3GPP TSG-T (Terminals) Meeting #16
Marco Island, USA 4 – 6 June 2002

Tdoc TP-020115

Source: T3

Title: Change Requests to TS 31.113 USAT interpreter; stage 3; Byte Codes

Document for: Approval

This document contains several change requests as follows:

T3 Doc	Spec	CR	Rel	Cat	Subject
T3-020341	31.113	008	5	F	Clarification of the Assign and Branch command
T3-020342	31.113	009	5	F	Miscellaneous corrections and clarifications on the specification.
T3-020343	31.113	010	5	F	Clarification of history management
T3-020344	31.113	011	5	F	Removal of ciphering of the One Time Password
T3-020345	31.113	012	5	F	Error on access to permanent variable
T3-020346	31.113	013	5	F	Clarification of the Terminal Response Handler Mechanism
T3-020349	31.113	014	6	B	Terminal Response Handler Modifier "remove" attribute enhancements
T3-020350	31.113	015	6	B	Addition of error handling
T3-020351	31.113	016	6	B	Addition of functionality for security plug-ins

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020341

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **008** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Clarification of the Assign and Branch command
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	22 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The current description of the Assign and Branch command needs further clarification. Questions on the usage and interpretation of the specified behaviour are the reason for the present clarification and correction.
Summary of change:	⌘	Clarification on the usage of the Ordered TLV List TLVs in Assign and Branch byte codes. An essential correction describes in addition the behaviour of the USAT Interpreter when the Ordered TLV List TLVs contained in an Assign and Branch byte code resulted in a SELECT ITEM command with only one item.
Consequences if not approved:	⌘	USAT Interpreter implementations may show a non interoperable, different behaviour when rendering Assign and Branch byte codes.

Clauses affected:	⌘	8.2, 8.2.3
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

8.2 Assign and Branch

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

The TLVs contained in the Ordered TLV List TLVs define whether the USAT Interpreter shall build a SELECT ITEM command according to 3GPP TS 31.111 [1] or perform an action immediately.

When a SELECT ITEM command is built by the USAT Interpreter, the command qualifier to be used shall be '03'.

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 TLV

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

Each of these TLVs encapsulate the

- "Inline Value 2",containing the text of a single item of the SELECT ITEM command;
- "Inline Value", containing a value to be 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 TLVs.

The "Assign and Branch" byte code can contain one or more Ordered TLV List TLVs. If more than one Ordered TLV List TLVs are present within the same "Assign and Branch" byte code, the following rules shall apply:

- If one or more Ordered TLV List TLVs containing an Inline Value 2 TLV are present in the same Assign and Branch TLV in addition to one or more Ordered TLV List TLVs not containing an Inline Value 2 TLV, the USAT Interpreter shall ignore the Ordered TLV List TLVs which do not contain the Inline Value 2 TLV. I.e. the items of the generated SELECT ITEM command are only determined by the Ordered TLV List TLVs which contain an Inline Value 2 TLV. Any actions defined by the Ordered TLV List TLVs not containing an Inline Value 2 TLV are ignored.

- If only Ordered TLV List TLVs not containing an Inline Value 2 TLV are present in the same Assign and Branch TLV, the USAT Interpreter shall take into account the first Ordered TLV List TLV only.

-When optional TLVs within the Ordered TLV List TLV 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 (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall assign the value of the Inline value TLV to the destination variable and branch to the page or 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 (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall assign the value of the Inline Value TLV to the destination variable and process next byte code.
present	not present	present	**"Go Selected"** ~~Generation of a SELECT ITEM command by the USAT Interpreter.~~ When the user has selected this item (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall branch to the page or the navigation unit specified within the Page ~~reference~~ Reference TLV. A destination variable identifier shall be ignored for this case.
present	not present	not present	**"Display and Process next byte code"** ~~Generation of a SELECT ITEM command by the USAT Interpreter.~~ When the user has selected this item (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall process the next byte code. A destination variable identifier shall be ignored for this case.
not present ~~(see NOTE)~~	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 page or the navigation unit specified within the Page ~~reference~~ Reference TLV.
not present ~~(see NOTE)~~	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 ~~(see NOTE)~~	not present	present	**"Direct Go"** ~~No generation of a SELECT ITEM command by the USAT Interpreter.~~ The USAT Interpreter shall directly branch to the page or the navigation unit specified within the Page ~~reference~~ 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.

~~NOTE : If Ordered TLVs containing an Inline Value 2 are present in the same Assign and Branch TLV, the USAT Interpreter shall ignore the Ordered TLVs which do NOT contain the Inline Value 2 TLV.~~

~~If only Ordered TLVs not containing an Inline Value 2 are present in the same Assign and Branch TLV, the USAT Interpreter shall take into account the first Ordered TLV only.~~

If the Ordered TLV List TLVs contained in the "Assign and Branch" byte code resulted in the generation of a SELECT ITEM command with only one item according to the above defined rules, the USAT Interpreter shall immediately perform the action assigned to this item but not generate the SELECT ITEM command. For this optimisation the assigned actions are as follows:

- **"Display, Assign and Branch":** Assign the value of the Inline value TLV to the destination variable and branch to the page or the navigation unit specified within the Page Reference TLV.
- **"Set Variable Selected":** Assign the value of the Inline Value TLV to the destination variable and process next byte code.
- **"Go Selected":** Branch to the page or the navigation unit specified within the Page Reference TLV.

- **"Display and Process next byte code":** Process the next byte code.

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020342

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **009** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Miscellaneous corrections and clarifications on the specification.
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	22 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	4.4: clarification on event handling behaviour and USAT Interpreter idle state 4.4: alignment with TS 31.114: starting of remote pages from menu structure 6.1: correction of indication of sizes of usage areas 6.1.2: modification of wording 6.1.3.1: correction of column title 6.3: correction of referenced TLV 7.2.1: clarification on variable type in ResetVar attribute flag description 7.5, 8.1: clarification on handling of DCS values 7.5, 7.6: correction of Inline Value content 7.9, 7.9.2: clarification of variables containing Anchor Reference 8: correction of TLV table
Summary of change:	⌘	
Consequences if not approved:	⌘	USAT Interpreter implementations could behave differently.

Clauses affected:	⌘	4.4, 6.1, 6.1.2, 6.1.3.1, 6.3, 7.2.1, 7.5, 7.6, 8.1, 7.9, 7.9.2, 8
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

4.4 Activation

Activation of USAT Interpreter depends on USAT Interpreter current state. The USAT Interpreter state corresponds to the presence or the origin of proactive session generated by USAT Interpreter. A state can be:

- Idle (i.e. no proactive session is running);
- Rendering a page (i.e. proactive session issued from byte code command);
- Wait state (see section 4.2.3).

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~~Identification TLV which identifies a locally ~~or remotely~~ stored page. When one of these identifiers is selected, and when USAT Interpreter is in idle state, 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 Identification~~Reference~~ TLV which identifies a locally ~~or remotely~~ stored page. When the UE sends an ENVELOPE command containing an event, and when USAT Interpreter is in idle state, 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,~~ is not in idle 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~~ enters idle state again.

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.

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

Except for the Page String Element usage area, t~~T~~he size of the different usage areas is to be defined by the card issuer and configured during the personalisation process of the USAT Interpreter.

6.1.2 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.

6.1.3.1 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	KeepAliveList 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

6.3 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 Data 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.

[…]

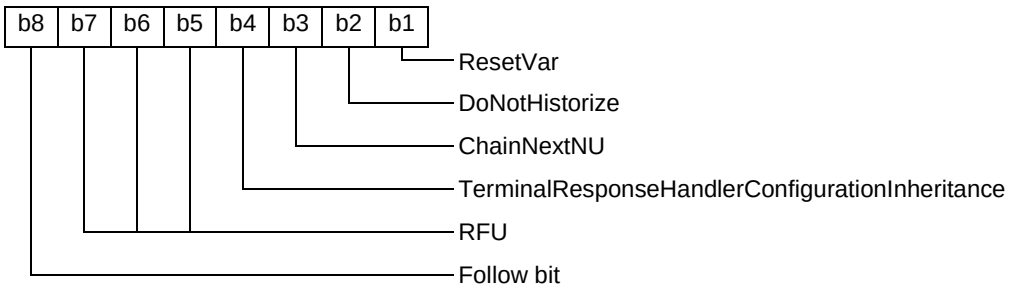
7.2 Navigation Unit

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 clauses specify the attributes and TLVs used in the navigation unit TLV.

7.2.1 Attributes



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.~~

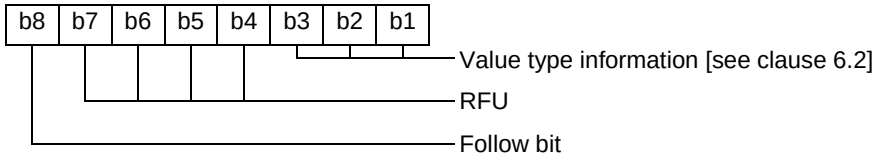
The Inline Value content may contain variable substitution indicators to indicate variable references. Therefore the Inline Value content has to be structured in Length-Value and Variable Substitution Indicator - Variable ID pairs. This structure shall be used even if the Inline Value content does not contain any variable substitution indicators. The possibly available constant data values and variable references have to be rendered according to clause 6.3 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:



If the value type information indicates "unknown", then the DCS attribute of the page shall be applied.

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.

7.9 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 Configuration 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 clause 7.10 .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 value part of an 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 clause 7.3.

7.9.2 Variable Identifier List

Referring to a variable containing the value part of an 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 clause 7.4.

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	Simple TLV Indicator
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 or Variable Identifier List 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.

If a Variable Identifier List TLV is used, the DCS of the variable, which stores the result of the concatenation, shall be set using the following rules:

- If all variables have the same type, then the DCS of the result variable shall be set to the same as the DCS of the first variable in the list;

- If variables have different types, then the DCS of the result variable shall be set to "unknown".

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020343

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **010** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Clarification of history management
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	22 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The behaviour of the history list is to be clarified.
Summary of change:	⌘	4: Removal of short description of history handling; goes into 4.6 4.2.2: The "go back" history navigation action can result in an outgoing message. 4.6: Create new clause. History mechanism is completely defined in this clause. 7.1.8.4.3: Removal of remark on "go back" history navigation action, because it is already explained in 4.6.
Consequences if not approved:	⌘	The USAT Interpreters implementations could behave differently.

Clauses affected:	⌘	4, 4.2.2, 4.6, 7.1.8.4.3
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

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.

[…]

4.2.2 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; and~~.~~

implicitly by a "go back" history navigation action.

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 clause 6.3. 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.

[…]

4.6 History list

The history list is a list of anchor references. This history list also owns an anchor reference pointer which points to a specific entry in the history list. When a navigation unit is completely rendered (i.e. when the USAT Interpreter starts to render another navigation unit), its anchor reference is added on the top of the history list, and the anchor pointer points on it. A navigation unit is not added to this list in following cases:

- If an appropriate attribute flag is set in the navigation unit;
- if the navigation unit does not have any anchor name.

The maximum number of entries in the history list is N (anchor references) where N is greater than or equal to zero. If N=0, the history list mechanism and related navigation actions become deactivated.

If the history list is full, the bottom-most entry is removed from the list in order to free space for a new top-most entry.

The history is reset (is emptied) whenever the USAT Interpreter is initialised.

The USAT Interpreter allows navigation based on the history list and the anchor reference pointer. The history navigation action "go back one entry in history list" means that the navigation unit corresponding to the pointed anchor reference shall be rendered, and the anchor reference pointer is immediately moved down in the list. The origin of this action can be either the system action '02' in terminal response handler configuration, or the Go Back byte code command.

The moving of this anchor reference pointer in the history list does not modify the history list itself.

If the anchor reference pointer reaches the bottom of the history list or the history list does not contain any entry, and if a "go back" history navigation action has to be performed in this situation, then the exception case of the terminal response handler mechanism shall be performed..

Retry-last-proactive-command, system action '03' of the terminal response handler configuration shall not modify the history list.

If, at any time, the anchor reference pointer does not point to the top-most anchor reference in the history list, and if a navigation action other than the "go back" history navigation action (e.g. Assign and Branch byte code command) is performed, then any anchor references between the anchor reference pointer and the top-most entry are deleted from the history list, that means the entry referenced by the anchor reference pointer becomes the top-most entry in the history list.

If the USAT Interpreter does not find the requested anchor locally while processing a "go back" history navigation action, an outgoing message shall be sent to the external system entity to retrieve the page the requested anchor reference belongs to. The Submit TLV shall be formatted in the same way as the previously used Submit TLV to retrieve this page and the USAT Interpreter shall start to render the navigation unit the anchor reference points to.

NOTE: Service providers should take care of that the "go back" history navigation action on remote pages could generate security issues.

[…]

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.

[…]

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020344

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **011** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Removal of ciphering of the One Time Password
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	22 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 *(GSM Phase 2)*
R96 *(Release 1996)*
R97 *(Release 1997)*
R98 *(Release 1998)*
R99 *(Release 1999)*
REL-4 *(Release 4)*
REL-5 *(Release 5)*

Reason for change:	⌘	Ciphering of the One Time Password does not provide additional security. The handling of keys for the OTP is currently not fully specified. The proposal is to remove ciphering of the One Time Password from TS 31.113 at all as this feature adds a lot of complexity for implementations without providing additional security.
Summary of change:	⌘	2: Removal of unused references 3.2: Removal of unused Kic abbreviation. 6.1.1.1: Change of reference number[]. 6.1.3, 6.1.3.2: Mention of ciphered OTP removed. 7.1.3: Change of Page Unlock TLV.
Consequences if not approved:	⌘	Ciphering of the one time password may cause interoperability problems.

Clauses affected:	⌘	2, 3.2, 6.1.1.1, 6.1.3, 6.1.3.2, 7.1.3		
Other specs affected:	⌘	[ ] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	
Other comments:	⌘			

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. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TS 31.111: "USIM Application Toolkit (USAT)".

[2] 3GPP TS 31.114: "USAT Interpreter protocol and administration".

[3] 3GPP TS 23.038: "Alphabets and language-specific information".

[4] ETSI 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.~~

[7~~9~~] IETF RFC 1738: "Uniform Resource Locators (URL)"

[...]

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

[...]

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 Command Filter	This includes the list of allowed USAT Commands. Coding as specified in TS 31.114 [2]. NOTE: Content is dynamic, i.e. it is impacted by the current configuration
'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 clause 12
'06'	Maximum page size for temporary storage of one page	Binary coded, most significant byte first: Number of bytes available for page storage.
'07'	USAT Interpreter issuer identification	URL of USAT Interpreter issuer, coding according to RFC 1738 [~~7~~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'	Reception Buffer Size	Binary coded, most significant byte first: - Receive buffer size in bytes available for messages to be received by the USAT Interpreter. This size includes all possibly needed space for transport headers, security, routing information, concatenation information and so on.
'0A'	USAT Interpreter Byte Code Filter	This includes the list of allowed USAT Interpreter byte codes. Coding as specified in TS 31.114 [2]. NOTE: Content is dynamic, i.e. it is impacted by the

		current configuration.
'0B'	Transmission Buffer Size	Binary coded, most significant byte first: - Transmit buffer size in bytes available for messages to be sent by the USAT Interpreter. This size includes all possibly needed space for transport headers, security, routing information, concatenation information and so on.
'0C'... '13'	RFU	

[...]

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). The following page shall present a Page Unlock TLV to get access to the shared variables. This TLV contains the OTP of the preceding page~~, ciphered or not~~.

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...).

[...]

6.1.3.2 Read access of the temporary variable area

A current page can freely access temporary variables stored by this current page. Variables of the previous page shall only be accessible according to the rules of the table in clause 6.1.3.

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.

[...]

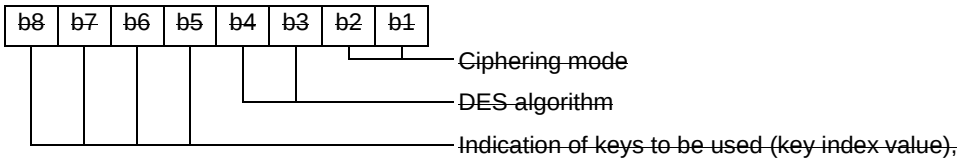
7.1.3 Page Unlock Code

The content of this TLV is a ~~key index and algorithm identifier KIc for ciphering and a~~ sequence of bytes (the page unlock code) ~~encrypted according to the information provided by the KIc. This sequence of bytes shall be decrypted by the USAT Interpreter and~~ to be compared and verified by the USAT Interpreter 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	'XX'~~KIc~~	Any one byte value. The USAT Interpreter shall ignore this byte.~~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 clause 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.~~

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020345

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **012** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Error on access to permanent variable
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	23 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	Only authorised Pages can access permanent variables. Currently, there are two interpretations of TS 31.113 possible: - 1) A permanent variable is identified only by its variable identifier. Each variable owns a "service ID". If a page tries to access to a variable initialised by another page whose service ID is different, then a "security error" could be generated - 2) A permanent variable is identified by the couple (variable ID, service ID). Therefore, in the permanent variable area, different variables can share the same variable ID
Summary of change:	⌘	During ad-hoc meeting #65, the USAT Interpreter group decided to specify the second interpretation. Section 6.1.2.2: add a clarification and provide an example.
Consequences if not approved:	⌘	The USAT Interpreters could behave differently in this case.

Clauses affected:	⌘	6.1.2.2
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

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. A variable is identified by the couple {variable ID, service ID}. Therefore, in the permanent variable area, two different variables can share the same variable ID. 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.

If a page, which does not provide a Service ID TLV, attempts to access a variable, the USAT Interpreter shall generate a "security error".

If a page attempts to read a variable, which has never been initialised by the service the page belongs to, the USAT Interpreter shall generate a "reference to undefined" error.

Example:

Step 1: page 1, with service ID "1111", creates a permanent variable. Its variable ID is '41' and its content is "Toto".

Step 2: page 2, with service ID "222222", attempts to read the variable '41' content. The USAT Interpreter generates a "reference to undefined" error because the variable {'41', "222222"} does not exist yet.

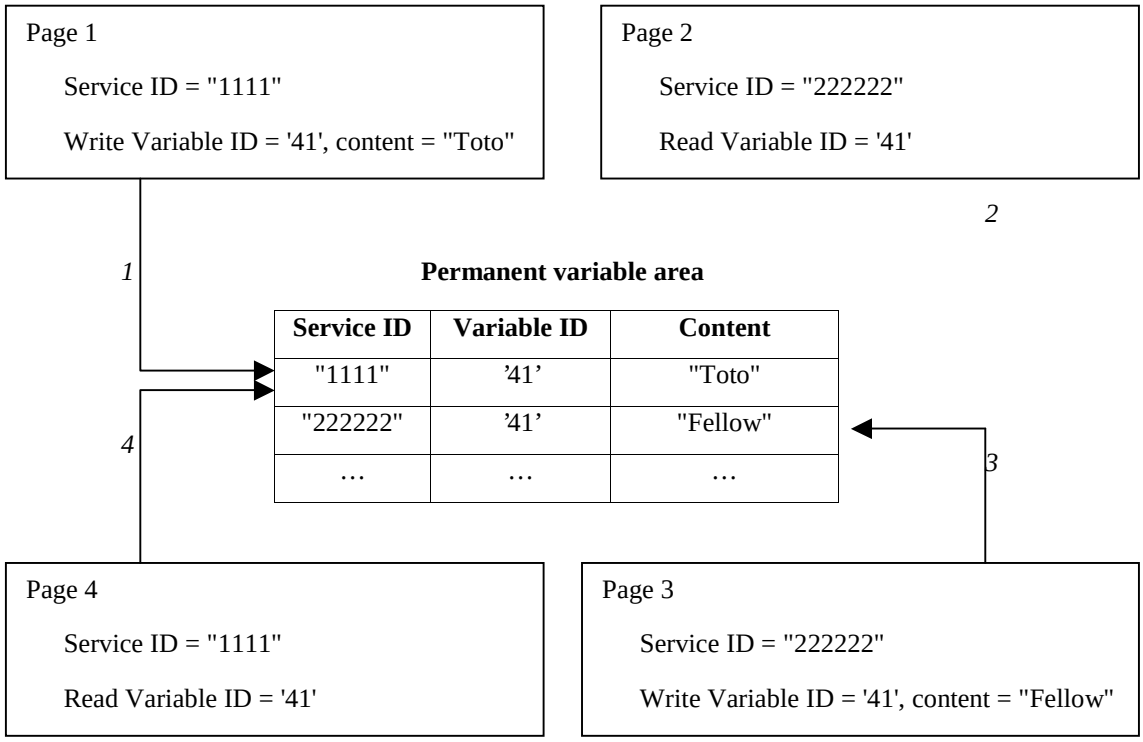
Step 3: page 3, with service ID "222222", creates a permanent variable. Its variable ID is '41' and its content is "Fellow".

Step 4: page 4, with service ID "1111", attempts to read the variable '41' content. The result is "Toto" and not "Fellow".

This example shows that page 2 does not overwrite the page 1 variables.

[Note from editor of the CR:

There has been an editorial problem with the change bar handling. The following picture is part of the additions of the CR and is to be incorporated into the specification.]



3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020346

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **013** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Clarification of the Terminal Response Handler Mechanism
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	23 May 2002
Category:	⌘	**F**
Release:	⌘	REL-5

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The terminal response handler mechanism might be implemented in different way as it is specified not detailed enough.
Summary of change:	⌘	3: some definitions added for clarification 4.1: corrected reference to system TRH configuration 4.2.3: clarification on wait state handling 4.3.x: more precise explanation of the Terminal Response Handler mechanism 7.1.8.x: more precise explanation of the TRH modifier mechanism 7.2.1: corrected reference to an exception case 7.2.3: behaviour is already defined in previous chapters Annex C: New informative annex with TRH modifier examples Annex D: Corrected chapter header for change history
Consequences if not approved:	⌘	USAT Interpreter implementations could behave differently.

Clauses affected:	⌘	3, 4.1, 4.2.3, 4.3, 4.3.1, 4.3.1.1, 4.3.1.2, 4.3.2, 7.8.1, 7.1.8.1, 7.1.8.2, 7.1.8.3, 7.1.8.4, 7.1.8.4.3, 7.1.8.4.4, 7.2.1, 7.2.3, Annex C, Annex D
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

3 Definitions, abbreviations and symbols

3.1 Definitions

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

anchor: named location on a page to which references can be made and at which rendering by the USAT Interpreter is initiated

NOTE: 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): sequence of consecutive bytes in the value part of a TLV containing the attributes of that TLV

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

current terminal response handler configuration: terminal response handler configuration currently valid

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

default terminal response handler configuration: the terminal response handler configuration as specified in clause 4.3.2

general result range: 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: block of a service description that can be referenced (by its anchor) and hence independently activated

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

service: 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: variable to be shared with the following page

NOTE: Shared variables can be provided to the next page in a protected or non protected manner.

string pool: list of predefined variables provided by the current page within the page TLV

NOTE: The string pool is mainly used for optimisation purposes.

system terminal response handler configuration: default terminal response handler configuration possibly modified by personalisation

terminal response handler configuration: configuration used by the terminal response handler mechanism to allow the mapping of actions to general results of USAT commands (see 4.3.1.1)

variable ID: identifier to reference a variable within a variable usage area

wait state: 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

[...]

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~~system terminal response handler configuration shall be used.

[...]

4.2.3 Wait State

When rendering a Page Reference TLV containing a Submit Configuration TLV having the "ProcessingBehaviour" attribute set, the USAT Interpreter shall perform the following actions:

- provide the Submit TLV to the protocol layer to be transmitted to the external system entity (see clause 4.2.2);
- If the transport layer successfully executed the given information
  - process next byte code.
- If the transport layer could not execute the given information successfully
  - enter the exception case of the terminal response handler mechanism.

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 clause 4.2.2).

If the transport layer successfully executed the given information

- enter the wait state.

If the transport layer could not execute the given information successfully

- enter the exception case of the terminal response handler mechanism.

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 figure 4.1.

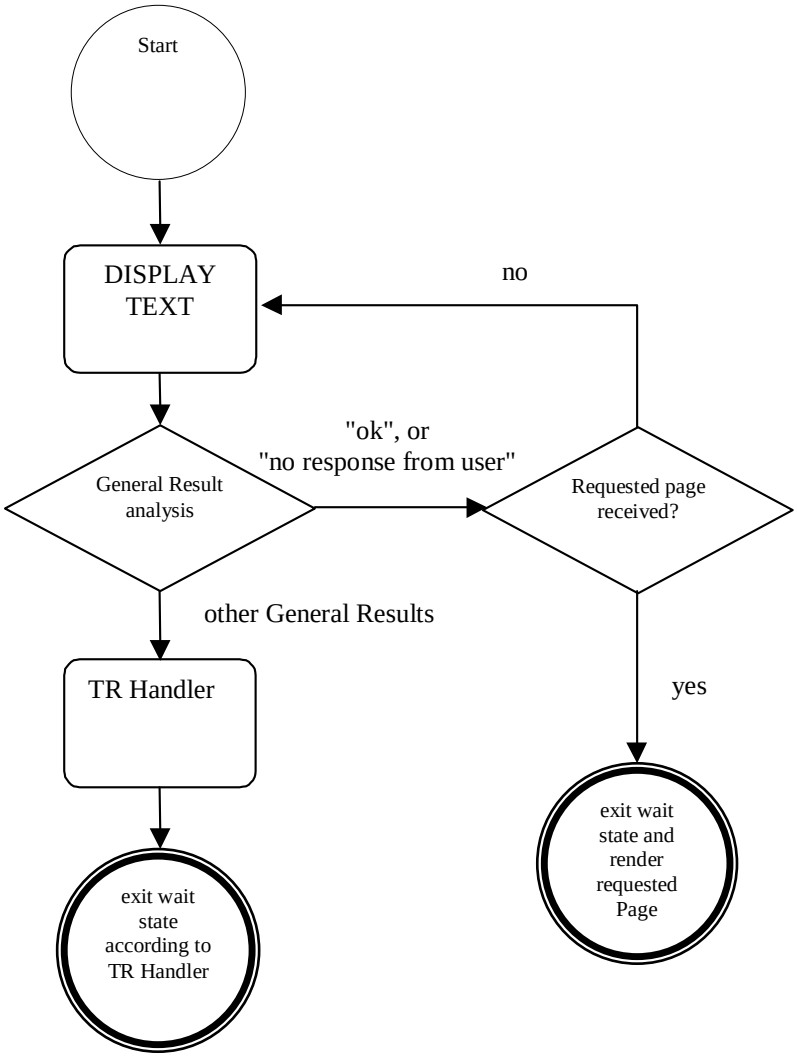


Figure 4.1: State diagram

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 (i.e. the configuration of the current navigation unit).

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 clause 11).

[...]

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 actions~~shall handle the general result according to the following rules.

~~.~~

4.3.1 Operation of the Terminal Response Handler

4.3.1.1 Definitions

For the description of the Terminal Response Handler Mechanism the following definitions and abbreviations apply:

Abbreviation	Item	Definition
AI	Action Identifier	a single value in the range of '00' to 'FF' identifying an action
GR	General Result	result of a USAT command; a single value in the range from '00' to 'FF'
GRR	General Result Range	multiple consecutive General Result (GR) values
a	Single Action	A single action identified by an external system or service defined Action Identifier(AI). $a_{'xx'}$ is a single action with the AI 'xx'.
A	Set of Actions	a collection of zero or more single actions (a).
A_{GR}	General Result Actions	A set of Actions (A) applying to a specific General Result (GR).
TRHC	Terminal Response Handler Configuration	A collection of A_{GR}, so that there is one Set of Actions for each General Result (GR).

4.3.1.2 Operation

The execution of any USAT command generates a general result (GR). The behaviour of the USAT Interpreter after the execution of a USAT command is determined by the generated general result and the current terminal response handler configuration as follows:

While the USAT Interpreter is in execution there is always one active terminal response handler configuration called the *current terminal response handle configuration*.

Let the generated general result be GR. The USAT Interpreter shall check the current terminal response handler configuration for the corresponding A_{GR} for that GR. By definition, for each GR an A_{GR} shall exist. As specified in 4.3.1.1 an A_{GR} might have no, one or more actions (a) applied to it.

If the A_{GR} contains only one action (a), then the single action (a) in A_{GR} shall be performed by the USAT Interpreter without user confirmation. If there are several actions in the A_{GR}, then the USAT Interpreter shall issue a SELECT ITEM command to let the user select one action (a) out of A_{GR} that shall be used by the USAT Interpreter. The handling of the SELECT ITEM command is described in clause 7.1.8.4.4.

If there is no action (a) in A_{GR} the exception action shall be performed by the USAT Interpreter.

In case of an exception the exception action will apply. This action can be changed by using the terminal response handler modifier with the reserved general result range 'FF FF'. In the default terminal response handler table (clause 4.3.2, table 4.1), this range is called "exception".

Exception examples:

- no more byte code when process next byte code (e.g. end of navigation unit);
- other exception cases of the USAT Interpreter not covered currently in the present document.

~~If several actions are assigned to the general result, a SELECT ITEM command shall be built by the USAT Interpreter from the action list using the action description to allow the user to choose between actions. If only one action is assigned to a general result range, this action shall be performed without user confirmation.~~

~~In case of an exception of the USAT Interpreter or no corresponding general result range to the general result of an USAT command, a specific action will apply. This action can be changed by using the terminal response handler modifier with the reserved general result range 'FF FF'. In the table below, this range is called "exception".~~

~~Exception example:~~

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

4.3.2 Default Terminal Response Handler configuration

A default terminal response handler configuration is defined in the present document (see table 4.1). The proposed default terminal response handler configuration ~~and may~~can be modified at personalization stage by the card issuer.

The possibly modified resulting terminal response handler configuration is called the system terminal response handler configuration, which shall be used by the USAT Interpreter. The system terminal response handler configuration can be the same as the default terminal response handler configuration or it can differ from it, depending on the decision of the card issuer.

NOTE: A service should take into account, that the system terminal response handler configuration might be different from the default terminal response handler configuration. The service might need to have knowledge of the system terminal response handler configuration in order to behave as intended.

The ~~default~~system 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,~~ see clause 7.1.8).

If the USAT Interpreter branches to another page due to the terminal response handler configuration, the standard inter page variable management shall apply (see clause 6.1.3.1).

Default terminal response handler configuration.

Table 4.1

		Action ID	General result range								
			'FF FF' exception	'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	'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 (note)	

NOTE: 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.

The USAT Interpreter may support storage of texts for user notification for the general result ranges of the system terminal response handler configuration. If texts for user notification are available, the texts shall be used according to clause 7.1.8.3.

For each of the system actions a text shall be assigned and shall to be used in the SELECT ITEM if more than one action is assigned to a general result (see clause 4.3.1.2). These texts shall be specified by the card issuer and shall be provided by personalisation.

[...]

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.

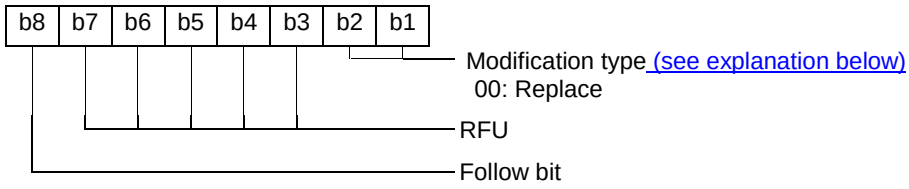
Coding of the terminal response handler modifier TLV:

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



Modification type

A terminal response handler modifier can be combined with a terminal response handler configuration to produce a new terminal response handler configuration using one of four operations:

- Replace operation
- Add/Append operation
- Restore operation
- Remove operation

Each of these operations given a current terminal response handler configuration and a terminal response handler modifier produces a new current terminal response handler configuration. For the following description, the following definitions apply:

Abbreviation	Item	Definition
AI	Action Identifier	a single value in the range of '00' to 'FF' identifying an action
GR	General Result	result of a USAT command; a single value in the range from '00' to 'FF'
GRR	General Result Range	multiple consecutive General Result (GR) values
A	Set of Actions	a collection of zero or more single actions; one Action TLV represents one single action

Replace operation

For the replace operation a GRR and A with at least one single action shall be provided. The GRR is the range of GR on which the operation applies. A is the set of actions which shall be linked with all GR within the given GRR.

This operation replaces all actions for all GR within the given GRR by the given action(s); all previously defined action(s) for all the GR within this GRR shall be erased by the USAT Interpreter.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

Add/Append operation

For the add/append operation a GRR and A with at least one single action shall be provided. The GRR is the range of GR on which the operation applies. A is the set of actions which shall be linked with all GR within the given GRR.

For every GR within the GRR, the given action(s) are appended to the existing ones for these GR. If action(s) with same action ID(s) exist already for a GR, the action(s) are replaced.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

Restore operation

For the restore operation a GRR shall be provided. The GRR is the range of GR on which the operation applies.

For every GR within the GRR, the action(s) shall be restored to the predefined action(s) of the system terminal response handler configuration.

For every GR within the GRR, the user notification text of the system terminal response handler configuration shall be restored. If the system terminal response handler configuration does not contain a text for a GR in the given GRR, the user notification text shall be removed for that GR.

If a text for user notification is provided within the terminal response handler modifier TLV, this user notification text TLV shall be ignored by the USAT Interpreter.

Remove Operation

For the Remove operation a GRR and AI within one or more Action TLV(s) shall be provided. The GRR is the range of GR on which the operation applies.

For every GR within the GRR, the action(s) indicated by the given AI are removed from the existing set of actions.

If the given action(s) to be removed do not exist in the existing set of actions(s) for a GR, the requested modification shall be ignored for that GR.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

Validity period of the terminal response handler modification:

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~~system terminal response handler configuration is valid (see clause 4.3).
- **Page context:** A terminal response handler modifier within the page context can modify the response handler configuration for the whole page. Just before entering another page, ~~After leaving the page~~ the modifications done by the terminal response handler modifier of the~~is~~current page are discarded and the terminal response handler configuration of the system context as defined in the paragraph above 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'.

The general result range specifies the general results for which the modification applies: for every general result within the general result range, corresponding operations shall be taken into account by the USAT Interpreter

~~General result ranges on the USAT Interpreter side shall be checked on a stack based model (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 matching 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 is part of ~~matches~~ the general result range the text for user notification is given for.

If a Terminal Response Handler modifier contains a text for user notification TLV, then the text is handled by the USAT Interpreter according to the operation descriptions in clause 7.1.8.1. The value part of this TLV may be empty (L of the Inline Value TLV is '00'. In this case, the text for user notification is to be removed for the respective general results.~~it replaces the text for user notification of the current terminal response handler configuration for all the general results within the Terminal Response Handler modifier general result range.~~

If this TLV is not available in the terminal response handler modifier TLV, the text for user notification remains unchanged for the respective general results.

After this DISPLAY TEXT command has been issued by the USAT Interpreter the actions defined for the general result are to be handled 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' ("wait for user to clear message" and "high priority").

7.1.8.4 Action

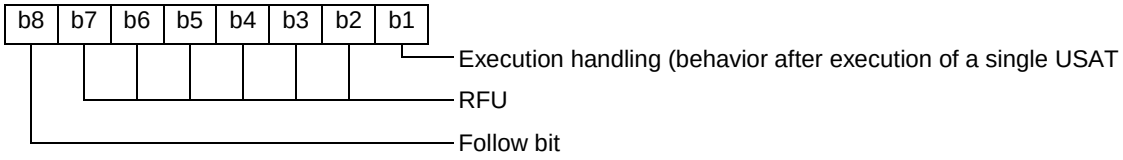
The action TLV defines the behaviour of the USAT Interpreter when the general result of the terminal response (TR) is part of the ~~the~~ associated general result range ~~matches the general result of the terminal response (TR)~~.

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

The following table gives an overview of conditions of presence for the Action to be performed TLV and the Inline Value TLV depending on the modification type indicated in the attributes of the terminal response handler modifier:

Modification Type	Action to be performed TLV	Inline Value TLV
Replace	shall be present	shall be present
Add / Append	shall be present	shall be present
Restore	not applicable, see clause 7.1.8	not applicable, see clause 7.1.8
Remove	need not to be present; to be ignored, if present	need not to be present; to be ignored, if present

7.1.8.4.1 Attributes



The following figure gives an overview of the return behaviour of the terminal response handler depending on the attribute value.

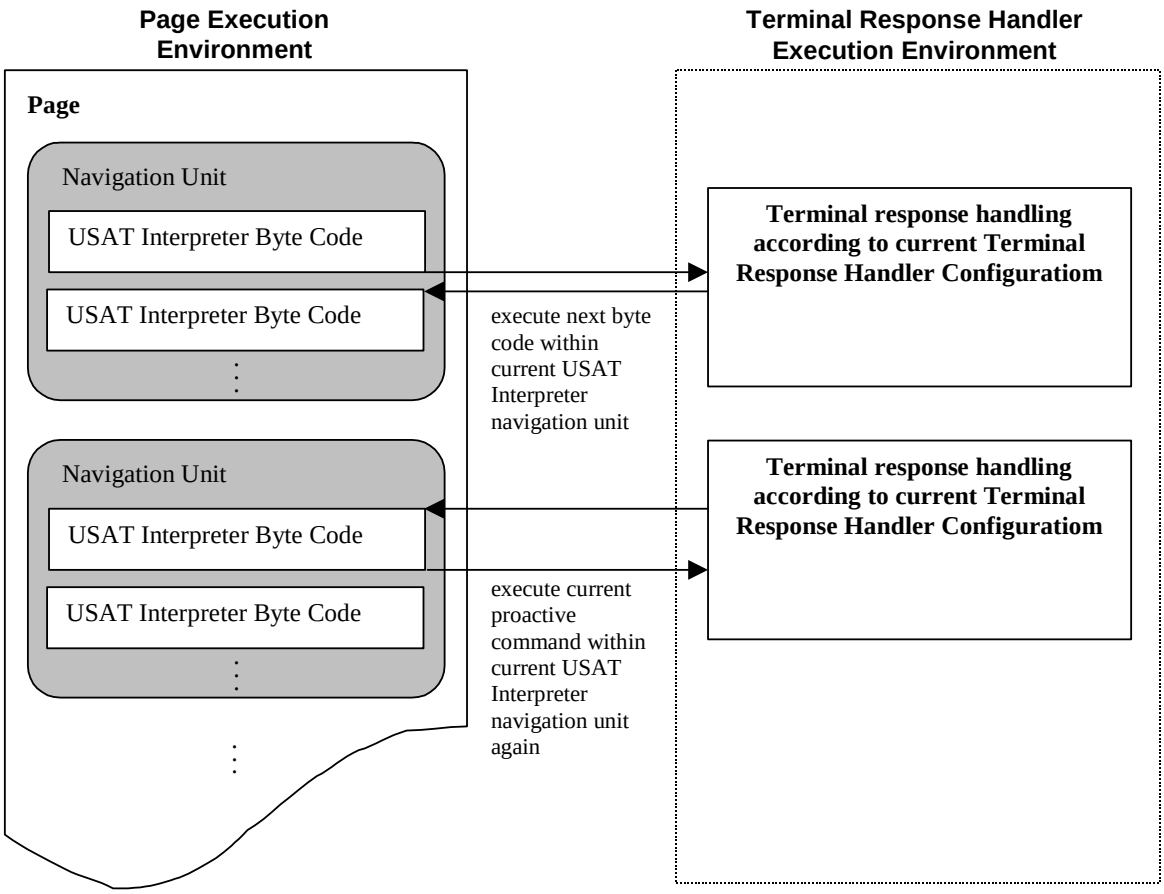


Figure 7.1

This attribute is to be considered only for certain types of actions to be performed (see table in clause 7.1.8.4.3).

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.

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

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 t~~The behavior of the USAT Interpreter after execution of the single USAT Interpreter byte code is given in the following table:~~.~~

General result for the USAT command	Comment
'00'...'0F' (ok)	behave as define in attribute of action TLV
'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' (Proactive SIM session terminated by the user)	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

Summary of action management in Terminal Response Handler mechanism:

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	defined by service ('20' to 'FF')	page reference TLV or anchor reference TLV	attribute byte shall be ignored
branch to another navigation unit			
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' (Proactive SIM session terminated by the user~~qui)t~~)	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' ("presentation type is specified in bit 2" and "presentation as a choice of navigation options").

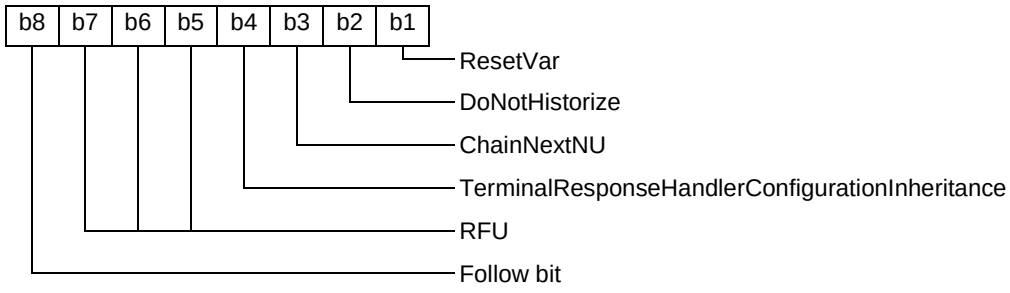
7.2 Navigation Unit

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 clauses 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 clause 7.1.8.

[…]

Annex C (informative): Terminal Response Handler Modifier examples

This annex provides examples for the operations of the terminal response header modifier. Starting point for the examples is the partly shown system terminal response handler configuration, which is in this case the unmodified default terminal response handler configuration as specified in table 4.1 with an assumed text for user notification for the exception case ("Error")..

The first row in the following tables shows the text for user notification assigned to a general result. A terminal response handler modifier can provide such a text for a whole range of general results. "--" indicates, that no user notification text is assigned to a general result.

The second row in the following tables shows the single action(s) assigned to a general result. For general results without an assigned action (indicated by "--" in the tables), the USAT Interpreter uses the exception case, which is indicated with the general result range 'FF FF'. If more than one action is assigned to a general result, the USAT Interpreter issues a SELECT ITEM command, using the action description texts of the actions as items to let the user choose between the options. A terminal response handler modifier can provide such a set of actions for a whole range of general results. $a_{'xx'}$ indicates an action a with the assigned Action ID 'xx'. For one general result, the Action ID uniquely identifies an action. For different general results, the same action ID in the service defined range (Action ID '20' to 'FF') could identify different actions. To distinguish between different actions with the same Action ID, the Action ID index is appended with a character. E.g. $a_{'20a'}$ represents a different action than $a_{'20b'}$, even if the Action ID '20' is the same.

The third row the following tables shows the general result values to which the user notification texts and actions are assigned to.

Starting configuration, partly reflection the default terminal response handler configuration as specified in table 4.1:

Table C.1

Text for user notification assigned to a general result	--	--	--	--	--	--	--	--	--	--	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	$a_{'00'}$	$a_{'00'}$	...	$a_{'00'}$	$a_{'01'}$	$a_{'02'}$	$a_{'01'}$	$a_{'03'}$	$a_{'01'}$	--	--	...	$a_{'01'}$ $a_{'03'}$	$a_{'01'}$ $a_{'03'}$	$a_{'01'}$ $a_{'03'}$	...	$a_{'01'}$
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

C.1 Replace Operation

The following terminal response handler modifier is applied as a replace operation to table C.1:

Table C.2

Text for user notification assigned to a general result range					"Proceed?"												
Single action(s) for a general result range; the index indicates the assigned Action ID					a'01' a'20a' a'21'												
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the general result range '10 11'. The new text for user notification for that result range is "Proceed?". The set of actions for that general result range is one system action ('Action ID '01': process next byte code) and two service defined actions with the Action IDs '20' and '21'. The result of a replace operation of table C.2 on table C.1 is shown in the following table:

Table C.3

Text for user notification assigned to a general result	--	--	--	--	"Proceed?"	"Proceed?"	--	--	--	--	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	a'00'	a'00'	...	a'00'	a'01' a'20a' a'21'	a'01' a'20a' a'21'	a'01'	a'03'	a'01'	--	--	...	a'01' a'03'	a'01' a'03'	a'01' a'03'	...	a'01'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

C.2 Add/Append Operation

The following terminal response handler modifier is applied as an add/append operation to table C.3:

Table C.4

Text for user notification assigned to a general result range						"Cont.?"											
Single action(s) for a general result range; the index indicates the assigned Action ID						a'20b' a'22' a'23' a'24'											
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the general result range '11 15'. The new text for user notification for that result range is "Cont.?". The set of actions for that general result range are four service defined actions with the Action IDs '20' and '22' to '24'. Note that for this example action '20b' represents another action than '20a' to show this specific case. The result of an add/append operation of table C.4 on table C.3 is shown in the following table:

Table C.5

Text for user notification assigned to a general result	--	--	--	--	"Proceed?"	"Cont.?"	"Cont.?"	"Cont.?"	"Cont.?"	"Cont.?"	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	$a_{'00'}$	$a_{'00'}$	...	$a_{'00'}$	$a_{'01'}$ $a_{'20a'}$ $a_{'21'}$	$a_{'01'}$ $a_{'20b'}$ $a_{'21'}$ $a_{'22'}$ $a_{'23'}$ $a_{'24'}$	$a_{'01'}$ $a_{'20b'}$ $a_{'22'}$ $a_{'23'}$ $a_{'24'}$	$a_{'03'}$ $a_{'20b'}$ $a_{'22'}$ $a_{'23'}$ $a_{'24'}$	$a_{'01'}$ $a_{'20b'}$ $a_{'22'}$ $a_{'23'}$ $a_{'24'}$	$a_{'20b'}$ $a_{'22'}$ $a_{'23'}$ $a_{'24'}$	--	...	$a_{'01'}$ $a_{'03'}$	$a_{'01'}$ $a_{'03'}$	$a_{'01'}$ $a_{'03'}$	...	$a_{'01'}$
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

Note, that in this specific case, for the general result '11' action $a_{'20a'}$ is replaced by $a_{'20b'}$, which are different. $a_{'20a'}$ for general result '10' remains unchanged and represents a different action than $a_{'20a'}$ for general result '10', even if the same Action ID is used.

C.3 Remove Operation

The following terminal response handler modifier is applied as a remove operation to table C.5:

Table C.6

Text for user notification assigned to a general result range					"GoOn?"												
Single action(s) for a general result range; the index indicates the assigned Action ID					$a_{'20'}$ $a_{'22'}$												
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the general result range '10 11'. The new text for user notification for that result range is "GoOn?". Actions with Action Ids '20' and '21' are to be removed. The result of a remove operation of table C.6 on table C.5 is shown in the following table:

Table C.7

Text for user notification assigned to a general result	--	--	--	--	"GoOn ?"	"GoOn ?"	"Cont. ?"	"Cont. ?"	"Cont. ?"	"Cont. ?"	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	a'00'	a'00'	...	a'00'	a'01' a'21'	a'01' a'21' a'23' a'24'	a'01' a'20b' a'22' a'23' a'24'	a'03' a'20b' a'22' a'23' a'24'	a'01' a'20b' a'22' a'23' a'24'	a'20b' a'22' a'23' a'24'	--	...	a'01' a'03'	a'01' a'03'	a'01' a'03'	...	a'01'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

C.3 Restore Operation

The following terminal response handler modifier is applied as a restore operation to table C.7:

Table C.8

Text for user notification assigned to a general result range													
Single action(s) for a general result range; the index indicates the assigned Action ID													
general result value	'00'	'01'	...	'0F'	'10'	'11' – ' 15'		'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the general result range '11 15'. No texts and no actions for the general result range are to be provided. All actions and user notification texts of the system terminal response handler are restored for the given general result range. The result of a restore operation of table C.8 on table C.7 is shown in the following table:

Table C.9

Text for user notification assigned to a general result	--	--	--	--	"GoOn ?"	--	--	--	--	--	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	a'00'	a'00'	...	a'00'	a'01' a'21'	a'02'	a'01'	a'03'	a'01'	--	--	...	a'01' a'03'	a'01' a'03'	a'01' a'03'	...	a'01'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

C.4 Special case: Empty text for user notification

For the operations add/append, replace and remove, the text for user notification may have an empty value part. In that case, the text for user notification is removed for the respective general results.

E.g. for an add/append operation:

Table C.10

Text for user notification assigned to a general result range					""												
Single action(s) for a general result range; the index indicates the assigned Action ID					a'20' a'22'												
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the general result range '10 11'. The text for user notification for that result range is to be removed. Actions with Action IDs '20' and '22' are to be added. The result of an add/append operation of table C.10 on table C.9 is shown in the following table:

Table C.11

Text for user notification assigned to a general result	--	--	--	--	--	--	--	--	--	--	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	a'00'	a'00'	...	a'00'	a'01' a'20' a'21' a'22'	a'02' a'20' a'22'	a'01'	a'03'	a'01'	--	--	...	a'01' a'03'	a'01' a'03'	a'01' a'03'	...	a'01'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

C.5 Special case: No text for user notification

For the operations add/append, replace and remove, the text for user notification is optional. If no text for user notification is given in the terminal response handler modifier, the text for user notification is remains unchanged for the respective general results.

E.g. for an add/append operation:

Table C.12

Text for user notification assigned to a general result range																	--
Single action(s) for a general result range; the index indicates the assigned Action ID																	a'34' a'35'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

This terminal response handler modifier is applied to the exception case 'FF FF'. The text for user notification for the exception case remains unchanged as no text for user notification TLV is provided. Actions with Action IDs '34' and '35' are to be added to the exception case. The result of an add/append operation of table C.12 on table C.11is shown in the following table:

Table C.13

Text for user notification assigned to a general result	--	--	--	--	--	--	--	--	--	--	--	...	--	--	--	...	"Error"
Single action(s) for a general result; the index indicates the assigned Action ID	a'00'	a'00'	...	a'00'	a'01' a'20' a'21' a'22'	a'02' a'20' a'22'	a'01'	a'03'	a'01'	--	--	...	a'01' a'03'	a'01' a'03'	a'01' a'03'	...	a'01' a'34' a'35'
general result value	'00'	'01'	...	'0F'	'10'	'11'	'12'	13'	'14'	'15'	'16'	...	'20'	'21'	'22'	...	'FF'

Annex ~~C~~D (informative): Change History

The table below indicates all CRs that have been incorporated into the present document since it was initially approved.

Change history								
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
2001-09	TP-13	TP-010208				Approved at TSG-T #13	2.0.0	5.0.0
2001-12	TP-14	TP-010245	001		F	Addition of SendAdditionalInformation attribute	5.0.0	5.1.0
		TP-010245	002		F	Collection of clarifications		
		TP-010245	003		C	Changes to USAT Interpreter system information partition table		
		TP-010245	004		B	comparison with a variable value		
2002-03	TP-15	TP-020066	005		B	Functional Additions to WML Annex	5.1.0	5.2.0
		TP-020066	006		F	Miscellaneous corrections and clarifications on the specification.		
		TP-020066	007		F	Clarification on behaviour on Single Actions for Terminal Response Handler		
		TP-020066	008		B	Addition of security plug-ins		

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020349

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **014** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Terminal Response Handler Modifier "remove" attribute enhancements
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	23 May 2002
Category:	⌘	**B**
Release:	⌘	REL-6

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The Terminal Response Handler Modifier allows to remove actions for some general results but all actions have to listed, which is not always possible
Summary of change:	⌘	Set list of action optional. If the list is present, the behaviour will be the one actually specified, but if the list is not present, then all the actions are to be removed.
Consequences if not approved:	⌘	

Clauses affected:	⌘	7.1.8, 7.1.8.1
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	This CR depends on the approval of the CR-013 to TS 31.113 in document T3-020346. It is valid only under the assumption that CR-013 will be approved.

[...]

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 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.

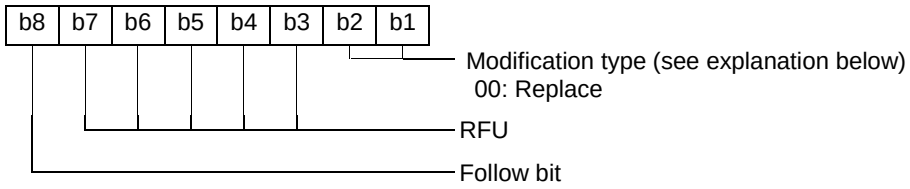
Coding of the terminal response handler modifier TLV:

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~~ optionally present

7.1.8.1 Attribute



Modification type

A terminal response handler modifier can be combined with a terminal response handler configuration to produce a new terminal response handler configuration using one of four operations:

- Replace operation
- Add/Append operation
- Restore operation
- Remove operation

Each of these operations given a current terminal response handler configuration and a terminal response handler modifier produces a new current terminal response handler configuration. For the following description, the following definitions apply:

Abbreviation	Item	Definition
AI	Action Identifier	a single value in the range of '00' to 'FF' identifying an action
GR	General Result	result of a USAT command; a single value in the range from '00' to 'FF'
GRR	General Result Range	multiple consecutive General Result (GR) values
A	Set of Actions	a collection of zero or more single actions; one Action TLV represents one single action

Replace operation

For the replace operation a GRR and A with at least one single action shall be provided. The GRR is the range of GR on which the operation applies. A is the set of actions which shall be linked with all GR within the given GRR.

This operation replaces all actions for all GR within the given GRR by the given action(s); all previously defined action(s) for all the GR within this GRR shall be erased by the USAT Interpreter.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

Add/Append operation

For the add/append operation a GRR and A with at least one single action shall be provided. The GRR is the range of GR on which the operation applies. A is the set of actions which shall be linked with all GR within the given GRR.

For every GR within the GRR, the given action(s) are appended to the existing ones for these GR. If action(s) with same action ID(s) exist already for a GR, the action(s) are replaced.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

Restore operation

For the restore operation a GRR shall be provided. The GRR is the range of GR on which the operation applies.

For every GR within the GRR, the action(s) shall be restored to the predefined action(s) of the system terminal response handler configuration.

For every GR within the GRR, the user notification text of the system terminal response handler configuration shall be restored. If the system terminal response handler configuration does not contain a text for a GR in the given GRR, the user notification text shall be removed for that GR.

If a text for user notification is provided within the terminal response handler modifier TLV, this user notification text TLV shall be ignored by the USAT Interpreter.

Remove Operation

For the Remove operation a GRR shall be provided. If one or more Action TLV(s) are provided, ~~and~~for each Action TLV an~~-~~ AI ~~within one or more Action TLV(s)~~ shall be provided. The GRR is the range of GR on which the operation applies.

If no Action TLV is provided, for all the GR within the given GRR, the USAT Interpreter shall remove all existing actions from the existing set of actions.

If at least one Action TLV is provided, ~~F~~for every GR within the GRR, the action(s) indicated by the given AI are removed from the existing set of actions.

If the given action(s) to be removed do not exist in the existing set of actions(s) for a GR, the requested modification shall be ignored for that GR.

If a text for user notification is provided within the terminal response handler modifier TLV, this operation replaces the existing text for all the GR within the given GRR by the given text.

3GPP T3 (USIM) Meeting #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020350

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **015** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Addition of error handling
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	23 May 2002
Category:	⌘	**B**
Release:	⌘	REL-6

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The error handling of the USAT Interpreter should be enhanced.
Summary of change:	⌘	A general chapter on error handling is incorporated. Additional error cases are introduced.
Consequences if not approved:	⌘	

Clauses affected:	⌘	5, 6.3, 7.5, 7.9, 8.1, 8.2, 8.2.3, 8.5, 8.7, 8.8, 8.11, 8.12, 12		
Other specs affected:	⌘	[ ] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	
Other comments:	⌘			

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~~ the execution result of the byte code shall be "Syntax error", as stated in chapter 12. Then the USAT Interpreter shall behave as described in chapter 12. TLVs not supported by the USAT Interpreter shall be ignored by the USAT Interpreter.

[...]

6.3 Variable substitution

[...]

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 a~~n~~ "Type mismatch" error unless the indicated type was set to '0' ("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 a~~n~~ "Type mismatch" 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.

[...]

7.5 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 clause 6.3 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 a~~n~~ "Type mismatch" error according to the following table:

[...]

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 "Jump to undefined" error is generated.

If the Submit Configuration 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 clause 7.10 .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

[...]

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 or Variable Identifier List 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
Type mismatch	Error in variables management	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
Type mismatch	Error in variables management	Stop

[...]

8.2.3 Ordered TLV List

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

Each of these TLVs encapsulate 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 (described by the Inline Value 2 TLV) 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 (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall assign the value of the Inline Value TLV to the destination variable and process next byte code.
present	not present	present	**"Go Selected"** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item (described by the Inline Value 2 TLV) 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	**Display and Process next byte code** Generation of a SELECT ITEM command by the USAT Interpreter. When the user has selected this item (described by the Inline Value 2 TLV) from the list, the USAT Interpreter shall process the next byte code. A destination variable identifier shall be ignored for this case.
not present (see NOTE)	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 (see NOTE)	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 (see NOTE)	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 a~~n~~ "Syntax error" shall be issued.
NOTE : If Ordered TLVs containing an Inline Value 2 are present in the same Assign and Branch TLV, the USAT Interpreter shall ignore the Ordered TLVs which do NOT contain the Inline Value 2 TLV. If only Ordered TLVs not containing an Inline Value 2 are present in the same Assign and Branch TLV, the USAT Interpreter shall take into account the first Ordered TLV only.			

[...]

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) or Inline Value TLV - 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
Type mismatch	Error in variables management	Stop

[...]

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
Type mismatch	Error in variables management	Stop

[...]

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 defined in clause 9. 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
Type mismatch	Error in variables management	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].

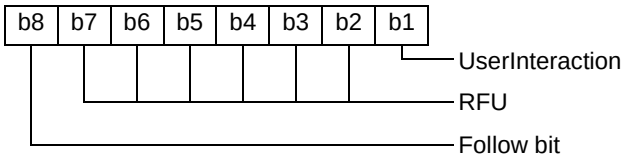
This command is used to display text of informational nature without require any input from user. The USAT Interpreter shall use the DCS value according to the indication given in the attributes of the Inline Value TLV. If no attributes are given in the Inline Value TLV, the coding scheme indication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

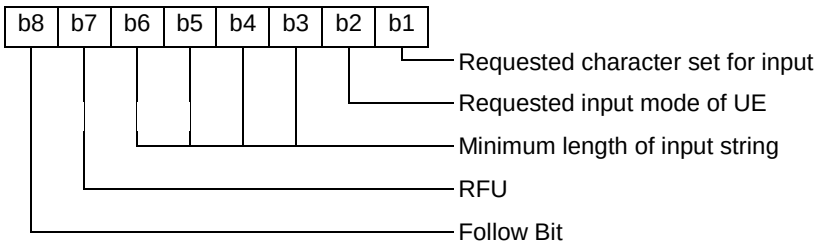
Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined variable	Stop
Type mismatch	Error in variables management	Stop

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 (e.g. user input shall not be revealed in any way), 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
Type mismatch	Error in variables management	Stop

[...]

12 Error handling and coding

This chapter describes how the USAT Interpreter shall behave when an error occurs. A table indicating the values for the different error codes is provided.

12.1 Setting of the environment variable "error code"

After having executed a byte code, the USAT Interpreter shall set the value of the environment variable "Error code generated by the last byte code command executed" ('05') according to the execution result. The possible execution results for a given byte code command are listed in the definition of this byte code command. The values for all possible error codes are listed in the table in clause 12.3.

12.2 User notification of the execution

In each byte code command description, for each possible error code, an action is indicated. This action can be either "continue" or "stop".If the action indicated is "continue", the USAT Interpreter shall process the next byte code without notifying the user.

If the action is "stop", the USAT Interpreter shall notify the user by displaying an error message to the user. For the DISPLAY TEXT USAT command used, the command qualifier options:

- "wait for use to clear message"

shall be used.

The error messages displayed by the USAT Interpreter

- shall be able to be modified by the operator at the personalisation stage;
- shall be able to be different for each error code.

After having displayed this message, for any general result of the terminal response, the USAT Interpreter shall quit.

12.3 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'

3GPP T3 #23
Espoo, Finland, 21 - 24 May 2002

Tdoc T3-020351

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **016** ⌘ rev ⌘ Current version: **5.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [X] ME/UE [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Addition of functionality for security plug-ins
Source:	⌘	T3
Work item code:	⌘	USAT Interpreter
Date:	⌘	23 May 2002
Category:	⌘	**B**
Release:	⌘	REL-6

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	The definition of functionality for security plug-ins is missing
Summary of change:	⌘	Technical functionality for PKI, Triple-DES and PIN-handling plug-ins is added
Consequences if not approved:	⌘	

Clauses affected:	⌘	2, 3.1, 3.3, 9, 9.1, Annex D, E and F		
Other specs affected:	⌘	[ ] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	
Other comments:	⌘			

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. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 31.111: "USIM Application Toolkit (USAT)".

[2] 3GPP TS 31.114: "USAT Interpreter protocol and administration".

[3] 3GPP TS 23.038: "Alphabets and language-specific information".

[4] ETSI 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)"

[10] RSA Laboratories: "PKCS #1 v2.0: RSA Cryptography Standard", www.rsasecurity.com/rsalabs/pkcs/

[11] ISO/IEC 9797-1:1999(E): "Information technology – Security techniques – Message Authentication Codes (MACs)"

[12] RSA Laboratories: "PKCS#9 v2.0: Selected Object Classes and Attribute Types", http://www.rsasecurity.com/rsalabs/pkcs/

[13] FIPS PUB 180-1: "Secure Hash Standard (SHS)"

[14] Wireless Application Forum: "Wireless Application Protocol – WMLScript Crypto Library Specification", Version 20-Jun-2001.

[15] Wireless Application Forum: "Wireless Application Protocol – Wireless Transport Layer Security Specification", Version 18-Feb-2000.

[16] IANA assigned character sets, http://www.iana.org/assignments/character-sets

[17] RSA Laboratories: "PKCS #5 v2.0: Password-Based Cryptography Standard", http://www.rsasecurity.com/rsalabs/pkcs/

3 Definitions, abbreviations and symbols

3.1 Definitions

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

anchor: named location on a page to which references can be made and at which rendering by the USAT Interpreter is initiated

NOTE: 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): sequence of consecutive bytes in the value part of a TLV containing the attributes of that TLV

current page: 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: 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: block of a service description that can be referenced (by its anchor) and hence independently activated

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

service: 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: variable to be shared with the following page

NOTE: Shared variables can be provided to the next page in a protected or non protected manner.

string pool: list of predefined variables provided by the current page within the page TLV

NOTE: The string pool is mainly used for optimisation purposes.

variable ID: identifier to reference a variable within a variable usage area

wait state: 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:

3DES	Triple DES
AKI	Asymmetric Key Index
AD	Asymmetric Decryption Plug-in
ASN.1	Abstract Syntax Notation One (1)
C	Conditional
CA	Certificate Authority
CBC	Cipher Block Chaining (Mode)
CHV	Card Holder Verification

CP	Change PIN Plug-in
DCS	Data Coding Scheme
DD	Triple DES Decrypt Plug-in
DE	Triple DES Encrypt Plug-in
DER	Distinguished Encoding Rules of ASN.1
DES	Data Encryption Standard
DS	Triple DES Sign Plug-in
DU	Triple DES Unwrap Plug-in
ECB	Electronic Code-book (mode)
EDE	Encrypt-Decrypt-Encrypt
FP	Fingerprint Plug-in
IANA	Internet Assigned Numbers Authority
ICCID	Integrated Circuit Card IDentification
ID	IDentifier
IV	Initialisation Vector
KIc	Key and algorithm Identifier for ciphering
LSB	Least Significant Bit
M	Mandatory
MAC	Message Authentication Code
MDC	Modification Detection Code
MSB	Most Significant Bit
NCI	Native Code Identifier
NU	Navigation Unit
O	Optional
OID	Object Identifier
OTA	Over-the-Air
OTP	One Time Password
P7	PKCS#7 Signature Plug-in
PIN	Personal Identification Number
PKCS	Public-Key Cryptography Standards
PS	Plug-in Status Code
PUK	PIN Unblocking Key
RFU	Reserved for Future Use
RP	Reset PIN Plug-in
RSA	Algorithm invented by Rivest, Adleman and Shamir
SHA-1	Secure Hash Algorithm 1
SMS	Short Message Service
SW1/SW2	Status Word 1 / Status Word 2
TLV	Tag Length Value
TTBS	Text To Be Signed
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
WAP	Wireless Application Protocol
WTLS	Wireless Transport Layer Security
WIM	Wireless Identity Module
XML	eXtensible Markup Language

3.3 Symbols

For the purposes of the present document, the following symbol applies:

'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'.

Symbols used in annexes:

<i..j>	Sub-string extraction operator. Extracts bytes i through j. $1 \le i \le j$.
X || Y	Concatenation of byte-strings X and Y (in that order).
|| . ||	Byte length operator.
bn	Individual bit in a byte. Range from bit 1 (least significant), denoted *b1*, to bit 8 (most significant), denoted *b8*.
Bn	Individual byte in a byte-string. Range from byte 1 (leftmost), denoted *B1*, to byte *n* (rightmost), denoted *Bn*.
c	Ciphertext representative. An integer between *0* and *n-1*.
C	Ciphertext. Input parameter to the AD plug-in.
DP	Decrypted PIN data.
DTBS	Data-to-be-signed. Input parameter to the FP plug-in.
EM	Encrypted message.
DM	Decrypted message.
EMSA-PKCS1-v1_5-ENCODE	PKCS#1 encoding function. See [10] for further reference
EP	Encrypted PIN data.
I2OSP	Integer-to-Octet-String conversion primitive. See [10] for further reference.
ICCID	Raw ICCID. 10 bytes length.
ISO_IEC_9797_ALG3	ISO/IEC 9797 MAC algorithm 3. See [11] for further reference.
ISO_IEC_9797_PAD2	ISO/IEC 9797 padding method 2. See [11] for further reference.
k	Length in bytes of the modulus.
K	RSA private key.
K_1, K_2, K , K'	DES keys.
KC	An 8 byte key checksum.
KH	SHA-1 hash of the public key. The hash shall be computed from the unsigned modulus to be in line with WAP WTLS and WAP WIM.
m	Message representative. An integer between *0* and *n-1*.
M	Message, a byte string.
MAC	A ISO/IEC 9797 message authentication code
MD	A SHA-1 hash value.
N	Modulus. An integer.
OS2IP	Octet-String-to-Integer conversion primitive. See [10] for further reference.
PC	An 8 byte PIN checksum.
PKCS5_PAD	PKCS#5 padding function. See [17] for further reference.

PKCS5_UNPAD	Inverse of *PKCS5_PAD*. See [17] for further reference.
PM	A padded message.
R	Random nonce. 8 bytes length.
RSADP	RSA decryption primitive. See [10] for further reference.
RSASP1	RSA signature primitive. See [10] for further reference.
RSASSA-PKCS1-v1_5-SIGN	PKCS#1 signature generation function. See [10] for further reference.
S	Raw signature of byte length *k*.
SHA1	SHA-1 hash function. See [13] for further reference.
TDEA_DECR	Triple DES decryption algorithm. See [8] for details regarding the algorithm.
TDEA_ENCR	Triple DES encryption algorithm. See [8] for details regarding the algorithm.
TTBS	Text-to-be-signed. Byte string. Input parameter to P7 plug-in.

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 NCI values '0000' to '7FFF' are specified in this clause. Other values may be used for proprietary implementations.

Native Commands defined below are optionally to be supported by the USAT Interpreter. If any of these 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:

NCI	Name	Chapter
'00 00'	RFU	
'00 01'	P7 – PKCS#7 Signature Plug-In	9.1.2.1
'00 02'	FP – Fingerprint Plug-In	9.1.2.2
'00 03'	AD – Asymmetric Decryption Plug-In	9.1.2.3
'00 04'	DE – Triple DES Encryption Plug-In	9.12.3.1
'00 05'	DD – Triple DES Decryption Plug-In	9.12.3.2
'00 06'	DS – Triple DES Sign Plug-In	9.12.3.3
'00 07'	DU – Triple DES Unwrap Plug-In	9.12.3.4
'00 08'	CP – Change PIN Plug-In	9.13.4.1
'00 09'	RP – Reset PIN Plug-In	9.13.4.2
'00 0A'-'7F FF'	RFU	

9.1 Security Plug-ins

9.1.1 Common Topics

9.1.1.1 Security Policy

Security policy related issues like

- principles of key management and key life cycle management
- practices and procedures to be followed when carrying out technical and administrative aspects of key management
- responsibilities and accountability of each party involved
- the types of records (i.e. audit trail information) to be kept

are all outside the scope of the present document.

9.1.1.2 Classification of PINs

The majority of plug-ins specified in subclause 9.1 normally (configuration dependent) include a PIN, and possibly also a PUK, verification step. This step is necessary to identify the user and obtain explicit authorisation before certain sensitive operations can be performed. The PIN(s) required by the security plug-ins bear no relation to the UICC PINs [4] (e.g. the USIM application PINs), and shall be completely controlled by the USAT Interpreter.

Theoretically, there can be as many PINs as there are keys, even if this seems unwise from a practical point of view.

9.1.1.3 Key Diversification

Key diversification is a technical term that signifies the possibility to associate a key with conditions stating for what purpose(s) the key may be used. Normally key diversification is used to improve the security of a system by eliminating certain security threats and reducing system complexity.

This specification mandates that:

- key diversification shall be implemented for all keys accessible to the security plug-ins
- key usage enforcement shall be implemented in every security plug-in that requires a key for it's operation

9.1.1.4 Output Parameters

The security plug-ins defined in subclause 9.1 conform to a model whereby a plug-in always generate one, or at most two, output variables. The first variable, called the Plug-in Status Code, indicates the status of the plug-in upon termination.

The second variable, called the Functional Output, is used to hold the result from the primary function of the plug-in, whenever this is applicable (not all plug-ins have a defined output).

Obviously this only applies when the Error Code returned by the Execute Native Command byte code is "No error"', otherwise the USAT Interpreter would unconditionally stop.

9.1.2 ~~9.1~~ PKI Plug-ins

9.1.2.1 P7 - PKCS#7 Signature Plug-In

9.1.2.1.1 Description

The P7 plug-in is used to provide a digital signature based on a private (RSA) key stored on the USIM card. The output of the plug-in is compliant with the WMLScript Crypto Library SignText function. As such, P7 will also be compliant with other important specifications like PKCS#1 and PKCS#7.

~~Plug-in follows WYSIWYS (what-you-see-is-what-you-sign) principle.~~

9.1.2.1.2 NCI

The NCI for this plug-in is '00 01'.

9.1.2.1.3 Arguments

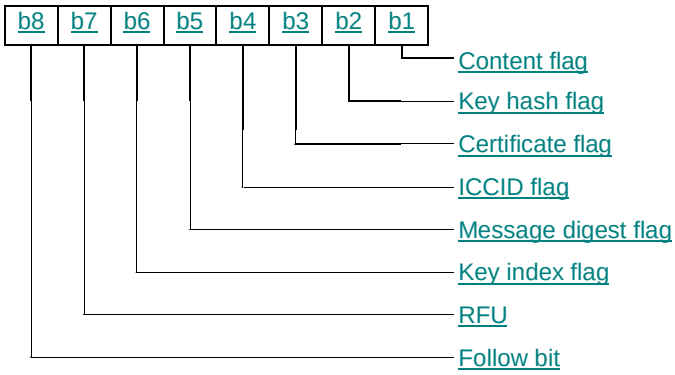
~~To be defined in REL-6.~~

The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	'00'/'01'/ '02'/'03'	Key identifier type. Indicates the type of the key identifier supplied in the next parameter: • '00' = No key identifier supplied. The plug-in shall choose a default key, if such a key exists, or abort with Plug-in Status Code "PS: No such key ". • '01' = User key hash. SHA-1 hash of the user public key is supplied in the next parameter. The plug-in shall use the private key that corresponds to the public key hash or, if this key is not available, abort with Plug-in Status Code "PS: No such key". • '02' = List of trusted key hashes. One or more SHA-1 hash values of trusted CA public key(s) are supplied in the next parameter. The plug-in shall use a signature key that is certified by the one of the indicated CAs or, if such a key is not available, abort with Plug-in Status Code "PS: No such key". • '03' = Index of RSA key.	M
1	Data	Index of RSA key (AKI).	C
20	Data	User key hash.	C
A	Data	List of trusted key hashes. The format of the field shall be LV, where the length is BER encoded onto 1, 2 or 3 bytes according ISO/IEC 7816-6 [5], and the value is the concatenation of all hash values.	C
1	'04'/'08'	Character encoding scheme • '04' = GSM default (unpacked). See 3GPP TS 23.038 ([3]) for further reference • '08' = UCS2	M
B	Data	Options.	M
C	Data	Text to be signed (TTBS). Represented in the indicated character encoding scheme.	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error"' and plug-in termination.

Coding of the "Options" field:



9.1.2.1.4 Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Contents
1	Plug-in Status Code (see subclause 9.1.2.1.6).
2	Functional Output. A SignedContent data structure as described in subclause D.1.2.3 or a textual error message.

9.1.2.1.5 Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause D.1.1.

9.1.2.3.6 Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.2.2 ~~9.1.2~~ FP – Fingerprint Plug-In

9.1.2.2.1 Description

The FP plug-in is used to provide a digital signature based on a private (RSA) key stored on the USIM card. The plug-in output contains a PKCS#1 compliant digital signature and is as such in line with important specifications like PKCS#1 and PKCS#7.

~~Plug-in is used to sign large amount of data (larger than few hundred bytes).~~

The plug-in follows a principle whereby an (encoded) excerpt of the data is displayed to the user before it is signed. The data itself would in a sensible application be represented as a DER encoded value.

9.1.2.2.2 NCI

The NCI for this plug-in is '00 02'.

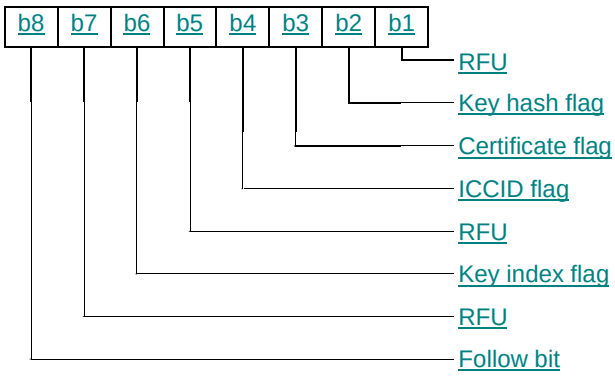
9.1.2.2.3 Arguments

~~To be defined in REL-6.~~The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	'00'/'01'/'03'	Key identifier type. Indicates the type of the key identifier supplied in the next parameter: • '00' = No key identifier supplied. The plug-in shall choose a default key, if such a key exists, or abort with Plug-in Status Code "PS: No such key". • '01' = User key hash. SHA-1 hash of the user public key is supplied in the next parameter. The plug-in shall use the private key that corresponds to the public key hash or, if this key is not available, abort with Plug-in Status Code "PS: No such key". • '03' = Index of RSA key.	M
1	Data	Index of RSA key (AKI).	C
20	Data	User key hash.	C
A	Data	Options.	M
B	Data	Data-to-be-signed. To be truly PKCS#1 compliant, this should be a DER encoded value of the *DigestInfo* ASN.1 type, as specified in PKCS#1. B shall be equal to or greater than 16.	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

Coding of the "Options" field:



9.1.2.2.4 Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Contents
1	Plug-in Status Code (see subclause 9.1.2.2.6).
2	Functional Output. A WrappedContent data structure as described in subclause D.2.2.2 or a textual error message.

9.1.2.2.5 Execution

~~Tobe defined in REL-6.~~

The detailed execution of the plug-in is described in subclause D.2.1.

9.1.2.2.6 Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.2.3 AD – Asymmetric Decryption Plug-In

9.1.2.3.1 Description

This plug-in is used for application-level asymmetric (RSA) decryption.

It is crucial that the application utilizing this plug-in protects the output from the plug-in in some way, e.g. by using (cryptographic) blinding.

9.1.2.3.2 NCI

The NCI for this plug-in is '00 03'.

9.1.2.3.3 Arguments

~~To be defined in REL-6.~~

The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	'00'/'01'/'03'	Key identifier type. Indicates the type of the key identifier supplied in the next parameter: • '00' = No key identifier supplied. The plug-in shall choose a default key, if such a key exists, or abort with Plug-in Status Code "PS: No such key". • '01' = User key hash. SHA-1 hash of the user public key is supplied in the next parameter. The plug-in shall use the private key that corresponds to the public key hash or, if this key is not available, abort with Plug-in Status Code "PS: No such key". • '03' = Index of RSA key.	M
1	Data	Index of RSA key (AKI).	C
20	Data	User key hash.	C
A	Data	Ciphertext. A byte string of the same (byte) length as the modulus of the decryption key. A shall be equal to or greater than 16.	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

9.1.2.3.4 Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.2.3.6).
2	Functional Output. The plaintext as described in subclause D.3.2 or a textual error message.

9.1.2.3.5 Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause D.3.1.

9.1.2.3.6 Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.3~~9.2~~ Triple DES Plug-ins

9.1.3.1~~9.2.1~~ DE – Triple DES Encryption Plug-In

9.1.3.1.1~~9.2.1.1~~ Description

The DE plug-in is used to encrypt arbitrary application-level data. It is typically called from a page to encrypt data before it is transmitted to a network application.

9.1.3.1.2~~9.2.1.2~~ NCI

The NCI for this plug-in is '00 04'.

9.1.3.1.3~~9.2.1.3~~ Arguments

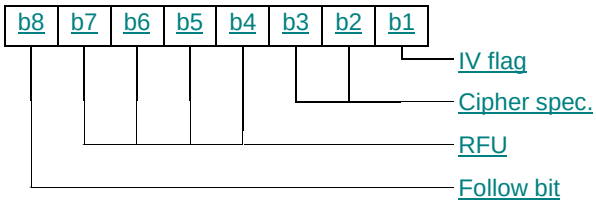
~~To be defined in REL-6.~~

The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	Data	Index of key.	M
A	Data	Options.	M
8	Data	IV (according to b1 of Options).	C
B	Data	Data to encrypt (plaintext).	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

Coding of the "Options" field:



ECB mode combined with IV shall be regarded as a "Syntax Error".

9.1.3.1.4~~9.2.1.4~~ Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.3.1.6).
2	Functional Output. The encrypted plaintext (i.e. ciphertext). 1 to 8 bytes longer than the length of the plaintext.

9.1.3.1.59.2.1.5 Execution

~~To be defined in REL-6~~

The detailed execution of the plug-in is described in subclause F.1.1.

9.1.3.1.6~~9.2.1.6~~ Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.3.2~~9.2.2~~ DD – Triple DES Decryption Plug-In

9.1.3.2.1~~9.2.2.1~~ Description

The DD plug-in is used to decrypt arbitrary application-level data. It is typically called from a page to decrypt data that has been encrypted by a network application.

9.1.3.2.2~~9.2.2.2~~ NCI

The NCI for this plug-in is '00 05'.

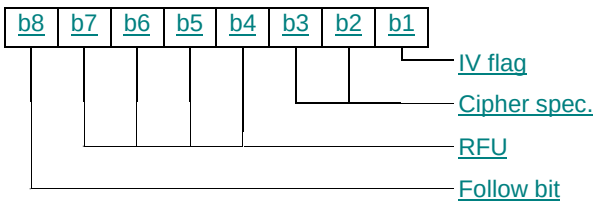
9.1.3.2.3~~9.2.2.3~~ Arguments

~~To be defined in REL-6.~~The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	Data	Index of key.	M
A	Data	Options.	M
8	Data	IV (according to b1 of Options).	C
B	Data	Data to decrypt (ciphertext).	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error"' and plug-in termination.

Coding of the "Options" field:



ECB mode combined with IV shall be regarded as a "Syntax Error".

9.1.3.2.49.2.2.4 Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.3.2.6).
2	Functional Output. The decrypted ciphertext (i.e. plaintext). 1 to 8 bytes shorter than the length of the ciphertext.

9.1.3.2.59.2.2.5 Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause F.2.1.

9.1.3.2.69.2.2.6 Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.3.39.2.3 DS – Triple DES Sign Plug-In

9.1.3.3.19.2.3.1 Description

The DS plug-in is used to calculate a message authentication code (MAC) for arbitrary application-level data. The MAC can be used as a data integrity mechanism to verify that data has not been altered in an unauthorised manner. It can also be used as a message authentication mechanism to provide assurance that a message has been originated by an entity in possession of the secret key.

The MAC is calculated according to ISO/IEC 9797 (algorithm 3, padding method 2) [11].

9.1.3.3.29.2.3.2 NCI

The NCI for this plug-in is '00 06'.

9.1.3.3.39.2.3.3 Arguments

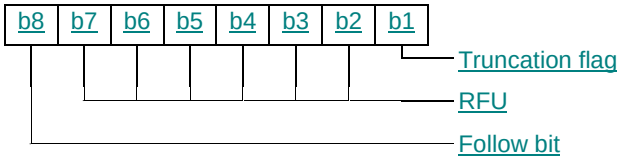
~~To be defined in REL-6.~~

The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	Data	Index of key	M
A	Data	Options	M
1	'04'/'08'	Character encoding scheme • '04' = GSM default (unpacked), see 3GPP TS 23.038 ([3]) • '08' = UCS2	M
B	Data	Text to be signed (TTBS). Represented in the indicated character encoding scheme.	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

Coding of the "Options" field:



9.1.3.3.49.2.3.4 Output Parameters

~~To be defined in REL-6.~~

The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.3.3.6).
2	Functional Output. The signature (MAC) on the text to be signed. The length of the signature is 4 or 8 bytes as indicated by the "Truncation flag".

9.1.3.3.59.2.3.5 Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause F.3.1.

9.1.3.3.69.2.3.6 Errors

~~To be defined in REL-6.~~Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.3.49.2.4 DU – Triple DES Unwrap Plug-In

9.1.3.4.19.2.4.1 Description

The DU plug-in is a key-management plug-in that enables a party in possession of a certain secret key, called a *key encryption key*, to replace an USAT Interpreter relatedapplication key stored in the USIM at its own desire.

9.1.3.4.29.2.4.2 NCI

The NCI for this plug-in is '00 07'.

9.1.3.4.39.2.4.3 Arguments

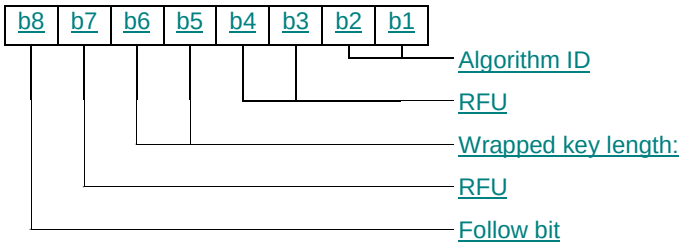
~~To be defined in REL-6.~~

The arguments (i.e. the value part of the inline value TLV within the input list TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	Data	Index of the key to be updated.	M
A	Data	Options.	M
B	Data	Encrypted key data.	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

Coding of the "Options" field:



9.1.3.4.4~~9.2.4.4~~ Output Parameters

~~To be defined in REL-6.~~The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.3.4.6).

9.1.3.4.5~~9.2.4.5~~ Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause F.4.1.

9.1.3.4.6~~9.2.4.6~~ Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: No such key"	'22'	The requested key is not available.

9.1.4~~9.3~~ PIN Management Plug-ins

These plug-ins shall be used to manage USAT Interpreter ~~application~~ related PINs.

9.1.4.1~~9.3.1~~ CP – Change PIN Plug-In

9.1.4.1.1~~9.3.1.1~~ Description

The CP plug-in shall be used to change a PIN to a value specified by the user. The user is requested to enter first the old PIN and then the new PIN twice, before the PIN is changed.

9.1.4.1.2~~9.3.1.2~~ NCI

The NCI for this plug-in is '00 08'.

9.1.4.1.3~~9.3.1.3~~ Arguments

~~To be defined in REL-6.~~The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	'01'/ '03'/'04'	Key identifier type. Indicates the type of the key identifier supplied in the next parameter: • '01' = User key hash. SHA-1 hash of the user public key is supplied in the next parameter. The plug-in shall use the private key that corresponds to the public key hash or, if this key is not available, abort with Plug-in Status Code "PS: No such key error". • '03' = Index of RSA key. • '04' = Index of secret key.	M
1	Data	Index of secret key.	C
1	Data	Index of RSA key.	C
20	Data	User key hash.	C

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

9.1.4.1.4~~9.3.1.4~~ Output Parameters

~~To be defined in REL-6.~~The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.4.1.6).

9.1.4.1.5~~9.3.1.5~~ Execution

~~To be defined in REL-6.~~The detailed execution of the plug-in is described in subclause E.1.1.

9.1.4.1.6~~9.3.1.6~~ Errors

~~To be defined in REL-6.~~Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: User cancel"	'21'	The user cancelled the operation.
"PS: No such key"	'22'	The requested key is not available.

9.1.4.2~~9.3.2~~ RP – Reset P IN Plug-In

9.1.4.2.1~~9.3.2.1~~ Description

The RP plug-in shall be used by a specially trusted party to set a PIN value OTA to a value of its own choice ~~remotely~~.

9.1.4.2.2~~9.3.2.2~~ NCI

The NCI for this plug-in is '00 09'.

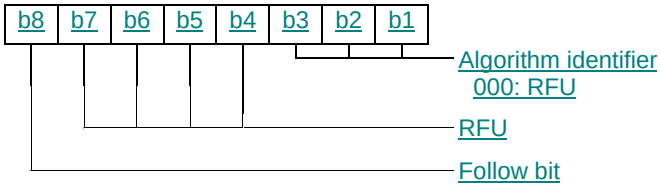
9.1.4.2.3~~9.3.2.3~~ Arguments

~~To be defined in REL-6.~~The arguments (i.e. the value part of the Inline Value TLV within the Input List TLV) shall be according to the following table:

Length	Value	Description	M/O/C
1	'01'/ '03'/'04'	Key identifier type. Indicates the type of the key identifier supplied in the next parameter: • '01' = User key hash. SHA-1 hash of the user public key is supplied in the next parameter. The plug-in shall use the private key that corresponds to the public key hash or, if this key is not available, or abort with Plug-In Status Code "PS: No such key". • '03' = Index of RSA key. • '04' = Index of secret key.	M
1	Data	Index of secret key.	C
1	Data	Index of RSA key.	C
20	Data	User key hash.	C
A	Data	Options.	M
B	Data	Encrypted PIN data (EP).	M

Malformed, out of range, or missing input parameters shall result in Error Code "Syntax Error" and plug-in termination.

Coding of the "Options" field:



9.1.4.2.4~~9.3.2.4~~ Output Parameters

~~To be defined in REL-6.~~The following table describes the output of the plug-in:

Output Variable #	Content
1	Plug-in Status Code (see subclause 9.1.4.2.6).

9.1.4.2.5~~9.3.2.5~~ Execution

~~To be defined in REL-6.~~

The detailed execution of the plug-in is described in subclause E.2.1.

9.1.4.2.6~~9.3.2.6~~ Errors

~~To be defined in REL-6.~~

Possible Plug-in Status Codes (see 9.1.1.4 for additional information):

Plugin Status Code	Coding	Description
"PS: OK"	'00'	There was no error.
"PS: No such key"	'22'	The requested key is not available.

Annex D (normative): PKI Plug-ins Implementation Specification

This annex provides a detailed description of the PKI plug-ins described in subclause 9.1.2.

D.1 P7

D.1.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the P7 execution.

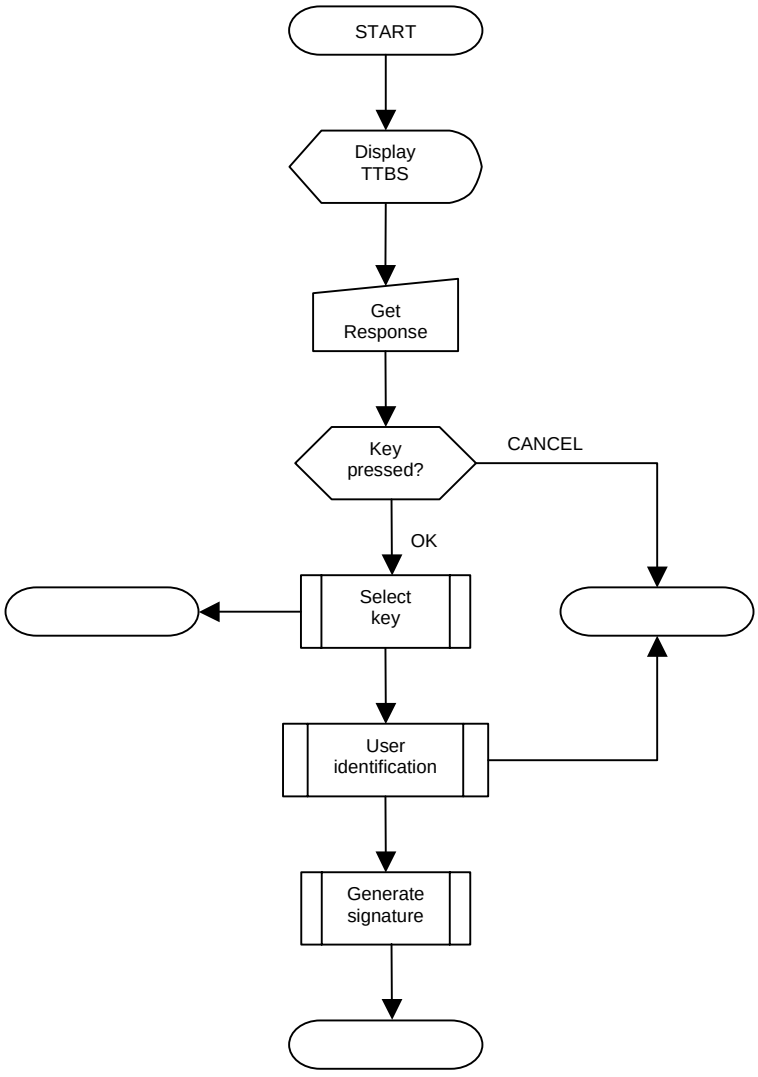


Figure D.1: P7 Flow diagram

The plug-in starts by showing the text-to-be-signed to the user and then awaits user confirmation. The user confirms by pressing a confirmation-button (any button resulting in a Terminal Response with a general result range '00 0F') or cancels by pressing a cancellation-button (any other general result value). If the user confirms, he shall be asked to enter his PIN and after that, if the PIN was valid, the plug-in calculates the signature.

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	SignedContent data	Indicates success.
CANCEL	"PS: User cancel"	“error:userCancel”	The user aborted the operation.
NO KEY	"PS: No such key"	“error:noCert”	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

D.1.1.1 User Identification

The "User identification" procedure is rather complex since it involves many states as well as alternative execution paths. The remainder of this subclause illustrates, using a combination of flow diagrams and sequence diagrams, the general characteristics of the user identification process.

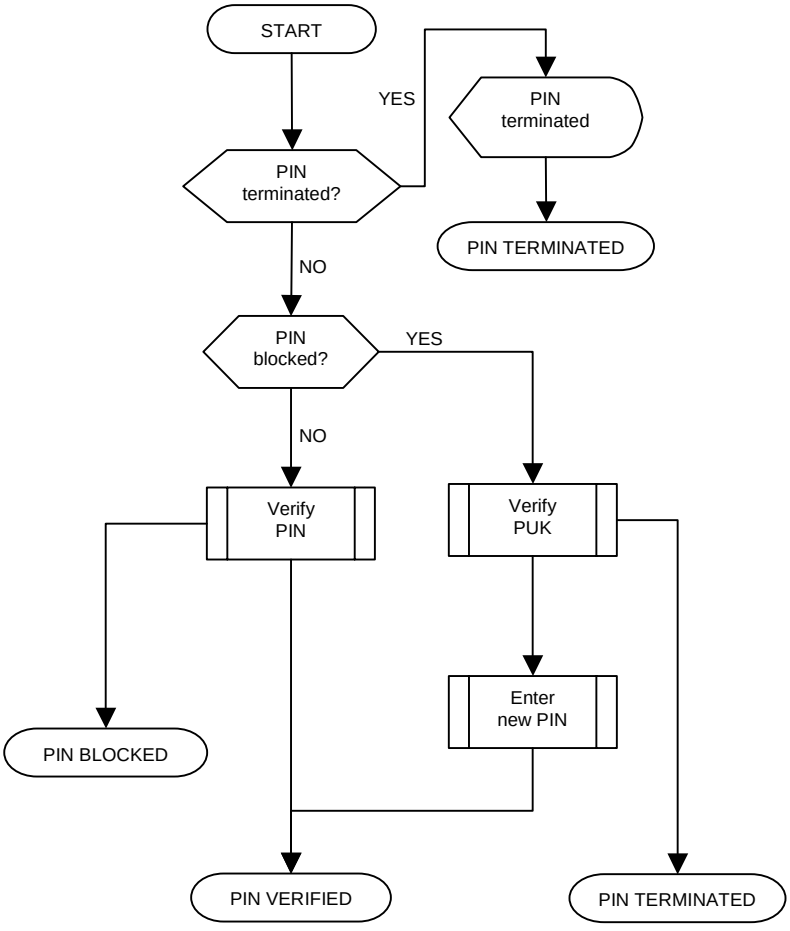


Figure D.2: User Identification Overview

If the execution stops in a "PIN TERMINATED" or "PIN BLOCKED" state, this shall lead to Error Code "Execution Error" and plug-in termination.

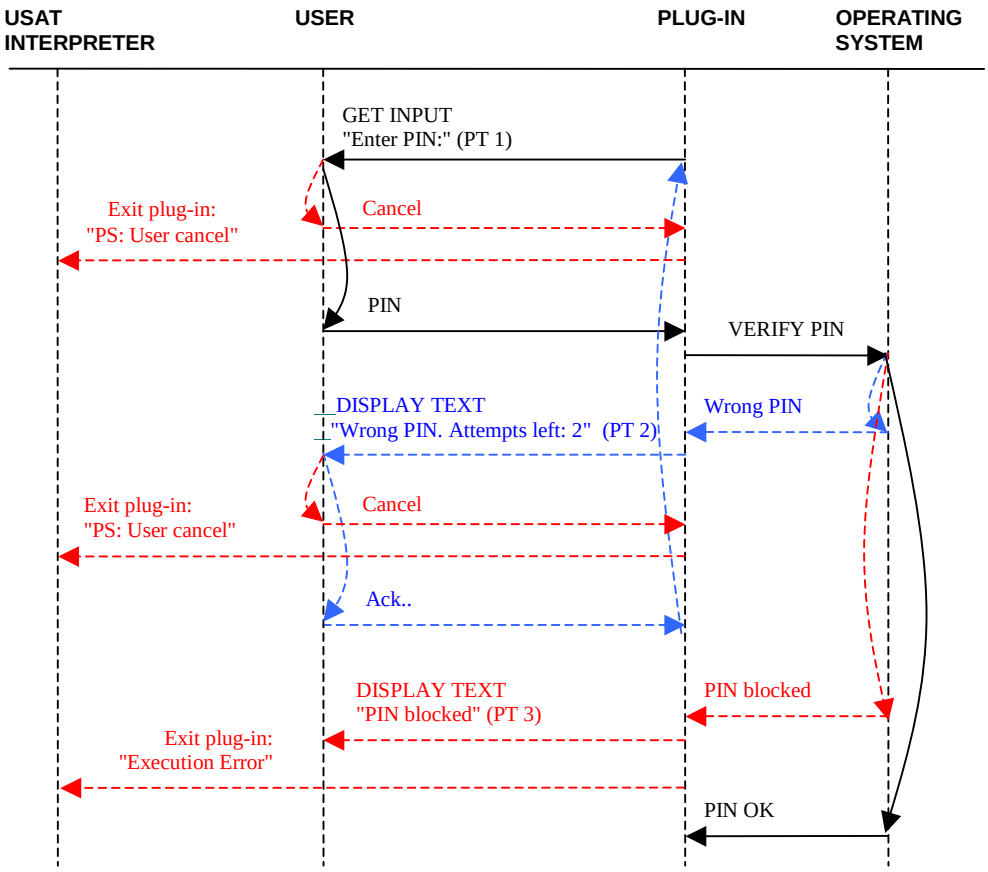


Figure D.3: Verify PIN

"Verify PIN" procedure is implemented according to the figure D.3.

The maximum and minimum length restrictions on the PIN value shall be included into the GET INPUT command and b3 of the command qualifier of the GET INPUT command shall be set to 1 (i.e. user input shall not be revealed in any way) in order to hide the PIN code entered by the user on the display of the UE.

If the PIN is entered incorrectly, the "Wrong PIN" (Prompt text nr 2) text shall be displayed concatenated with the number of attempts left. E.g. if the "Wrong PIN" message is "Wrong PIN. Attempts left: "and there are two attempts left before blocking, the message displayed on the screen shall be "Wrong PIN. Attempts left: 2".

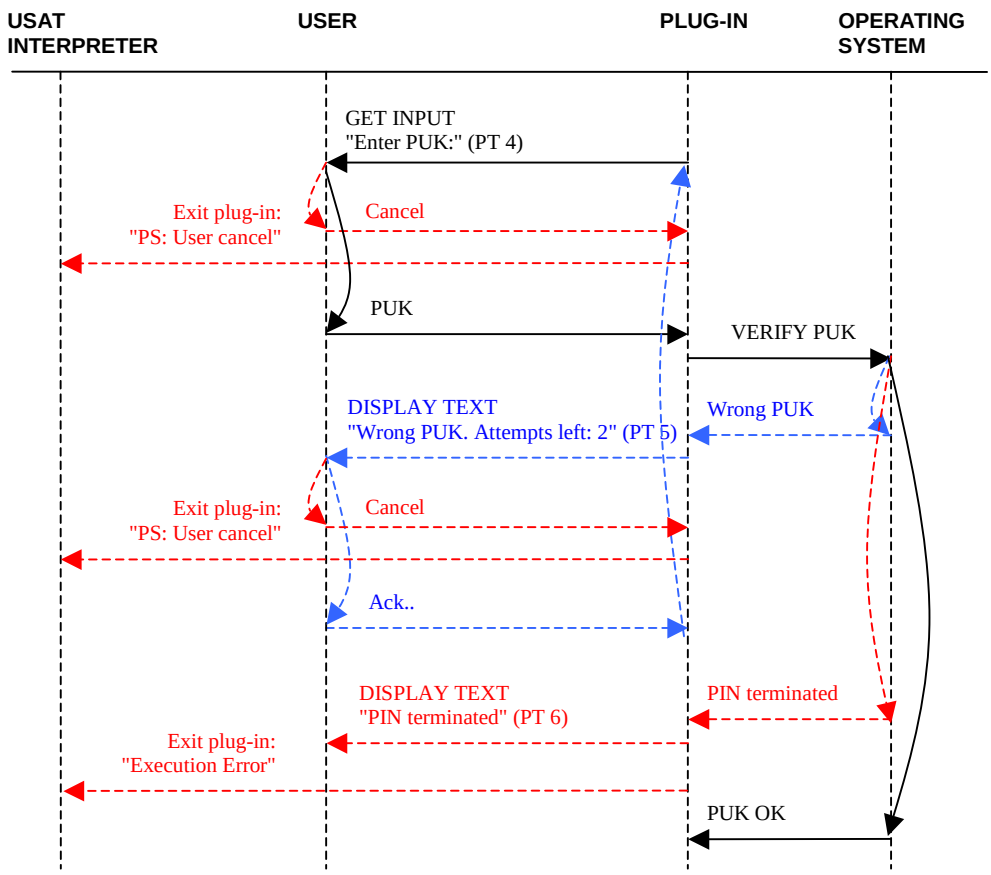


Figure D.4: Verify PUK

"Verify PUK" procedure is implemented according to the figure D.4.

The maximum and minimum length restrictions on the PUK value shall be included into the GET INPUT command and b3 of the command qualifier of the GET INPUT command shall be set to 1 (i.e. user input shall not be revealed in any way) in order to hide the PUK code entered by the user on the display of the UE.

If the PUK is entered incorrectly, the "Wrong PUK" (prompt text no 5) message shall be displayed concatenated with the number of attempts left. E.g. if the "Wrong PUK" message is "Wrong PUK. Attempts left: " and there are two attempts left before blocking, the message displayed on the screen shall be "Wrong PUK. Attempts left: 2".

PUK functionality is an optional feature of the present specification.

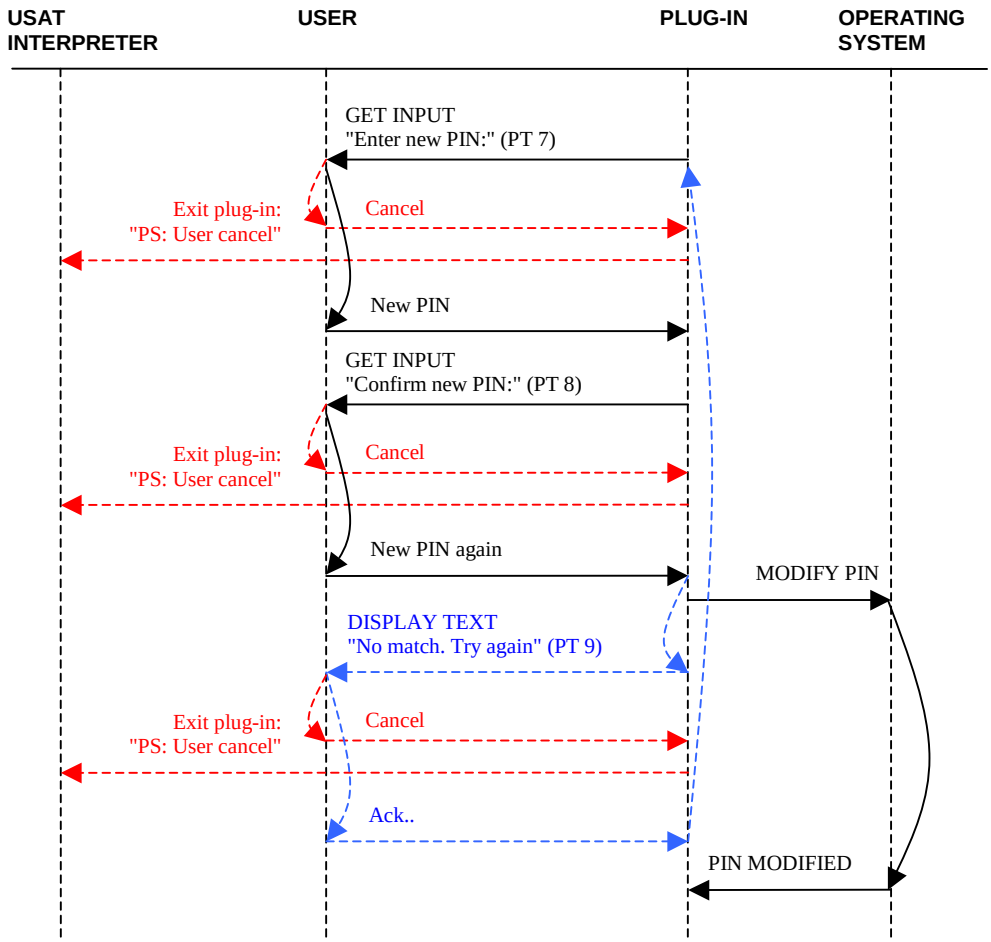


Figure D.5: Enter New PIN

"Enter New PIN" procedure is implemented according to the figure D.5.

The user is requested to enter the new PIN twice. If the two PIN entries does not match, the procedure shall restart. The use may abort the procedure (and the plug-in) at any time by pressing a cancellation-button (a button with a Terminal Response not in the general result range '00 0F'. If the user enters two identical PIN values, the plug-in shall modify the corresponding PIN value to the value entered.

Following prompt texts are used in the "User Identification" procedure:

Prompt Text #	Prompt Text example	Command type	Associated procedure
1	"Enter PIN:"	GET INPUT (digits only, hidden, max. and min. length set accordingly)	Verify PIN
2	"Wrong PIN. Attempts left: 2"	DISPLAY TEXT (high priority, wait for user to clear message)	Verify PIN
3	"PIN blocked"	DISPLAY TEXT (high priority, wait for user to clear message)	Verity PIN
4	"Enter PUK:"	GET INPUT (digits only, hidden, max. and min. length set accordingly)	Verify PUK
5	"Wrong PUK. Attempts left: 2"	DISPLAY TEXT (high priority, wait for user to clear message)	Verify PUK
6	"PIN terminated"	DISPLAY TEXT (high priority, wait for user to clear message)	Verify PUK
7	"Enter new PIN:"	GET INPUT (digits only, hidden, max. and min. length set accordingly)	Enter new PIN
8	"Confirm new PIN:"	GET INPUT (digits only, hidden, max. and min. length set accordingly)	Enter new PIN
9	"No match. Try again."	DISPLAY TEXT (high priority, wait for user to clear message)	Enter new PIN

D.1.2 Signature Calculation

The output from the P7 plug-in is a SignedContent data structure as specified in [14]. The (ordered) steps to produce this data structure are as follows:

1. Template expansion
2. Signing
3. Output formatting

Each step is described thoroughly in the following sections.

D.1.2.1 Template Expansion

The template expansion constructs the signer's authenticated attributes. These are:

Attribute	OID	Binary OID
contentType	pkcs-9 3	'2A 86 48 86 F7 0D 01 09 03'
messageDigest	pkcs-9 4	'2A 86 48 86 F7 0D 01 09 04'
signerNonce	pkcs-9 25 3	'2A 86 48 86 F7 0D 01 09 19 03'

See [12] for further information regarding these attributes.

First, construct the following 91-byte buffer ('xx' indicates an undefined value):

```
31 59
    30 18
        06 09 2A 86 48 86 F7 0D 01 09 03    -- contentType
        31 0B
            06 09 2A 86 48 86 F7 0D 01 07 01 -- data
    30 18
        06 0A 2A 86 48 86 F7 0D 01 09 19 03 -- signerNonce
        31 0A
            04 08 xx xx xx xx xx xx xx xx    -- random nonce
    30 23
        06 09 2A 86 48 86 F7 0D 01 09 04    -- messageDigest
        31 16
            04 14 xx xx xx xx xx xx xx xx xx xx -- SHA-1 digest
            xx xx xx xx xx xx xx xx xx xx xx xx
```

The authenticated attributes are included in ascending order compared as byte strings.

Now perform the following steps.

1. Generate *R*, an 8 byte nonce, and replace B47 to B54 of the buffer with *R*. Recommended standards for implementing pseudorandom bit generators are ANSI X9.19 or FIPS 186.

   NOTE: The nonce should be a pseudorandom number generated securely in the USIM and of good quality.

2. Generate

   MD = SHA-1(TTBS).

   Replace B72 to B91 of the buffer with *MD*.

The expanded buffer constitutes the input to the signature generation operation.

D.1.2.2 Signature Generation Operation

Generate the signature

S = RSASSA-PKCS1-v1_5-SIGN(K, M)

where *K* is the selected private key and *M* is the output from the previous step.

The hash function required in *EMSA-PKCS1-v1_5-ENCODE* shall be *SHA-1*. See [10] for further details.

D.1.2.3 Output data formatting

The SignedContent data-structure may be encoded in a one-pass encoding operation. The pseudo-code below covers the required steps.

B := '01'
B := *B* || '01'
B := *B* || *k* || *S*
siLen := 0
IF key hash flag is set
    siLen := *siLen* + 21
END
IF ICCID flag is set
    siLen := *siLen* + 11
END
IF key index flag is set
    siLen := *siLen* + 2
END
IF certificate flag is set
    z := 0
    FOR all certificate URLs
        urlLen = ||URL||
        z := *z* + *urlLen* + 2
    END
    siLen := *siLen* + *z*
END
B := *B* || *siLen*
IF ICCID flag is set
    B := *B* || '80' || *ICCID*
END
IF key index flag is set
    B := B || '81' || *AKI*
END
IF key hash flag is set
    B := *B* || '01' || *KH*
END
IF certificate flag is set
    FOR all certificate URLs
        urlLen = ||URL||
        B := *B* || '05' || *urlLen* || *URL*
    END
END
B := *B* || '01'
IF character encoding scheme is UCS2
    B := *B* || '03E8'
ELSE
    B := *B* || '07D0'
END
IF content flag is set
    ttbsLen = ||*TTBS*||
    B := *B* || '01' || *ttbsLen* || *TTBS*
ELSE
    B := *B* || '00'
END
IF message digest flag is set
    B := *B* || '1E' || '80' || *MD*
ELSE
    B := *B* || '09'
END
B := *B* || '02' || *R*

After the last step, the variable B contains the Functional Output.

k, *siLen* and *ttbsLen* shall all be encoded in two bytes, big endian.

NOTE: Using ICCID as a SignerInfo has no equivalent in [14].

NOTE: The value '07 D0' (2000 decimal) is used due to fact that IANA [16] has not assigned a character set number for the GSM default character set.

D.2 FP

D.2.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the FP execution.

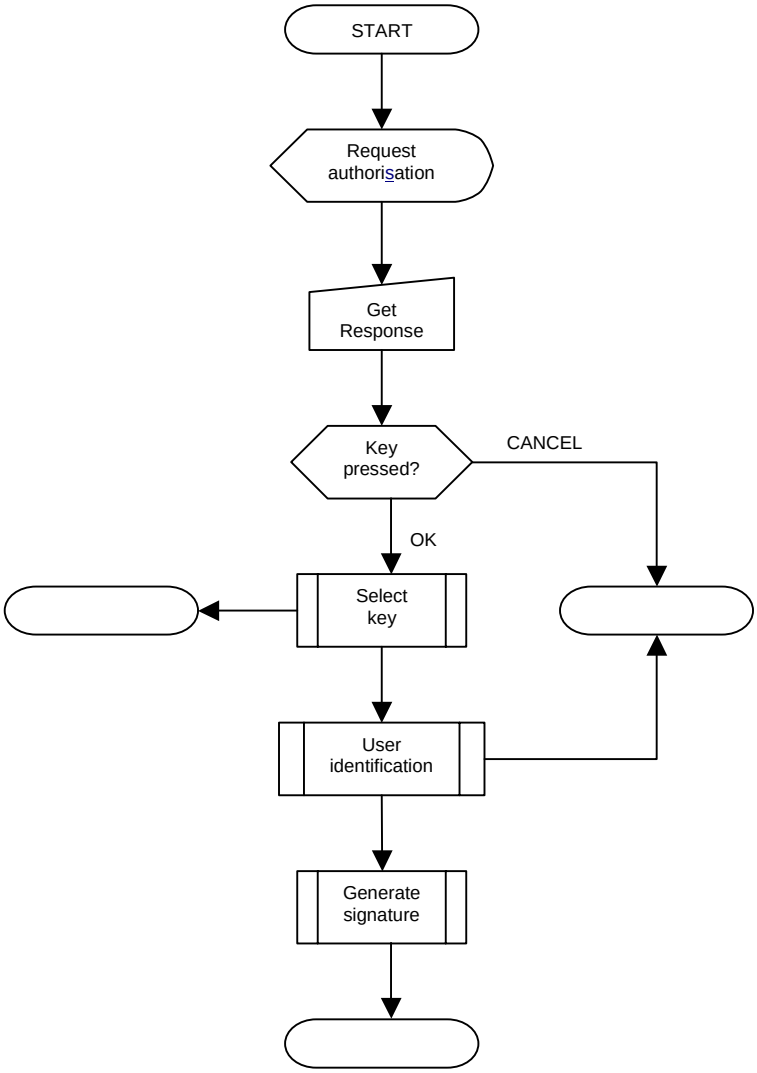


Figure D.6: FP Flow Diagram

The plug-in starts by displaying the authorisation request to the user and the await user confirmation.

The authorisation request itself consists of the *authorisation prompt* concatenated with the *authorisation value*, which is an excerpt of the data-to-be-signed (DTBS). The authorisation value shall be displayed using a two-digit hexadecimal representation for every byte. The digits of the hexadecimal alphabet shall be "0123456789ABCDEF", i.e. lower-case letters are not allowed. If DTBS is longer than 16 bytes, only the 16 least significant bytes shall be shown, starting with the most significant byte. To improve readability, the hexadecimal digits shall be grouped 4-and-4, with space between the groups. Splitting a group over two consecutive lines should be avoided if possible.

After explicitly validating the authorisation value with information received via some other channel, the user confirms by pressing a confirmation-button (any button resulting in a Terminal Response with general result range '00 0F') or cancels by pressing a cancellation-button (any other general result value). If the user confirms, he shall be asked to enter his PIN and after that, if the PIN was valid, the plug-in calculates the signature.

The "User identification" procedure is identical to the procedure described in subclause D.1.1.1.

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	WrappedContent data	Indicates success.
CANCEL	"PS: User cancel"	“error:userCancel”	The user aborted the operation.
NO KEY	"PS: No such key"	“error:noCert”	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

D.2.2 Signature Calculation

The output from the FP plug-in is a WrappedContent data structure as specified in subclause D.2.3. The (ordered) steps to produce this data structure are as follows:

1. Signing
2. Output formatting

Each step is described thoroughly in the following subclauses.

D.2.2.1 Signature Generation Operation

Generate the signature

$S = RSASSA\text{-}PKCS1\text{-}v1\_5\text{-}SIGN(K, DTBS)$

where *K* is the selected private key and *DTBS* is supplied as an input parameter.

In *EMSA-PKCS1-v1_5-ENCODE*, only steps from (including) step 3 shall be executed. The following equality (using PKCS#1 terminology) apply for the computation of the remaining steps:

$T = DTBS$ and $||T|| = ||DTBS||$

D.2.2.2 Output data formatting

The WrappedContent data-structure may be encoded in a one-pass encoding operation. The pseudo-code below covers the required steps.

B := '02'
B := *B* || *k* || *S*
siLen := 0
IF key hash flag is set
    siLen := *siLen* + 21
END
IF ICCID flag is set
    siLen := *siLen* + 11
END
IF key index flag is set
    siLen := *siLen* + 2
END
IF certificate flag is set
    z := 0
    FOR all certificate URLs
        urlLen = ||URL||
        z := *z* + *urlLen* + 2
    END
    siLen := *siLen* + *z*
END
B := *B* || *siLen*
IF ICCID flag is set
    B := *B* || '80' || *ICCID*
END
IF key index flag is set
    B := B || '81' || *AKI*
END
IF key hash flag is set
    B := *B* || '01' || *KH*
END
IF certificate flag is set
    FOR all certificate URLs
        urlLen = ||URL||
        B := *B* || '05' || *urlLen* || *URL*
    END
END

k and *siLen* shall be encoded in two bytes, big endian.

After the last step, the variable B contains the Functional Output.

D.2.3 Format of WrappedContent

For completeness, the formal definition of WrappedContent is included below (it is described using the same presentation language as used in [14]).

```
struct {
  opaque signature<0.. 2^16-1>;
} Signature;

enum {
  sha_key_hash(1),
  certificate_url(5),
  iccid (128),
  aki (129),
  (255)
} SignerInfoType;
```

Item	Description
sha_key_hash	The SHA-1 hash of the public key, encoded as specified in [15].
certificate_url	A URL where the certificate is located.
iccid	The (raw) ICCID.
aki	The Index of the used private key.

```
struct {
  SignerInfoType signer_info_type;
  switch (signer_info_type) {
   case sha_key_hash: opaque hash[20];
   case certificate_url: opaque url<0..255>;
   case iccid: opaque iccid[10];
   case aki: uint8;
  };
} SignerInfo;


struct {
  uint8 version;
  Signature signature;
  SignerInfo signer_infos<0..2^16-1>;
} WrappedContent;
```

Item	Description
version	Version of the WrappedContent structure. The current version is 2.
signature	Signature
signer_infos	Information about the signer. This may contain zero items (in case the signer is implicit). Also, there may be multiple items of SignerInfo present (public key hash and a certificate).

D.3 AD

D.3.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the AD execution.

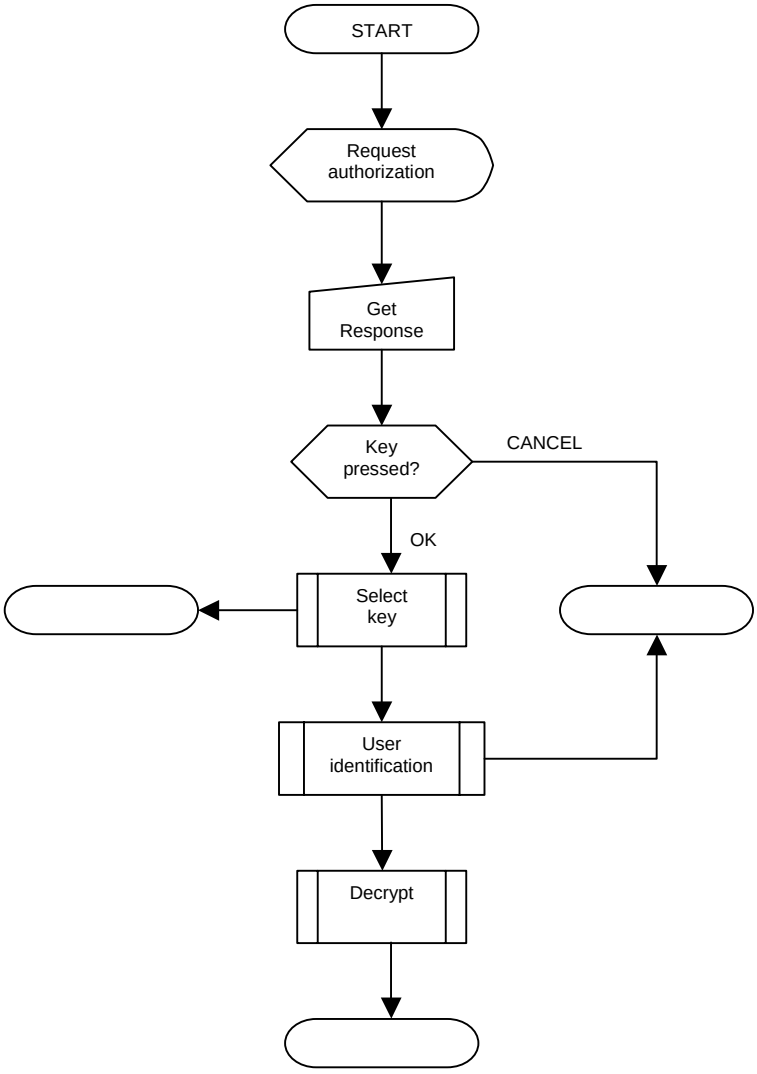


Figure D.7: AD Flow Diagram

The plug-in starts by displaying the authorisation request to the user and the await user confirmation.

The authorisation request itself consists of the *authorisation prompt* concatenated with the *authorisation value*, which is an excerpt of the ciphertext (C). The authorisation value shall be displayed using a two-digit hexadecimal representation for every byte. The digits of the hexadecimal alphabet shall be "0123456789ABCDEF", i.e. lower-case letters are not allowed. If C is longer than 16 bytes, only the 16 least significant bytes shall be shown, starting with the most significant byte. To improve readability, the hexadecimal digits shall be grouped 4-and-4, with space between the groups. Splitting a group over two consecutive lines should be avoided if possible.

After explicitly validating the authorisation value with information received via some other channel, the user confirms by pressing a confirmation-button (any button resulting in a Terminal Response with a general result range '00 0F') or cancels by pressing a cancellation-button (any other general result value). If the user confirms, he shall be asked to enter his PIN and after that, if the PIN was valid, the plug-in decrypts the data.

The "User identification" procedure is identical to the procedure described in subclause D.1.1.1.

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	decrypted data	Indicates success.
CANCEL	"PS: User cancel"	“error:userCancel”	The user aborted the operation.
NO KEY	"PS: No such key"	“error:noCert”	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

D.3.2 Decryption calculation

The decrypted ciphertext (i.e. plaintext), is generated by computing the following steps.

1. Convert the ciphertext *C* to an integer ciphertext representative *c*:

   $c = OS2IP(C)$

2. Calculate the integer message representative m:

   $m = RSADP(K, c)$

   where K is the selected private key.

3. Convert the message representative *m* to an encoded message *M* of length *k* bytes:

   $M = I2OSP(m, k)$

M represents the decrypted ciphertext, and hence the Functional Output.

D.4 Non-functional Requirements

D.4.1 Customisation Requirements

1. All customisation requirements with regard to PINs and PUKs listed in E.3.1 apply equally here.
2. It shall be possible to enable or disable the "Authorisation request" and the subsequent user confirmation by performing an administrative task at personalisation time.
3. The authorisation prompt shall be configurable through an administrative task at personalisation time. UCS2 and GSM default alphabets shall be supported.
4. It should be possible to configure the number of digits displayed in the authorisation value through an administrative task at personalisation time. The number of digits displayed shall be 4, 8, 12 or 16, with 16 as the default.
5. The list of URL(s) linked to a private key shall be updatable through an administrative task at personalisation time.
6. The list of trusted key hashes linked to a private key shall be updatable through an administrative task at personalisation time.

D.4.2 Architectural Requirements

1. All architectural requirements with regard to PINs and PUKs listed in E.3.2 apply equally here.

Annex E (normative): PIN Management Plug-ins Implementation Specification

This annex provides a detailed description of the PIN management plug-ins defined in subclause 9.1.4.

E.1 CP

E.1.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the CP execution.

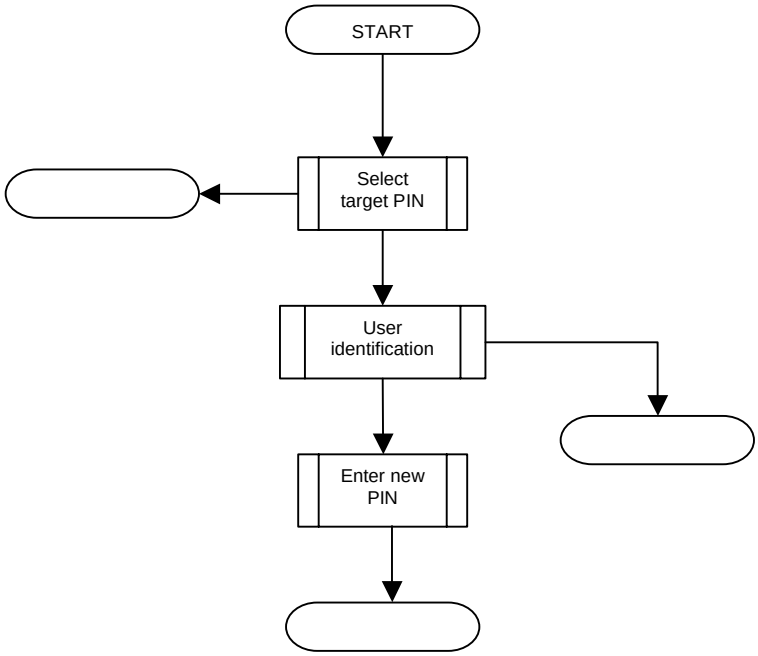


Figure E.1: CP Flow Diagram

The plug-in execution starts with locating the PIN to be changed based on the key identifier input parameter.

After locating the target PIN, the user is requested to enter the PIN (if the PIN is not blocked) and thereafter prompted twice for a new PIN as described in subclause D.1.1.1.

If the user is subjected to a PUK verification due to blocked PIN, the "Enter new PIN" procedure shall only be executed once.

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	-	Indicates success.
CANCEL	"PS: User cancel"	“error:userCancel”	The user aborted the operation.
NO KEY	"PS: No such key"	“error:noKey”	Can not locate target PIN.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

Sub procedures "User identification" and "Enter new PIN" are all described in detail in subclause D.1.1.1.

The maximum and minimum length restrictions on the PIN value shall be checked before PIN modification. If violated, the plug-in shall set the Error Code to "Execution Error" and terminate.

E.2 RP

E.2.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the RP execution.

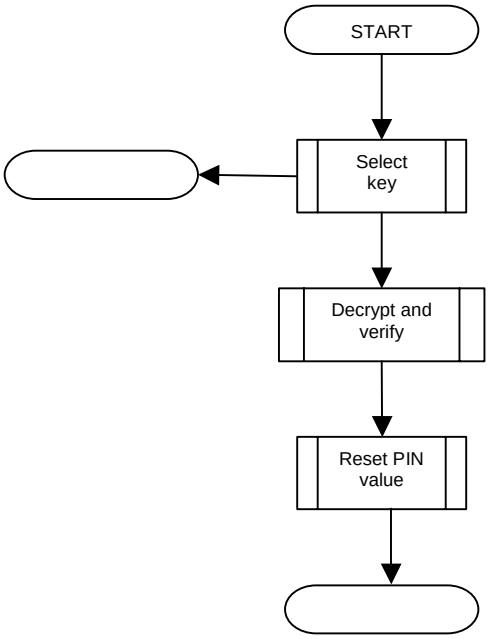


Figure E.2: RP Flow Diagram

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	-	Indicates success.
NO KEY	"PS: No such key"	“error:noKey”	Can not locate target PIN.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

Changing the PIN value is simply copying the new PIN value to the appropriate location, possibly stripping of the padding bytes and/or converting the PIN value to an internal format. The "remaining attempts" counter shall always be reset to its maximum value at the same time.

The maximum and minimum length restrictions on the PIN value shall be checked. If violated, the plug-in shall set the Error Code to "Execution Error" and terminate.

E.2.2 Decryption and Verification

This procedure includes decryption of the encrypted PIN data, as well as verification of it’s authenticity.

To decrypt and verify the encrypted PIN data, select the correct algorithm based on the algorithm identifier and thereafter decrypt and verify according to the selected algorithm.

An implementation shall support at least one algorithm.

Algorithms employing SHA-1 are preferred prior to algorithms employing ISO/IEC 9797.

E.2.2.1 3DES EDE CBC with two keys + SHA-1 MDC

The decrypted PIN data shall be formatted according to the table below:

Bytes	Description	M/O	Length
1 – 8	Nonce. 8 bytes of random data.	M	8
9 – 16	PIN value. Each digit in the PIN shall be encoded with its corresponding GSM default alphabet value. All unused digits at the end shall be encoded as 'FF'.	M	8
17 – 24	PIN checksum. Truncated SHA-1 MDC.	M	8

To decrypt and verify the PIN data, do the following:

1. Calculate the decrypted PIN data

   DP = TDEA_DECR(EP)

   using the following cipher parameterisation:

Keys	K_1, K_2
Cipher mode	Outer CBC using two keys in EDE operation.
IV	'00 … 00' (this is not a weakness since the nonce effectively becomes a randomly chosen IV).

   a) Calculate

      MD = SHA1(unencrypted parameters || DP<1..16>).

      The unencrypted parameters ("Key identifier type", "Key identifier" and "Options") shall be included in the checksum calculation to avoid certain replay attacks.

   b) Calculate the PIN checksum

      PC = MD<1..8>

   c) Compare *PC* with *DP<17..24>*. If identical, proceed to the next step. Otherwise, set Error Code to "Execution Error" and terminate.

   d) Success. The new PIN is *DP<9..16>*.

E.2.2.2 3DES EDE CBC with two keys + ISO/IEC 9797 MAC

The decrypted PIN data shall be formatted according to the table below:

Bytes	Description	M/O	Length
1 – 8	Nonce. 8 bytes of random data.	M	8
9 – 16	PIN value. Each digit in the PIN shall be encoded with its corresponding GSM default alphabet value. All unused digits at the end shall be encoded as 'FF'.	M	8
17 – 24	PIN checksum . ISO/IEC 9797 MAC.	M	8

To decrypt and verify the PIN data, do the following:

1. Calculate the decrypted PIN data

   DP = TDEA_DECR(EP)

   using the following cipher parameterisation:

Keys	K_1, K_2
Cipher mode	Outer CBC using two keys in EDE operation.
IV	'00 … 00' (this is not a weakness since the nonce effectively becomes a randomly chosen IV).

2. Calculate

   PM = ISO_IEC_9797_PAD2(unencrypted parameters || DP<1..16>).

   The unencrypted parameters ('Key identifier type', 'Key identifier' and 'Options') shall be included in the checksum calculation to avoid certain replay attacks.

3. Calculate

   PC = ISO_IEC_9797_ALG3(PM).

   Using terminology from [11], keys *K* and *K'* shall be derived by complementing alternate sub-strings of four bits of K_1 and K_2 respectively, commencing with the four most significant bits.

   8 bytes of output from the MAC calculation shall be used (i.e. *m=64* using ISO/IEC 9797 terminology).

4. Compare *PC* with *DP<17..24>*. If identical, proceed to the next step. Otherwise, set the Error Code to 'Execution Error' and terminate.

5. Success. The new PIN is *DP<9..16>*.

E.2.2.3 3DES EDE CBC with three keys + SHA-1 MDC

This algorithm is identical to the algorithm described in E.6.2.1, except that the 3DES cipher shall be parameterized with three DES keys.

E.2.2.4 3DES EDE CBC with three keys + ISO/IEC 9797 MAC

This algorithm is identical to the algorithm described in E.6.2.2, except that the 3DES cipher shall be parameterized with three DES keys. For the MAC calculation, only K_1 and K_2 shall be used.

E.3 Non-functional Requirements

E.3.1 Customisation Requirements

1. Maximum number of attempts before blocking/termination for PINs and PUKs shall be configurable through an administrative task at personalisation time.

2. PIN and PUK values shall be configurable through administrative tasks at personalisation time.

3. All prompts displayed to the user during PIN/PUK verification shall be configurable through an administrative task at personalisation time. UCS2 and GSM default alphabets shall be supported.

4. All prompts displayed to the user during the PIN change procedure shall be configurable through an administrative task at personalisation time. UCS2 and GSM default alphabets shall be supported.:

5. The possibility to use the "Reset PIN" plug-in to reset a PIN shall be configurable on a per PIN basis, using an administrative task at personalisation time. I.e. some PINs may not be allowed to be reset via the "Reset PIN" plug-in, while others are.

6. Minimum and maximum PIN lengths shall be configurable using an administrative task at personalisation time. The same boundaries shall be shared by all PINs.

E.3.2 Architectural Requirements

1. It shall be possible to associate every key (private or secret) with a unique PIN. It shall also be possible for keys to share PINs, if so desired. The associations between keys and PINs shall be configurable through an administrative task at personalisation time. A key that is not linked to a PIN shall not be subjected to PIN verification before it is accessed.

2. Is shall be possible to associated a unique "Enter PIN" prompt (i.e. the first prompt displayed in the PIN verification procedure) to every PIN, and thereby to every key. This is to ensure that the user is given the possibility to recognize a key before using it. All other prompts may be shared between PINs.

3. It shall be possible to associate every PIN with a unique PUK.

4. PIN lengths between 4 and 8 digits shall be supported.

5. Successfully entering a PIN shall only grant access to the underlying key (private or secret) for the remaining duration of the plug-in execution. I.e. the next time the plug-in is executed, a new PIN verification is required.

6. A "terminated" PIN, i.e. a PIN who’s PUK has be unsuccessfully exercised for the maximum allowed number of times, shall not be usable, changeable or reset-able by any means. In other words, it shall be unconditionally unrecoverable.

Annex F (normative): Triple DES Plug-ins Implementation Specification

This annex provides a detailed description of the triple DES plug-ins outlined in subclause 9.1.3 of this document.

F.1 DE

F.1.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the DE execution.

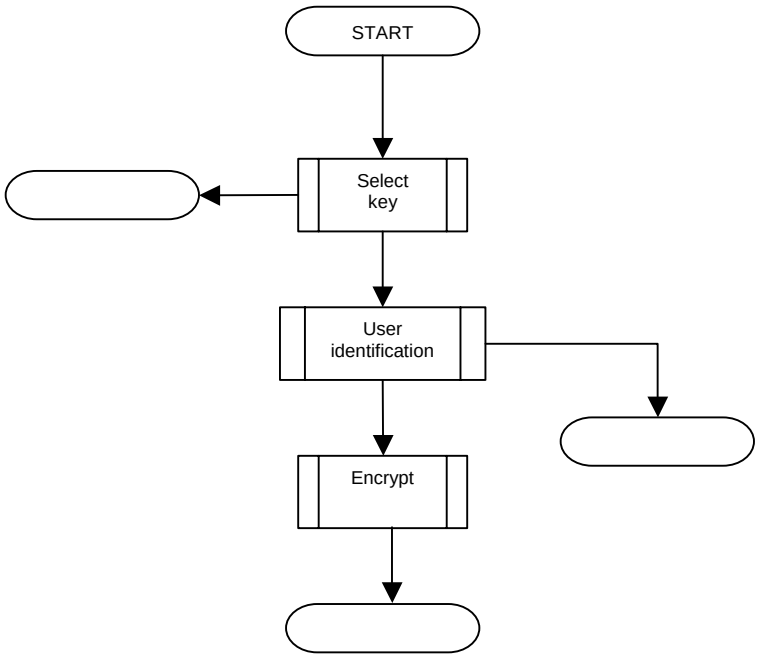


Figure F.1: DE Flow Diagram

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	encrypted data	Indicates success.
CANCEL	"PS: User cancel"	“error:userCancel”	The user aborted the operation.
NO KEY	"PS: No such key"	“error:noKey”	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

The "User identification" procedure is identical to the procedure described in subclause D.1.1.1.

F.1.2 Encrypt Procedure

To encrypt the plaintext, do the following:

1. Calculate the padded message

   PM = PKCS5_PAD(Plaintext).

2. Calculate the encrypted message

   EM = TDEA_ENCR(PM)

   using the following cipher parameterisation:

Keys	K_1, K_2 and possibly K_3 as indicated by 'Cipher spec'.
Cipher mode	ECB or CBC as indicated by "Cipher spec".
IV	Indicated by "IV flag".

3. *EM* is the Functional Output.

F.2 DD

F.2.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the DD execution.

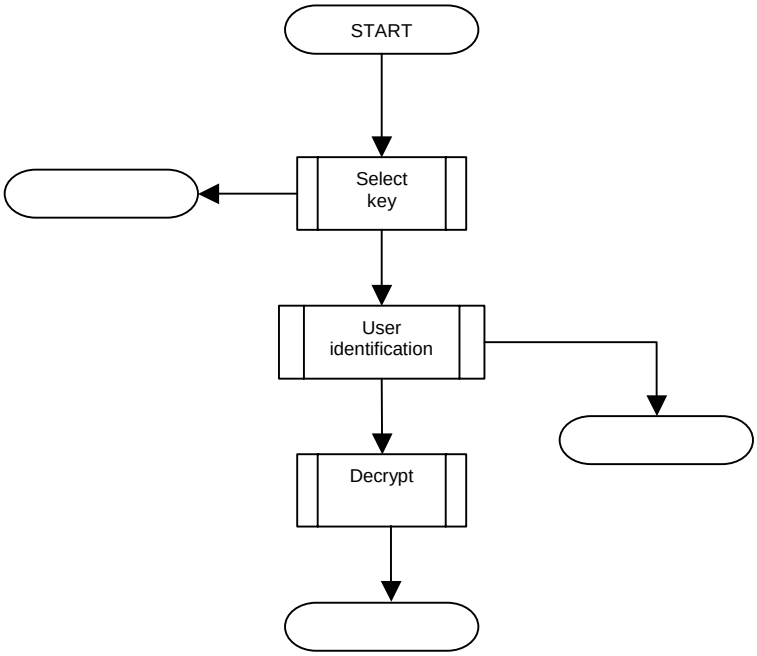


Figure F.2: DD Flow Diagram

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	'PS: OK'	decrypted data	Indicates success.
CANCEL	'PS: User cancel'	"error:userCancel"	The user aborted the operation.
NO KEY	'PS: No such key'	"error:noKey"	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

The "User identification" procedure is identical to the procedure described in subclause D.1.1.1.

F.2.2 Decrypt Procedure

To decrypt the ciphertext, do the following:

1. Calculate the padded plaintext message

   DM = TDEA_DECR(Ciphertext)

   using the following cipher parameterisation:

Keys	K_1, K_2 and possibly K_3 as indicated by "Cipher spec".
Cipher mode	ECB or CBC as indicated by "Cipher spec".
IV	Indicated by "IV flag".

2. Calculate the plaintext message

   M = PKCS5_UNPAD(DM).

3. *M* is the Functional Output..

F.3 DS

F.3.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the DS execution.

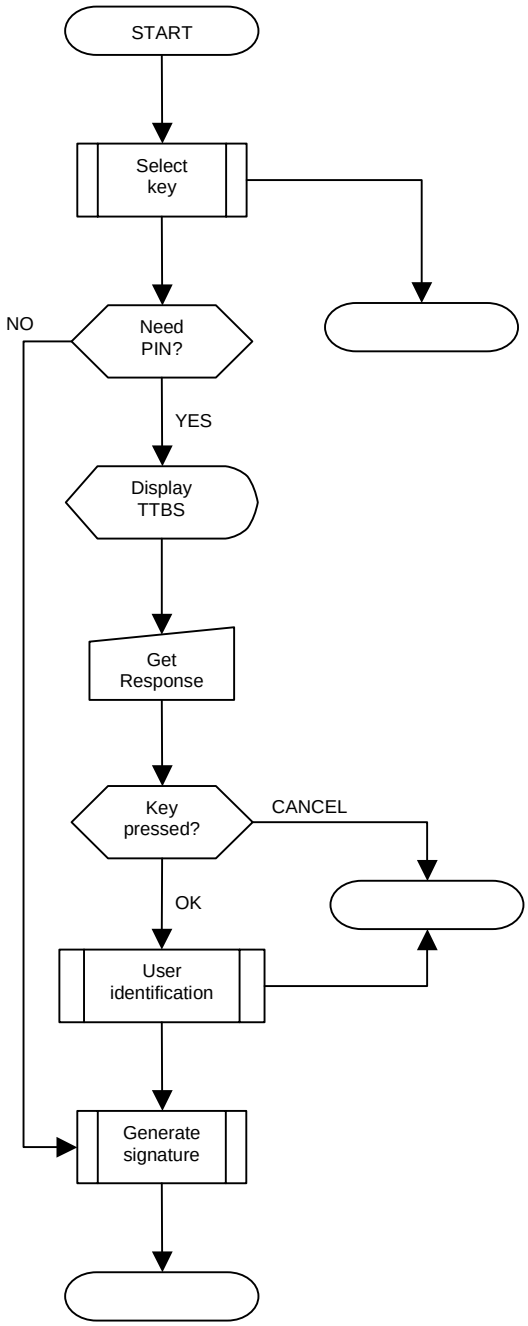


Figure F.3: DS Flow Diagram

As the figure illustrates, the plug-in shall check if the selected key has an associated PIN, and in this case display the text-to-be-signed to the user using the indicated character encoding scheme, and await user confirmation. The user confirms by pressing a confirmation-button (any button resulting in a Terminal Response with a general result range '00 0F') or cancels by pressing a cancellation-button (any other general result value).

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	signed data	Indicates success.
CANCEL	"PS: User cancel"	"error:userCancel"	The user aborted the operation.
NO KEY	"PS: No such key"	"error:noKey"	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

The "User identification" procedure is identical to the procedure described in subclause D.1.1.1.

F.3.2 MAC Calculation Procedure

To calculate the MAC, do the following:

1. Calculate the padded message

   PM = ISO_IEC_9797_PAD2(TTBS)

2. Calculate the MAC

   MAC = ISO_IEC_9797_ALG3(PM)

   using the following cipher parameterisation:

Keys	K_1, K_2
Truncation	As indicated by "Truncation flag".

3. *MAC* is the Functional Output.

F.4 DU

F.4.1 Plug-in Execution

The flow diagram below illustrates briefly the different steps of the DU execution.

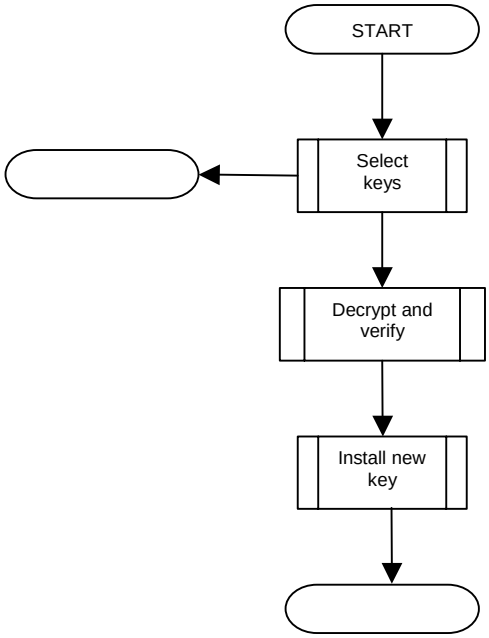


Figure F.4: DU Flow Diagram

The termination states shall be mapped to output variables according to:

State	Plug-in Status Code	Functional Output	Description
FINISHED	"PS: OK"	-	Indicates success.
NO KEY	"PS: No such key"	“error:noKey”	The requested key was not available.

In case of a serious error not listed above, an implementation may use any of the Error Codes listed in the error code table in subclause 8.8.

Installing the new key means simply copying the key material to the location referenced by key index input parameter.

F.4.2 Decryption and Verification Procedure

This procedure includes decryption of the encrypted key data, as well as verification of its authenticity.

To decrypt and verify the key data, select the correct algorithm based on the algorithm identifier field and thereafter proceed according to the selected algorithm.

An implementation shall support at least one algorithm.

Algorithms employing SHA-1 are preferred prior to algorithms employing ISO/IEC 9797.

F.4.2.1 3DES EDE CBC with two keys + SHA-1 MDC

The decrypted key data shall be formatted according to the table below.

Bytes	Description	M/O	Length
1 – 8	Random nonce.	M	8
9 – P	Key material	M	16 or 24
Q – R	Key checksum.	M	8

The values P,Q and R are calculated from wrapped key length according to the following table:

Wrapped key length	P	Q	R
16	24	25	32
24	32	33	40

To decrypt and verify the key data, do the following:

2. Select the key pointed to by the key index input parameter. This is the *destination key* ,K_D.

   a) Based on the key index parameter, locate the *unwrap key*, K_U.

   b) Calculate the decrypted key data

   DK = TDEA_DECR(Encrypted key data)

   using the following cipher parameterisation:

Keys	K_1 and K_2 of K_U.
Cipher mode	Outer CBC in EDE operation.
IV	'00 … 00' (this is not a weakness since the nonce effectively becomes a randomly chosen IV).

   a) Calculate the message digest

   MD = SHA1(unencrypted parameters || DK<1..P>)

   The unencrypted parameters ('Index of secret key' and 'Options') shall be included in the checksum calculation to avoid certain replay attacks.

   b) Calculate the key checksum

   KC = MD<1..8>

   c) Compare *KC* with *DK<Q..R>*. If identical, proceed to the next step. Otherwise, the plug-in shall set the Error Code to 'Execution Error' and terminate.

   d) Success.

F.4.2.2 3DES EDE CBC with two keys + ISO/IEC 9797 MAC

The format of the decrypted key data is the same as in the previous subclause (F.4.2.1).

To decrypt and verify the key data, do the following:

3. Select the key pointed to by the key index input parameter. This is the *destination key*, K_D.

   a) Based on the key index parameter, locate the *unwrap key*, K_U.

   b) Calculate the decrypted key data

      DK = TDEA_DECR(Encrypted key data)

      using the following cipher parameterisation:

Keys	K_1 and K_2 of K_U.
Cipher mode	Outer CBC in EDE operation.
IV	'00 … 00' (this is not a weakness since the nonce effectively becomes a randomly chosen IV).

   a) Calculate the padded message

      PM = ISO_IEC_9797_PAD2(unencrypted parameters || DK<1..P>)

      The unencrypted parameters ('Index of secret key' and 'Options') shall be included in the checksum calculation to avoid certain replay attacks.

   b) Calculate the key checksum

      KC = ISO_IEC_9797_ALG3(PM)

      Using terminology from [11], keys *K* and *K'* shall be derived by complementing alternate sub-strings of four bits of K_1 and K_2 respectively, commencing with the four most significant bits.

      8 bytes of output from the MAC calculation shall be used (i.e. *m=64* using ISO/IEC 9797 terminology).

   c) Compare *KC* with *DK<Q..R>*. If identical, proceed to the next step. Otherwise, the plug-in shall set the Error Code to "Execution Error" and terminate.

   d) Success.

F.4.2.3 3DES EDE CBC with three keys + SHA-1 MDC

This algorithm is identical to the algorithm described in F.4.2.1, except that the 3DES cipher shall be parameterized with three DES keys.

F.4.2.4 3DES EDE CBC with three keys + ISO/IEC 9797 MAC

This algorithm is identical to the algorithm described in F.4.2.2, except that the 3DES cipher shall be parameterized with three DES keys. For the MAC calculation, only K_1 and K_2 shall be used.

F.5 Non-functional Requirements

F.5.1 Customisation Requirements

1. All customisation requirements with regard to PINs and PUKs listed in E.3.1 apply equally here.
2. OTA modifiability of a key using the DU plug-in shall be configurable through an administrative task at personalisation time.

F.5.2 Architectural Requirements

1. All architectural requirements with regard to PINs and PUKs listed in E.3.2 apply equally here.