3GPP TSG-T (Terminals) Meeting #15
Jeju, Korea, 6 – 8 March 2002

Tdoc TP-020066

Source: T3

Title: Change Request to USAT interpreter specifications (TS 31.112 / 31.113)

Document for: Approval

This document contains several change requests as follows:

T3 Doc	Spec	CR	Rel	Cat	Subject
T3-020074	31.113	005	5	B	Extended WML Annex
T3-020092	31.113	006	5	F	Collection of Corrections
T3-020109	31.113	007	5	F	Clarification on behaviour on Single Actions for Terminal Response Handler
T3-020129	31.113	008	5	B	Addition of Security plug-ins

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **005** ⌘ rev **-** ⌘ Current version: **5.1.0** ⌘

*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:	⌘	Functional Additions to WML Annex
Source:	⌘	T3
Work item code:	⌘	USAT1-Interpr
Date:	⌘	23.01.2002
Category:	⌘	**B**
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 parts of the functionality of the USAT Interpreter is currently reflected in annex B of TS 31.113. This CR adds important missing functionality to annex B.
Summary of change:	⌘	Addition of functionality for accessing variables and handling of the Terminal Response Handler Modifiers.
Consequences if not approved:	⌘	

Clauses affected:	⌘	B2.1, B.5.2, B.5.2.2, B5.2.3 - B.5.2.10, B6, B7		
Other specs affected:	⌘	[ ] Other core specifications [ ] Test specifications [ ] O&M Specifications	⌘	
Other comments:	⌘			

B.2 Namespace

The WML code makes use of the concept of namespace to address the functionality. The WML code in the present document uses the `efi` scheme, as defined by WAP Forum in reference [B4], to address USAT commands, Card plug-ins and other explicitly addressed functionality. The concepts used in the namespace for addressing this functionality is described in that specification.

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

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

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

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

B.2.1 The USAT Interpreter EF Class

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

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

The service grouping used is listed in the below table.

Service Type	Service Group
USAT commands	atk/
Client side plug-in	cpi/
Server side Plug-In	spi/
USIM Manufacturer specifics	Ssp/
Interpreter Internals	ipi/

The present document only specifies specific forms for the `atk`, `spi` and `ipi` groups of services.

B.5.2 Services for Interpreter Commands

These are commands that are directed to the Interpreter itself and thus are internally handled by the interpreter. Unless otherwise stated, the encoding of the result variables match the format of the information as specified in other parts of this specification.

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

Service Name
ipi/getInterpreterVersion
~~-~~ipi/getBufferSize
ipi/getNativeCommandList
ipi/getTerminalProfile
ipi/getErrorCode
ipi/getMaxPageSize
ipi/getIssuerUrl
ipi/getIssuerUrlHash

B.5.2.1 Get Interpreter Version Information

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

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

Argument	Argument value	
outputVar	Variable to contain output data.	M

B.5.2.2 Get Interpreter Buffer Size

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

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

Argument	Argument value	
~~OutputVar~~ outputVar	Variable to contain output data.	M

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

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

B.5.2.3 Get Native Command List

This command reads the list of supported native commands.

Service name: **ipi/getNativeCommandList?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the output list of supported Native Commands	M

B.5.2.4 Get Terminal Profile

This command gets the Terminal Profile as got at runtime by the USAT Interpreter.

Service name: **ipi/getTerminalProfile?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the binary encoded terminal profile	M

B.5.2.5 Get Error Code for Last Byte Code Command

This command gets the Error Code generated by the last executed byte code command.

Service name: **ipi/getErrorCode?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the error code	M

B.5.2.6 Get Maximum Size for Temporary Storage of Page

This command gets the maximum page size for temporary storage of one page.

Service name: **ipi/getMaxPageSize?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the maximum size of a page	M

B.5.2.7 Get USAT Interpreter Issuer URL

This command gets the URL of the issuer of the USAT Interpreter.

Service name: **ipi/getIssuerUrl?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the URL of the issuer of the USAT Interpreter	M

B.5.2.8 Get USAT Interpreter Issuer URL Hash

This command gets the 4 most significant byte of the SHA-1 hash of the URL of the issuer of the USAT Interpreter.

Service name: **ipi/getIssuerUrl?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the hash of the URL	M

B.5.2.9 Get User Name

This command gets the name of the end user, if the end user has set the values.

Service name: **ipi/getUserName?outputVar=**

Argument	Argument value	
outputVar	Variable to contain the name of the end user	M

B.5.2.10 Get User Email

This command gets the email of the end user, if the user has chosen to set it.

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

Argument	Argument value	
outputVar	Variable to contain the email of the end user.	M

B.6 Access to Special Features

This chapter describes how to modify the behaviour of the USAT Interpreter. This includes modifying the Terminal Response Handler and variable management.

B.6.1 Variable Management

The byte code of the USAT Interpreter provides mechanisms for sharing access to variables between pages. The behaviour can be initiated from WML by using the constructs exemplified in this chapter.

Service Name
spi/keepAlive

B.6.1.1 Keep Alive and Protect Variables

The functionality to control saving of variables between decks is reached through a service. What is given is a list of variables that are to be shared with the next deck. Up to 64 variables can be indicated.

In the context of variable management, the one time password is used to control access to variables. Together with the Page Unlock Code, it provides a possibility for sharing variable values between decks in a protected manner. This is controlled by giving an argument to control password protection of the variables.

Service name:**spi/keepAlive?variableList=&password=**

Argument	Argument value	
variableList	List of the variables that are to be made available to the following page. If the argument is not present, all variables will be kept	O
usePassword	Indicates if the variables are to be protected by a usage of the combination of a one-time password and a page unlock. Values can be “yes” or “no”. The default value is “no”.	O
password	Gives the application provider the possibility to explicitly specify the password to be used for protecting the variables	O

The service is valid for the whole deck and is thus called in a template at deck level.

```
<wml>
 <template>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/spi/keepAlive?
            variableList='A, B, NAME'&amp;usePassword=yes
            &amp;password=gurksmorgas"/>
   </do>
 </template>
```

B.6.2 Terminal Response Handler Modifier

This chapter illustrates how the Terminal Response Handler can be modified. The Terminal Response Handler Modifier allows modification of the default behaviour for the Terminal Response Handler. In this context, modification includes addition to and overriding of the default behaviour. The Terminal Response Handler can be modified for the whole page and/or for each Navigation Unit.

When the service for modifying the Terminal Response Handler is called from a card, the scope is card. When the call is handled as a template at the deck level, it is valid for the whole deck.

The following table lists the logical group of services used for performing Terminal Response Handler modification.

Service Name
trh/replace
trh/add
trh/restore
trh/remove

The arguments to be supplied vary for the services.

B.6.2.1 Replace

Service name:
trh/replace?start=&end=&text=&actionDesc=&actionId=&href=&displayText=&variableName=&setvarValue=&getInputString

The replace operation erases all previously defined actions for a result range and adds the one supplied as an argument

Argument	Argument value	
start	The start of the general result range that is to be modified	M
end	The end of the general result range that is to be modified	M
text	Text to display to the user when handling this general result range.	O
actionDesc	Text to describe the action. To be used in User Interface for select item when asking the user which action to perform when multiple actions are defined for the general result range.	C
actionId	Unique identifier of the action to be performed	M
href	Indicates where to branch execution if the intended action is a navigation action. The href argument can also be used if the intended action is to execute a native command, call a USAT Command or perform another action as specified in this appendix.	C_1
displayText	Text to be displayed if the desired action is to execute a DISPLAY TEXT	C_1
variableName	Name of variable to set. If this argument is present, either the setvarValue or getInputString is to be supplied. In the case where setvarValue is supplied, as set variable is executed. If getInputString is supplied, the user is asked for input by supplying the string.	C_1
setvarValue	Value to assign to the variable. This argument is to be present only if the setvarName is given.	C_1
getInputString	Text to display to the user when asking for input.	C_1

The principle is to express the range that is to be modified and an action to be performed for that range. The actions that can be used require somewhat different arguments. The arguments having the "C_1"-property are mutually dependent as described above. If the actions are system actions, which means that the actionId is '00' – '03', none of the "C_1" arguments are to be supplied. If the action to be performed is a navigation action, the argument href is used. This attribute is also used for calling USAT Commands and Native Commands as defined elsewhere in this appendix.

The example below will modify the Terminal Response Handler by replacing the action for the general result value of '10' with a call to a USAT Command for setting a new idle mode text. The change is valid for the current card.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/trh/replace?
            start=10&amp;end=10&amp;
            text=Changing%20Idle Mode Text&amp;
            actionId=42&amp;href='efi://vnd.3gpp.interpreter/atk/setIdleModeText?
            text=Welcome'"/>
   </do>
 </p>
</card>
```

In the following example, the same change is applied to the whole deck.

```
<wml>
 <template>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/trh/replace?
             start=10&amp;end=10&amp;
             text=Changing%20Idle%20Mode%20Text&amp;
             actionId=42&amp;href='efi://vnd.3gpp.interpreter/atk/setIdleModeText?
             text=Welcome'"/>
   </do>
 </template>

 <card>
  <p>
   This is the card
  </p>
 </card>
</wml>
```

B.6.2.2 Add

Service name: **trh/add?start=&end=&text=&actionDesc=&actionId=&href=&displayText=&variableName=&setvarValue=&getInputString**

The add operation adds a new action for an existing general result range or defines a new general result range and the corresponding action.

Argument	Argument value	
Start	The start of the general result range that is to be modified	M
End	The end of the general result range that is to be modified	M
Text	Text to display to the user when handling this general result range.	O
ActionDesc	Text to describe the action. To be used in User Interface for select item when asking the user which action to perform when multiple actions are defined for the general result range.	C
ActionId	Unique identifier of the action to be performed	M
Href	Indicates where to branch execution if the intended action is a navigation action. The href argument can also be used if the intended action is to execute a native command, call a USAT Command or perform another action as specified in this appendix.	C_1
DisplayText	Text to be displayed if the desired action is to execute a DISPLAY TEXT	C_1
VariableName	Name of variable to set. If this argument is present, either the setvarValue or getInputString is to be supplied. In the case where setvarValue is supplied, as set variable is executed. If getInputString is supplied, the user is asked for input by supplying the string.	C_1
SetvarValue	Value to assign to the variable. This argument is to be present only if the setvarName is given.	C_1
GetInputString	Text to display to the user when asking for input.	C_1

The principle is exactly the same as for the replace modification.

B.6.2.3 Restore

Service name: **trh/restore?start=&end=&**

The operation restores the general result range.

Argument	Argument value	
Start	The start of the general result range that is to be modified	M
End	The end of the general result range that is to be modified	M

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/trh/restore?
            start=10&amp;end=10”/>
   </do>
 </p>
</card>
```

B.6.2.4 Remove

Service name: **trh/remove?start=&end=&actionId=**

The remove operation removes the specified action from the general result range that is specified.

Argument	Argument value	
start	The start of the general result range that is to be modified	M
end	The end of the general result range that is to be modified	M
actionId	Unique identifier of the action to be performed	M

This service will modify the Terminal Response Handler by removing the action of changing idle mode text for the general result value of ‘10’.

```
<card>
 <p>
  <do type="vnd.3gpp.org">
   <go href="efi://vnd.3gpp.interpreter/trh/remove?
            start=10&amp;end=10&amp;
            actionId=42"/>
   </do>
 </p>
</card>
```

B.~~6~~7 References

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **006** ⌘ rev **-** ⌘ Current version: **5.1.0** ⌘

*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:	⌘	USAT1-Interpr
Date:	⌘	23.01.2002
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:	⌘	4.2: Reference to TS31.114 for clarification. 4.3: The meaning of word "default" was unclear and not consistent. 5.3: Error on attribute bytes definition. 6.1.1.1: Inconsistency between coding of different buffer sizes. 6.1.1.3.1: Unclear access rights of the user information partition. 6.1.3: Inconsistent description of the OTP mechanism. 6.1.3.2: Clarification on read access of temporary variable area. 7.1.7: Missing information of length coding for string pool data. 7.1.8.1: One attribute was only partly defined. Page context definition clarified. 7.1.8.3: Missing information on several user notification texts. 7.2.1: Clarification on attribute handling. 7.3: Clarification on AnchorReference coding. Unclear usage of "#" character. 7.5: Value type information description for InlineValue already somewhere else. 7.9: Correction on PageReference handling. Reference to wrong TLV. 7.9.3.1: UI behaviour described in detail somewhere else in the document. 7.9.3.4: The information was already specified in another document. 8.1: Variable Identifier List TLV not used consistently in SetVariable command. 8.2.3: Clarification on usage of Ordered TLVs command. 8.7.1: Clarification on Attributes for Execute USAT command. Missing '11' case. 8.12: Clarification on parameter handling for GetInput.
Summary of change:	⌘	
Consequences if not approved:	⌘	USAT Interpreter implementations could behave differently.

Clauses affected:	⌘	4.2, 4.3, 5.3, 6.1.1.1, 6.1.1.3.1, 6.1.3, 6.1.3.2, 7.1.7, 7.1.8.1, 7.1.8.3, 7.2.1, 7.3, 7.5, 7.9, 7.9.3.1, 7.9.3.4, 8.1, 8.2.3, 8.7.1, 8.12
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

4.2 Communication with the external system entity

This clause provides an overview of the communication of the USAT Interpreter with the external system entity. ~~overview.~~ The present document describes the format of content exchanged between the external system entity and the USAT Interpreter. The protocol and bearer used for the communication with a USAT Interpreter Gateway System is ~~outside the scope of the present document~~ specified in TS 31.114 [2]. The protocol and bearer used for the communication with other external system entities is out of the scope of the present document.

4.3 Terminal response handler mechanism

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

If several actions are assigned to the~~a~~ general result ~~range~~, 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.

In case of an exception of the USAT Interpreter or no corresponding general result range to the general result of an USAT command, a ~~default~~ specific action will apply. This ~~default~~ action can ~~ould~~ 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).

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

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

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

Default terminal response handler configuration.

Table 4.1

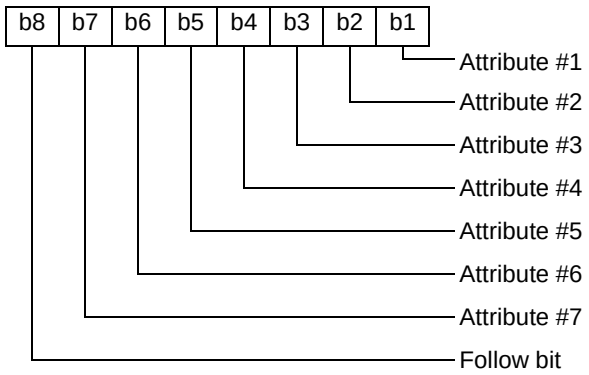
		Action ID	General result range								
			'FF FF' default exception	'14 14' USSD/SS transaction terminated	'00 0F' ok	'13 13' help request	'12 12' no response from user	'11 11' backward move requested	'10 10' quit	'20 2F' worth to re-try	'30 3F' not worth to re-try
System actions	process next byte code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

5.3 Coding of attribute bytes

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

The context, namely the tag, completely determines the order, span and semantics of the bit-packed attribute values. An attribute consisting of more than 1 bit may span two attribute bytes.

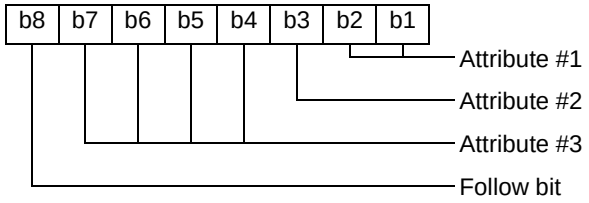
General coding:



Follow bit coding:

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

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



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: nNumber of bytes available for page storage~~., coded in 2 bytes indicating the number of blocks of memory consisting of 128 bytes each, most significant byte first~~
'07'	USAT Interpreter issuer identification	URL of USAT Interpreter issuer, coding according to RFC 1738 [9] <host> of URL.
'08'	Hash Value of URL of USAT Interpreter issuer identification	4 most significant (left most) bytes of SHA-1 hash of the content of variable '07'
'09'	Reception Buffer Sizes	~~First 2 bytes, b~~Binary coded, most significant byte ~~MSB~~ first: ~~-~~Receive buffer size in bytes available for messages to be received by the USAT Interpreter. ~~Second 2 bytes, binary coded, MSB first:~~ ~~- Transmit buffer size in bytes available for messages to be sent by the USAT Interpreter.~~ ~~Both~~This sizes 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.
'0CB'... '13'	RFU	

6.1.1.3.1 Write access to the partition

This area can only be updated ~~locally~~ by the end user. ~~by an user interface provided by the USAT Interpreter. Most likely, the user interface will be realised by a locally stored application of the USAT Interpreter accessible by the end user.~~How this is implemented is out of the scope of the present document.

6.1.3 Temporary variable area

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

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

The current page shall define, which variables are to be kept for access of the following page. To ensure, that only a dedicated following page can access the variables defined to be sharable, the current page may protect them with a One Time Password (OTP)~~., which has to be presented by the following page in order to get access to the shared variables. The OTP to be presented by the following page may be ciphered.~~ The following page shall present a Page U~~u~~nlock ~~code"~~ 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. ~~Protected v~~Variables of ~~a~~the previous page shall only be accessible according to the rules of the table in clause 6.1.3 ~~after a successful verification of the One Time Password set by the previous page to protect temporary variables to be shared between these two pages.~~

In order to unlock the shared protected variables the Page Unlock TLV has to be present within the Page TLV. The Page Unlock TLV shall contain the OTP (ciphered or in clear as indicated by the KIc of the TLV) of the previous page. If the OTP in the Page Unlock TLV matches the OTP stored with the protected variables, the protected variables are made available to the current page as regular temporary variables.

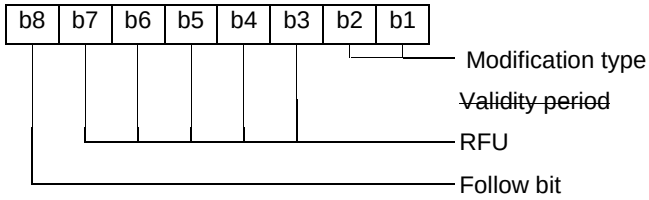
7.1.7 String Pool

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

Coding:

Length	Value	Description	M/O
1	'07'	String Pool Tag	M
1 – 3	L	Length	M
L	Data	LV values of each string element in the string pool with the length L is BER coded onto 1, 2 or 3 bytes according to ISO/IEC 7816-6 [5].	M

7.1.8.1 Attribute



Validity period:

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

- **System context:** In this context the default terminal response handler configuration is valid (see clause 4.3). The default terminal response handler configuration can only be changed by administrative means.
- **Page context:** A terminal response handler modifier within the page context can modify the response handler configuration for the whole page. After leaving the page the modifications done by the terminal response handler modifier of this page are discarded and the terminal response handler configuration of the system context as defined in the paragraph above clause 4.1.2 is restored.
- **Navigation unit context:** A terminal response handler modifier within the navigation unit context can modify the response handler configuration for the navigation unit containing the modifier. After leaving a navigation unit the modifications done by the terminal response handler modifier of this navigation unit are discarded and the terminal response handler configuration of the page context is restored.

7.1.8.3 Text for user notification

This text is displayed by a DISPLAY TEXT command whenever a general result in response to a proactive command is received, that matches the general result range.

If a Terminal Response Handler modifier contains a text for user notification TLV, then it replaces the text for user notification of the current terminal response handler configuration for all the general results within of the Terminal Response Handler modifier general result range.

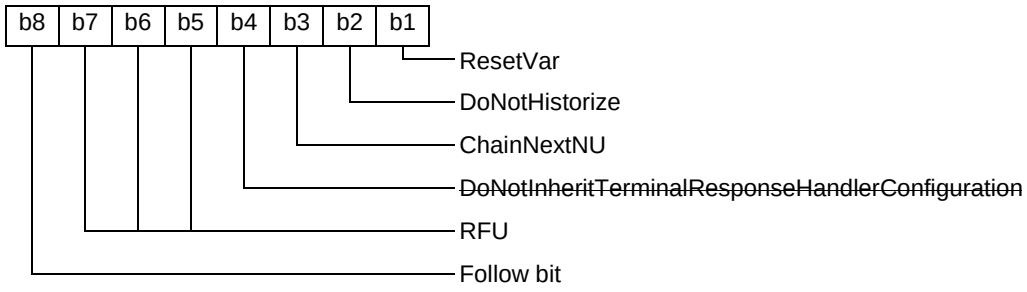
After this DISPLAY TEXT command has been issued by the USAT Interpreter the actions defined for the general result range are to be performed handled regardless of the general result of the DISPLAY TEXT command itself.

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").

7.2.1 Attributes



7.3 Anchor Reference

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

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

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

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.

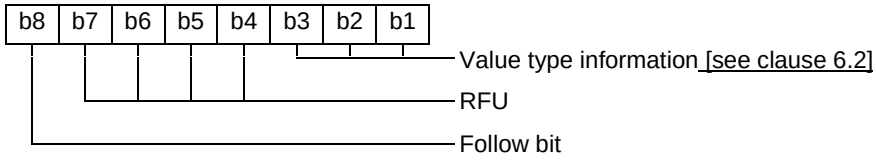
The Inline Value content may contain variable substitution indicators to indicate variable references. Therefore the Inline Value content has to be structured in Length-Value and Variable Substitution Indicator - Variable ID pairs. The possibly available constant data values and variable references have 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 an error according to the following table:

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

Coding of the Inline Value TLV:

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

Coding of the attributes:



7.9 Page Reference

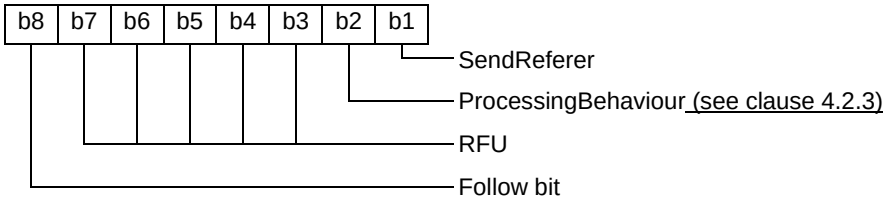
This TLV can represent a page, an anchor within the current page, or an anchor within another page.

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 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

7.9.3.1 Attributes



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

~~If the ProcessingBehaviour attribute is not set the USAT Interpreter shall enter the wait state (see clause 4.2.2) after transmission.~~

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

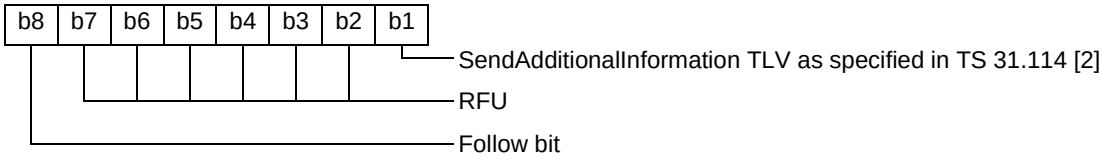
7.9.3.4 Gateway Address

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

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

Length	Value	Description	M/O
1	'15' / '95'	Gateway Address Tag	M
1-3	A+B	Length	M
A	Data	Attributes	O
B	Data	Gateway Address Information	O

Attributes:



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

8.2.3 Ordered TLV List

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

Each of these ~~This~~ TLVs encapsulates~~s~~ 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.

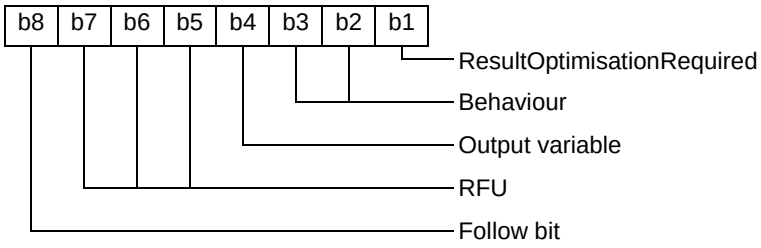
General variable assignments and navigation operations may be performed by the "Assign and Branch" byte code dependent on the data provided in the Ordered TLV List TLV. When optional TLVs are omitted, special cases can be encoded according to the following table:

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 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.

8.7.1 Attributes



8.12 Get Input

[...]

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

3GPP T3 (USIM) Meeting #22
Marbella, Spain, 22-25 January 2002

Tdoc T3-020109
supersedes T3-020105

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** **007** ⌘ rev ⌘ Current version: **5.1.0** ⌘

*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 on behaviour on Single Actions for Terminal Response Handler
Source:	⌘	T3
Work item code:	⌘	USAT1-Interpr
Date:	⌘	24.01.2002
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:	⌘	Behaviour of USAT Interpreter is not specified clearly when only a single action is defined for a general result range.
Summary of change:	⌘	A clarification is made on the process when only one action is assigned to a general result.
Consequences if not approved:	⌘	Possible inconsistent implementations of the USAT Interpreter.

Clauses affected:	⌘	4.3
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

4.3 Terminal response handler mechanism

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

If several actions are assigned to a general result range, a SELECT ITEM command shall be built 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 default action will apply. This default action could be changed by using the terminal response handler modifier with the reserved general result range 'FF FF'.

Exception example:

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

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

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

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

Default terminal response handler configuration.

Table 4.1

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

3GPP T3 #22
Marbella, Spain, 22 – 25 January 2002

Tdoc T3-020129
Revised T3-020114

CR-Form-v3

CHANGE REQUEST

⌘ **31.113** **CR** 008 ⌘ rev **2** ⌘ Current version: **5.1.0** ⌘

*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 security plug-ins
Source:	⌘	T3
Work item code:	⌘	USAT1-Interpr
Date:	⌘	2002-01-25
Category:	⌘	**B**
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:	⌘	To define dedicated Native Code Identities for security plug-ins
Summary of change:	⌘	Native Commands (Plug-Ins) for PKI, Triple-DES and PIN-handling are added
Consequences if not approved:	⌘	

Clauses affected:	⌘	8.8, 9
Other specs affected:	⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications
Other comments:	⌘	

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~~RFU~~. Other values may be used for proprietary implementations.

Possible errors:

Error Code	Description	Action
No error	OK	Continue
Reference to undefined	Reference to undefined	Stop
Jump to undefined	Execute element does not exist	Stop
Problem in memory management	Memory problem in the preparation of the structure	Stop
User Abort	Execute was aborted by user	Stop
Syntax Error	Incorrect number of arguments passed to the execute element.	Stop
Execution Error	Execute element generated an internal error.	Stop

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 ~~RFU for native commands~~ specified in this clause~~ present document~~. Other values may be used for proprietary implementations.

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

Native commands specified by the present document:

NCI	**Name**	**Chapter**
'00 00'	RFU	
'00 01'	P7 – PKCS#7 Signature Plug-In	9.1.1
'00 02'	FP – Fingerprint Plug-In	9.1.2
'00 03'	AD – Asymmetric Decryption Plug-In	9.1.3
'00 04'	DE – Triple DES Encryption Plug-In	9.2.1
'00 05'	DD – Triple DES Decryption Plug-In	9.2.2
'00 06'	DS – Triple DES Sign Plug-In	9.2.3
'00 07'	DU – Triple DES Unwrap Plug-In	9.2.4
'00 08'	CP – Change PIN Plug-In	9.3.1
'00 09'	RP – Reset PIN Plug-In	9.3.2
'00 0A'-'7F FF'	RFU	

~~This is for further study.~~

9.1 PKI Plug-ins

9.1.1 P7 - PKCS#7 Signature Plug-In

9.1.1.1 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, PKCS#7 and CMS.

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

9.1.1.2 NCI

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

9.1.1.3 Arguments

To be defined in REL-6.

9.1.1.4 Output Parameters

To be defined in REL-6.

9.1.1.5 Execution

To be defined in REL-6.

9.1.3.6 Errors

To be defined in REL-6.

9.1.2 FP – Fingerprint Plug-In

9.1.2.1 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, PKCS#7 and CMS.

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

9.1.2.2 NCI

The NCI for this plug-in is ’00 02’.

9.1.2.3 Arguments

To be defined in REL-6.

9.1.2.4 Output Parameters

To be defined in REL-6.

9.1.2.5 Execution

Tobe defined in REL-6.

9.1.2.6 Errors

To be defined in REL-6.

9.1.3 AD – Asymmetric Decryption Plug-In

9.1.3.1 Description

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

9.1.3.2 NCI

The NCI for this plug-in is ’00 03’.

9.1.3.3 Arguments

To be defined in REL-6.

9.1.3.4 Output Parameters

To be defined in REL-6.

9.1.3.5 Execution

To be defined in REL-6.

9.1.3.6 Errors

To be defined in REL-6.

9.2 Triple DES Plug-ins

9.2.1 DE – Triple DES Encryption Plug-In

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.

9.2.1.2 NCI

The NCI for this plug-in is ’00 04’.

9.2.1.3 Arguments

To be defined in REL-6.

9.2.1.4 Output Parameters

To be defined in REL-6.

9.2.1.5 Execution

To be defined in REL-6

9.2.1.6 Errors

To be defined in REL-6.

9.2.2 DD – Triple DES Decryption Plug-In

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.

9.2.2.2 NCI

The NCI for this plug-in is ’00 05’.

9.2.2.3 Arguments

To be defined in REL-6.

9.2.2.4 Output Parameters

To be defined in REL-6.

9.2.2.5 Execution

To be defined in REL-6.

9.2.2.6 Errors

To be defined in REL-6.

9.2.3 DS – Triple DES Sign Plug-In

9.2.3.1 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.

9.2.3.2 NCI

The NCI for this plug-in is ’00 06’.

9.2.3.3 Arguments

To be defined in REL-6.

9.2.3.4 Output Parameters

To be defined in REL-6.

9.2.3.5 Execution

To be defined in REL-6.

9.2.3.6 Errors

To be defined in REL-6.

9.2.4 DU – Triple DES Unwrap Plug-In

9.2.4.1 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 application key stored in the USIM at its own desire.

9.2.4.2 NCI

The NCI for this plug-in is ’00 07’.

9.2.4.3 Arguments

To be defined in REL-6.

9.2.4.4 Output Parameters

To be defined in REL-6.

9.2.4.5 Execution

To be defined in REL-6.

9.2.4.6 Errors

To be defined in REL-6.

9.3 PIN Management Plug-ins

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

9.3.1 CP – Change PIN Plug-In

9.3.1.1 Description

The CP plug-in shall be used to change a PIN to a value specified by the user.

9.3.1.2 NCI

The NCI for this plug-in is ’00 08’.

9.3.1.3 Arguments

To be defined in REL-6.

9.3.1.4 Output Parameters

To be defined in REL-6.

9.3.1.5 Execution

To be defined in REL-6.

9.3.1.6 Errors

To be defined in REL-6.

9.3.2 RP – Reset PIN Plug-In

9.3.2.1 Description

The RP plug-in shall be used to set a PIN remotely.

9.3.2.2 NCI

The NCI for this plug-in is ’00 09’.

9.3.2.3 Arguments

To be defined in REL-6.

9.3.2.4 Output Parameters

To be defined in REL-6.

9.3.2.5 Execution

To be defined in REL-6.

9.3.2.6 Errors

To be defined in REL-6.