Signaling No.7 Scenario Constructing, Analysis, Log-booking and Execution Suit

Version 0.3.1

Application Program Interface Description

Sofia, 2013

Table of Content

Document history

Date	Authors	Doc Rev.	SW Rel.	Subject/Reason for change	Status
2012-11	Ivelin Atanasov	1	0.2.2	Initial version	Complete
2013-03	Ivelin Atanasov	2	0.3.1	New SW release	Complete

1 Decoding messages

1.1 dmtcap

1.1.1 Signature

```
int dmtcap(int child, struct TCAPmsu* ptcm, int msuLEN, uint8_t msuBUF[], int s7AP, void* pvoid);
```

1.1.2 Arguments

`struct TCAPmsu` – an empty structure that is filled in by `dmtcap`

`msuBUF` – message buffer, containing an MTP3 message to be decoded

`msuLEN` – length of the MTP3 message

`s7AP` – SS7 Application Part that is needed in some cases

`pvoid` – a void pointer used to address TCAP session storage (`TCses_stg`) or to carry AC code, depending on the value of `s7AP`.

1.1.3 Description

This function performs ASN.1 syntax decoding of a TCAP message as well as semantic decoding of any Application Part down to Operation Argument Tag. It takes an empty `TCAPmsu` object and fills in this object as much as possible. It creates a complete `thisIE` struct for all operations in the message, incl. `FCI`, `ACR`, etc, which bear some peculiarity.

For any message that has a Dialogue Portion this function sets `TCAPmsu.s7ap` based on the Dialogue Portion, no matter if message is received or is to be sent and no matter if `s7AP` argument is set or not.

For a received message w/o a Dialogue Portion, the `TCAPmsu.s7ap` and `TCAPmsu.accod` are taken from an appropriate `TCAPses` object in the `TCses_stg[]` that is chosed on the basis of `OTID`/`DTID` of the message. Note that `dmtcap` determines the TCAPsesion structure the message belongs to, but does not write anything there.

For messages to be sent that have no Dialogue Portion, `TCAPmsu.s7ap` is provided by the argument `s7AP` and must be `!=0`. The `TCAPmsu.accod` is required in this case and is provided in `pvoid`, which is explicitly cast to integer if `s7AP!=0`.

To ignore `TCses_stg`, use `pvoid=NULL` as an argument ; to ignore `s7AP`, set it to `0` (`UAP_ID`), but never ignore both of them.

1.1.4 Return codes

If `dmtcap < 0`, there is a severe problem that makes impossible to decode the message in full (usually length missmatch).

If `dmtcap > 0` there is a problem that may need further analysis. In this case you should investigate `TCAPmsu.cm[i].rcod` that contains the return code, associated with that particular Component.

If `dmtcap==0` the decoding has passed without any problem.

This function produces another set of return codes that is written in the TCAPmsu object - `.dp.rcod` and `.cm[i].rcod`. These return codes address Dialogue portion and Components. In addition, each I.E. in a Component or Dialogue portion has a return code saved in `thisIE[Rc]`.

1.1.5 Relations to other functions

This function is related to all functions that explore or modify TCAP messages.

2 Exploring messages

2.1 dmgsie

2.1.1 Signature

```
int dmgsie(int* opi, int* idx, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int
tie_ID);
```

2.1.2 Arguments

`int* opi` – an index of the Operation in the set of all Operations in the message

`int* idx` – an index of the I.E. in the dynamic `thisIE` structure of the Operation

`struct TCAPmsu* ptm` - the message

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

2.1.3 Description

At first, this function retrieves the index of the Operation in the array `ptm.cm[]`,identified by `opcod` and places it in the argument `opi`. Then it retrieves the index of the I.E., identified by `tie_ID` within the `thisIE` structure addressed by `ptm.cm[opi].ie_p` (the one, associated with the operation `opcod`) and places it in the argument `idx`. This is done by mapping the Static thisIE structure associated with the Operation `opcod` to the Dynamic thisIE structure of the same Operation in TCAPmsu.

This function is applicable if the I.E. has only one appearance in the Operation.

2.1.4 Return codes

`dmgsie < 0` that the Static or Dynamic structures are bad or the Operation is not found in the message.

`dmgsie > 0` means that the I.E. sought is not found. It could be because it is not present in the Dynamic `thisIE` structure. This happens also when the Target I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

`dmgsie == 0` means that no errors or warning are encountered

2.1.5 Relations to other functions

This function is used in relation with `getptr_ieval` and `get2ptr_ieval` that retrieves a pointer to the Value of the I.E. in the message.

2.2 dmgmie

2.2.1 Signature

```
int dmgmie(int* opi, int* pidx, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int
tie_ID, int oie_ID, int cie_ID, struct XUBuffer* pxcnd)
```

2.2.2 Arguments

`int* opi` – an index of the Operation in the set of all Operations in the message

`int* pidx` – an index of the I.E. in the dynamic `thisIE` structure of the Operation

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int oie_ID` – the name of the Owner I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int cie_ID` – the name of the Conditiont I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`struct XUBuffer* pxcnd` – a pointer to the Value of the Condition I.E. that has to be met

2.2.3 Description

This function is used to retrieve the indexes of an I.E. and an Operations under more complex conditions compared to `dmgsie`. The I.E. may have multiple appearance in the Operation and the one that is sought is identified by possible another I.E. and its Value. Both I.E.s need to have the same Owner and in the context of this Owner they are unique.

At first, this function retrieves the index of the Operation in the array `ptm.cm[]`, identified by `opcod` and places it in the argument `opi`. Then it retrieves the index of the I.E., identified by `tie_ID` within the `thisIE` structure addressed by `ptm.cm[opi].ie_p` (the one, associated with the operation `opcod`) and places it in the argument `pidx`. This is done by mapping the Static thisIE structure associated with the Operation `opcod` to the Dynamic thisIE structure of the same Operation in TCAPmsu and checking the Value of the Condition I.E. to the Value provided by `pxcnd`.

2.2.4 Return codes

`dmgmie < 0` means that the Static or Dynamic structures are bad or the Operation is not found in the message.

`dmgmie > 0` means that the I.E. sought is not found. It could be because it is not present in the Dynamic `thisIE` structure or the Value of Condition I.E. is not found in the message. This happens also when the Target I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

`dmgmie == 0` means that no errors or warning are encountered

2.2.5 Relations to other functions

This function is used in relation with `getptr_ieval` and `get2ptr_ieval` that retrieves a pointer to the Value of the I.E. in the message.

3 Modifying messages

3.1 dmnew

3.1.1 Signature

```
int dmnew(int child, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int tie_ID, struct XUBuffer* pxnew)
```

3.1.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`struct XUBuffer* pxnew` - a pointer to the new Value of the Target I.E. that has to placed in the message

3.1.3 Description

This function sets new value for an I.E. identified by `tie_ID` in an Operation identified by `opcod`. The I.E. has to be unique within the Operation. If old and new values have the same length, message reconstruction is by-passed and the process is very fast. If the lengths are different, a temporary copy of `ptm` is created in the Heap and all modifications are done on it. At the end, it is copied onto `ptm`.

The temporary copy is freed from the Heap.

3.1.4 Return codes

`dmnew < 0` means that the Static or Dynamic structures are bad, or the Operation is not found in the message, or it is not possible to expand raw message buffer (`TCAP.msu.pbuf`) because the limit of this buffer, controlled by `BUFSIZE` will be exceeded when modification is done, or there is no enough room in the Heap. It is also possible to have unsuccessful result from underlying functions, for example `doGetLngITAG`, which constructs Tags of the I.E.s along the path from the root down to the Target I.E.

`dmnew > 0` means that the Target I.E. is not found in the Dynamic thisIE structure associated with the Operation. This happens also when the Target I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

3.1.5 Relations to other functions

If applied to a message template, the template must be cloned first in the Heap by `cloneTCAPmsu`.

3.2 dmadd

3.2.1 Signature

```
int dmadd(int child, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int tie_ID, int
oie_ID, struct XUBuffer* pxnew, int UNIQ)
```

3.2.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int oie_ID` – the name of the Owner of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`struct XUBuffer* pxnew` - a pointer to the I.E. to be inserted into the message. This means a sequence of bytes that present the I.E. in TLV format

`int UNIQ` - the determines the mode of insertion: unique – only one appearance of the I.E. is allowed in this Operation and multiple – many appearances are allowed. In this case the new I.E. is inserted before the existing I.E.s of the same type.

3.2.3 Description

This function inserts new I.E. identified by `tie_ID` in an Operation identified by `opcod`. A temporary copy of `ptm` is created in the Heap and all modifications are done on it. At the end, it is copied onto `ptm`.

The temporary copy is freed from the Heap.

Message templates after being cloned expand the length of thisIE arrays by `OPPARMADD8`. Therefore, in a single run a message can be expanded by up to `OPPARMADD8` times before special measurements needs to be taken. This is very unlikely to happen.

3.2.4 Return codes

`dmadd < 0` means that the Static or Dynamic structures are bad, or the Operation is not found in the message, or it is not possible to expand raw message buffer (`TCAP.msu.pbuf`) because the limit of this buffer, controlled by `BUFSIZE` will be exceeded when modification is done, or the Dynamic thisIE structure for the Operation cannot be expanded or there is no enough room in the Heap. It is also possible to have unsuccessful result from underlying functions, for example `doGetLngITAG`, which constructs the Tags of the I.E.s along the path from the root down to the Target or Owner I.E.

`dmadd > 0` means that the Owner I.E. is not found in the Dynamic thisIE structure associated with the Operation. This happens also when the Owner I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

3.2.5 Relations to other functions

If applied to a message template, the template must be cloned first in the Heap by `cloneTCAPmsu`.

3.3 dmdel

3.3.1 Signature

```
int dmdel(int child, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int tie_ID)
```

3.3.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

3.3.3 Description

This function deletes an I.E. from a message. If more than one I.E.s of the that type are present in the message, the first I.E. will be deleted.

3.3.4 Return codes

`dmdel < 0` means that the Static or Dynamic structures are bad, or the Operation is not found in the message. . It is also possible to have unsuccessful result from underlying functions, for example `doGetLngITAG`, which constructs the Tags of the I.E.s along the path from the root down to the Target I.E. Creating a temporary copy in the Heap may lead to an error if there is no enough space there.

`dmdel > 0` means that the Target I.E. is not found in the Dynamic thisIE structure associated with the Operation. This happens also when the Target I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

3.3.5 Relations to other functions

If applied to a message template, the template must be cloned first in the Heap by `cloneTCAPmsu`.

3.4 dmcnd

3.4.1 Signature

```
int dmcnd(int child, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int tie_ID, int
oie_ID, int cie_ID, struct XUBuffer* pxnew, struct XUBuffer* pxcnd)
```

3.4.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int oie_ID` – the name of the Owner of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int cie_ID` – the name of the Condition I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`struct XUBuffer* pxnew` - a pointer to the new Value of the Target I.E. that has to placed in the message

`struct XUBuffer* pxcnd` - a pointer to the Value of the Condition I.E. that has to be found in the message

3.4.3 Description

This function modifies the Value of an I.E. of a type for which multiple I.E.s are allowed in the Operation. It is identified by its name and by the Value of the Condition I.E. – another I.E. that goes together with the Target I.E, and by the Owner I.E. – an I.E. that always embraces a Target and a Condition I.E.s and is repeated for each set of Target and Condition I.E. presented in the message.

A temporary copy of `ptm` is created in the Heap and all modifications are done on it. At the end, it is copied back onto `ptm`.

The temporary copy is freed from the Heap.

3.4.4 Return codes

`dmcnd < 0` means that the Static or Dynamic structures are bad or the Operation is not found in the message. It is also possible to have unsuccessful result from underlying functions, for example `doGetLngITAG`, which constructs the Tags of the I.E.s along the path from the root down to the Target or Condition or Owner I.E. Creating a temporary copy in the Heap may lead to an error if there is no enough space there.

`dmcnd > 0` means that the I.E. sought is not found. It could be because it is not present in the Dynamic `thisIE` structure or the Value of Condition I.E. is not found in the message. This happens also when the Target I.E. or Condition I.E. or Owner I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

3.5 dmgnb

3.5.1 Signature

```
int dmgnb(int child, struct TCAPmsu* ptm, int opcod, struct XArray* pst, int tie_ID, int
oie_ID, struct XUBuffer* pxnew, uint8_t qual)
```

3.5.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`struct XArray* pst` – static `thisIE` structure of the Operation

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int oie_ID` – the name of the Owner of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`struct XUBuffer* pxnew` - a pointer to the new Value of the Target I.E. that has to placed in the message

`uint8_t qual` - a qualifier taken from the `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

3.5.3 Description

This function modifies the Value of an I.E. of a type Generic Number or Generic digit with a given qualifier. It is identified by its name and by the value of the qualifier, which is also located in `xxxxopdefs.h`. The Owner I.E. in this case is of type Generic Number or Generic Digits. The value to be placed in the message has to be prepared upfront.

A temporary copy of `ptm` is created in the Heap and all modifications are done on it. At the end, it is copied back onto `ptm`.

The temporary copy is freed from the Heap.

3.5.4 Return codes

`dmgnb < 0` means that the Static or Dynamic structures are bad or the Operation is not found in the message.

`dmgnb > 0` means that the I.E. sought is not found. It could be because it is not present in the Dynamic `thisIE` structure or the Value of Condition I.E. is not found in the message. This happens also when the Target I.E. is not found in the Static thisIE structure for this Operation. Note that if the Static thisIE structure were complete, including all I.E. that the Operation may have, this should be treated as an error (< 0).

3.5.5 Relations to other functions

If applied to a message template, the template must be cloned first in the Heap by `cloneTCAPmsu`. This function can is used by `dmfci`, that has been developed especially to deal with the FCI peculiarities. It is also related to the `mod73gen_any_int` that modify various telephony numbers from Unknown or National (Significant) Number format to international one.

3.6 dmfci

3.6.1 Signature

```
int dmfci(int child, struct TCAPmsu* ptm, int tie_ID, int oie_ID, int newlen, uint8_t* newval,
uint8_t qual)
```

3.6.2 Arguments

`struct TCAPmsu* ptm` - the message itself

`int opcod` – the name of the Operation, taken from `xxxxbase.h` (`xxx=map_, cs1p`, …)

`int tie_ID` – the name of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int oie_ID` – the name of the Owner of the Target I.E. taken from `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

`int newlen – the length of the new value`

`uint8_t* newval` - a pointer to the new value of the Target I.E. that has to placed in the message

`uint8_t qual` - a qualifier taken from the `xxxxopdefs.h` (`xxx=map_, cs1p`, …)

3.6.3 Description

This function packs `newlen` and `newval` into `struct XUBuffer` and calls `dmgnb`.

3.6.4 Return codes

It passes return codes from `dmgnb`.

3.6.5 Relations to other functions

If applied to a message template, the template must be cloned first in the Heap by `cloneTCAPmsu`. It is also related to the `mod73gen_any_int` that modify various telephony numbers from Unknown or National (Significant) Number format to international one.

3.7 mod73gen_any_int

3.7.1 Signature

```
static int mod73gen_any_int(int* plen, uint8_t* vnew, uint8_t* vold, uint8_t gentyp)
```

3.7.2 Arguments

`int* plen` – pointer to the length of the array containing the number

`uint8_t* vnew` – pointer to the array containing the modified value

`uint8_t* vold` – pointer to the array containing the initial value

`uint8_t gentyp` – generic type, taken from xxxxopdefs.h (xxx=map_, cs1p, …)

3.7.3 Description

This functon converts any number I.E. into Generic number of type `gentyp` under the following condition:

- number type is 7-bit field in the `vold[0]`,

- we place 3-digit Country Code `359` for Bulgaria

The resulting Generic number is of type `gentyp` and the number is in International format.

3.7.4 Return codes

`mod73gen_any_int < 0` if the number addressed by vold is too short or if the number type is not International, National (Significant) or Unknown Number.

3.7.5 Relations to other functions

`dmgnb, dmfci.`

4 Dealing with sessions

`TCAPses` structure is defined in header file `tcapbase.h`.

4.1 `addRAARQ_sesTCAPremi`

4.1.1 Signature

```
int addRAARQ_sesTCAPremi(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.1.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.1.3 Description

This function fills in ‘remote’ Dialogue and Component portions data for a remotely initiated Session based on a TCAP message that is received from the remote end. It is expected a `BEGIN` with `AARQ APDU`.

The `TCAPses` structure `ptcs`, must have been seized by `seize_sesTCAP()` upfront.

`TCAPses.rSPC`, `TCAPses.rCallPty` and `TCAPses.lCallPty` are set as per OPC and Call Parties in the received message `ptcm`. They can be changed by `setSPC_sesTCAPalli` and `setGT_sesTCAPalli`.

Invoke IDs are set from 0 and incremented by 1.

It is advised that the reader goes through the comments in the function body.

4.1.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.1.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addLOCAL_sesTCAPalli`, `addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli`.

4.2 `addLAARE_sesTCAPremi`

4.2.1 Signature

```
int addLAARE_sesTCAPremi(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.2.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.2.3 Description

This function fills in the ‘local’ Dialogue and Component portions data for a remotely initiated Session based on a message that is to be sent to the remote end. It is expected to be a `CONTINUE` or `END` with `AARE APDU`.

Note that SPCs and GTs are already set by `addRAARQ_sesTCAPremi`, but may need to be modified by `setSPC_sesTCAPalli` and `setGT_sesTCAPalli` before calling this function; here they are filled in TCAP message to be sent (`ptcm`) as they are set already.

Note also that the local TID is determined automatically.

4.2.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.2.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addLOCAL_sesTCAPalli`, `addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli`.

4.3 `addLAARQ_sesTCAPloci`

4.3.1 Signature

```
int addLAARQ_sesTCAPloci(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.3.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.3.3 Description

This function creates and fills in Dialogue and Component portions data in a locally-initiated Session based on a message template or cloned message that is going to be send to the remote end. SPCs as well as local and remote CallParties should be set already by the calling process. Originating TID and starting invokeID are determined here automatically.

After that the message to be sent is synchronized with the info in the Session (CallParties, etc).

Note that a message with AARQ APDU is expected.

Note that `lSPC`, `rSPC`, `lCallPty` and `rCallPty` are set already.

4.3.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.3.5 Relations to other functions

`setSPC_sesTCAPalli, setGT_sesTCAPalli, addRAARQ_sesTCAPremi, addLAARE_sesTCAPremi, addRAARE_sesTCAPloci, addREMOT_sesTCAPalli, addLOCAL_sesTCAPalli, addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli.`

4.4 `addRAARE_sesTCAPloci`

4.4.1 Signature

```
int addRAARE_sesTCAPloci(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.4.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.4.3 Description

This function fills in Dialogue and Component portions data in a locally-initiated Session based on the first response message from the remote end. The remote CallPty and the remote TID are also set here.

Note that AARE APDU is expected.

rSPC may have to be modified by setSPC_sesTCAPalli to match the new rCallPty after return from this function.

4.4.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.4.5 Relations to other functions

`setSPC_sesTCAPalli, setGT_sesTCAPalli, addRAARQ_sesTCAPremi, addLAARE_sesTCAPremi, addLAARQ_sesTCAPloci, addREMOT_sesTCAPalli, addLOCAL_sesTCAPalli, addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli.`

4.5 addREMOT_sesTCAPalli

4.5.1 Signature

```
int addREMOT_sesTCAPalli(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.5.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.5.3 Description

This function fills in a Component portion data in a Session for a CONT/END message that has been received from the remote end. It is expected that no APDU is available in the message.

Note that SPCs and CallPtys are not changed.

4.5.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.5.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addLOCAL_sesTCAPalli`, `addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli`.

4.6 addLOCAL_sesTCAPalli

4.6.1 Signature

```
int addLOCAL_sesTCAPalli(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.6.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.6.3 Description

This function fills in the Component portion data for a message in a Session that is to be sent to the remote end. It is expected expect here a CONTINUE or END with no APDU.

Note that SPCs and CallPtys are not changed in this function!

4.6.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.6.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addRABRT_sesTCAPalli` and `addLABRT_sesTCAPalli`.

4.7 addRABRT_sesTCAPalli

4.7.1 Signature

```
int addRABRT_sesTCAPalli(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.7.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.7.3 Description

This function fills in the ‘remote’ ABRT data for a Session based on a message containing ABRT PDU that has been received from the remote end.

4.7.4 Return codes

This function returns 0.

4.7.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addLOCAL_sesTCAPalli` and `addLABRT_sesTCAPalli`.

4.8 `addLABRT_sesTCAPalli`

4.8.1 Signature

```
int addLABRT_sesTCAPalli(int child, struct TCAPses* ptcs, struct TCAPmsu* ptcm)
```

4.8.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`struct TCAPmsu* ptcm` – the message itself

4.8.3 Description

This function fills in the ‘local’ ABRT data for a Session based on a message containing ABRT PDU that is to be sent to the remote end.

4.8.4 Return codes

This function returns 0.

4.8.5 Relations to other functions

`setSPC_sesTCAPalli`, `setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addLOCAL_sesTCAPalli` and `addRABRT_sesTCAPalli`.

4.9 `setSPC_sesTCAPalli`

4.9.1 Signature

```
int setSPC_sesTCAPalli(int child, struct TCAPses* ptcs, uint32_t rSPCdft, uint32_t lSPCdft,
int OVRD)
```

4.9.2 Arguments

`struct TCAPses* ptcs` – the session associated with the message

`uint32_t rSPCdft` – default remote SPC

`uint32_t lSPCdft` – default local SPC

`int OVRD` – mode of remote SPC setting

4.9.3 Description

This function sets rSPC and lSPC for the session, pointered by ptcs. There are two modes:

- Override (`OVRD=1`): put rSPCdft always, no matter of GT indicators of the rCallPty, and

- No override (`OVRD=0`): set rSPC, depending on the GT indicators of the rCallPty.

In `OVRD=0`, `rSPC` is set as follows:

- If `rCallPty` is available, and if `rCallPty.indSPC` and `rCallPty.indRtg` are set, then put `rCallPty.SPC` as `rSPC`;

- If `rCallPty` is available, and if `rCallPty.indSPC` is set but `rCallPty.indRtg` isn't, use `rSPCdft`

- If `rCallPty` is available and if `rCallPty.indRtg` is set but `rCallPty.indSPC` isn't, or if `rCallPty` isn't available, put for `rSPC` the remote SPC taken from the Routing label of the last processed message

Routing label of the last processed message means:

- Take the Routing label from the 2nd message in the session, if available

- If there is no 2nd message in the session, take the Routing label from the 1st message in the session and set `rSPC` accordingly.

4.9.4 Return codes

This function returns either 0 (success) or <0 (failure).

4.9.5 Relations to other functions

`setGT_sesTCAPalli`, `addRAARQ_sesTCAPremi`, `addLAARE_sesTCAPremi`, `addLAARQ_sesTCAPloci`, `addRAARE_sesTCAPloci`, `addREMOT_sesTCAPalli`, `addLOCAL_sesTCAPalli` and `addRABRT_sesTCAPalli`.

5 Other useful functions

You are advised to read the comments in the following functions: `move_sccp_tcap`, `move_tcap_sccp`, `put2udt_QS`, `get_sccpQR`, `gen_sccpGT15`, `get_sccpGT`, `sendrecv_stu`, `sendrecv_stu` and `PrintPacketInfo`.

Note that the named constants for `PrintPacketInfo` are defined in `sctpbase.h`.

Pay attention to `put2udt_QS` – it takes information from the structure `sccp_udt` and places this information in the raw message. Keep in mind also that the array of chars you pass to `gen_sccpGT15` should be at least 15 bytes.