3GPP TSG-T (Terminals) Meeting #13
Beijing, China, 19 - 21 September, 2001

Tdoc TP-101208

Presentation of Specification

Presentation to:	**TSG-T Meeting #13**
Document for presentation:	**TS 31.113 v2.0.0 "USAT Interpreter; Byte Codes" Rel-5**
Presented for:	**Approval**

Abstract of document:

The document specifies the byte codes that are recognised by an USAT Interpreter. The byte codes primary purpose is to provide efficient programmatic access to the SIM Application Toolkit commands.

Changes since last presentation to TSG-T Meeting #11:

The main changes are enhancements to the sections on "USAT Interpreter data structures" and "Example of Accessing USAT Interpreter Functionality in Wireless Mark-up Language".

Outstanding Issues:

none

Contentious Issues:

none

3GPP TS 31.113 V2.0.0 (2001-09)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Terminals; USAT Interpreter Byte Codes (Release 5)

Keywords
USIM, Toolkit

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document specifies the byte codes that are recognised by an USAT Interpreter. The byte codes primary purpose is to provide efficient programmatic access to the SIM Application Toolkit commands.

The design objectives of the byte code set are:

- Compact representation for efficient transmission over the air interface.
- Minimisation of USAT Interpreter complexity to minimise SIM footprint and ease compliance testing.
- Easily configured and extended.
- Source language independent although XML-style mark-up languages are explicitly envisioned.
- Transport bearer independent (e.g. SMS, GPRS...)
- Transport protocol independent.
- Independent from design of external entities.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.

For a non-specific reference, the latest version applies.

[1] 3GPP TS 31.111: "3rd Generation Partnership Project (3GPP); USIM Application Toolkit (USAT)"

[2] 3GPP TS 31.114: "3rd Generation Partnership Project (3GPP); USAT Interpreter Protocol and Administration"

[3] 3GPP TS 23.038: "3rd Generation Partnership Project (3GPP); Alphabets and language-specific information"

[4] SCP TS 102 221: "Smart cards; UICC-Terminal interface; Physical and logical characteristics"

[5] ISO/IEC 7816-6 (1995): "Identification cards – Integrated circuit(s) cards with contacts, Part 6: Inter-industry data elements"

[6] ISO 8731-1:1987 "Banking – Approved algorithms for message authentication – Part 1: DEA".

[7] ISO/IEC 10116:1997 "Information technology – Security techniques – Modes of operation for an n-bit block cipher".

[8] Schneier, Bruce: "Applied Cryptography Second Edition: Protocols, Algorithms and Source code in C", John Wiley & Sons, 1996, ISBN 0-471-12845-7.

[9] IETF RFC 1738: "Uniform Resource Locators (URL)" : T. Berners-Lee, et al., December 1994.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

anchor: A named location on a page to which references can be made and at which rendering by the USAT Interpreter is initiated. Anchors can be referenced by anchor reference TLVs.

attribute: A property assigned to a TLV. The attribute can consist of a single bit or of a sequence of consecutive bits within the attribute bytes of a TLV.

attribute byte(s): A sequence of consecutive bytes in the value part of a TLV containing the attributes of that TLV.

current page: The page which is currently rendered by the USAT Interpreter.

external system entity: Any entity outside the USAT Interpreter, able to communicate with the USAT Interpreter (e.g. USAT Gateway, content/application system).

general result range: A general result range is a range of general results in the terminal response of an USAT command (refer to 3GPP TS 31.111 [1]).

navigation unit: A block of a service description that can be referenced (by its anchor) and hence independently activated.

page: The context of an USAT Interpreter rendering, the default scope of USAT Interpreter variables and the unit of transmission between an external system entity and the USAT Interpreter.

protected variable: A shared variable, which is protected by an one time password.

service: A collection of pages that defines an unitary capability of the mobile equipment from the point of view of the user. Examples include remote database access, electronic mail, and alerts.

service ID: Unique ID to identify a service on the external system entity.

shared variable: A variable to be shared with the following page. Shared variables can be provided to the next page in a protected or non protected manner.

string pool: A list of predefined variables provided by the current page within the page TLV. The string pool is mainly used for optimisation purposes.

variable ID: Identifier to reference a variable within a variable usage area.

wait state: The state which is possibly entered by the USAT Interpreter to wait for a response from the external system entity after information has been submitted to the external system entity.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply

C	Conditional
DCS	Data Coding Scheme
ID	IDentifier
KIc	Key and algorithm Identifier for ciphering
LSB	Least Significant Bit
M	Mandatory
MSB	Most Significant Bit
NCI	Native Code Identifier
NU	Navigation Unit
O	Optional
OTP	One Time Password
SMS	Short Message Service

SW1/SW2	Status Word 1 / Status Word 2
TLV	Tag Length Value
TR	Terminal Response
TS	Technical Specification
UCS2	Universal two byte coded Character Set
UE	User Equipment
URL	Uniform Resource Locators
USAT	USIM Application Toolkit
USIM	Universal Subscriber Identity Module
XML	eXtensible Markup Language

3.3 Symbols

'0' to '9' and 'A' to 'F' The sixteen hexadecimal digits.

Single bits are identified by b1 to b8, where b1 is the LSB and b8 is the MSB of the byte containing the bit.

RFU bits and bytes are to be set to '0'.

4 Model of computation

A *service* is mobile device (user equipment) functionality as seen by the user, for example e-mail, information access or order entry.

A service is composed of one or more *pages*. Pages describe information presented to the subscriber and retrieve input from the subscriber. The unit of transmission to the user equipment as well as the unit of USAT Interpreter interpretation is the page. The set of all pages describing a service is called the *service description*.

Pages are composed of navigation units. Anchors reference the beginning of navigation units. Therefore anchors are points in a service description that can be branched to from other points in the service description. Each page has an implicit anchor at the beginning of the page.

In some mark-up languages pages are known as decks and anchors are known as cards.

The USAT Interpreter renders pages and provides a way to navigate from within pages to anchors belonging to the same page or other pages. The requirements of the USAT Interpreter include a way to automatically go back to previously visited anchors.

The USAT Interpreter manages a stack of N last anchors visited. Each anchor visited is added to this history list, except if an appropriate flag is set in the anchor. The back operation is interpreted relative to this history list and means go to the preceding anchor in the list.

When reaching the last byte code of a page, the USAT Interpreter shall behave like ending a navigation unit.

4.1 Navigation

A page expressed as compiled byte code instructions is stored as a unit in the USAT Interpreter. The page is the smallest unit that the external system entity can provide to the USAT Interpreter. A page is partitioned into one or more navigation units each of which can be referenced using anchors. In other words, navigation units and anchors are included in pages.

The anchor is defined as being the elementary navigation target. The USAT Interpreter can skip from one anchor to another, backwards and forwards based either on control flow constructs or user interaction. If a navigation unit contains no instructions to branch to an anchor within the current page or another page, the behaviour of the USAT Interpreter is defined by the terminal response handler mechanism. This keeps the proactive session alive and allows further navigation.

Pages are stored in the USAT Interpreter. The structure of pages is described later in the present document. These pages are stored either permanently in the USAT Interpreter or received and interpreted on the fly.

Pages and navigation units are referenced using anchor references as described below.

To be able to create multiple-page services, page references within USAT Interpreter commands are used to fetch new pages or to link pages together.

The behaviour of the USAT Interpreter in response on user interaction (e.g. backward move, proactive session terminated, help information requested) is defined by the current terminal response handler configuration. The terminal response handler configuration can be modified by a terminal response handler modifier within the page or navigation unit context.

If no terminal response handler modifier is defined in the page context or in the navigation unit context, the default terminal response handler configuration shall be used.

4.2 Communication with the external system entity

This chapter provides an overview of the communication of the USAT Interpreter with the external system overview. The present document describes the format of content exchanged between the external system entity and the USAT Interpreter. The protocol and bearer used for the communication is outside the scope of the present document.

4.2.1 Incoming pages from the external system entity

Any information obtained by the USAT Interpreter from the external system entity shall be formatted as a Page TLV. After obtaining a Page TLV from the external system entity the USAT Interpreter shall start rendering the obtained page according to the present document.

4.2.2 Outgoing data to the external system entity

The submission of outgoing data can be triggered by the USAT Interpreter byte codes

- Assign and Branch and
- Branch on Variable Value.

A service can trigger the submission of outgoing data by providing a Page Reference TLV containing a Submit Configuration TLV within the byte codes mentioned above.

The Submit Configuration TLV contains the parameters to be used to build a Submit TLV structure, which will be provided to the external system entity then.

The Submit TLV structure is used only in the direction from the USAT Interpreter to the external system entity. All information provided by the USAT Interpreter to the external system entity shall be formatted as a Submit TLV structure. The Submit TLV structure consists of a Submit Data TLV and optionally of a Page Identification TLV.

The Submit Data TLV is used in two forms:

- In the direction from the external system entity to the USAT Interpreter, the value part of the Submit Data TLV contained in the Submit Configuration TLV may consist of any byte sequence possibly containing variable references.
- In the direction from the USAT Interpreter to the external system entity, all variable references within the Submit Data TLV contained in the Submit Configuration TLV are substituted according to method 2 in chapter "6.3 Variable substitution". The resulting Submit Data TLV containing the substituted variable references with variable content shall then be used within the Submit TLV to be submitted by the USAT Interpreter to the external system entity.

4.2.2 Wait State

When rendering a Page Reference TLV containing a Submit Configuration TLV having the "ProcessingBehaviour" attribute not set, the USAT Interpreter shall perform the following actions:

- generate a new RequestID value, by incrementing the RequestID value. If the Request ID value reaches its maximum value, the RequestID value shall start at 0 again.

- provide the RequestID to the protocol layer to be incorporated into the transport protocol (refer to 3GPP TS 31.114 [2])
- provide the Submit TLV to the protocol layer to be transmitted to the external system entity (see chapter "4.2.2 Outgoing data to the external system entity")
- enter the wait state

In the wait state, the USAT Interpreter shall keep the proactive session alive. Therefore, a DISPLAY TEXT USAT command shall be issued by the USAT Interpreter to notify the user that the USAT Interpreter has entered the wait state.

The text to be used for the text string of the DISPLAY TEXT command shall be taken from the Inline Value TLV of the Submit Configuration TLV requesting the wait state.

If this Inline Value TLV is not available in the Submit Configuration TLV when entering the wait state, then a default text shall be taken by the USAT Interpreter. This default text can be personalised and later on changed by administrative means.

For the DISPLAY TEXT USAT command the command qualifier option

"clear message after delay"

shall be used.

The USAT Interpreter shall handle the wait state according to the following state diagram:

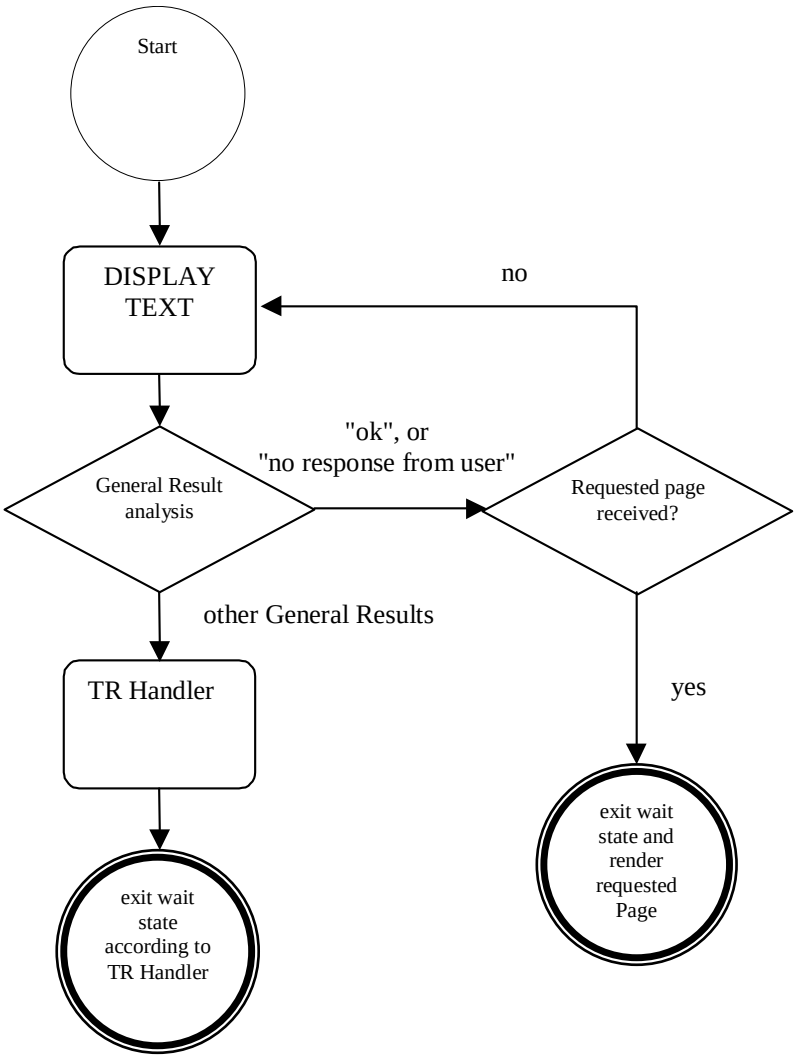


Figure 4.1: State diagram

The terminal response handler is activated by the USAT Interpreter, when the general result range of the DISPLAY TEXT command is not '00 0F' ("ok") and not '12 12' ("no response from user"). The terminal response handler shall use the current terminal response handler configuration.

Incoming pages shall be handled as follows:

When getting a page during the wait state being active, the protocol layer shall check the received RequestID:

- If the provided RequestID does not match the expected RequestID, the page is discarded and the wait state remains active. The current page is not affected by the discarded page.

- If the provided RequestID does match the expected RequestID, the wait state is terminated by the USAT Interpreter and the received page is rendered.

If the wait state has been terminated before the expected RequestID has been received (e.g. the wait state was cancelled by the user, the UE was switched off...), the protocol layer shall discard all incoming PULLed pages (see chapter "11 Modes of operation").

4.3 Terminal response handler mechanism

For any general result of an USAT command, the USAT Interpreter shall branch to the terminal response handler. The terminal response handler will match the general result with the currently defined general result ranges and process the corresponding action.

If several actions are assigned to a general result range, a SELECT ITEM command shall be built from the action list using the action description by the USAT Interpreter to allow the user to choose between actions.

In case of an exception of the USAT Interpreter or no corresponding general result range to the general result of an USAT command, a default action will apply. This default action could be changed by using the terminal response handler modifier with the reserved general result range 'FF FF'.

Exception example:

- no more byte code when process next byte code (e.g. end of navigation unit)

A default terminal response handler configuration is defined in the present document and can be modified by administrative means or at personalization stage.

The default terminal response handler configuration can be modified temporarily by the terminal response handler modifier (e.g. to hide default entries by using action IDs, to add new ones or to modify existing entries).

If the USAT Interpreter branches to another page due to the terminal response handler configuration, the standard inter page variable management shall apply.

Default terminal response handler configuration:

		Action ID	General result range								
			'FF FF' default	'14 14' USSD/SS transaction terminated	'00 0F' ok	'13 13' help request	'12 12' no response from user	'11 11' backward move requested	'10 10' quit	'20 2F' worth to re-try	'30 3F' not worth to re-try
System actions	process next byte code	'00'			X						
	quit USAT Interpreter without user confirmation	'01'	X	X			X		X	X	X
	go back one entry in history list	'02'						X			
	retry last proactive command within current USAT Interpreter navigation unit	'03'				X				X (*1)	

NOTE (*1): In the case of SET UP CALL, the system action "retry last proactive command within current USAT Interpreter navigation unit" should be deactivated by the service.

4.4 Activation

The USAT Interpreter can be activated in different ways:

- locally from the UE using menu selection,
- locally from the UE as the result of an event,
- by an incoming page as a result of a previous page request from the USAT Interpreter,
- by an incoming page initiated by an external system entity ("push") or
- optionally by an internal application using a proprietary interface.

The rendering of a page shall be independent of the means of activation.

With respect to activation locally from the UE using menu selection, the SETUP MENU command as described in 3GPP TS 31.111 [1] can contain one or more links to a Page Reference TLV which identifies a locally or remotely stored page. When one of these identifiers is selected, the USAT Interpreter is activated and renders the referenced page, local or remote. Registering of pages to the main menu is up to administrative means.

An event (as specified in 3GPP TS 31.111 [1] or proprietary events defined by the card issuer) is linked to a Page Reference TLV which identifies a locally or remotely stored page. When the UE sends an ENVELOPE command containing an event, the USAT Interpreter is activated and renders the referenced page, local or remote. If an event is received not referencing to a page, the event shall be ignored by the USAT Interpreter. For security reasons, setting up events is up to administrative means.

If an event occurs while the USAT Interpreter renders another page or while the USAT Interpreter is in wait state, the USAT Interpreter shall queue the event and shall postpone executing the event until the USAT Interpreter is neither in wait state nor rendering a page.

The USAT Interpreter shall be able to queue at least one event. Events shall be executed in the order the events have been occurred.

If the USAT Interpreter is not able to store an event (e.g. because the event queue is full already), it is up to the implementation of the USAT Interpreter to handle this situation.

4.5 Page format overview

Figure 4.2 gives an overview of the construction and elements of a page to be rendered by the USAT Interpreter.

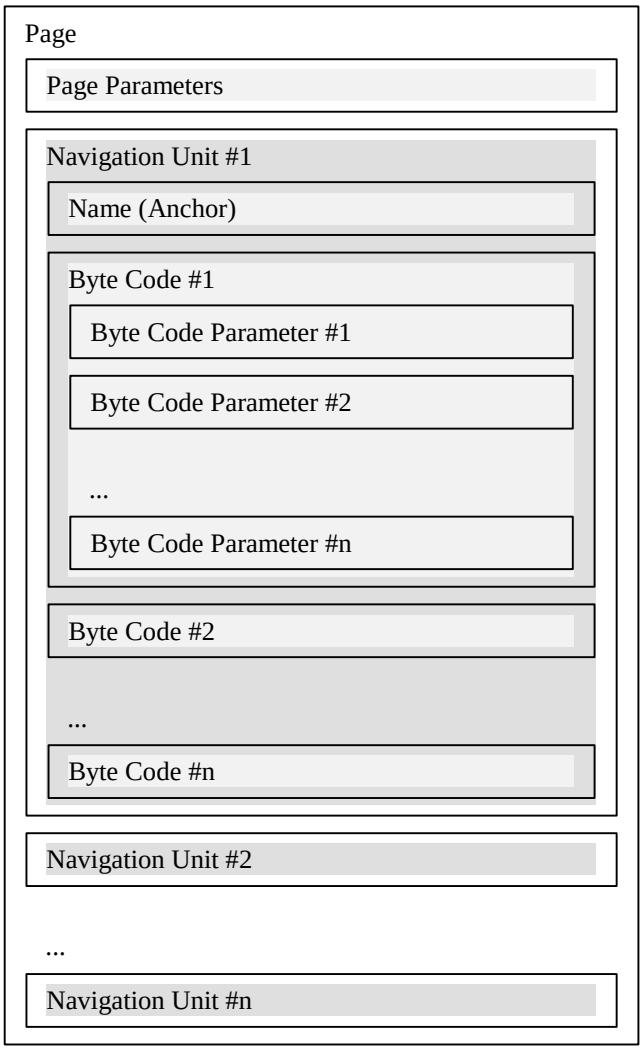


Figure 4.2: Overview of page format

A transmission initiated by the USAT Interpreter to the external system entity is performed when the USAT Interpreter executes a byte code containing a Page Reference TLV containing a Submit Configuration TLV (see chapter "7.9.3 Submit Configuration") referring to a page which is not locally stored.

Page Reference TLVs are used in the following byte code commands:

- Assign and Branch
- Branch on Variable Value

5 TLV Format

The Tag Length Value (TLV) is the basic data structure element. If the value part of a TLV contains other TLV elements it is called a BER-TLV or a template TLV. If not, it is called a simple TLV. Refer to ISO/IEC 7816-6 [5] for more information on data objects.

The tag byte contains a seven-bit tag value and an attribute byte-present bit in the MSB. If the attribute byte-present bit is set then the leading byte(s) in the value field contain attribute information for the element identified by the tag.

Length	Value	Description	M/O
1	T	Tag	M
1-3	L	Length of following data, a length value of '00' is allowed	M
L	V	The data value associated with the tag	O

The length is BER coded onto 1, 2 or 3 bytes according to ISO/IEC 7816-6 [5].

The value of a TLV is the content of its value field and therefore *evaluation* of a TLV yields its value.

TLVs shall appear in the order given in the present document. Additional TLVs may be appended to the TLVs given in the present document. If TLVs are expected by the USAT Interpreter and are missing, the USAT Interpreter shall generate an error message to the user. TLVs not supported by the USAT Interpreter shall be ignored by the USAT Interpreter.

5.1 Coding of the tag byte

The tag byte of all TLVs described in the present document is as follows:

b8	b7	b6	b5	b4	b3	b2	b1
Attribute byte present bit	Tag value coded on 7 bits						

Attribute byte present bit	Value
Attribute byte present as first byte of V	1
Attribute byte not present as first byte of V	0

5.2 Attributes in TLVs

Every TLV can have one or more attributes bytes if indicated by the attribute byte present bit of the tag byte. The coding of an attribute byte is shown below. Attributes provided in the attribute byte shall be related to the belonging TLV. The meaning of the attributes of a TLV is TLV specific and specified in the TLV descriptions.

An attribute given in an attribute byte can consist of a single bit or a combination of consecutive bits forming an attribute value.

The default value of an attribute value or an attribute bit within an attribute byte is always '0'. The '0' value of an attribute shall be used by the USAT Interpreter, if the attribute is not available in the TLV.

Whenever the attributes for a tag require more than 7 bits within an attribute byte, the number of attribute bytes will be extended. The extension of the attribute byte shall be indicated by the MSB of the attribute byte, which is called the follow bit.

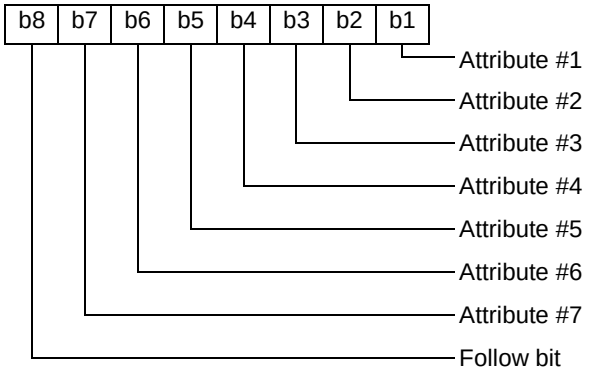
Attributes or attribute bytes not expected or not known by the USAT Interpreter shall be ignored by the USAT Interpreter.

5.3 Coding of attribute bytes

The MSB of each attribute byte indicates if another attribute byte follows or not. The MSB is called follow bit. The remaining seven bits of an attribute byte contain TLV specific attributes, either coded as a single bit or as a combination of consecutive bits.

The context, namely the tag, completely determines the order, span and semantics of the bit-packed attribute values. An attribute consisting of more than 1 bit may span two attribute bytes but this situation should be avoided if possible.

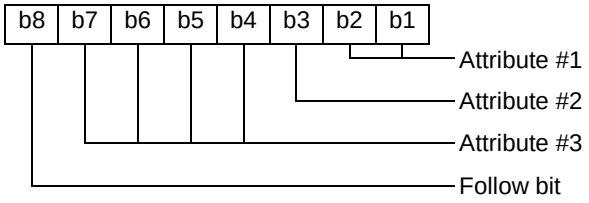
General coding:



Follow bit coding:

Follow bit	Value
Another attribute byte available as next byte of V	1
No more attribute bytes available	0

Other coding example where attribute #1 consists of a single bit, attribute #2 consists of a 4 bit value and attribute #3 consists of a 2 bit value.



6 Variables

Variables are name-value pairs. The name is called the variable identifier (ID) and the value is called the variable value. Operations are provided to refer to a variable value by using its variable ID and for setting and resetting the value associated with a variable.

Variables can be stored in the following usage areas:

- Environment variable area;
- Permanent variable area;
- Temporary variable area;
- Page string element.

Variables have one of the following variable types:

- SMS default 7-bit coded alphabet as specified in 3GPP TS 23.038 [3] with bit 8 set to 0;
- SMS default 7-bit coded alphabet as specified in 3GPP TS 23.038 [3] packed;
- Binary;
- UCS2 coded string.

The list can be extended.

6.1 Usage areas

Variables are referred by using an unified one byte notation. The one byte variable reference is called the variable ID. b8 and b7 of the variable ID are used to indicate the belonging of a variable to a certain usage area. The remaining 6 bits are used to reference a certain variable within the usage area.

Due to the used coding, the number of variables per area is restricted to 64.

The coding of the variable ID is as follows:

b8	b7	b6	b5	b4	b3	b2	b1	
0	0							belongs to Environment usage area
0	1							belongs to Permanent usage area
1	0							belongs to Temporary usage area
1	1							belongs to Page String Element usage area
		x	x	x	x	x	x	identifier of the variable within the usage area

The size of the different usage areas is to be defined by the card issuer and configured during the personalisation process of the USAT Interpreter.

6.1.1 Environment variable usage area

This usage area consists of 3 different partitions:

- USAT Interpreter system information partition;
- USIM issuer information partition;
- End user information partition.

6.1.1.1 USAT Interpreter system information partition

The USAT Interpreter partition is preloaded during the manufacturing process of the USIM or during the runtime of the USAT Interpreter.

At least the following information shall be stored:

Variable ID	Description	Coding
'00'	ICCID of UICC	Binary coding as for EF_{ICCID} specified in SCP TS 102 221 [4]
'01'	USAT Interpreter version	**Byte 1: Issuer Version** USAT Interpreter issuer specific version. The coding and value of this byte depends on the USAT Interpreter issuer. The USAT Interpreter issuer is stored in variable '07' and variable '08'. **Bytes 2-3: TS 31.113, Version (this TS)** Byte 2: first digit (x according to the foreword of the present document) of the version of the supported 3GPP TS 31.113; BCD coded Byte 3: second digit (y according to the foreword of the present document) of the version of the supported 3GPP TS 31.113; BCD coded **Bytes 4-5: Version of TS 31.114 [2]** Byte 2: first digit (x according to the foreword of the present document) of the version of the supported 3GPP TS 31.114; BCD coded Byte 3: second digit (y according to the foreword of the present document) of the version of the supported 3GPP TS 31.114; BCD coded further bytes are RFU Example: Issuer version: '22' TS 31.113 version: 5.2.0 TS 31.114 version: 5.12.3 resulting coding: '22 05 02 05 12'
'02'	USAT Interpreter profile (features)	According to administrative configuration coding as specified in TS 31.114 [2]. This includes the list of supported and forbidden USAT Commands.
'03'	USAT Interpreter Native Commands	List of supported native commands. Coding: Sequence of NCIs. Each NCI coded in 2 bytes.
'04'	Terminal Profile as got at runtime	Binary coded as defined in 3GPP TS 31.111 [1] for TERMINAL PROFILE
'05'	Error Code as generated by the last byte code command executed	Binary coded as specified in chapter "12 Error coding"
'06'	Maximum page size for temporary storage of one page	Binary coded, number of bytes available for page storage, coded in 2 bytes indicating the number of blocks of memory consisting of 128 bytes each, most significant byte first
'07'	USAT Interpreter issuer identification	URL of USAT Interpreter issuer, coding according to RFC 1738 [9] <host> of URL.
'08'	Hash Value of URL of USAT Interpreter issuer identification	4 most significant (left most) bytes of SHA-1 hash of the content of variable '07'
'09'	Buffer Sizes	First 2 bytes, binary coded, MSB first: - Receive buffer size in bytes available for messages to be received by the USAT Interpreter. Second 2 bytes, binary coded, MSB first: - Transmit buffer size in bytes available for messages to be sent by the USAT Interpreter. Both sizes include all possibly needed space for transport headers, security, routing information, concatenation information and so on.
'0A'... '13'	RFU	

6.1.1.1.1 Write access to the partition

This partition shall not be updated by administrative means after the personalisation process. The variables in this partition may be changed by the USAT Interpreter itself, if e.g. the configuration of the USAT Interpreter changes (e.g. addition of a new native code functionality).

6.1.1.1.2 Read access of the partition

The information stored in this partition can be freely accessed by any page executed by the USAT Interpreter.

6.1.1.2 USIM issuer information partition

The information stored in this partition is under the control of the USIM issuer. The USIM issuer is responsible to allocate variable IDs for his own purposes in the range from '14' to '28'. The used variable IDs shall be published to content providers.

6.1.1.2.1 Write access to the partition

This partition can be updated by the USIM issuer by administrative means.

6.1.1.2.2 Read access of the partition

The information stored in this partition can be freely accessed by any page executed by the USAT Interpreter.

6.1.1.3 End user information partition

The information stored in this partition is under the control of the end user. If the user decides to store information in this partition, the following variable IDs shall be used:

Variable ID	Description	Coding
'29'	User name	SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] with bit 8 set to 0 or UCS2 coded
'2A'	User e-mail address	SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] with bit 8 set to 0
'2B' ... '3F'	RFU	

6.1.1.3.1 Write access to the partition

This area can be updated locally by the end user by an user interface provided by the USAT Interpreter. Most likely, the user interface will be realised by a locally stored application of the USAT Interpreter accessible by the end user.

6.1.1.3.2 Read access of the partition

The information stored in this partition can be freely accessed by any page executed by the USAT Interpreter.

6.1.2 Permanent variable area

This area is used to store permanently variables which can be accessed even after the USIM was reset. This area is organised as a cyclic variable buffer. If the buffer is full, a new entry shall delete the most oldest entries until enough space is made available to store the new entry.

Each entry consists of the service ID of the page storing the variable in this area, the variable ID and the content of the variable. For pages using this variable area, it is mandatory to provide the service ID in the Page TLV. The assignment of service IDs is up to an external system entity.

6.1.2.1 Write access to the permanent variable area

Any page which provides a service ID may store permanent variables.

6.1.2.2 Read access of the permanent variable area

The information in this area can be freely accessed by pages providing a service ID within the Page TLV which is contained in the list of permanently stored variables. A page shall have access to those variables only, which have the same service ID as stored in the Page TLV.

6.1.3 Temporary variable area

Temporary variables are used during the execution of the current page. They may be shared with the following page. Temporary variables are used for 2 purposes:

- as variables defined and used within the current page,
- as variables to be shared between the current page and the following page.

The current page shall define, which variables are to be kept for access of the following page. To ensure, that only a dedicated following page can access the variables defined to be sharable, the current page may protect them with a One Time Password (OTP), which has to be presented by the following page in order to get access to the shared variables. The OTP to be presented by the following page may be ciphered.

If this mechanism is used to protect shared variable, it might happen that a page is not able to access the protected shared variables, if the sequence of pages provided to the USAT Interpreter is disturbed (e.g. by using backward navigation between pages...).

6.1.3.1 Write access to the temporary variable area

Only the current page can allocate temporary variables. The current page can allocate temporary variables as many as it is space available in this area.

To indicate how to provide variables to the next page, the KeepAll flag in the attribute of the current page and the OTP TLV and the Keep Alive List TLV within the current Page TLV is used according to the following table:

KeepAll flag	OTP TLV	KeepAlive TLV	Actions
set	present	present	not valid, if occurs, the KeepAll attribute shall be ignored, variables listed in the Keep Alive List TLV shall be kept for the following page and shall be protected by OTP
set	present	not present	all temporary variables shall be kept for the following page and shall be protected by OTP
set	not present	present	not valid, if occurs, the variables listed in the Keep Alive List TLV shall be kept for the following page and shall not be protected by OTP
set	not present	not present	all temporary variables shall be kept for the following page and shall not be protected by OTP
not set	present	present	variables listed in the Keep Alive List TLV shall be kept for the following page and shall be protected by OTP
not set	present	not present	not valid, no variables to be kept for the following page
not set	not present	present	variables listed in the Keep Alive List TLV shall be kept for the following page and shall not be protected by OTP
not set	not present	not present	no variables to be kept for the following page

6.1.3.2 Read access of the temporary variable area

A current page can freely access temporary variables stored by this current page. Protected variables of a previous page shall only be accessible after a successful verification of the One Time Password set by the previous page to protect temporary variables to be shared between these two pages.

In order to unlock the shared protected variables the Page Unlock TLV has to be present within the Page TLV. The Page Unlock TLV shall contain the OTP (ciphered or in clear as indicated by the KIc of the TLV) of the previous page.

If the OTP in the Page Unlock TLV matches the OTP stored with the protected variables, the protected variables are made available to the current page as regular temporary variables.

6.1.3.3 Lifetime of temporary variables

By default, all variables which are not kept explicitly to be shared by the following page are deleted, after the page is processed.

If there are protected variables, but the current page does not contain a matching OTP, the protected variables are deleted before processing the current page.

6.1.4 Page string element

This area is provided optionally by the current page. It can be used to store e.g. strings that are used several times in the current page.

The first string element in the String Pool TLV shall be identified by the variable reference 'C0', the next with 'C1' and so on.

6.1.4.1 Write access to page string elements

The information contained in this area is read only.

6.1.4.2 Read access of page string elements

The information can be accessed by the current page.

6.2 Variable values

The value associated with a variable identifier is a length-byte string pair. The type of a variable value is determined by the usage context. The USAT Interpreter shall keep track of the type of a variable. How the type of the variable is stored internally within the USAT Interpreter is up to the implementation of the USAT Interpreter.

The length of the variable value is restricted to 65535 ('FFFF') bytes. Each variable has one of the following types:

Type of variable	coding (3 bits)
Unknown	'000'
SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] with bit 8 set to 0	'001'
SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] packed	'010'
Binary format	'011'
UCS2 coded string	'100'
Other types	RFU

The coding specified shall be used to indicate the type of variable, when variable substitution is used.

6.3 Variable substitution

Variable IDs may appear in fields explicitly labelled as containing a variable identification. Variable substitution can take place in the following TLVs:

- Simple TLV Indicator (see clause "Execute USAT Command");
- Inline Value TLV;
- Inline Value 2 TLV;
- Submit TLV.

The value part of TLVs, where variable substitution can take place, consists of sequences of

- length - value pairs to indicate constant text

or

- variable substitution indicator - variable ID pairs to indicate variable substitutions.

Such sequences may appear in any order in value parts of TLVs where variable substitution may take place.

The variable substitution indicators are used to indicate that the next byte is a variable ID.

Length - Value pair

Length	Value	Description	M/O
1-3	L	Length of the following data	M
L	V	data	O

The length L is BER coded onto 1, 2 or 3 bytes according to ISO/IEC 7816-6 [5]. If L indicates a length of '00', no data shall be available.

Variable Substitution Indicator - Variable ID Pair

Length	Value	Description	M/O
1	‘C0’ or ‘C1’ or ... or ‘C7’	Variable substitution indicator, see table below	M
1	ID	Variable ID	M

The least significant 3 bits of the variable substitution indicators shall be used to indicate the type of the variable coded according to the table below.

Coding of variable substitution indicators:

Coding of variable substitution indicator	Type of variable referenced to
‘C0’	unknown
'C1'	SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] with bit 8 set to 0
'C2'	SMS default 7-bit coded alphabet as defined in 3GPP TS 23.038 [3] packed
'C3'	Binary format
'C4'	UCS2 coded string
'C5' ... 'C7'	RFU

Whenever TLVs, where variable substitutions may take place, are encountered by the USAT Interpreter at runtime, one of the following mechanisms are used, to replace the respective Length - Value pair(s) or the Variable Substitution Indicator - Variable ID pair(s) depending on the context:

Method 1:

Length - Value pair:

- the length is removed from the running text;
- the value part remains unchanged;

Variable Substitution Indicator - Variable ID pair:

- the variable substitution indicator is removed from the running text;
- the type of the value corresponding to the following variable reference shall be checked against the type indicated in the variable substitution indicator. If the type of the value is different from the indicated type, the USAT Interpreter shall generate an error unless the indicated type was set to 'C0' ("unknown");

- the following variable reference is replaced by:
  - the current content of the variable (that means inserting the variable content into the running text).

Method 2:

Length - Value pair:

- the length is *not* removed from the running text;
- the value part remains unchanged;

Variable Substitution Indicator - Variable ID pair:

- the variable substitution indicator is *not* removed from the running text:
- the type of the value corresponding to the following variable reference shall be checked against the type indicated in the variable substitution indicator. If the type of the value is different from the indicated type, the USAT Interpreter shall generate an error unless the indicated type was set to 'C0' ("unknown");
- if the indicated type was set to 'C0' ("unknown"), the type information of the variable substitution indicator in the running text is updated with the actual type of the variable;
- the following variable reference is replaced by:
  - the length of the content of the variable. The length is coded onto 1, 2 or 3 bytes according to ISO/IEC 7816-6 [5];
  - the current content of the variable (inserting the variable content into the text).

A variable value shall not contain a variable substitution, i.e. an inserted variable value is not rescanned for variable IDs.

7 Used USAT Interpreter data structures

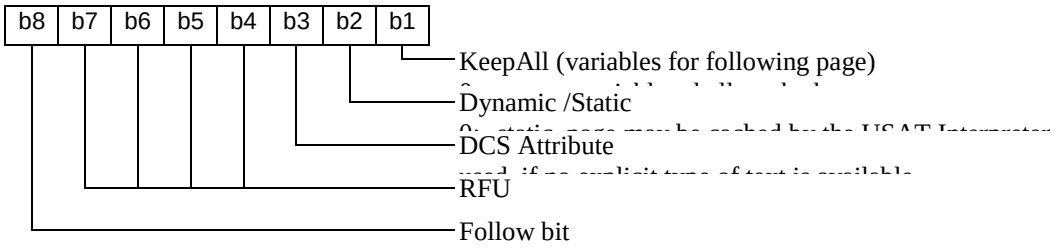
7.1 Page

A page is the unit which the USAT Interpreter does render and the default name scope of the temporary variables.

Length	Value	Description	M/O
1	'01' / '81'	Page Tag	M
1-3	A+B+C+D+ E+F+G+H+I	Length	M
A	Data	Attributes	O
B	TLV	Page Identification	M
C	TLV	Page Unlock Code	O
D	TLV	One Time Password	O
E	TLV	Keep Alive List	O
F	TLV	Service ID	O
G	TLV	String Pool	O
H	TLVs	Terminal response handler modifier - one or more TLVs	O
I	TLVs	Navigation Units – one or more TLVs	M

The following chapters specify the attributes and TLVs used in the Page TLV.

7.1.1 Attributes



7.1.2 Page Identification

The content of this TLV is a sequence of bytes to uniquely identify the page. This reference may later on be used by the USAT Interpreter to reference the page (e.g. for caching mechanisms or accessing the page by the end-user from the menu structure).

Coding:

Length	Value	Description	M/O
1	'02'	Page Identification Tag	M
1-3	L	Length	M
L	Data	Unique identification of the page. A sequence of bytes to uniquely identify the Page. This identification shall not contain a #-character (coded '23') and is coded by the external system entity.	M

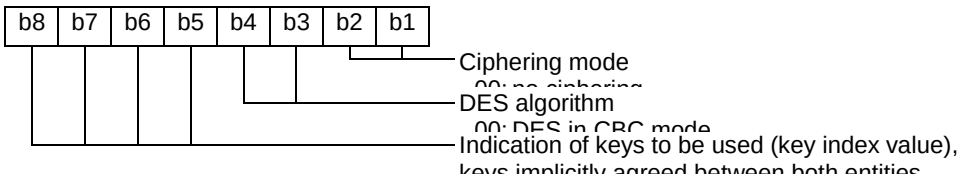
7.1.3 Page Unlock Code

The content of this TLV is a key index and algorithm identifier KIc for ciphering and a sequence of bytes encrypted according to the information provided by the KIc. This sequence of bytes shall be decrypted by the USAT Interpreter and verified against an OTP provided by a previous page.

Coding:

Length	Value	Description	M/O
1	'03'	Page Unlock Code Tag	M
1	L+1	Length (up to 1+8 bytes)	M
1	KIc	Key index and algorithm identifier for ciphering	M
L	Data	Page unlock code (one time password of the previous page)	M

The KIc is coded as follows:



DES is the algorithm specified as DEA in ISO 8731-1 [6]. DES in CBC mode is described in ISO/IEC 10116 [7]. Triple DES in outer-CBC mode is described in section 15.2 of [8]. DES in ECB mode is described in ISO/IEC 10116 [7].

The initial chaining value for CBC modes shall be zero.

7.1.4 One Time Password

The content of this TLV is a sequence of bytes generated by random to protect the temporary variables of the current page against unauthorised access.

Coding:

Length	Value	Description	M/O
1	'04'	One Time Password Tag	M
1	L	Length (up to 8 bytes)	M
L	Data	One time password (random value generated by an external system entity)	M

7.1.5 Keep Alive List

The content of this TLV is a list of variable IDs indicating which variables of the current page may be shared with the following page. The list shall not contain other variable IDs than variable IDs referring to temporary variables.

Coding:

Length	Value	Description	M/O
1	'05'	Keep Alive List Tag	M
1	L	Length (number of temporary variable IDs, up to 64 variables)	M
L	Data	Variable IDs	M

7.1.6 Service ID

The content of this TLV is a sequence of bytes to indicate that the current page shall belong to a certain service and is mainly used to handle permanent variable management The assignment and coding of service IDs is up to an external system entity. The length of a service ID shall not exceed 8 bytes.

Coding:

Length	Value	Description	M/O
1	'06'	Service ID Tag	M
1	L	Length (number of bytes of the service ID, <= 8 bytes)	M
L	Data	Service ID, unique identification of a service	M

7.1.7 String Pool

The content of this TLV is a list of strings coded in with the alphabet indicated in the DCS attribute used within the page. Within the page the strings are referenced by using their variable references (range 'C0' to 'FF') within the page string element area.

Coding:

Length	Value	Description	M/O
1	'07'	String Pool Tag	M
1 – 3	L	Length	M
L	Data	LV values of each string element in the string pool	M

7.1.8 Terminal response handler modifier

The current terminal response handler configuration can be modified temporarily by this TLV (e.g. to hide default entries by using action IDs, to add new ones or to modify existing entries).

This TLV can be used at the page level and at the navigation unit level. If this TLV is present at the page level and also at the navigation unit level, the last one will modify the first one. The content describes the action which shall be performed after the USAT Interpreter has received a matching general result byte of the terminal response within a proactive session. If a syntax error or a logical error occurs in the terminal response handler modifier, the current terminal response handler configuration remains unchanged.

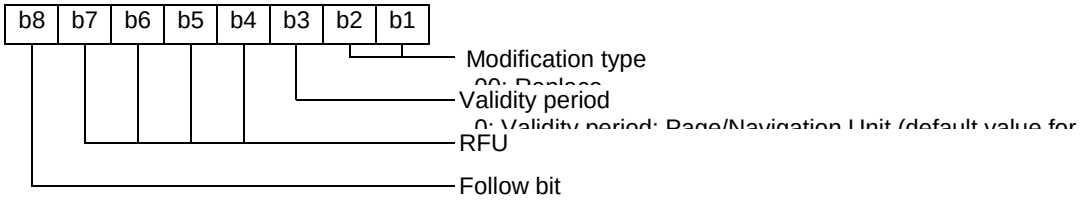
Coding:

Length	Value	Description	M/O/C
1	'08' / '88'	Terminal response handler modifier tag	M
1-3	A+2+B+C	Length	M
A	Attributes	Data	O
2	Data	General result range	M
B	TLV	Inline Value TLV, containing text for user notification	O
C	TLVs	Action TLVs – one or more TLVs	C

The following table gives an overview of conditions of presence for the Action TLVs depending on the modification type indicated in the attributes:

Modification Type (see Attributes)	Action TLV
Replace	shall be present
Add / Append	shall be present
Restore	need not to be present; to be ignored, if present
Remove	shall be present

7.1.8.1 Attribute



Validity period:

All terminal response handler modifications are valid only within the context they have been introduced. There are 3 different contexts:

- **System context:** In this context the default terminal response handler configuration is valid (see chapter"4.3 Terminal response handler mechanism"). The default terminal response handler configuration can only be changed by administrative means.

- **Page context:** A terminal response handler modifier within the page context can modify the response handler configuration for the whole page. After leaving the page the modifications done by the terminal response handler modifier of this page are discarded and the terminal response handler configuration of the system context as defined in [4.1.2 is restored.

- **Navigation unit context:** A terminal response handler modifier within the navigation unit context can modify the response handler configuration for the navigation unit containing the modifier. After leaving a navigation unit the modifications done by the terminal response handler modifier of this navigation unit are discarded and the terminal response handler configuration of the page context is restored.

7.1.8.2 General result range

A general result range defines subsequent values of the general result in the terminal response to an USAT command.

- A range consisting of only one value of the general result is coded by setting both bytes to the desired value.

- A range is coded by setting the first byte to the lowest value of the range and the second byte to the highest value of the range.

For example:

- general result '10' shall be coded: '10 10'

- general result '1X' shall be coded: '10 1F'

- general result 'XX' shall be coded: '00 FF'

- general result between '11' and '13' shall be coded: '11 13'

General result ranges on the USAT Interpreter side shall be checked by order of occurrence in the byte code (order of TLVs within the navigation unit, order of TLVs within the page), meaning the last defined modification shall be checked first. The action(s) defined for the first general result range found, which contains the received general result of the USAT command, shall be executed.

7.1.8.3 Text for user notification

This text is displayed by a DISPLAY TEXT command whenever a general result in response to a proactive command is received, that matches the general result range. After this DISPLAY TEXT command has been issued by the USAT Interpreter the actions defined for the general result range are to be performed regardless of the general result of the DISPLAY TEXT command itself.

The parameters for the DISPLAY TEXT command shall be as follows:

- The DCS for the DISPLAY TEXT command shall be set according to the value type information of the Inline Value TLV.

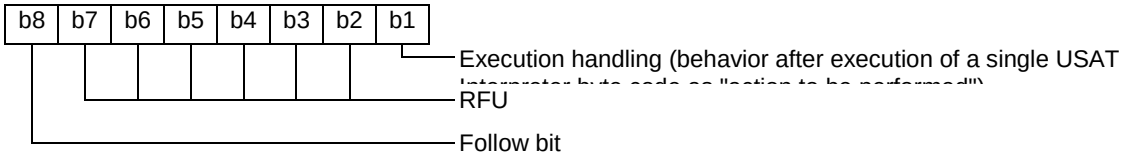
- The command qualifier to be used for the DISPLAY TEXT command shall be '81'.

7.1.8.4 Action

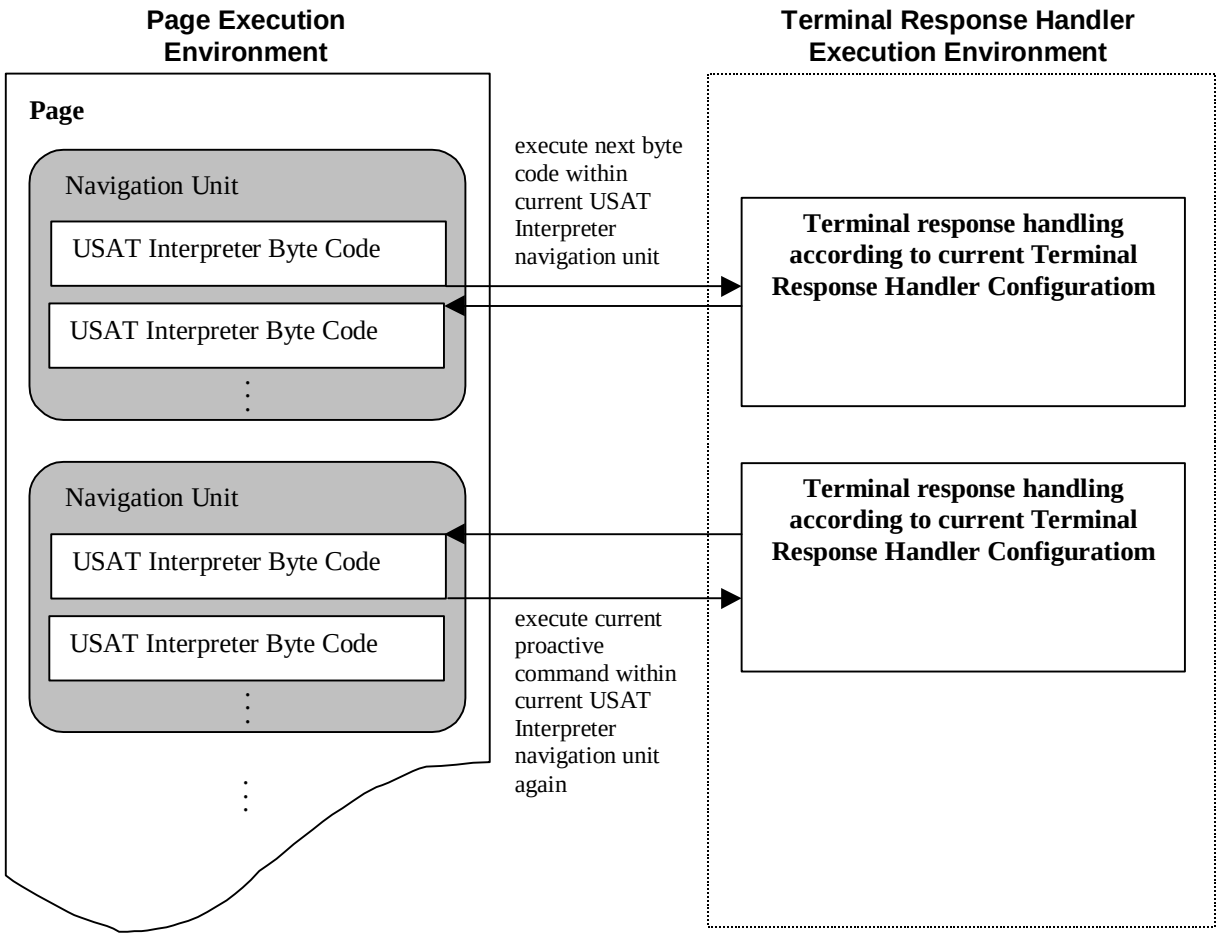
The action TLV defines the behaviour of the USAT Interpreter when the associated general result range matches the general result of the terminal response (TR).

Length	Value	Description	M/O/C
1	'09' / '89'	Action TLV tag	M
1-3	A+1+B+C	Length	M
A	Attributes	Data	O
1	1	Action ID	M
B	TLV	Action to be performed	C
C	TLV	Inline Value TLV, containing the action description of this action. This is a text assigned to this action to be used as text string of item within an item data object of a SELECT ITEM command.	C

7.1.8.4.1 Attributes



The following figure gives an overview of the return behaviour of the terminal response handler.



7.1.8.4.2 Action ID

Every action shall be uniquely identified by an action ID. This allows to remove or to update a targeted item in the action list without reconstructing the whole action list.

IDs are separated into two ranges:

- '00' - '1F' predefined system action IDs
- '20' - 'FF' service defined action IDs for navigation and other commands. These action IDs shall be uniquely assigned to the actions defined for a general result range by the service.

7.1.8.4.3 Action to be performed

The action to be performed is either predefined by the USAT Interpreter system (system action) or flow control information (navigation action) or a single USAT Interpreter byte code to be executed.

A system action is indicated within the action TLV by a predefined system action ID only:

- process next byte code
- quit USAT Interpreter without user confirmation
- go back one entry in history list. If the history list is empty, this action shall be ignored.
- retry last proactive command within current USAT Interpreter navigation unit (the command which generated the current general result)

For system actions the attribute of the action TLV shall be ignored by the USAT Interpreter.

A navigation action is indicated by a service given action ID and one of the following USAT Interpreter data structures as "action to be performed":

- page reference TLV
- anchor reference TLV

For navigation actions the attribute of the action TLV shall be ignored by the USAT Interpreter.

A single USAT Interpreter byte code to be executed is indicated by a service given action ID and one of the following USAT Interpreter byte codes as "action to be performed":

- Display Text
- Get Input
- Set Variable
- Execute USAT Command
- Execute Native Command

The attribute of the action TLV determines the behavior of the USAT Interpreter after execution of the single USAT Interpreter byte code.

Action to be performed			
	Action ID	**used TLV**	**attribute handling**
System actions			
process next byte code	'00'	none	attribute byte shall be ignored
quit USAT Interpreter without user confirmation	'01'	none	
go back one entry in history list	'02'	none	
retry last proactive command within current USAT Interpreter navigation unit	'03'	none	
RFU system actions	'04' to '1F'	RFU	
Navigation actions			
branch to another page branch to another navigation unit	defined by service (‘20’ to ‘FF’)	page reference TLV or anchor reference TLV	attribute byte shall be ignored
Single USAT Interpreter byte codes			
Execute Native Command byte code	defined by service ('20' to 'FF')	Execute Native Command byte code TLV	behavior after execution of a single USAT Interpreter byte code as "action to be performed": - execute next byte code within current USAT Interpreter navigation unit - execute current proactive command within current USAT Interpreter navigation unit again
Execute Display Text byte code		Display Text byte code TLV	
Execute Set Variable byte code		Set Variable byte code TLV	
Execute Get Input byte code		Get Input byte code TLV	
Execute USAT Command byte code		Execute USAT Command byte code TLV	

NOTE: The retry action should be used only in conjunction with other actions or a notification text for a general result range to avoid the immediate repetition of the USAT command causing retry (possible senseless loop).

7.1.8.4.4 Action description

In the case of several actions (action list) assigned to the same general result range, a SELECT ITEM command shall be constructed by the USAT Interpreter using the corresponding action descriptions as items.

This TLV is mandatory if the modification type within the attribute byte of the terminal response handler modifier indicates "Replace" or "Add / Append".

If only one action is defined for the general result range, the action is executed by the USAT Interpreter without building the SELECT ITEM command.

After this SELECT ITEM command has been issued by the USAT Interpreter, an action shall be performed depending on the general result of the SELECT ITEM command itself:

General result for the SELECT ITEM	Comment
'00'...'0F' (ok)	the action defined for the option selected by the user shall be performed
'11' (backward move requested)	execute current proactive command within current USAT Interpreter navigation unit again or return to the wait state if the wait state is currently active
'10' (quit)	quit USAT Interpreter without user confirmation
'12'...'1F'	quit USAT Interpreter without user confirmation
'20'...'2F' (worth to retry)	quit USAT Interpreter without user confirmation
'30'...'3F' (not worth to retry)	quit USAT Interpreter without user confirmation

The parameters for the SELECT ITEM command shall be as follows:

- Alpha identifier not used
- The command qualifier to be used for the SELECT ITEM command shall be '03'.

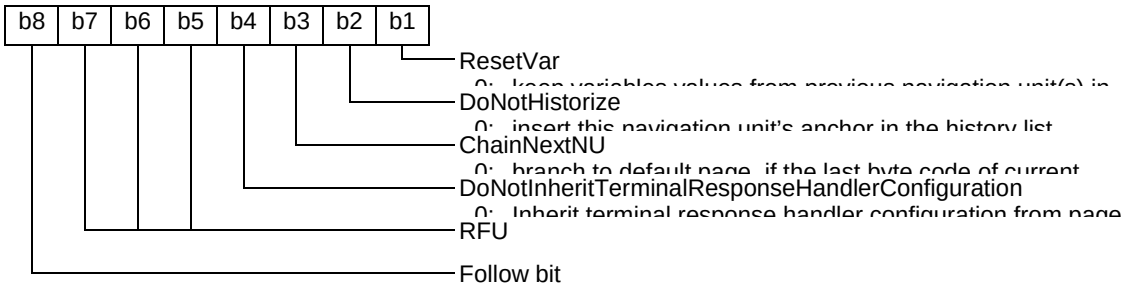
7.2 Navigation Unit

A navigation unit is a component of a page. It is named using an anchor. A navigation unit is referenced using an anchor reference.

Length	Value	Description	M/O
1	'0A' / '8A'	Navigation Unit Tag	M
1-3	A+B+C+D	Length	M
A	Data	Attributes	O
B	TLV	Anchor (name of a navigation unit)	O
C	TLVs	Terminal response handler modifier - one or more TLVs	O
D	TLVs	USAT Interpreter Byte Codes – one or more TLVs	O

The following chapters specify the attributes and TLVs used in the navigation unit TLV.

7.2.1 Attributes



7.2.2 Anchor

The content of this TLV is a sequence of bytes identifying the navigation unit. It is mandatory to provide this TLV, if a navigation unit of the current page or another page needs to branch to this navigation unit.

Coding:

Length	Value	Description	M/O
1	'0B'	Anchor Tag	M
1-3	L	Length	M
L	Data	Unique identification of navigation unit within the page. A sequence of bytes to uniquely identify the Anchor. This identification shall not contain a #-character (coded '23') and is coded by the external system entity.	M

7.2.3 Terminal response handler modifier

The current terminal response handler configuration can be modified temporarily by this TLV (e.g. to hide default entries by using action IDs, to add new ones or to modify existing entries).

The content describes the action which shall be performed after the USAT Interpreter has received a matching general result byte of the terminal response within a proactive session. If a syntax error or a logical error occurs in the terminal response handler modifier, the current terminal response handler configuration remains unchanged.

Coding:

See chapter "7.1.8 Terminal response handler modifier".

7.2.4 USAT Interpreter Byte Codes

These TLVs contain the executable part of the page.

Coding:

See chapter "8 USAT Interpreter byte codes".

7.3 Anchor Reference

This TLV is used to refer to a navigation unit in the current page or in another page.

Length	Value	Description	M/O
1	'0C'	Anchor Reference Tag	M
1-3	L	Length	M
L	Data	Anchor Reference Name	M

An anchor reference name is the value part of a page identification TLV (unique identification of the page, see "7.1.2 Page Identification") followed by a '23' ("#") and the value part of the anchor TLV (unique identification of navigation unit, see "7.2.2 Anchor") within the page. Either the page identification part or the anchor part (but not both) can be omitted. If the page identification part is omitted the reference is to an anchor on the current page. If the anchor name part is omitted the reference is to the beginning of the referenced page.

7.4 Variable Identifier List

This TLV is used to list a sequence of variables.

Length	Value	Description	M/O
1	'0D'	Variable Identifier List Tag	M
1	L	Length	M
L	Data	Variable IDs (up to 64 Variable IDs)	M

7.5 Inline Value

This TLV inserts a byte array, which often is simply running text, at the point of its appearance. The TLV is thus simply a way to encapsulate an immediate value.

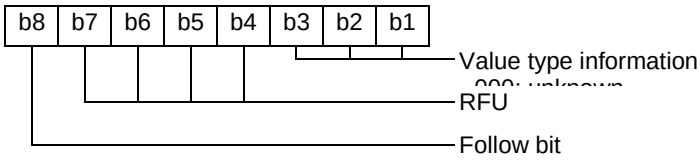
The Inline Value content may contain variable substitution indicators to indicate variable references. Therefore the Inline Value content has to be structured in Length-Value and Variable Substitution Indicator - Variable ID pairs. The possibly available constant data values and variable references have to be rendered according to chapter "6.3 Variable substitution" Method 1 during processing of this TLV by the USAT Interpreter. If the type of the possibly substituted variable values is different from the type indicated in the attribute of this TLV, the USAT Interpreter shall perform a type conversion or generate an error according to the following table:

from DCS	to DCS	comment
SMS default	SMS default	*
SMS packed		not supported, error generated
binary		cast allowed, no change of sequence of bytes
UCS2		not supported, error generated
unknown		cast allowed, no change of sequence of bytes
SMS default	SMS packed	not supported, error generated
SMS packed		*
binary		cast allowed, no change of sequence of bytes
UCS2		not supported, error generated
unknown		cast allowed, no change of sequence of bytes
SMS default	binary	cast allowed, no change of sequence of bytes
SMS packed		cast allowed, no change of sequence of bytes
binary		*
UCS2		cast allowed, no change of sequence of bytes
unknown		cast allowed, no change of sequence of bytes
SMS default	UCS2	conversion supplied, according to TS 102 221[4]
SMS packed		not supported, error generated
binary		cast allowed, no change of sequence of bytes
UCS2		*
unknown		cast allowed, no change of sequence of bytes
SMS default	unknown	cast allowed, no change of sequence of bytes
SMS packed		cast allowed, no change of sequence of bytes
binary		cast allowed, no change of sequence of bytes
UCS2		cast allowed, no change of sequence of bytes
unknown		*

Coding of the Inline Value TLV:

Length	Value	Description	M/O
1	‘0E’ / '8E'	Inline Value Tag	M
1-3	A+B	Length	M
A	Data	Attributes	O
B	Data	Inline value content	O

Coding of the attributes:



7.6 Inline Value 2

This TLV inserts a byte array, which often is simply running text, at the point of its appearance. The TLV is thus simply a way to encapsulate an immediate value. Usage and syntax and behaviour of this TLV is identical to the Inline Value TLV, but another tag value is used.

Length	Value	Description	M/O
1	‘0F’ / '8F'	Inline Value 2 Tag	M
1-3	A+B	Length	M
A	Data	Attributes	O
B	Data	Inline Value 2 content	O

Coding :See Inline Value TLV.

7.7 Input List

This TLV contains a list of Variable Identifier List TLVs and Inline Value TLVs.

Length	Value	Description	M/O
1	‘10’	Input List Tag	M
1-3	L	Length	M
L	TLVs	Any sequence of - Variable Identifier List TLVs and / or - Inline Value TLVs	M

7.8 Ordered TLV List

This TLV is used to associate a list of other TLVs. The order and the possible types of contained TLVs within an ordered TLV list is specified within the byte codes using this TLV. The number of actual contained TLVs is implicitly given by the length indication of the Ordered TLV List. It is allowed, that the ordered TLV list does not contain any TLV.

Depending on the context (the byte code using this TLV) each optional TLV within the Ordered List of TLVs shall have a different tag value.

Length	Value	Description	M/O
1	'11'	Ordered TLV List Tag	M
1-3	A+...+Z	Length	M
A	TLV	First TLV	O/M
...	...	...	
Z	TLV	Last TLV	O/M

7.9 Page Reference

This TLV can represent a page, an anchor within the current page, or an anchor within another page.

If the Anchor Reference TLV or the Variable Identifier List TLV is available, then the USAT Interpreter shall start rendering the requested locally stored Anchor. If the Anchor is not found locally, an error is generated.

If the Submit TLV is available (that indicates that the page is not locally stored on the USIM, i.e. e.g. stored at an external system entity), then the USAT Interpreter shall build a request to the external system entity according to chapter "7.10 Submit" .If the transmission to the external system entity fails, an USAT Interpreter transmission error shall be generated by the USAT Interpreter and the execution shall stop.

Length	Value	Description	M/O
1	'12' / '92'	Page Reference Tag	M
1-3	A	Length	M
A	TLV	either - Anchor Reference TLV or - Variable Identifier List TLV (referring to a variable containing the Anchor Reference, only the first variable ID shall be considered by the USAT Interpreter, remaining variable IDs shall be ignored) or - Submit Configuration TLV	M

7.9.1 Anchor Reference

Reference to a locally stored anchor.

Coding:

See chapter "7.3 Anchor Reference"

7.9.2 Variable Identifier List

Referring to a variable containing the Anchor Reference. Only the first variable ID within the variable ID list shall be considered by the USAT Interpreter. Possibly remaining variable IDs shall be ignored.

Coding:

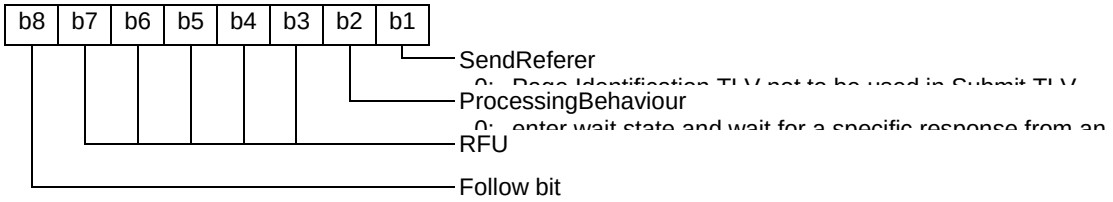
See chapter "7.4 Variable Identifier List".

7.9.3 Submit Configuration

This TLV describes the information which shall be sent to the external system entity.

Length	Value	Description	M/O
1	'13' / '93'	Submit Configuration Tag	M
1-3	A+B+C+D	Length	M
A	Data	Attributes	O
B	TLV	Submit Data TLV (submit information, text possibly containing variable references)	M
C	TLV	Inline Value TLV, text to be displayed during the wait state active.	O
D	TLV	Gateway Address TLV, to be incorporated into the operational layer, refer to TS 31.114 [2]	O

7.9.3.1 Attributes



If the SendReferer attribute is set, the Page Identification TLV of the current page shall be incorporated into the generated Submit TLV prior to the transmission to the external system entity.

If the ProcessingBehaviour attribute is not set the USAT Interpreter shall enter the wait state (see chapter "4.2.2 Wait State") after transmission.

If the ProcessingBehaviour attribute is set the USAT Interpreter shall render the next byte code after transmission.

7.9.3.2 Submit Data

The submit data information is a sequence of bytes possibly containing constant data values and variable references to be substituted according to chapter "6.3 Variable substitution" method 2. The sequence of bytes shall be structured into Length - Value and Variable Substitution Indicator - Variable ID pairs to ensure, that variable references can be detected. The content of the submit information is coded by the external system entity and possibly contains a request for the next page to be transmitted to the USAT Interpreter by an external system entity.

After variable substitution this TLV is used within the Submit TLV to provide information to the external system entity. See chapter "7.10Submit" for the structure of data provided to the external system entity.

Length	Value	Description	M/O
1	'14'	Submit Data Tag	M
1-3	A	Length	M
A	Data	Byte sequence, according to chapter "6.3 Variable substitution" containing possibly variable references	O

7.9.3.3 Text to be displayed during the active wait state

This TLV shall only be considered by the USAT Interpreter if the wait state is entered.

If this Inline Value TLV is given in the Submit Configuration TLV, the value part of this Inline Value TLV shall override the default Text String of the DISPLAY TEXT command to notify the user about the wait state (see chapter "4.2.2 Wait State").If the Inline Value TLV is not given in the Submit Configuration TLV, the default text shall be taken for the Text String of the DISPLAY TEXT command to notify the user about the wait state.

7.9.3.4 Gateway Address

The Gateway Address TLV contains data (the Gateway Address Information) to address a specific Gateway in the USAT Interpreter Gateway System. The coding of the Gateway Address Information is out of the scope of the present document.

If the Gateway Address TLV is available in the Submit Configuration TLV, the USAT Interpreter shall put the Gateway Address Information into the Gateway Address TLV of the operational layer of the USAT Interpreter transport protocol (see TS 31.114 [2]).

Length	Value	Description	M/O
1	'15'	Gateway Address Tag	M
1-3	A	Length	M
A	Data	Gateway Address Information	O

7.10 Submit

This TLV is used to provide information from the USAT Interpreter to the external system entity. It shall be used only in the direction from the USAT Interpreter to the external system entity.

Length	Value	Description	M/O/C
1	'16'	Submit Tag	M
1-3	A+B	Length	M
A	TLV	Submit Data TLV	M
B	TLV	Page Identification TLV (if indicated in attribute of Submit Configuration TLV)	C

7.10.1 Submit Data

The submit data information is a sequence of bytes with substituted variables if contained. The origin of this TLV is the Submit Data TLV in the Submit Configuration TLV with variables substituted according to chapter "6.3 Variable substitution" method 2.

Length	Value	Description	M/O
1	'14'	Submit Data Tag	M
1-3	A	Length	M
A	Data	Byte sequence, according to chapter "6.3 Variable substitution" containing substituted variable references	O

7.10.2 Page Identification

This TLV shall be available if and only if the SendReferer bit in the attributes of the Submit Configuration TLV was set. It containes the page identification of the current page.

8 USAT Interpreter byte codes

Each USAT Interpreter byte code is a TLV. Each byte code has its own byte code tag value, optional attributes and a list of arguments. Arguments, if present, shall appear in the order given.

The byte codes make use of the USAT Interpreter TLVs as follows:

	Attribute Bytes	Variable References	Variable Identifier List TLV	Inline Value TLV	Inline Value 2 TLV	Page Reference TLV	Ordered TLV List TLV	Input List TLV
Set Variable		✓		✓				
Assign and Branch		✓		✓	✓	✓	✓	
Extract		✓						
Go Back	1							
Branch on Variable Value		✓	✓			✓	✓	
Exit	1		✓					
Execute USAT Command	1	✓						
Execute Native Command	1		✓					✓
Get Length		✓	✓					
Get TLV Value		✓	✓					
Display Text	1			✓				
Get Input	1	✓		✓	✓			

8.1 Set Variable

This byte code sets one or more variables either to a value contained in the corresponding Inline Value TLV or to the concatenated contents of the referenced variables in the Variable Identifier List TLV. This byte code can be used to e.g. copy the content of one variable to another variable or to concatenate a list of variables and/or constant text into another variable. All pairs of Variable ID and Inline Value TLV or Variable Identifier List TLVs are used independently, i.e. the Variable ID is used to store the result of the following TLV only.

Length	Value	Description	M/O
1	'40'	Set Variable Tag	M
1-3	1+A+...+1+X	Length	M
1	Data	Variable ID to store the result of the following TLV	M
A	TLV	Inline Value TLV or Variable Identifier List TLV	M
...	...	...	
1	Data	Variable ID to store the result of the following TLV	O
X	TLV	Inline Value TLV	O

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Syntax error	Syntax error	Stop
Reference to undefined	Reference to undefined variable	Stop
Problem in memory management	Memory allocation problem	Stop

At least one pair of Variable ID and Inline Value TLV or Variable Identifier List TLV shall be present in the Set Variable byte code.

8.2 Assign and Branch

This byte code displays a menu on the UE and assigns a selected value to a variable according to the selection of the user.

Length	Value	Description	M/O
1	'41'	Assign and Branch Tag	M
1-3	1+A+...+1+X	Length	M
1	Data	Destination Variable ID, identifier of the variable to be set	M
A	TLV	Inline Value TLV: Contains the select item alpha-identifier (according to 3GPP TS 31.111 [1])	O
B	TLV	Ordered TLV List TLV (see description below) containing possibly: - Inline Value 2 TLV - Inline Value TLV - Page Reference TLV	M
...	...	...	
X	TLV	Ordered TLV List TLV (see description below)	O

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined variable	Stop
Problem in memory management	Memory allocation problem	Stop
Syntax error	Syntax error	Stop
USAT command failed	USAT command failed.(SELECT ITEM could not be built)	Stop

Explanation of used arguments:

8.2.1 Destination Variable Identifier

The content of this value identifies the destination variable. The value contained in the selected Inline value TLV within the Ordered TLV List TLV will be assigned to this destination variable by the USAT Interpreter.

8.2.2 Inline TLV containing Select Item Title

The content of this TLV is running text which specifies the alpha identifier to be used by the USAT Interpreter when generating a SELECT ITEM command from the "Assign and Branch" byte code according to 3GPP TS 31.111 [1].

8.2.3 Ordered TLV List

One or more of this TLVs shall be contained in the "Assign and Branch" byte code.

This TLV encapsulates the

- "Inline Value 2",containing the text of a single item of the SELECT ITEM command
- "Inline Value", containing a value to the assigned to the destination variable, if the item is selected and
- "Page Reference", containing a destination for a branch, if the item is selected.

TLVs in the given order, which determine the action to be performed.

General variable assignments and navigation operations may be performed by the "Assign and Branch" byte code dependent on the data provided in the Ordered TLV List TLV. When optional TLVs are omitted, special cases can be encoded according to the following table:

Inline Value 2	Inline value (to be assigned to destination variable)	Page Reference	
present	present	present	**"Display, Assign and Branch"** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item from the list, the USAT Interpreter shall assign the value of the Inline value TLV to the destination variable and branch to the navigation unit specified within the Page Reference TLV.
present	present	not present	**"Set Variable Selected"** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item from the list, the USAT Interpreter shall assign the value of the Inline Value TLV to the destination variable.
present	not present	present	**"Go Selected"** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item from the list, the USAT Interpreter shall branch to the navigation unit specified within the Page reference TLV. A destination variable identifier shall be ignored for this case.
present	not present	not present	**"Select Item"** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item from the list, the USAT Interpreter shall process the next byte code. A destination variable identifier shall be ignored for this case.
not present	present	present	**"Assign and Branch"** No generation of a SELECT ITEM command by the USAT Interpreter. The USAT Interpreter shall assign the value of the Inline Value TLV to the destination variable and branch to the navigation unit specified within the Page reference TLV.
not present	present	not present	**"Set Variable"** No generation of a SELECT ITEM command by the USAT Interpreter. The USAT Interpreter shall assign the value of the Inline value TLV to the destination variable.
not present	not present	present	**"Direct Go"** No generation of a SELECT ITEM command by the USAT Interpreter. The USAT Interpreter shall directly branch to the navigation unit specified within the Page reference TLV. The destination variable identifier shall be ignored for this case.
not present	not present	not present	not valid, if occurs an error shall be issued.

8.3 Extract

This byte code extracts a byte array from a value and stores the result in a variable.

Length	Value	Description	M/O
1	'42'	Extract Tag	M
1	4	Length	M
1	Data	Variable ID, which shall contain the result	M
1	Data	Variable ID, containing the source data	M
1	I	Zero based start index in the byte array	M
1	N	Maximum number of bytes to extract, '00' indicates to retrieve all remaining bytes	M

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Syntax error	Syntax error	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

8.4 Go Back

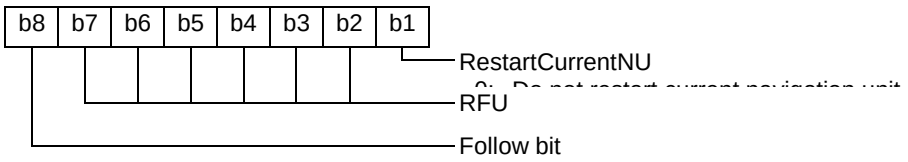
This byte code forces branching to the last anchor pushed on the history list. It has no impact on the history list itself.

Length	Value	Description	M/O
1	'43' / 'C3'	Go Back Tag	M
1	A	Length	M
A	Data	Attributes	O

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Jump to undefined	Reference to undefined (case of history containing no previous anchors any more)	Stop

Attributes:



8.5 Branch On Variable Value

This byte code compares a variable to a list of values that have an associated Page Reference. When a match is found, the referenced page shall be executed. If no match is found, the first Page Reference after the Ordered TLV List shall be used to branch. If this last Page Reference TLV is not contained in the byte code, no branch shall be executed and the USAT Interpreter shall continue to render the next byte code after the Branch on Variable Value byte code.

Length	Value	Description	M/O
1	'44'	Branch on Variable Value Tag	M
1	1+A+...+X+Y	Length	M
1	Data	Variable ID (containing the value to match)	M
A	TLV	Ordered TLV List TLV (see description below) containing: - Variable Identifier List TLV (referring to one variable containing the value to be compared with the match value, additional Variable IDs to be ignored) - Page Reference TLV, to branch to, if value matches	M
...	...	...	
X	TLV	Ordered TLV List TLV	O
Y	TLV	Page Reference TLV, if no match is found, go to this reference	O

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined variable	Stop
Jump to undefined	Page Reference not found.	Stop

Explanation of used arguments:

8.5.1 Variable ID

This variable shall contain the value to be compared.

8.5.2 Ordered TLV List

In each of these TLVs the following TLVs are encapsulated:

- Variable Identifier List TLV (referring to one variable containing the value to be compared with the match value; additional Variable IDs to be ignored)

- Page Reference TLV.

The Page Reference TLV contains the location to be branched to, if the comparison is successful.

8.5.3 Page Reference

If no match was found, the reference contained in here is used to branch. If this TLV is not available, no branch is executed and the USAT Interpreter continues to render the next byte code after the Branch on Variable Value byte code.

8.6 Exit

If the TerminateSession Attribute is not set, the USAT Interpreter shall behave as defined by the current terminal response handler configuration for the case of "Proactive SIM session terminated by the user".

If the TerminateSession Attribute is set, the proactive session is terminated immediately by the USAT Interpreter. The USAT Interpreter shall respond to the UE with SW1/SW2='9000' in this case.

If the USAT Interpreter had been called USIM internally (by an proprietary internal interface), the Variable Identifier List TLV may be used to provide return values to the calling function. Handling of these internal return values isout of the scope of the present document.

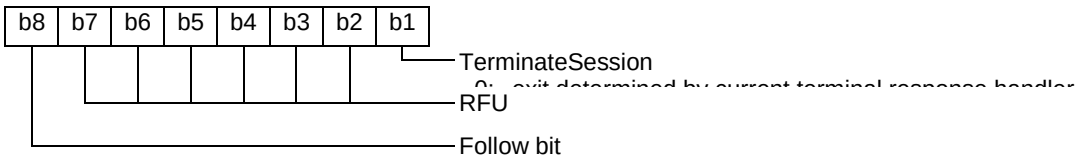
If the USAT Interpreter does not support the mechanism of providing return values, it shall ignore the possibly available Variable Identifier List TLV.

Length	Value	Description	M/O
1	'45' / 'C5'	Exit Tag	M
1	A+B	Length	M
A	Data	Attributes	O
B	TLV	Variable Identifier List TLV (containing return values)	O

Possible errors:

Error Code	Description	Action
No error	OK	Stop
Reference to undefined	Variables in Variable Identifier list are not available	Stop

Attributes:



8.7 Execute USAT Command

This byte code executes an USAT command using the provided arguments.

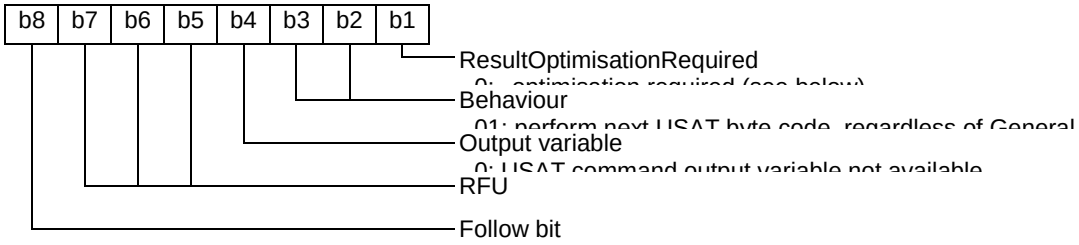
Length	Value	Description	M/O/C
1	'46' / 'C6'	Execute USAT Command Tag	M
1	A+B+C+3+D	Length	M
A	Data	Attributes	O
B	Variable ID	Variable to hold the General Result code from Terminal Response, depending on the Behaviour bits of the attribute byte	C
C	Variable ID	Variable to hold the output of the USAT command (only available, if indicated in attributes)	C
1	Cmd type	Command type value according to 3GPP TS 31.111 [1]	M
1	Cmd qual.	Command qualifier value according to 3GPP TS 31.111 [1]	M
1	Dest dev.	Destination device according to 3GPP TS 31.111 [1]	M
D	TLVs and Simple TLV Indicators	Sequence of - simple TLVs of the proactive command as defined in 3GPP TS 31.111 [1] - and / or Simple TLV Indicators	O

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined	Stop
Problem in memory management	Memory problem in the preparation of the USAT command	Stop
Syntax error	Try to initialise a text element	Stop
USAT command failed	USAT Command could not be delivered to UE	Stop
USAT command not allowed	due to configuration of the USAT Interpreter	Stop

Explanation of used arguments:

8.7.1 Attributes



8.7.2 Simple TLV

This TLV shall be a simple TLV coded as described in 3GPP TS 31.111 [1] for the USAT proactive command to be executed.

8.7.3 Simple TLV Indicator

A Simple TLV Indicator is a placeholder for a Simple TLV. A Simple TLV Indicator is coded as follows:

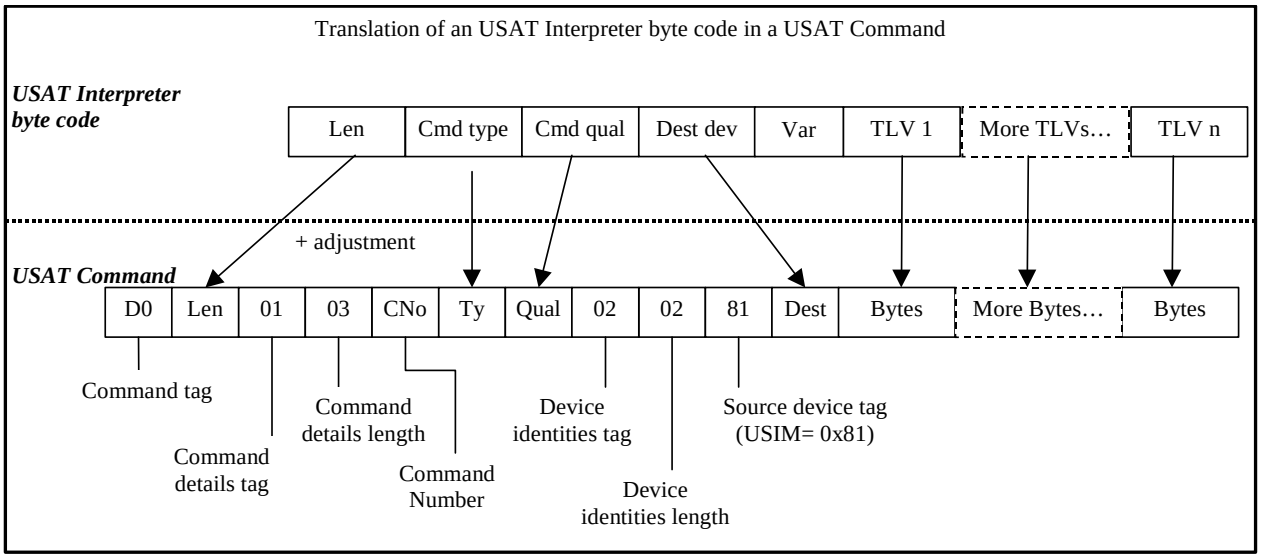
Coding	Description
'00'	This value indicates the Simple TLV Indicator
Length	This value indicates the length of the following data belonging to the Simple TLV Indicator
Result Tag	This value represents the Tag value of the resulting Simple TLV
Simple TLV Indicator content	The Simple TLV Indicator content may contain variable substitution indicators to indicate variable references. Therefore the Simple TLV Indicator content has to be structured into Length - Value and Variable Substitution Indicator - Variable ID pairs. The possibly available variable references have to be expanded according to chapter "6.3 Variable substitution" Method 1 during processing of this indicator by the USAT Interpreter.

The result of processing the Simple TLV Indicator shall be a Simple TLV. When the USAT Interpreter processes a Simple TLV Indicator the Result Tag shall be the Tag of the resulting Simple TLV. The value part shall be formed of the Simple TLV Indicator content and the length of the resulting Simple TLV is the length of the Simple TLV Indicator content after possible variable substitution.

8.7.4 Sequence of Simple TLVs and Simple TLV Indicators

The sequence of these Simple TLVs and Simple TLV Indicators is translated by the USAT Interpreter to form the sequence of Simple TLVs of an USAT command (3GPP TS 31.111 [1]). When expanding Simple TLV Indicators to Simple TLVs the length of the BER-TLV of the resulting USAT command shall be adjusted by the USAT Interpreter before issuing the command to the UE.

When executing a Execute USAT command byte code, the USAT Interpreter issues a regular USAT command to the UE using the USAT protocol. The translation procedure from the Execute USAT Command TLV to an USAT command can be visualised in principle as follows:



8.7.5 Result of an Execute USAT Command

The result of executing an USAT command is a Terminal Response structure containing a list of Simple TLVs as defined in 3GPP TS 31.111 [1].

The General Result of the USAT command shall be stored in the variable to hold the General Result code from the Terminal Response if indicated by the Behaviour bits of the attribute byte

The Terminal Response structure itself is processed by the USAT Interpreter as specified in the following 2 chapters.

8.7.5.1 Optimisation not Required

The complete proactive command specific response data (Terminal Response structure according to 3GPP TS 31.111 [1]) is stored in the result variable.

8.7.5.2 Optimisation Required

Only the first TLVs after the Result Simple TLV within a Terminal Response (see 3GPP TS 31.111 [1]) shall be processed by the USAT Interpreter as follows:

- If the first TLV after the Result Simple TLV is a Text String TLV according to 3GPP TS 31.111 [1], the value part without the DCS byte is assigned to the result variable. The DCS is removed from the V field, but used for variable management internally the USAT Interpreter.

- In all other cases, the value part of the first TLV after the Result Simple TLV is assigned to the result variable.

8.8 Execute Native Command

This byte code is used to execute an operating system call, "plug-in" or an application external to the USAT Interpreter.

The attribute indicates if the execution returns to the USAT Interpreter or not. Arguments are passed for input and output. The output is stored in a list of variables.

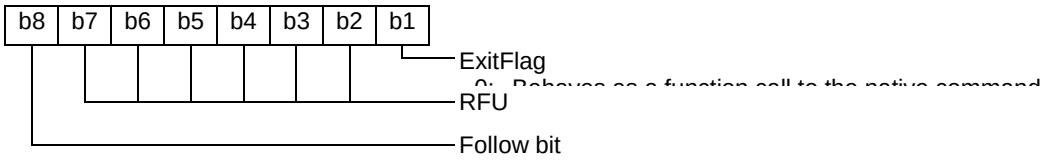
Length	Value	Description	M/O
1	'47' / 'C7'	Execute Native Command Tag	M
1	A+2+B+C	Length	M
A	Data	Attributes	O
2	Data	NCI of application or plug-in	M
B	TLV	Input List TLV containing arguments	O
C	TLV	Variable Identifier List TLV for output of application or plug-in	O

The NCI (Native Code Identifier) has a size of 2 bytes and is binary coded, most significant byte first. The values '0000' to '7FFF' are RFU. Other values may be used for proprietary implementations.

Possible errors:

Error Code	Description	Action
No error	OK	Continue
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	Stop
Syntax Error	Incorrect number of arguments passed to the execute element.	Stop
Execution Error	Execute element generated an internal error.	Stop

8.8.1 Attributes



8.8.2 Result of a Native Function Call

If the native function call returns, the values produced by the call are the values associated with the variable references in the output list.

8.9 Get Length

This byte code instructs the USAT Interpreter to calculate the length of all variable contents of the variables in the Variable List and to assign the result to the output variable.

Length	Value	Description	M/O
1	'48'	Get Length Tag	M
1-3	1+A	Length	M
1	Data	Variable ID (output, containing the BER encoded result length in binary format according to ISO/IEC 7816-6 [5])	M
A	TLV	Variable Identifier List TLV, Variable List for length calculation	M

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Problem in memory management	Memory allocation problem	Stop
Reference to undefined	Reference to undefined variable	Stop

8.10 Get TLV Value

This byte code instructs the USAT Interpreter to extract the value part of a TLV from a sequence of TLVs and to assign the resulting value to the output variable.

If the requested tag value is not found in the sequence of TLVs, the output variable is generated with no content (i.e. the length of content of the variable is 0).

Length	Value	Description	M/O
1	'49'	Get TLV Value Tag	M
1-3	2+A	Length	M
1	Data	Variable ID (output, containing the value of the requested TLV)	M
1	Data	Tag value to search for	M
A	TLV	Variable Identifier List TLV, each referenced variable shall contain a list of TLVs (e.g. generated Terminal Response of Execute USAT Command)	M

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Problem in memory management	Memory allocation problem	Stop
Reference to undefined	Reference to undefined variable	Stop

8.11 Display Text

This byte code instructs the USAT Interpreter to issue a DISPLAY TEXT command according to 3GPP TS 31.111 [1].

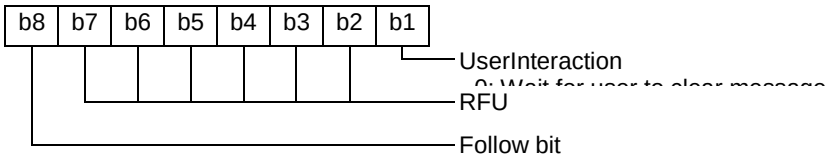
This command is used to display text of informational nature without require any input from user. The USAT Interpreter shall use the DCS value according to the indication given in the attributes of the Inline Value TLV. If no attributes are given in the Inline Value TLV, the coding scheme indication of the current page shall be used.

Length	Value	Description	M/O
1	'4A' / 'CA'	Display Text Tag	M
1-3	1+A	Length	M
1	Data	Attributes	O
A	TLV	Inline Value TLV, containing text to be displayed	M

The following parameters shall be used for the generated DISPLAY TEXT command:

Field	Comment
Command Details according to 3GPP TS 31.111 [1]	High Priority shall always be used. Other command parameters shall be used according to the information provided in the attribute byte.

Coding of the attributes:



Possible errors:

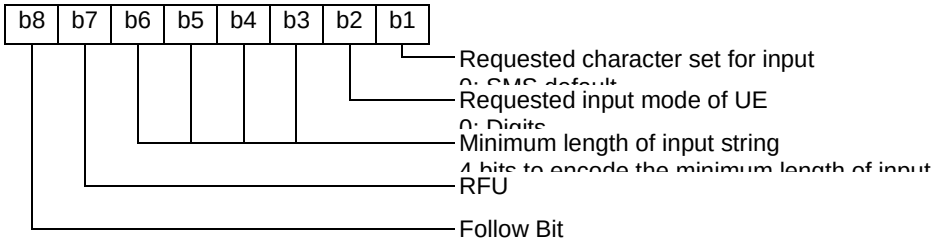
Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined variable	Stop

8.12 Get Input

This command is used to request multiple character input from user.

Length	Value	Description	M/O
1	'4B' / 'CB'	Get Input Tag	M
1-3	A+1+B+C	Length	M
A	Data	Attributes	O
1	Data	Variable ID (for storing the entered characters, the variable type information of the variable is set according to the DCS indication received from the UE. The DCS received from the UE is not stored in the variable value.)	M
B	TLV	Inline Value TLV, containing text to be displayed (e.g. the question, to be used in the text string TLV of the GET INPUT USAT command)	M
C	TLV	Inline Value 2 TLV, containing the default text for the default text TLV of the GET INPUT USAT command	O

Coding of the attributes:



The following parameters shall be used for the generated GET INPUT command:

Field	Comment
Response length	Minimum length: the value supplied by the attribute byte is to be used; Maximum length: 'FF' shall be used
Command Qualifier	UE may echo user input on the display; User input to be in unpacked format; No help information available;

If more parameters are necessary for the Get Input command, for security reasons, the Execute USAT command byte code shall be used.

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Problem in memory management	Memory allocation problem	Stop
Reference to undefined	Reference to undefined variable	Stop

9 Native Commands

Native Commands or "plug-ins" shall be used to provide specific functionality not contained in the USAT Interpreter byte code set. This can be e.g. operating system calls, execution of specific security algorithms, calculation routines or conversion routines. All native commands are called using the Execute Native Command byte code.

Each native command shall have a Native Code Identifier. The Native Code Identifier has a size of 2 bytes and is binary coded, most significant byte first. The values '0000' to '7FFF' are RFU for native commands specified in the present document. Other values may be used for proprietary implementations.

Native Commands are optionally to be supported by the USAT Interpreter. If Native Commands are supported by the USAT Interpreter, which are specified within the present document (using a NCI specified in the present document), they shall be implemented according to the present document.

Native commands specified by the present document:

This is for further study.

10 End to End Security

10.1 Encrypt

This is for further study.

10.2 Decrypt

This is for further study.

11 Modes of operation

This is for further study.

11.1 Pull

This is for further study.

11.2 Push / Cell Broadcast

This is for further study.

12 Error coding

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 Error Code variable ('05') in the system information partition.

Type of error	Coding
No error	‘0000’
Syntax error	‘6F01’
Jump to undefined	‘6F02’
Problem in memory management	‘6F03’
Security problem	‘6F04’
Reference to undefined	‘6F05’
Out of range	‘6F06’
User abort	‘6F07’
Execution error	‘6F08’
USAT command failed	'6F09'
USAT command not allowed	'6F0A'
USAT Interpreter transmission error	'6F0B'
Type mismatch	‘6F0C’
General unspecific error	‘6FFF’

13 Tag Values

The present document uses the following Tag values:

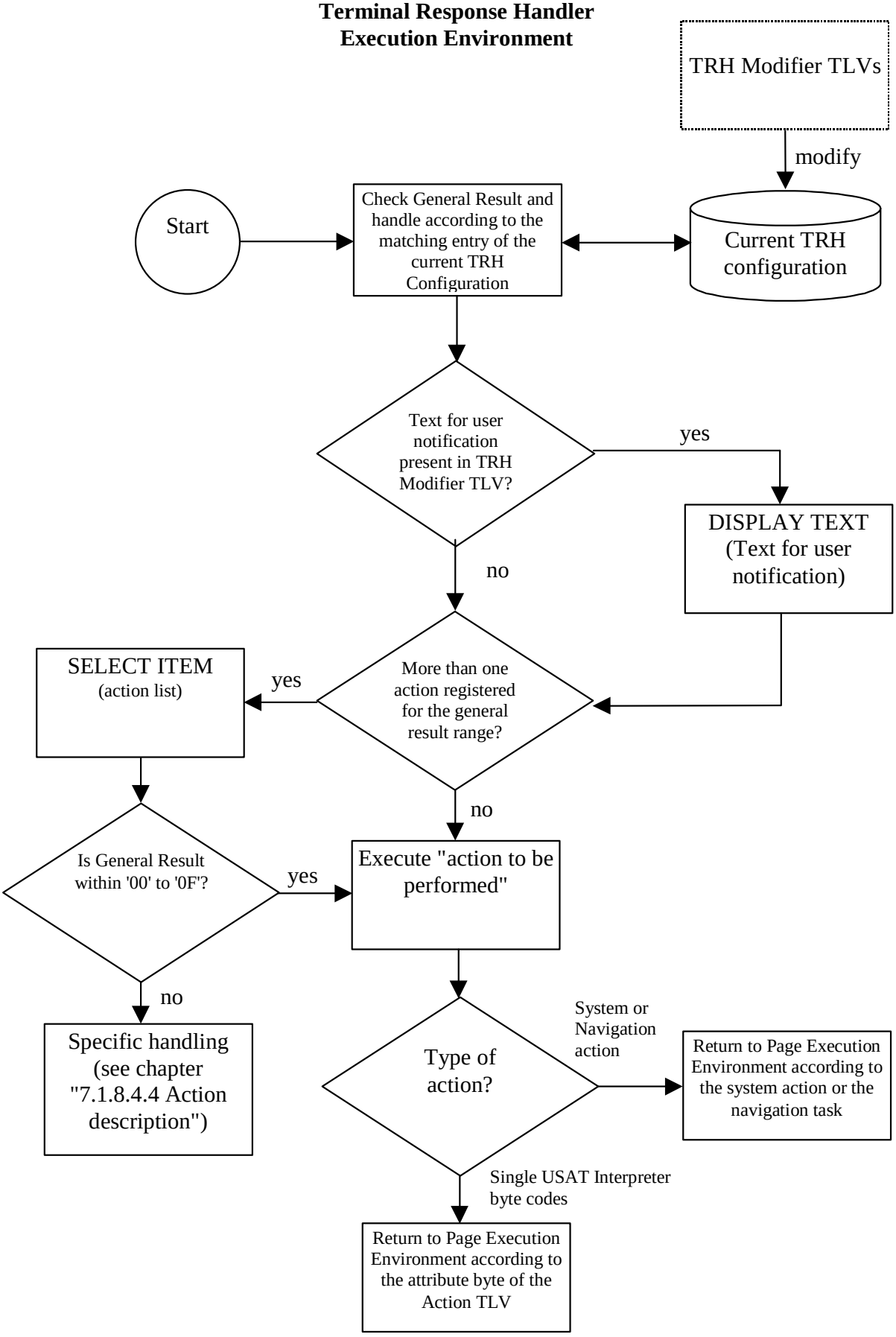
Tag Value	Usage
'01' / '81'	Page Tag
'02'	Page Identification Tag
'03'	Page Unlock Code Tag
'04'	One Time Password Tag
'05'	Keep Alive List Tag
'06'	Service ID Tag
'07'	String Pool Tag
'08' / '88'	Terminal response handler modifier tag
'09' / '89'	Action TLV tag
'0A' / '8A'	Navigation Unit Tag
'0B'	Anchor Tag
'0C'	Anchor Reference Tag
'0D'	Variable Identifier List Tag
'0E' / '8E'	Inline Value Tag
'0F' / '8F'	Inline Value 2 Tag
'10'	Input List Tag
'11'	Ordered TLV List Tag
'12' / '92'	Page Reference Tag
'13' / '93'	Submit Configuration Tag
'14'	Submit Data Tag
'15'	Gateway Address Tag
'16'	Submit Tag
'17' to '3F'	RFU for data structures
'40'	Set Variable Tag
'41'	Assign and Branch Tag
'42'	Extract Tag
'43' / 'C3'	Go Back Tag
'44'	Branch on Variable Value Tag
'45' / 'C5'	Exit Tag
'46' / 'C6'	Execute USAT Command Tag
'47' / 'C7'	Execute Native Command Tag
'48'	Get Length Tag
'49'	Get TLV Value Tag
'4A' / 'CA'	Display Text Tag
'4B' / 'CB'	Get Input Tag
'4C' to '7F'	RFU for commands

All other Tag values are RFU.

Annex A (Informative): Terminal Response Handler Flow Charts

After an USAT command a General Result is returned and the returned General Result is checked according to the current Terminal Response Handler Configuration. The further processing depends on the current Terminal Response Handler Configuration which may have been modified by Terminal Response Handler Modifier TLVs.

Terminal Response Handler Execution Environment



Annex B (Informative): Example of Accessing USAT Interpreter Functionality in Wireless Mark-up Language

B.1 Introduction

B.1.1 Purpose

The annex demonstrates how USAT Interpreter functionality can be provided to the application developer by usage of a mark-up language without requiring in-depth knowledge of USAT Commands. The annex is informative and the functionality does not have to be limited to what is proposed here.

The annex proposes how to form WML [B5] code to address USIM Application Toolkit commands and Plug-In extensions. The WML code constitutes the deck delivered from an application provider as a response to a request for an application.

The intention is to provide a necessary base for developing applications in WML. The annex thus describes a limited set of WML that can be regarded as the minimal support needed for application development.

B.1.2 Terminology

This document uses the terms Implicit and Explicit calls when discussing access to USAT and Plug-In functionality. The distinction is that when the term Implicit is made it refers to cases where the WML code does not indicate that a specific command is called but the rendering of the WML will be encoded to use specific commands.

When using the term explicit, it refers to cases where the WML code specifically states that it intends to call a specific function.

As an example, one can say that the following WML code is an implicit call of the USAT command displayText since that function will be used to render the WML

```
<wml>
 <card id="implicit">
  <p>
   Displayed
  </p>
 </card>
</wml>
```

The explicit version of that WML would be

```
<wml>
 <card id="explicit">
  <p>
   <do type="vnd.3gpp.org">
    <go href="efi://vnd.3gpp.interpreter/atk/displayText?text=Displayed"/>
   </do>
  </p>
 </card>
</wml>
```

B.1.3 Definitions and abbreviations

Acronym	Definition
DCS	Data Coding Scheme
PID	Protocol IDentifier
WAP	Wireless Application Protocol
WML	Wireless Mark-up Language
UCS	Universal Character Set
URL	Uniform Resource Locators
USIM	Universal Subscriber Identity Module
UTF	Unicode Transformation Format
XML	eXtensible Mark-up Language

B.2 Namespace

The WML code makes use of the concept of namespace to address the functionality. The WML code in this document uses the `efi` scheme, as defined by WAP Forum in reference [B6], to address USAT commands, Card plug-ins and other explicitly addressed functionality. The concepts used in the namespace for addressing this functionality is described in that specification.

According to the terminology of the EFI Framework specification, the USAT Interpreter can be introduced as an EF Class. The addressing is then fully compliant with those ideas, regardless of future development.

According to the EFI Framework specification, the WML namespace used for addressing services from WML is structured according to the below.

`efi://vnd.3gpp.interpreter/atk/sendSm`

In the terminology used in the EFI Framework, the above URL uses the default implementation of the vnd.3gpp.interpreter class as the server and calls the service named `atk/sendSm`.

B.2.1 The USAT Interpreter EF Class

The USAT Interpreter is viewed as an EF Class with the name vnd.3gpp.interpreter. Its services are named using an internally hierarchical structure to group the command types.

According to the EFI Framework, service names can contain the "/" which can be used to give a logical grouping to the services supplied by the class. The USAT Interpreter class uses this notation to place services in logical groups. The service groups address USAT Commands, Card resident plug-ins and interpreter internal functionality in appropriate groups.

The service grouping used is listed in the below table.

Service Type	Service Group
USAT commands	atk/
Client side plug-in	cpi/
USIM Manufacturer specifics	ssp/
Interpreter Internals	ipi/

This document only specifies specific forms for the `atk` and `ipi` groups of services.

B.2.2 Examples

The following lists a few examples of URLs that are used to address different type of functionality.

The following URL addresses the USAT command `powerOffCard` with argument `card`

```
efi://vnd.3gpp.interpreter/atk/powerOffCard?card=<value>
```

The following URL addresses a client side plug-in with name `sign`, which is called with argument `doc` containing the data to be signed and `keyId` identifying the key to be used.

```
efi://vnd.3gpp.interpreter/cpi/sign?doc=<text>&keyId=<value>
```

The following URL addresses the USIM Manufacturer specific function `doSpecifics` with data as contained by `data`.

```
efi://vnd.3gpp.interpreter/ssp/doSpecifics?data=11624
```

Here are some examples of more complete code using the addressing principles.

```
<wml>
 <card id="play">
  <p>
   I will play you a tone!
   <do type="vnd.3gpp.org">
    <go href="efi://vnd.3gpp.interpreter/atk/playTone?toneId=03&amp;
              timeUnit=01&amp;duration=10&amp;text=Hej" />
   </do>
  </p>
 </card>
</wml>
```

```
<wml>
 <card id="test">
  <p>
   Calling funny plugin
   <do type="vnd.3gpp.org">
    <go href="efi://vnd.3gpp.interpreter/cpi/doGuess?
              age=$(age)&amp;outputVar=output">
     <setvar name="age" value="35"/>
    </go>
   </do>
  Olle has a mobile of the brand $(output)!
  </p>
 </card>
</wml>
```

B.3 WML

This chapter gives an introduction to the WML and extended functionality.

B.3.1 WML Syntax

B.3.1.1 The WML page

A WML page is either stored at an application provider, or stored in compiled form on the USIM.

B.3.1.2 Entities

Entities are used to specify characters in the document character set which either need to be escaped in WML or may be difficult to enter in a text editor. WML text can contain numeric or named character entities. All entities begin with an ampersand and end with a semicolon.

The following predefined named entities are supported:

Entity	Character
&amp;	&
&apos;	*apostrophe*
&lt;	<
&gt;	>
	non-breaking space
&shy;	*soft hyphen*
&quot;	"

B.3.1.3 Elements

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

<tag/> or <tag>content</tag>

B.3.1.4 Attributes

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

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

All attribute values are quoted using double quotation marks (").

B.3.1.5 Variables

Variables can be used in the place of strings and are substituted at run-time with their current values. Anywhere the variable syntax is legal, an `$` character followed by `(VARIABLENAME)` indicates a variable substitution:

$(VARIABLENAME)

The `setvar`, `input` and `select` elements can be used to set a variable.

Different variables may contain characters from different character sets. The type of a variable is set the first time the variable is defined in the WML document (for instance in a `setvar`, `input` or `select` element).

Variables have to be named with characters supported by ISO-8859-1.

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

B.3.2 Extended functionality interface

Some commands on the USAT Interpreter are not possible to address using WML [B5] tags. In those cases, an EFI [B6] syntax is used according to the following example:

```
<go href="efi://vnd.3gpp.interpreter/atk/functionName?arg1=a1"/>
```

The syntax is described in Chapter B.2.

The function name is unique for the command. All commands are called with different arguments, see chapter B.5, and the arguments are used for both input and output data. The name of the function defines which command to be called.

B.4 Implicit calls using WML syntax

Supported WML tags are described in this chapter.

B.4.1 Prologue

A WML document always starts with an XML declaration and a document type declaration.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE wml PUBLIC "-//3GPP//DTD USAT-WML 1.0//EN"
          "http://www.3gpp.org/DTD/USAT-WML10.dtd">
```

B.4.2 Character encoding

The document always begins with an XML declaration containing the `encoding` attribute.

The following examples show the declarations for two different character encoding:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

or

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

This example shows how Unicode can be used for text that are to be input and output on the telephone, and for the content of variables. It also shows that the Unicode variable content can be passed to the application provider as a parameter value to the "`go href`" command. The whole URL in "`go href`" is limited to contain valid URL characters. However, the content of the variables that are passed in the query string can be Unicode, e.g. in the example, the content of the variable `DRINK` is Unicode.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE wml PUBLIC "-//3GPP/DTD USAT-WML 1.0//EN"
     "http://www.3gpp.org/DTD/USAT-WML1.0.dtd">
<wml>
  <card>
    <p>
        <select name="DRINK" title="喝什么">
        <option value="可乐" >可乐</option>
        <option value"雪君" >雪君</option>
        <option value"芬达" >芬达</option>
        </select>
        你选的是$(DRINK)
      <do type="accept">
         <go href="http://webserver/path/page.asp?drink=$(DRINK)" />
      </do>
    </p>
  </card>
</wml>
```

B.4.3 Elements

The order of elements in a WML document is significant since the USAT interpreter will interpret the elements in sequence.

In the following subsections, the last column in the attribute tables indicates if the attribute is Optional(O) or Mandatory(M).

The mapping of implicit WML tags to USAT commands are explained in the following table.

WML tag	USAT Command
wml	-
p	If containing text, DISPLAY TEXT is used.
br	-
input	GET INPUT
card	-
option	SELECT ITEM (In the select tag.)
select	SELECT ITEM
go	SELECT ITEM / SEND SM
setvar	-
noop	-
do	-
refresh	-

B.4.3.1 wml element

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

Syntax

`<wml>content</wml>`

B.4.3.2 card element

The card element defines a container of text and elements in a WML document. A document may contain multiple card elements but card elements may not be nested. The first card element in a document is the start card.

Syntax

`<card>content</card>`

Attribute	Explanation	
id	This attribute specifies a unique id of the card within the deck.	O
newcontext	This attribute specifies if the current USAT interpreter context is to be re-initialised. Allowed values: `true` or `false` (Default).	O

```
<card id="card1">
.
.
</card>
```

B.4.3.3 p element

The p element, or paragraph element, delimits a text section.

No arguments are supported for the p element.

Syntax
** content **

B.4.3.4 br element

The br element inserts a line break in the displayed text.

The <br/> element can not take any arguments.

Syntax
**
**

B.4.3.5 input element

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

Syntax
<input/>

Attribute	Explanation	
name	This attribute specifies the name of variable to set.	M
value	This attribute specifies the default value of the variable named in the `name` attribute.	O
format	This attribute Expected data format entered by the user. The following values are allowed: `*M` - Any character. (Default) `*N` - Any numeric character.	O
emptyok	This attribute specifies whether or not empty input will be accepted Allowed values: `true` or `false` (Default).	O
maxlength	This attribute specifies the max number of bytes that can be entered by the user.	O
title	This attribute specifies the prompting string.	O
class	This attribute specifies the type of the variable. The following values are allowed: `SMSDefault` – Default for an ISO-8859 WML document. `UCS2` - Default for an UTF-8 WML document.	O

```
<input title="Please enter your phone number" name="PHONE" format="*N" maxlength="20"/>
```

B.4.3.6 select Element

The `select` element defines and displays a set of optional list items from which the user can select an item. An `option` element is required for each item in the list, see Section B.4.3.7. The name of the menu, normally displayed by the telephone, is specified by the `title` attribute.

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

Syntax
<select>content</select>

Attribute	Explanation	
title	This attribute specifies the title of the menu.	O
name	This attribute specifies the name of the variable to set.	O
iname	This variable specifies the name of the variable to set with the index result of the selection. See the WAP WML specification [B5].	O

B.4.3.7 option element

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

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

Syntax

`<option>content</option>`

Attribute	Explanation	
value	This attribute specifies what the variable named in `select` attribute `name` is set to, if this `option` element is selected.	O
onpick	This attribute specifies a destination URI to go to, if this `option` element is selected.	O

This example illustrates the use of `select` and `option`. If the user selects the "Banking" option, a jump will occur to "card2". If the user selects the "Gambling" option, a jump will occur to "card3". If "[Home]" is selected a GET request will be sent for the "home.wml" document. Note that the `value` attribute in the `option` element can not be used for anything if the corresponding `onpick` attribute refers to an external URL.

```
<select title="Please choose service" name="SELECTION">
 <option value="BANKING" onpick="#card2">Banking</option>
 <option value="GAMBLING" onpick="#card3">Gambling</option>
 <option value="Not used." onpick="http://www.3gpp.org/home.wml">[Home]</option>
</select>
```

B.4.3.8 go element

The `go` element declares a `go` task to a URL or to a specified card in the document. The `go` element may also be used for performing USAT interpreter or Gateway specific commands.

Note that after each "`go href`" referring to an external URL, no more WML elements will be executed. Using text or WML tags after a "`go href`" referring to an external URL may cause problems for the application.

The URL may contain variable references.

The URL starts with "`https://`", if SSL is to be used for connecting to the application server.

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

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

Syntax

`<go/>`

`<go>content<go>`

Attribute	Explanation	
href	This attribute identifies the destination URI.	M
method	This attribute specifies the http submission method to be used by the Gateway. The following values are allowed: `get` - HTTP GET will be used. (Default) `post` - HTTP POST will be used.	O

```
<card>
 <p>
  <input title="Variable" name="VARIABLE"/>
  <do type="accept">
   <go href="http://www.3gpp.org/page.jsp?f=$(VARIABLE)&amp;l=StaticText "/>
   </do>
 </p>
</card>
```

```
<card>
 <p>
  <input title="First name" name="FIRSTNAME"/>
  <input title="Last name" name="LASTNAME"/>
  <input title="Age" name="AGE"/>
  <do type="accept">
   <go method="post" href="http://www.3gpp.org/page.jsp?
    f=$(FIRSTNAME)&amp;l=$(LASTNAME)&amp;a=$(AGE)"/>
  </do>
 </p>
</card>
```

A card reference starts with the character '#'.

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

B.4.3.9 setvar element

The `setvar` element sets the value of a variable.

The `class` attribute is used for setting the type of the variable according to the present document.

Syntax

`<setvar/>`

Attribute	Explanation	
name	This attribute specifies the name of the variable to be set.	M
value	This attribute specifies the value the variable is set to. May only contain fixed text. Variables are not allowed.	M

class	This attribute specifies an optional type specification of the variable, used for conversion purposes in the Gateway. The following values are allowed:
	`SMSDefault`
	`SMSDefault.packed`
	`UCS2`
	`binary.base64` - The variable contains binary data coded according to base64 encoding. This is the default value if the "class" attribute is omitted. The "binary.base64" class is used for instance when encrypted data is sent to the content. The type in the USAT interpreter will be "Binary" (Default).
		O

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

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

`setvar` is contained in a `refresh` element

```
<card id="setexample2">
 <p>
  <do type="accept">
   <refresh>
    <setvar name="HEXVARIABLE" class="binary.base64" value="A678F5D3"/>
   </refresh>
  </do>
  <do type="accept">
   <go href="http://www.3gpp.org?a=$(HEXVARIABLE)"/>
  </do>
 </p>
</card>
```

`setvar` is contained in a `go` element. The variables are set before the go element is executed.

```
<card id="setexample3">
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3ggp.interpreter/cpi/encrypt?
       a1=$(KEY1)&amp;a2=$(KEY2)&amp;outputVar=out">
    <setvar name="KEY1" class="binary.base64" value="F5FF34FF"/>
    <setvar name="KEY2" class="binary.base64" value="90AB45DA"/>
   </go>
  </do>
 </p>
</card>
```

B.4.3.10 noop element

The noop element specifies that nothing will be done. The noop element requires a start tag only.

Syntax

<noop/>

B.4.3.11 do element

The `do` element is a general mechanism for the user to act upon the current card. The supported types are `accept` and `vnd.3gpp.org`. Both of these imply that the task following the `do` element is always executed.

This means that the execution of the script does not stop at the do element. If a stop before the do element is desired, a construction as in 0 can be used.

Syntax
<do>content</do>

Attribute	Explanation	
type	This attribute specifies the type of the do element. The following values are allowed: vnd.3gpp.org - When the do element contains a USAT interpreter specific command. accept - All other cases.	M

```
<wml>
 <card id="command">
  <p>
   <input title="Enter your age:" name="AGE"/>
   <do type="accept">
    <go href="http://www.3gpp.org/survey.asp?f=$(AGE)&amp;name=Martin"/>
   </do>
  </p>
 </card>
</wml>
```

B.4.3.12 refresh Element
The refresh element surrounds the setvar tag. The refresh tag has no function in itself.

Syntax
<refresh>content</refresh>

B.5 Explicit calls using WML syntax

This chapter demonstrates how the namespace can be used to explicitly address USAT Commands, USAT Interpreter specific functions and Plug-ins. The purpose is to demonstrate how this can be done rather than to describe how the complete command set of the USAT Interpreter is addressed.

Mandatory parameters need always be present in an explicit call and the optional attributes may be left out. The last column in the following tables indicates if the attribute is M-mandatory or O-optional.

An argument value can include a variable, which is substituted at run-time with its current value.

B.5.1 Services for USAT Commands

Access to USAT commands is grouped into the service group `atk`. Anything that belongs to this group of services can be coded, by the gateway, by using generic coding on the byte code level.

The following table lists the logical group of services used for calling USAT commands.

Service Name
atk/launchBrowser
atk/playTone
atk/provideLocalInfo
atk/refresh
atk/runATCommand
atk/sendUSSD
atk/sendSM
atk/setupCall
atk/setIdleModeText

For detailed information on the parameters and data format, see 3G TS 31.111 [B3]. Although the "GO" tag is used, no message is sent to the server, as the commands are executed locally on the USIM.

The following chapters handle these functions in detail. The parameter names as listed in the tables below are the same as the ones that are to be used in the URL query string. The parameter names are case sensitive.

B.5.1.1 Launch Browser

This command causes the USIM to request that the ME start a browser to interpret the content corresponding to the URL.

Service name: **`atk/launchBrowser?qualifier=&URL=`**

Argument	Argument value	
qualifier	The Command Details to use (see [B3]). The value is given in decimal format. The default value is 0.	O
URL	The URL whose contents is to be displayed.	M

A browser will be launched and the URL "http://www.3gpp.org/page.wml" will be fetched.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/launchBrowser?
       URL=http://www.3gpp.org/page.wml"/>
  </do>
 </p>
</card>
```

B.5.1.2 Play tone

This command makes the mobile station play a tone.

Service name: **atk/playTone?toneId=&timeUnit=&duration=&text=**

Argument	Argument value	
toneId	01: Dial tone	M
	02: Called subscriber busy	
	03: Congestion	
	04: Radio path acknowledge	
	05: Radio path not available	
	06: Error / special information	
	07: Call waiting time	
	08: Ringing tone	
timeUnit	00: minutes	M
	01: seconds	
	02: tenths of seconds	
duration	Coded as integer multiples of the time unit used. Decimal value. Allowed values: 0-255.	M
text	Text to display. (Corresponds to the alpha identifier according to [B3])	O

In this example, the mobile phone is requested to play a congestion tone with duration of 10 seconds. Since text string is empty, no text will be displayed.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/playTone?
        toneId=03&amp;timeUnit=01&amp;duration=10"/>
  </do>
 </p>
</card>
```

B.5.1.3 Provide Local Information

This command is used to get location information from the mobile station. Different location parameters can be fetched from the mobile phone and put into a variable.

Service name: **atk/provideLocalInfo?qualifier=&outputVar=**

Argument	Argument value	
qualifier	00: location information (7 bytes)	M
	01: IMEI of ME (8 bytes)	
	02: Network measurement results and BCCH list (16 bytes)	
	03: Date, time and time zone (7 bytes)	
	04: Language setting (2 bytes)	
	05: Timing advance (2 bytes)	
outputVar	Variable to contain output data.	M

In this example, the IMEI is fetched and put in the variable `imeiOutput`. On the next line, the IMEI is sent to a content provider.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/provideLocalInfo?
       qualifier=01&amp;outputVar=imeiOutput"/>
  </do>
  <do type="accept">
   <go href="http://www.arne.se?IMEI=$(imeiOutput)"/>
  </do>
 </p>
</card>
```

B.5.1.4 Refresh

This command makes the USIM notify the mobile phone of changes in the USIM configuration as the result of USIM application activity. Depending on the command qualifier, different tasks will be performed. For more information see [B3].

Service name: **`atk/refresh?qualifier=&numberOfFiles=&fileList=`**

Argument	Argument value	
qualifier	`00`: USIM Initialisation and Full File Change Notification `01`: File Change Notification (requires file list) `02`: USIM Initialisation and File Change Notification (requires file list) `03`: USIM Initialisation `04`: USIM Reset	M
numberOfFiles	Number of files included in `filelist`. Decimal value. Default: `0`.	O
fileList	List of files.	O

In the example, a USIM initialisation is requested, and in addition, the mobile phone in notified that two files on the USIM have been updated, 3F00/2F05 and 3F00/7F10/6F3A.

```
<card id="command">
 <p>
  <do type="vnd.3gpp.org">
   <go href ="efi://vnd.3gpp.interpreter/atk/refresh?qualifier=02&amp;
        numberOfFiles=02&amp;fileList=3F002F053F007F106F3A"/>
  </do>
 </p>
</card>
```

Full paths are given to files. Each file path is at least 4 octets in length. An entry in the file description begins with '3FXX' and there is no delimiters between files.

B.5.1.5 Run AT Command

This command makes the USIM request the ME to execute an AT Command.

Service name: **`atk/runATCommand?command=&text=&iconId=`**

Argument	Argument value	
command	The AT Command string that is to be executed	M
text	Text to be displayed to the user.	O
iconId	The identifier of an icon to show instead of text.	O

```
<card id="command">
 <p>
  <do type="vnd.3gpp.org">
   <go href ="efi://vnd.3gpp.interpreter/atk/runATCommand?
     command=ATD0706746151&amp;text=Calling"/>
  </do>
 </p>
</card>
```

B.5.1.6 Send USSD

This command sends a byte string by the Unstructured Supplementary Service.

Service name: **`atk/sendUSSD?text=&ussd=`**

Argument	Argument value	
text	Text to display.	O
ussd	According to [B1].	M

In this example, a USSD message with the contents "*21*1222#" is sent to the network.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/sendUSSD?
             text=MessageText&amp;ussd=*21*1222#"/>
  </do>
 </p>
</card>
```

B.5.1.7 Send SM

This command sends a plain text SM to a particular destination.

Service name: **atk/sendSM?userData=&pid=&dcs=&bNumber=&smscAddress=**

Argument	Argument value	
userData	Text in the SM.	O
pid	Protocol identifier. Decimal value. Default: 0.	O
dcs	Data Coding Scheme, according to [B4]. Decimal value.	O
bNumber	The called party number.	M
smscAddress	The number of the service center.	O

In this example, a text SM, with contents as entered by the user, is sent to MSISDN "0706754321". As "PID" and "DCS" are omitted, the default values "0" and "242" decimally are used. The Service Centre "+46705008999" is used, regardless of the default value in the mobile phone.

```
<card>
 <p>
  <input title="Please enter message" name="m"/>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/sendSM?userData=$(m)&amp;
             bNumber=0706754321&amp;smscAddress=+46705008999"/>
  </do>
 </p>
</card>
```

B.5.1.8 Set up call

This command requests the mobile phone to initiate a call.

Service name:
atk/setupCall?qualifier=&text=&capability=&timeUnit=&duration=&bNumber=

Argument	Argument value	
qualifier	`00`: only if not currently busy `01`: only if not currently busy, with redial `02`: putting all other calls on hold `03`: putting all other calls on hold, with redial `04`: disconnecting all other calls `05`: disconnecting all other calls, with redial	M
text	Text to display. (Corresponds to the alpha identifier according to [B3].)	O
capability	Capability Configuration Parameters. For coding, see [B2]. Default: None.	O
timeUnit	This argument is mandatory if `duration` attribute is used. Default: Not used. `00`: minutes `01`: seconds `02`: tenths of seconds	O
duration	Coded as integer multiples of the time unit used. Decimal value. Allowed values: 0-255. Default: Not used.	O
bNumber	The called party number.	M

In this example, the USIM requests the mobile phone to set up a call to "0707789613", if not currently busy with another call. No text is displayed, no Capability Configuration Parameters are attached, and no automatic retries to set up the call will be made.

B.5.1.8.1 <card>

```
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/setupCall?
             qualifier=00&amp;bNumber=0707789613"/>
  </do>
 </p>
</card>
```

B.5.1.9 Set Idle Mode Text

This command sets a text on the idle screen of the mobile station.

If no text attribute is included or the text attribute consists of an empty string, the existing idle mode text on the mobile phone will be removed.

Service name: **`atk/setIdleModeText?text=`**

Argument	Argument value	
text	The idle mode text to display.	O

This example will set the idle mode text to "Welcome".

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/atk/setIdleModeText?
             text=Welcome"/>
   </do>
 </p>
</card>
```

B.5.2 Services for Interpreter Commands

These are commands that are directed to the Interpreter itself and thus are internally handled by the interpreter.

The following table lists the logical group of services used for calling interpreter internal functions.

Service Name
ipi/getInterpreterVersion
ipi/getBufferSize

B.5.2.1 Get Interpreter Version Information

This command reads the version information of the USAT Interpreter and assigns it to the specified variable.

Service name: **`ipi/getInterpreterVersion?outputVar=`**

Argument	Argument value	
outputVar	Variable to contain output data.	M

B.5.2.2 Get Interpreter Buffer Size

This command reads the size of the receive and send buffer of the USAT Interpreter and assigns it to the specified variable.

Service name: **`ipi/getBufferSize?outputVar=`**

Argument	Argument value	
OutputVar	Variable to contain output data.	M

In the following example, the interpreter buffer size and version information are put into the variables "bufferSize" and "version" respectively. On the next line, the information is sent back to the Application Provider.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/ipi/getInterpreterVersion?
             outputVar=version"/>
  </do>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/ipi/getBufferSize?
             outputVar=bufferSize"/>
  </do>
  <do type="accept">
   <go href="http://www.server.com?VERSION=$(version)&BUFFER=$(bufferSize)"/>
  </do>
 </p>
</card>
```

B.5.3 Services for Calling Client Plug-Ins

This chapter illustrates the way the addressing for calling a card plug-in is done and the principles for handling the arguments to the plug-in. The addressing enables the application to call any plug-in that is available for the application. The actual plug-ins that are available for the application depends on the configuration of the USAT Interpreter. On the byte code level, the card plug-ins are called in a generic way. The translation to generic format is done by the gateway.

To exemplify the calling of plug-ins from the application, an example plug-in with the name `myPlugin` is used. It is assumed that there are seven arguments to the plug-in as described in the table below.

a#	Argument	Argument value	
a1	homeTown	The home town of the user	M
a2	currentTown	The town where the user currently is.	M
a3	homePhone	The home phone number of the user	O
a4	buyTicket	This parameter acts as a Boolean value. If it is set to 1, a ticket will be reserved. If set to 0, only timetable is provided. The default behaviour is to provide timetable information only.	O
a5	timeToLeave	If set, the parameter gives a date when the user wishes to start travelling.	O
a6	timeToArrive	If set, this parameter gives a date when the user wishes to arrive.	O
a7	transport	The desired means of transport for the user.	O

As a calling convention for plug-ins, the parameter names are enumerated using **a** as a prefix. The enumeration order indicates the order in which the arguments are sent to the plug-in. Optional parameters that are not used are left out from the URL query string.

The order of the parameters in the query string is insignificant. It is the naming of the parameters that control the order when calling the plug-in.

This service will call the plug-in `myPlugin`. Any other plug-in is called in the same manner based on its documentation. The plug-in services are always placed in the `cpi` service group.

Service name: **`cpi/myPlugin`**

In this example, the plug-in `myPlugin` is called using only arguments 1,2 and 7 as described by the documentation.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/cpi/myPlugin?
      a2=Stockhom&amp;a7=Train&amp;a1=Paris"/>
  </do>
 </p>
</card>
```

The WML code above causes the gateway to construct a call to the generic plug-in mechanism to call a plug-ins whose name is `myPlugin`. The arguments to the generic call are inserted in the order the naming enumerates them.

B.6 References

[B1] 3GPP TS 22.030: "Technical Specification Group Services and System Aspects;Man-Machine Interface (MMI) of the User Equipment (UE)"

[B2] 3GPP TS 24.008: " Technical Specification Group Core Network; Mobile radio interface layer 3 specification;Core Network Protocols – Stage 3"

[B3] 3GPP TS 31.111: "3rd Generation Partnership Project (3GPP); USIM Application Toolkit (USAT)"

[B4] 3GPP TS 23.038: "3rd Generation Partnership Project (3GPP); Alphabets and language-specific information."

[B5] Wireless Application Protocol Forum: "Wireless Markup Language Specification. Version 1.3. 19 February 2000. Available: http://www.wapforum.org/"

[B6] Wireless Application Protocol: "EFI Framework. Draft Version 0.15."

History

Document history		
V0.0.9	January 2001	First full ETSI style version
V0.0.10	February 2001	Prepared version for ad-hoc #25 05-07.02.2001 Stockholm
V0.0.11	February 2001	New version after ad-hoc #25 and #29
V1.0.0	March 2001	Presented for information to TSG-T #11
V1.1.0	April 2001	New version after ad-hoc #33 and #35
V1.2.0	May 2001	New version after ad-hoc during T3 #19
V1.3.0	June 2001	New version after ad-hoc #38
V1.4.0	July 2001	New version after ad-hoc #43
V1.5.0	July 2001	New version after ad-hoc #45
V1.6.0	August 2001	New version for ad-hoc #48
V1.6.1	August 2001	Revised version for ad-hoc #48
V1.7.0	August 2001	New version after ad-hoc #48
V1.7.1	August 2001	Version for presentation to T3 #20
V1.7.2	September 2001	Editorial changes of T3 #20 incorporated
V2.0.0	September 2001	For presentation to TSG-T #13 for approval