EF_{ADN} of GSM 11.11

Errors :

EXECUTE tag will return "Execution Failure" in case of conversion error. (bad character set, invalid character detected,…)

Remark :

- The input variable must contain only digit characters and optionally "+",."*", "#", and "," characters in SMS alphabet
- The '+' sign or a double zero ("00") as first part of the input variable indicates an international number (TON/NPI will be set to 0x91), in the other case the TON/NPI will be set to national (0xA1).
- The resulting value contains the length in its first byte (the total number of following bytes, including the TON/NPI byte) and is composed of digit characters and "A", "B", "C" signs, as specified in GSM 11.11 (EF_{ADN} coding and extension 1).

Example :

- The execution of the plug-in with an input variable containing the string "+33442365000" will produce in the resulting variable the bytes 0x07 0x91 0x33 0x44 0x32 0x56 0x00 0xF0 and no error will be produced.
- The execution of the plug-in with an input variable containing the string "0033442365000" will produce in the resulting variable the bytes 0x07 0x91 0x33 0x44 0x32 0x56 0x00 0xF0 and no error will be produced.
- The execution of the plug-in with an input variable containing the string "0442365000" will produce in the resulting variable the bytes 0x06 0xA1 0x40 0x24 0x63 0x05 0x00 and no error will be produced.
- The execution of the plug-in with an input variable containing the string "04Z42365000" will not produce anything in the resulting variable and the error "Execution failure" will be produced.

8.2 ComputeValueLength

Description : Allows to compute the length of the input parameter, and store it in a result variable.

Attribute byte : Exit Attribute not set (call)

Input parameters :

Variable containing the text whoes length must be returned

Output parameters :

Variable to contain the length of the input parameter. Range is 0x00 to 0xFF

Errors :

EXECUTE tag will return "Execution Failure" in case of length greater than 255

Remark :

- The input variable may contain any character. The result is the number of bytes contained in the variable, not regarding the variable DCS coding.

Example :

- The execution of the plug-in with an input variable containing the string "Hello World" will set in the resulting variable the byte 0x0B and no error will be produced.
- The execution of the plug-in with an input variable containing the bytes 0x0A 0x0B 0x00 0xFF will set in the resulting variable the byte 0x04 and no error will be produced.
- The execution of the plug-in with an input variable containing more than 255 bytes will not set the resulting variable and the error "Execution failure" will be produced.

9 LIST OF TAGS

Since bit 7 is reserved for attribute indication the tag values can be in range [0x00..0x7F]

Tag	Value
Deck tag	0x01
Deck identifier tag	0x02
SPS tag	0x03
Text element table tag	0x04
Card tag	0x05
Card identifier tag	0x06
Card template tag	0x07
Variable reference tag	0x08
Variable reference list tag	0x09
Inline value tag	0x0A
Input list tag	0x0B
Parameter tag	0x0C
URL tag	0x0D
Address reference tag	0x0E
Constant parameter tag	0x0F

Secure message tag	0x10
Couple tag	0x11
Plugin tag	0x12
Byte code tags	
Initialise variable tag	0x20
Initialise variable selected tag	0x21
Get Environment tag	0x22
Set help tag	0x23
Concatenate tag	0x24
Extract tag	0x25
Encrypt tag	0x26
Decrypt tag	0x27
Go back tag	0x28
Go selected tag	0x29
Switch case tag	0x2A
Exit tag	0x2B
Manage contextual menu item tag	0x2C
STK generic macro tag	0x2D
Execute tag	0x2E

10 Annex: Environment Variables

10.1 Variable Names

The names of the variables are defined without consideration of any naming conventions, e. g. WML naming.

10.2 Variable Attributes

The following attributes define the access rights for the environment variables for any access within a temporary or resident deck with the GET ENV macro.

- Read only

If this attribute is set, the variable can only be read by any deck with the GET ENV macro.

10.3 System Variables

The term “system” here includes the SIM software - especially the implemented browser - and the mobile phone.

10.3.1 SIM Characteristics

The SIM characteristics define the characteristics of the SIM software. The related system variables are “read only” for the gateway.

10.3.1.1 ICCID

The ICCID of the SIM card.

Variable name:	*ICCID*
Access:	Read only
Availability:	Mandatory
Length min:	10

Coding according GSM 11.11 as in EF_{ICCID}. This variable may be useful for key management.

10.3.1.2 SIM Browser Supplier Identifier

The SIM Browser Supplier Identifier defines the SIM browser supplier.

Variable name:	*SIMBrowserSupplier*
Access:	Read only
Availability:	Mandatory
Length:	1

Coding:

0x01 : Giesecke&Devrient

0x02 : Gemplus / Gemalto

0x03 : Sagem Orga

0x04 : Axalto / Gemalto

0x05 : Oberthur Card Systems

0x06 : XPonCard

0x07 : Prism

0xFF : Other

10.3.1.3 SIM Browser Settings

The SIM browser settings are used by the gateway to check and to configurate the browser behaviour concerning application management, memory management, performance and interaction between gateway and browser.

Browser Version:

Variable name: *BrowserVersion*

Access: Read only

Availability: Mandatory

Length: 1

Coding:

High nibble: S@T Version number
Low nibble: Manufacturer Release number

Browser Profile:

The browser profile defines the features supported by the Browser. Each supported feature is indicated by a corresponding flag.

Variable name: *BrowserProfile*

Access: Read only

Availability: Mandatory

Length: 1

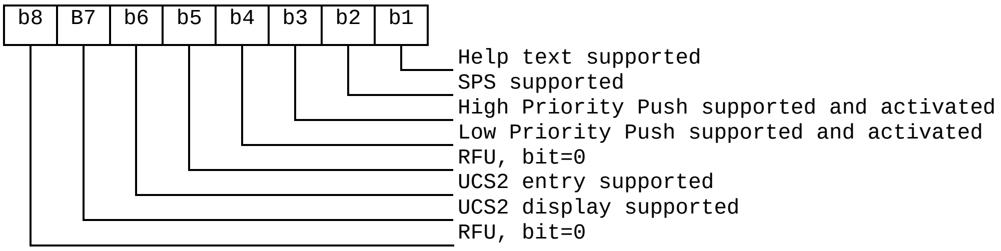
Coding:

1 bit is used to code each facility:

bit = 1: facility supported by the Browser

bit = 0: facility not supported by the Browser

First byte:



During each initialisation procedure the terminal profile is evaluated and a logical ‘and’ operation is being executed between the browser and the mobile facility indication flags, e.g. an indication of an UCS2 entry support means, that both the browser and the mobile phone support this facility.

10.3.1.4 Network Operator Name

The network operator name defines the name of the network operator issuing the SIM.

Variable name:	*OperatorName*
Access:	Read only
Availability:	Optional
Length max:	20

Coding according GSM 03.38

10.3.2 ME Characteristics

The ME characteristics define the characteristics of the mobile phone.

Terminal Profile:

The Terminal Profile contains the response data to the command “Terminal Profile”. This variable is set by the SIM at each Reset during the initialisation procedure.

Variable name:	*TerminalProfile*
Access:	Read only
Availability:	Mandatory
Length :	variable

Coding according GSM 11.14

10.4 Runtime Variables

Status Word:

The status word indicates the result of the previous executed byte code macro.

Variable name:	*StatusWord*
Access:	Read only
Availability:	Mandatory
Length	2

Coding according to table in chapter 7.2.

Location Information:

The location information defines the location of the mobile user at the time of using the service.

Variable name:	*LocationInformation*
Access:	Read only
Availability:	Optional
Length	7

Coding according GSM 11.14 (Location Information):

Byte 1-3: Mobile Country and Network Codes (MCC & MNC)

Byte 4-5: Location Area Code (LAC)

Byte 6-7: Cell Identity Value (Cell ID)

Each time this variable is accessed, the browser will use a 'Provide Local Information' command with command qualifier 0x00 to get the actual value of this variable

10.5 User Preferences

10.5.1 User Type

The user type defines the mode, dynamic applications are to be executed.

Three possible user types “Beginner”, ”Advanced User” and “Expert” result in three execution modes:

- Beginners mode
- Advanced mode
- Experts mode

In the beginners mode the application provides full help information to the mobile user. In the advanced mode help information is only partly available. In the experts mode help information is very limited.

This variable can be accessed by a resident application provided to the mobile user.

Variable name:	*UserType*
Access:	Read only
Availability:	OptionalMandatory
Availability:	Optional
Length:	1

Coding:

0x00: Beginner
0x01: Advanced
0x02: Expert

10.6 List of all Environment Variables

Variable name	Availability	Access	Description	Id
ICCID	Mandatory	Read only	ICCID of the SIM	0x00
SIMBrowserSupplier	Mandatory	Read only	Identifier of the SIM browser supplier	0x01
BrowserVersion	Mandatory	Read only	Version of the SIM browser	0x02
BrowserProfile	Mandatory	Read only	List of supported facilities of the browser	0x03
OperatorName	Optional	Read only	Name of the network operator (SIM card issuer)	0x04
TerminalProfile	Mandatory	Read only	Terminal Profile of the currently used ME	0x05
StatusWord	Mandatory	Read only	Result of the previous macro	0x06
LocationInformation	Optional	Read only	Location information	0x07
UserType	Optional	Read only	Mode of the user interface	0x08
IMEI	Optional	Read only	IMEI of the mobile equipment	0x09
NMR	Optional	Read only	Network Measurement Results	0x0A

Identifiers from 0x09 to 0x3F are reserved values.

Note:

Outside the scope of this specification the following identifier ranges are defined:

0x40 – 0x4F	Administrative Mode Identifiers
0x50 – 0x5F	Operational Mode Identifiers
0x60 – 0x6F	Administrative Mode Proprietary Identifiers
0x70 – 0x7F	Operational Mode Proprietary Identifiers
0x80 –0xFF	RFU

11 Annex: Optional Features

The following features of SBC are optional:

- SPS (Service Permanent Store)
- Implementation of Execute Elements
- Bookmark in Contextual Menu

- Pause item in Contextual Menu
- Push (low and high priority)

12 History

Document history		
Release	Approved by	Comment
1.0.0	SIM Alliance TDG	First internal release
1.0.1	SIM Alliance TDG	Changes after meeting 16 (new and unified error codes, CRs: G+#1, SLB#1)
1.0.2	SIM Alliance TDG	Changes after meeting 17, 18 CRs: 10001, 10002, 10003, 10005, 10008, 10009, 10011, 10012, 10013, 10021, 10022, 10023, 10024, 10026, 10027, 10028, 10029, 10019, 10020, 10031, 10032, 10033, 10034, 10035, 10039, 10041, 10019, 10040 integrated
1.0.3	SIM Alliance TDG	after meeting 18a and 19 CRs: 10042, 10030, 10038, 10044, 10047, 10049, 10050, 10054, 10055, 10056 integrated
1.0.4	SIM Alliance TDG	after meeting 25 CRs: 10084, 10090, 10091, 10096, 10099 integrated
1.0.5	SIM Alliance TDG	after meeting 27 CRs: 10083, 10098, 10106, 10107, 10108, 10112 integrated
1.0.6	SIM Alliance TDG	after meeting 29 CRs: 10121, 10122 integrated
1.0.7	SIM Alliance TDG	Editorial changes for publication
2.0.0	SIM Alliance TDG	Clarifications according to change request forum in section 5.3.7, 6.2.10, 6.2.6, June 2004
3.0.0	SIM Alliance TDG	CRs: 2006-12, 2006-32, 2006-33, 2006-34 integrated

12.1 Annex: List of CRs [informative]

CR Number	CR Identifier	Subject	Document Reference	Status / Meeting No.
10001	SLB-WG1-APRIL-2000#1	REMOVE THE START CARD REFERENCE	S@T 1.00 V1.0.1	Accepted #17
10002	SLB-WG1-APRIL-2000#2	CHANGE URL FORCED LOCAL TO FORCED RESIDENT	S@T 1.00 V1.0.1	Accepted #17
10003	SLB-WG1-APRIL-2000#3	ADD A TAG FOR SECMSG	S@T 1.00 V1.0.1	Accepted #17
10005	G&D-1-04-2000#1	“#”-CHARACTER IN LONG DECK NAMES	S@T 1.00 V1.0.1	Accepted #17
10008	G&D-1-04-2000#4	Reset Help String	S@T 1.00 V1.0.1	Accepted #17
10009	G&D-1-04-2000#5	Extract with Length	S@T 1.00 V1.0.1	Accepted #17
10011	ORGA-WG1-APRIL-2000#1	Additional Error – Unknown tag	S@T 1.00 V1.0.1	Accepted #17
10012	GEMPLUS-WG1-MAY-2000#1	ADD AN ITEM FOR PAUSE IN CONTEXTUAL MENU	S@T 1.00 V1.0.1	Accepted #17
10013	GEMPLUS-WG1-MAY-2000#2	REMOVE A SENTENCE	S@T 1.00 V1.0.1	Accepted #17
10019	GEMPLUS-WG1-MAY-2000#17.6	ENCAPSULATION OF COUPLE STRUCTURES	S@T 1.00 V1.0.1	Accepted #18
10020	GEMPLUS-WG1-MAY-2000#17.7	CARD ONLY ATTRIBUTE IN URL REFERENCE	S@T 1.00 V1.0.1	Accepted #18
10021	G&D-WG1- MAY-2000#1	Editorial change for SET Help	S@T 1.00 V1.0.1	Accepted #18
10022	G&D-WG1- MAY-2000#2	Editorial change for Deck	S@T 1.00 V1.0.1	Accepted #18
10023	G&D-WG1- MAY-2000#3	Editorial change for init_variables_Selected and go_selected	S@T 1.00 V1.0.1	Accepted #18
10024	G&D-WG1- MAY-2000#4	Editorial change for variable ref list	S@T 1.00 V1.0.1	Accepted #18
10026	G&D-WG1- MAY-2000#6	ICCID Mandatory	S@T 1.00 V1.0.1	Accepted #18
10027	G&D-WG1- MAY-2000#7	Clarify coding of Location Information	S@T 1.00 V1.0.1	Accepted #18
10028	G&D-WG1-	Remove Substitution, Output, security	S@T 1.00	Accepted #18

	MAY-2000#8		V1.0.1	
10029	GEMPLUS-WG1-MAY-2000#18.1	Contents of the Length byte in TL[A]V struct	S@T 1.00 V1.0.1	Accepted #18
10030	GEMPLUS-WG1-MAY-2000#18.2	Editorial change for init_variables_Selected and go_selected	S@T 1.00 V1.0.1	Pending
10031	GEMPLUS-WG1-MAY-2000#18.3	RFU indication after Follow bit in Attribute field	S@T 1.00 V1.0.1	Accepted #18
10032	ORGA-WG1-MAY-2000#18.1	Term “result string” in STK Generic Macro	S@T 1.00 V1.0.1	Accepted #18
10033	ORGA-WG1-MAY-2000#18.2	Browser behaviour on empty variables	S@T 1.00 V1.0.1	Accepted #18
10034	SLB-WG1-MAY-2000#2	ORDER OF SIGNATURE VS ENCRYPTION IN ENCRYPT/DECRYPT MACROS	S@T 1.00 V1.0.1	Accepted #18
10035	SLB-WG1-MAY-2000#1	PRECISE USE OF FORCE RESIDENT FLAG IN URL REFERENCE	S@T 1.00 V1.0.1	Accepted #18
10038	ORGA-WG1-MAY-2000#18.4	ADDRESS REFERENCE IN CARD ID ELEMENT OF A CARD ELEMENT	S@T 1.00 V1.0.1	Accepted #18a
10039	GEMPLUS-WG1-MAY-2000#18.8	OUTPUT VARIABLE AS DEFAULT VARIABLE	S@T 1.00 V1.0.1	Accepted #18
10040	GEMPLUS-WG1-MAY-2000#18.5	COUPLE USAGE IN CONTEXTUAL MENUS	S@T 1.00 V1.0.1	Accepted #18
10041	GEMPLUS-WG1-MAY-2000#18.6	GET ENV MACRO PARAMETERS ORDER	S@T 1.00 V1.0.1	Accepted #18
10042	GEMPLUS-WG1-MAY-2000#18.7	OPTIONAL / MANDATORY FEATURES	S@T 1.00 V1.0.1	Accepted #18
10044	GEMPLUS-WG1-MAY-2000#18.10	EXTRACT MACRO PARAMETERS ORDER	S@T 1.00 V1.0.1	Accepted #18a
10047	ORGA-WG1-JUNE-2000#18a.1	ADD AN OPTIONAL TYPE INFORMATION IN THE INLINE VALUE STRUCTURE	S@T 1.00 V1.0.2	Accepted #18a
10049	GEMPLUS-Thread1-June-2000#18a.1	Standard Plug-In Definition (Phone Number Converter)	S@T 1.00 V1.0.2	Accepted #18a
10050	GEMPLUS-Thread1-June-2000#18a.2	Change the Default Visibility of the "Go To Home Page" Item in Contextual Menu to True	S@T 1.00 V1.0.2	Accepted #18a
10054	GIESECKE & DEVRIENT-WG1-JUNE28-2000#1	Editorial change for dynamic / static	S@T 1.00 V1.0.3	Accepted #19
10055	SCHLUMBERGER-WG1-JUNE-2000#19.1	ERROR: Change of tag value for byte code	S@T 1.00 V1.0.3	Accepted #19
10056	GIESECKE & DEVRIENT-WG1-	Clarification and changes for crypto macros	S@T 1.00	Accepted #19

	JUNE28-2000#2		V1.0.3	
10084	SCHLUMBERGER-WG1-OCTOBER-2000#1	Clarification of 'go back' macro	S@T 1.00 V1.0.4	Accepted #22
10090	ORGA-WG1-OCTOBER-2000#22.1	Change Range for Extract Length	S@T 1.00 V1.0.4	Accepted #25
10091	ORGA-WG1-OCTOBER-2000#22.2	Consideration of DCS for Extract Length and Startindex	S@T 1.00 V1.0.4	Accepted #25
10096	SCHLUMBERGER-WG1-JANUARY-2001#1	DCS for 'SMS Alphabet'	S@T 1.00 V1.0.4	Accepted #25
10099	GEMPLUS-WG1-JAN-2001#1	SecMsg Format	S@T 1.00 V1.0.4	Accepted #25
10083	SCHLUMBERGER-WG1-SEPTEMBER 2000#5	PAUSE ITEM IN CONTEXTUAL MENU OPTIONAL	S@T 01.00 V1.0.5	Accepted #26

10098	GEMPLUS-FEB-2001#26-0	Standard Plug-in definition (COMPUTE VALUE LENGTH**)**	S@T 1.00 V1.0.5	Accepted by eMail 05.03.01
10106	SCHLUMBERGER-WG1-MAR-2001#2	clarification of “SET_HELP” macro	S@T 01.00 V1.0.5	Accepted by eMail 05.03.01
10107	10107 GEMPLUS-WG1-MAR-2001#1	VARIABLE/TEXT ELEMENT length	S@T 01.00 V1.0.5	Accepted by eMail 05.03.01
10108	GEMPLUS-WG1-MAR-2001#2	Enciphered or ClearText DataBlock field in Sec msg Tag	S@T 01.00 V1.0.5	Accepted by eMail 05.03.01
10112	GEMPLUS-WG1-MAR-2001#3	Optional MAC Element in SecMsg Tag	S@T 01.00 V1.0.5	Accepted #27
10121	GEMPLUS-WG1-MAY-2001#29	Push Support Indication in Browser Profile	S@T 01.00 V1.0.6	Accepted #29
10122	GEMPLUS-WG1-JUNE-2001#1	System Item in Contextual Menu for Push Activation/Deactivation	S@T 01.00 V1.0.6	Accepted by eMail 21.06.01
2004-009	CR_AXALTO_S@T_NUMBER009	6.2.6 EDITORIAL CHANGE IN ENCRYPT DECRYPT MACRO	S@T 01.00 V2.0.0	Accepted by email voting Feb 2004
2004-011	CR_AXALTO_S@T_NUMBER011	Replace Schlumberger by Axalto	S@T 01.00 V2.0.0	Accepted by email voting Feb 2004
2004-023	CRG&D01_01.00_SBC_5.3.7-S@T_SPECS2003	5.3.7 ALLOWED MACROS IN TEMPLTES	S@T 01.00 V2.0.0	Accepted by email voting Feb 2004
2004-026	CRG&D04_01.00_SBC_6.2.10-S@T_SPECS2003	6.2.10 DEFINITION OF FUNCTIONALITY “EXIT”	S@T 01.00 V2.0.0	Accepted by email voting Feb 2004
2006-032	CR_GEMPLUS_004_IMEI_V002	10.6 IMEI environment variable	S@T 01.00 V3.0.0	Accepted by email voting Feb 2006
2006-033	CR_GEMPLUS_005_NOWAIT_V002	5.5.7 Extend the URL reference macro with nowait attributes	S@T 01.00 V3.0.0	Accepted by email voting Feb 2006
2006-034	CR_GEMPLUS_006_PLUG-INSVARTYPE_V002	6.2.10 Plug-ins Variable Types	S@T 01.00 V3.0.0	Accepted by email voting Feb 2006
2006-012	CR_AXALTO December 2004 #015	Administrative Plug-in Management cmd	S@T 01.00 V3.0.0	Accepted by email voting Feb 2006