CEPTEST Application Note

Version 1.0

Running Stress Tests

CELSI AG
Museumstr. 76
CH-8400 Winterthur
Tel. 052 212 63 03
Fax 052 212 66 78
www.celsi.ch

1 Introduction

This document is a step by step description, how to set up a stress test with CEPTEST. The goal is to set up the following stress test:

- Send 10 parallel authorisation requests from different terminals and cards to a server with chip, magstripe and manual key entry as pos entry mode
- Send Reversals for these requests.
- Send again 10 requests
- Add 10 off-line authorisations
- Send submissions by terminal for these transactions
- Send again 10 requests
- Add 10 off-line authorisations
- Send a PMS-submission file for these requests
- Repeat the above for 10 times

Note that CEPTEST allows up to 100 parallel requests. The limit here with 10 is just to have shorter sample files.

The intention is to use this example to provide all information needed to implement a stress test according to your needs.

2 Preparing the Server

The server system must be prepared with terminal and card data for 10 terminals and cards. Terminals should have different component secrets. Card data may use on-line PIN verification or off-line PIN verification.

3 Basic CEPTEST Parameters

This chapter shows the parameters needed for setup of the views found in menu item [Parameters]

3.1 Menu [Parameters, Keys...]

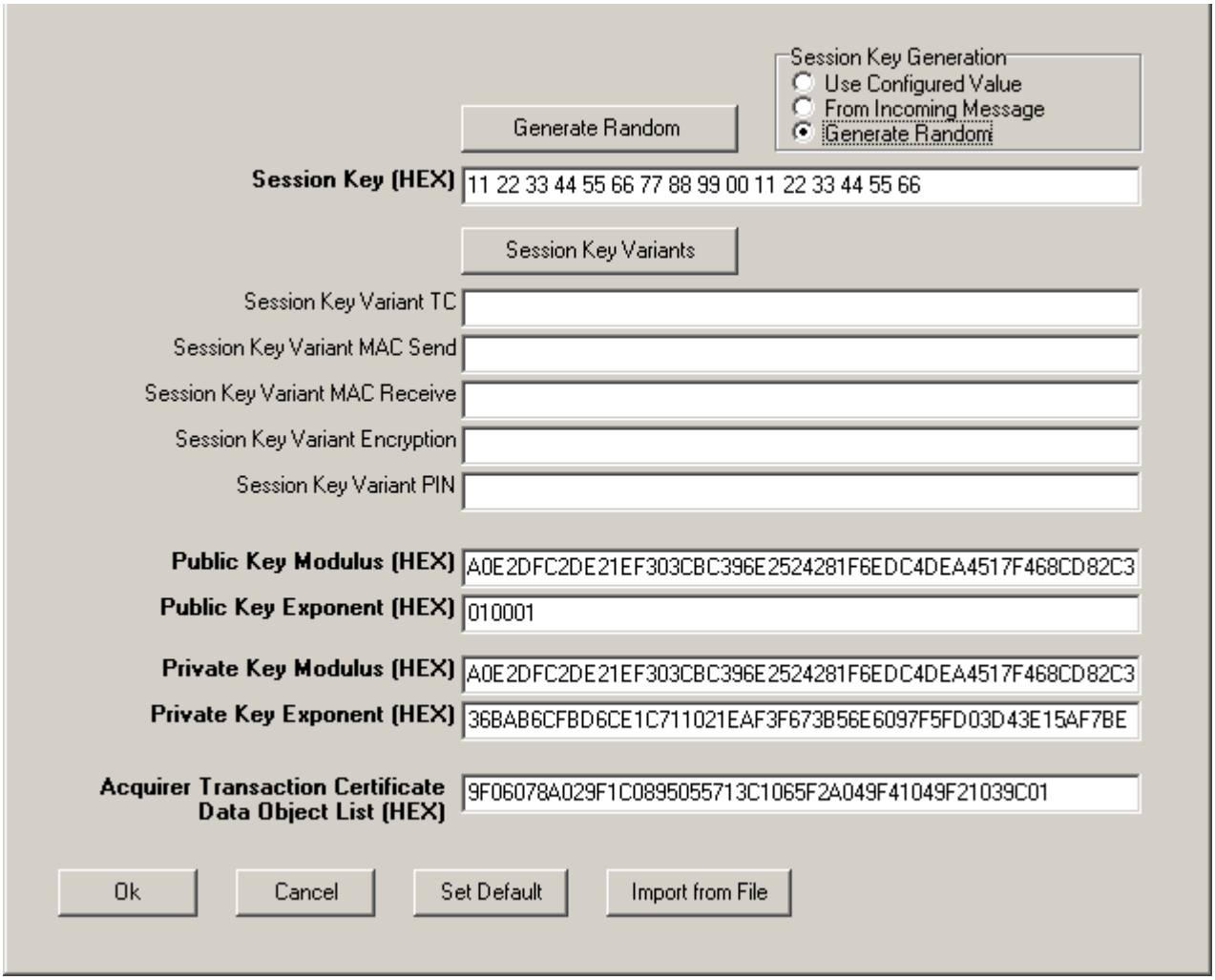

Figure 1 Menu [Parameters, Keys...]

- The Session key shall be generated at random
- The Public Key Modulus and Exponent shall be those of the server. If you have a formatted key on a file, click on the button 'Import from File' to import it.
- The Private Key will not be used and may be left empty.
- The Acquirer Transaction Certificate Data Object List will be needed for data submission.

3.2 Menu [Parameters, Communication...]

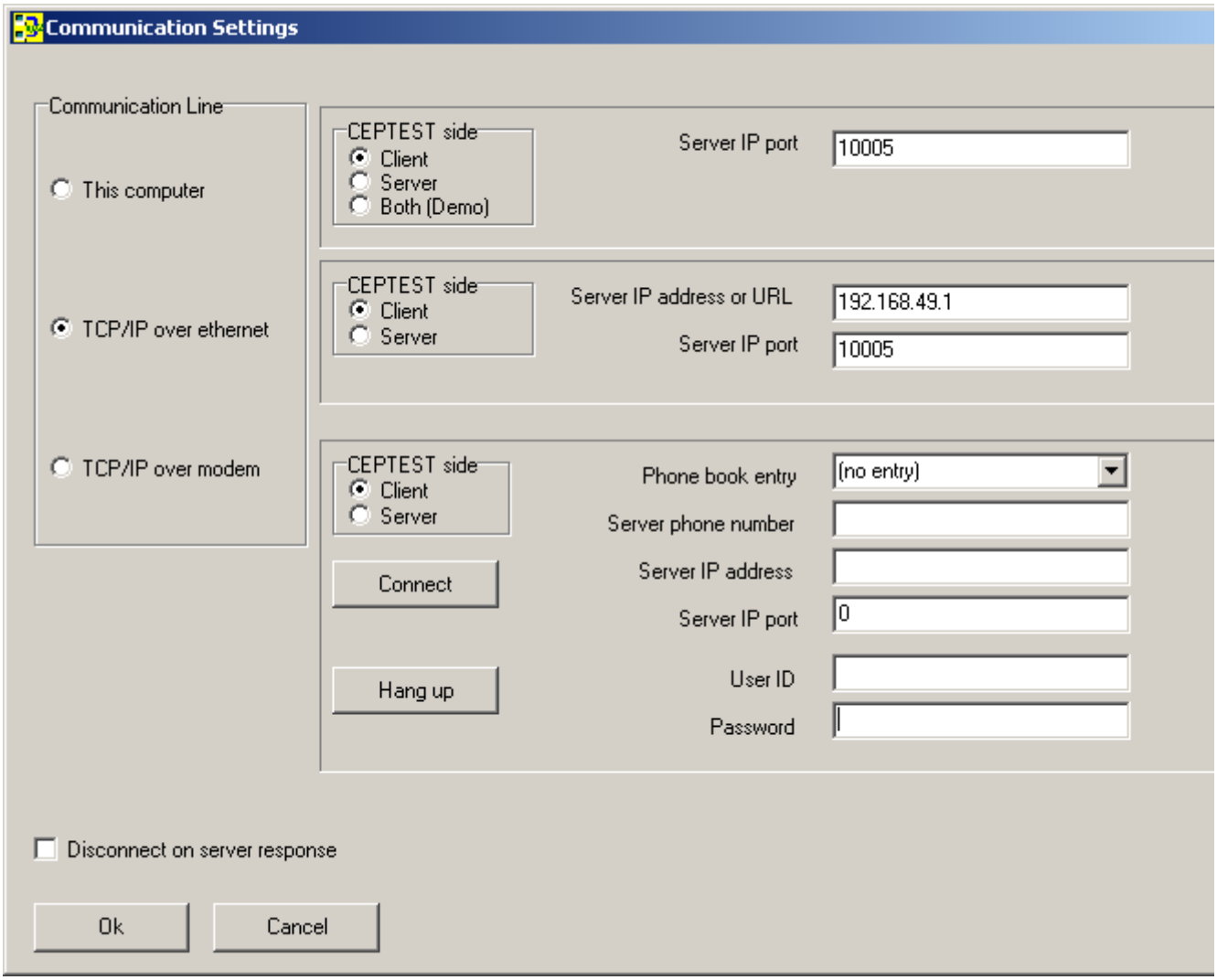

Figure 2 Menu [Parameters, Communication...]

- This example shows TCP/IP communication over ethernet. CEPTEST has to be set as client.
- The check box 'Disconnect on server response' should not be set. If this check box is set, CEPTEST will close a communication line after each server message. But it is more flexible to decide later when to close the communication line.

3.3 Menu [Parameters, Message Header...]

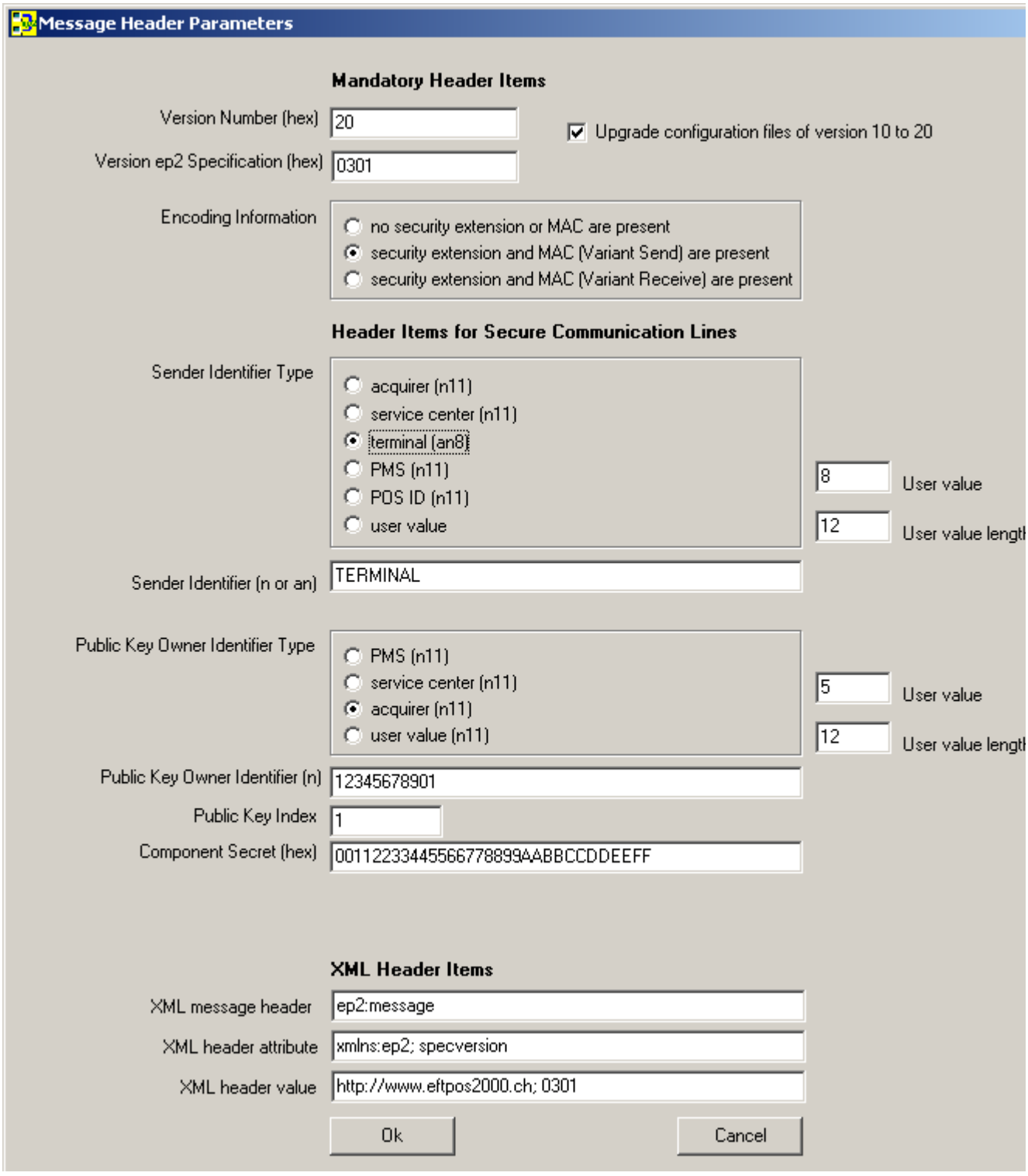

Figure 3 Menu [Parameters, Message Header...]

- Encoding Information shall be set to ’security extension and MAC (Variant Send) are present’ since CEPTEST is simulating a terminal or a PMS, using session key variant MAC Send for MAC generation.
- Sender Identifier Type and Sender ID need to be varied according to the different Terminal and PMS, which we need for our stress test. So this value is not used. We will see later, how the sender ID is taken from the message data instead of the configured value.
- Public Key Owner ID Type and Public Key Owner ID, on the other side, are the same for all messages. Select ’acquirer’ for Public Key Owner ID Type and enter the Public Key Owner ID of the host. Do the same for the public key index.
- The Component Secret is also different for each terminal and it will be set in the message.

3.4 Menu [Parameters, Timeouts...]

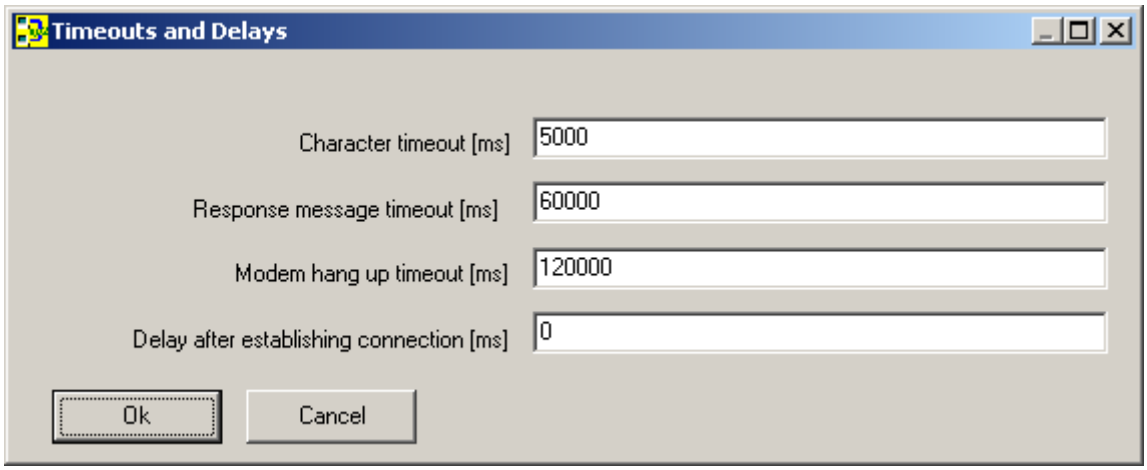

Figure 4 Menu [Parameters, Timeouts...]

These timeouts are ok in a wide range and the above values should do. The only point is that ’Delay after establishing connection’ should be set to 0. Otherwise, CEPTEST holds its message back for the time value, which makes it easier to generate overload situations, but slows down our stress test here.

3.5 Menu [Parameters, Plug Ins...]

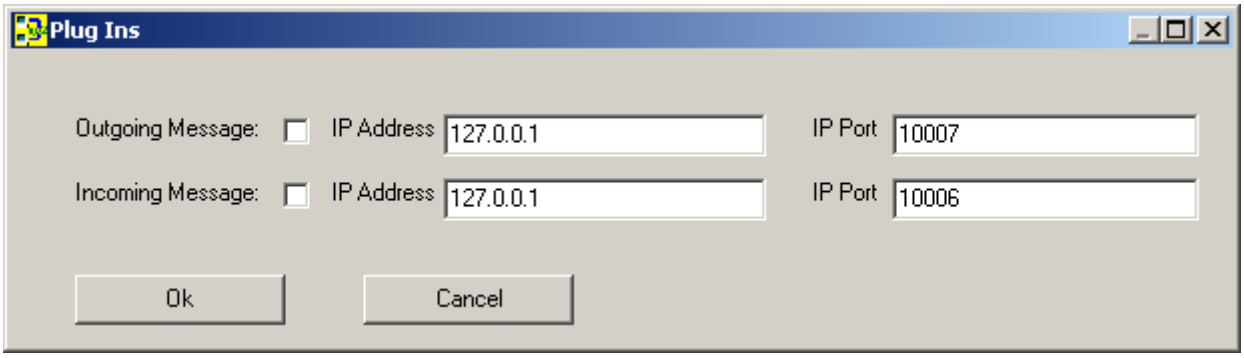

Figure 5 Menu [Parameters, Plug Ins...]

Plug Ins are user supplied applications, which compute proprietary data elements such as ICC transaction certificates or issuer authentication data. If you use such applications for message generation (e.g. ICC transaction certificate), click the check box 'Outgoing Message' and enter the TCP/IP address and port data, where the application listens to. See the on-line help for details. If you use such applications for checking incoming messages from the server, click on the check box 'Incoming Message' and enter the TCP/IP address.

3.6 Menu [Parameters, Directory Aliases...]

Aliases are names for directories, used for easier exchange of CEPTEST files with other users.

Before using this menu, we need to decide, where we want to store files. In our example, we create a directory K:\ceptest and 3 subdirectories:

- Subdirectory 'messages', where we store all message files
- Subdirectory 'logs', where log information shall be written during our stress test
- Subdirectory 'Value pools', where we store value files for elements, which change their value from one message to the next

You are free to organize files differently and create as many aliases as needed, provided, all users of shared files use the same subdirectories, which, however, may have different locations. If this is the case, other users need only to change aliases to the different locations and may use the shared files.

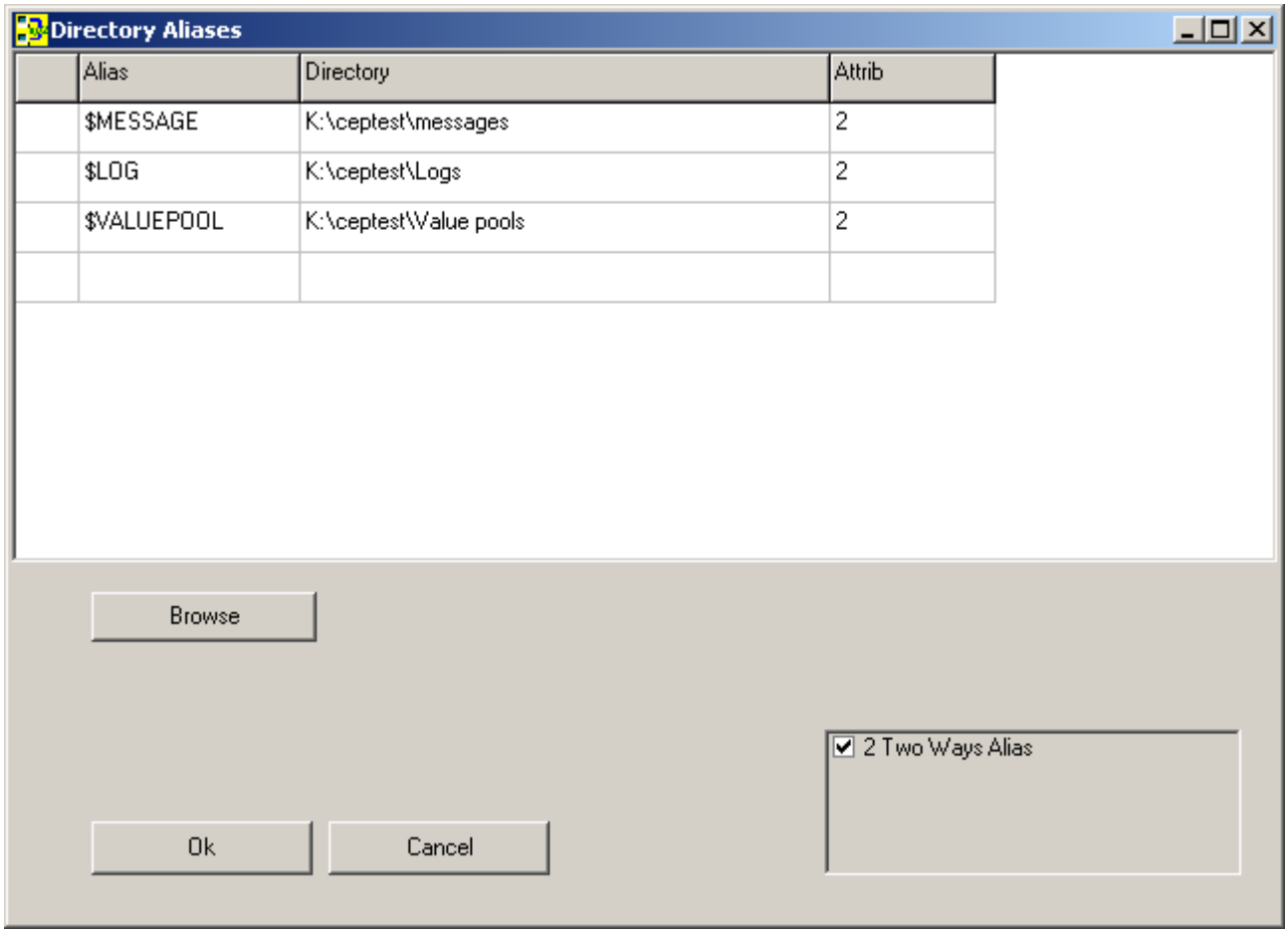

Figure 6 Menu [Parameters, Directory Aliases...]

- Set the Two Ways Alias flag if you want to have CEPTEST replace directory locations by aliases while saving files. One way aliases only replace aliases by directory locations while loading files. In most cases, two way aliases are the right choice, except if you want to keep the location data as is without replacing it by an alias.
- Tip: If you load and save existing files which were created without alias (e.g. with older CEPTEST versions), the files will be stored with alias information if you use Two Ways Aliases

4 Value Pools

In this step, we want to set values of transaction data, that are not equal for all transactions.

There are 2 ways to vary the data of a transaction:

- Automatic Update: Values like Transaction Date, Time or Sequence Counters can be marked for automatic update by CEPTEST during message definition (see below)
- Value Pools: This are text files, which contain the possible values of an element, e.g. Terminal Identification. This chapter handles this type of elements.

Some of the value pools affect only one data element of the message. Others value pools contain several elements, which need to be used in the same request message. Example: Track 2 Data and On Line PIN.

Depending on the type of a request (ICC, magstripe...) elements are also present or not.

4.1 Amount Authorised

Assuming, we want to have different amounts for authorisations, we generate a value pool file with one value per line. The number of entries is not relevant, since we will use the values randomly. Example:

1000
2000
4000
4100
4200
100
2345
1600
4300
5600
6620
12300
200000

It is a good idea to have all values above floor limit of all terminals. Otherwise, Amount Authorised needs to be set with the other transaction parameters below.

If other data elements shall be varied, similar value pools can be created.

Since we decided to have also off-line authorised transactions, we may need a second value pool file for small amounts, which looks similar to the above.

All value pool files shall be stored in the directory, which is defined by the alias $VALUEPOOL.

4.2 Terminal Identification and Component Secret

We have to consider, that Terminal Identification and component secret are closely related. In our example, we need (at least) 10 terminals defined by Terminal Identifications and their Component Secret. We create a text file of the form: 'Terminal Identification';'Component Secret'.

Example:

TERM0001; 1E976701E4866EFE25829975B467E13C

TERM0002; 4782396005E3A559F6F7C2B3A9ACC391
TERM0003; 647E6860D937D56E255B8B0BC3FE60B1
TERM0004; 9A56D2320F3EBF83FCB1F11A483DCAAA
TERM0005; 90AED3597E529CC33B5536568A8692A0
TERM0006; 76C4990627FD5CDF7909477E7038912F
TERM0007; 23705B7F6991122CDB5D6C29487A5788
TERM0008; AA3F6D005C60FCB55B4BC67A9557A64B
TERM0009; 543B8019CA89ABCAD9DB7E2290318FDD
TERM0010; EE004CF7F062B82D84597F9468EEDB5C

We store this file by any name in the directory, which is assigned to the alias '$VALUEPOOLS'.

4.3 PAN, Track 2 Data and other Transaction Parameters

There is a number of relevant elements, which must be considered if we want to have ICC, magstripe and manual key entry transactions. In our example, we make the assumptions below, which must be adapted (e.g. if your ICC uses on-line PIN verification).

Note that value pools are only needed for elements, which are different from one transaction to the next. Elements, which remain stable (e.g. Acquirer Identifier) are not mentioned here.

4.3.1 Elements for ICC Transactions

The following elements are present only for ICC transactions:

<Application Cryptogram>
<Application Effective Date>
<Application Interchange Profile>
<Application PAN Sequence Number>
<Application Transaction Counter (ATC)>
<Cryptogram Information Data>
<Track 2 Equivalent Data>
<Unpredictable Number>

CEPTEST can not generate a valid <Application Cryptogram>, because this element is generated by the ICC according to the issuers specification. Using CEPTESTS plug in feature however, you may add a correct value for this element.

4.3.2 Elements for Magstripe Transactions

The following elements are available only for magstripe transactions:

<Enciphered Personal Identification Number (PIN)
<Track 2 Data>

4.3.3 Elements for Manual Data Entry

The following data elements are available only for manual data entry:

<Application Expiration Date>
<Application Primary Account Number (PAN)>
<Card Verification Code 2>

4.3.4 Elements Depending on POS Entry Mode

The following data elements vary depending on the POS entry mode:

<CVM Results>
<POS Entry Mode>
<Terminal Verification Results (TVR)>

Element	ICC	Magstripe	Manual
<CVM Results>	410302[1)]	420302[3)]	5E0002[5)]
<POS Entry Mode>	05	02	01
<Terminal Verification Results (TVR)>	0000008000[2)]	0000048000[4)]	0000008000[2)]

Table 1 EMV Data Elements for POS Entry Modes

1) Plain text PIN verification by ICC
2) Transaction exceeds floor limit
3) On line PIN verification
4) Transaction exceeds floor limit, On-line PIN entered
5) Signature

4.3.5 Using the EMV Elements Editor

For several EMV elements, there is an EMV editor available in CEPTEST. It might be more convenient to use this editor to find the right bits and bytes for these elements.

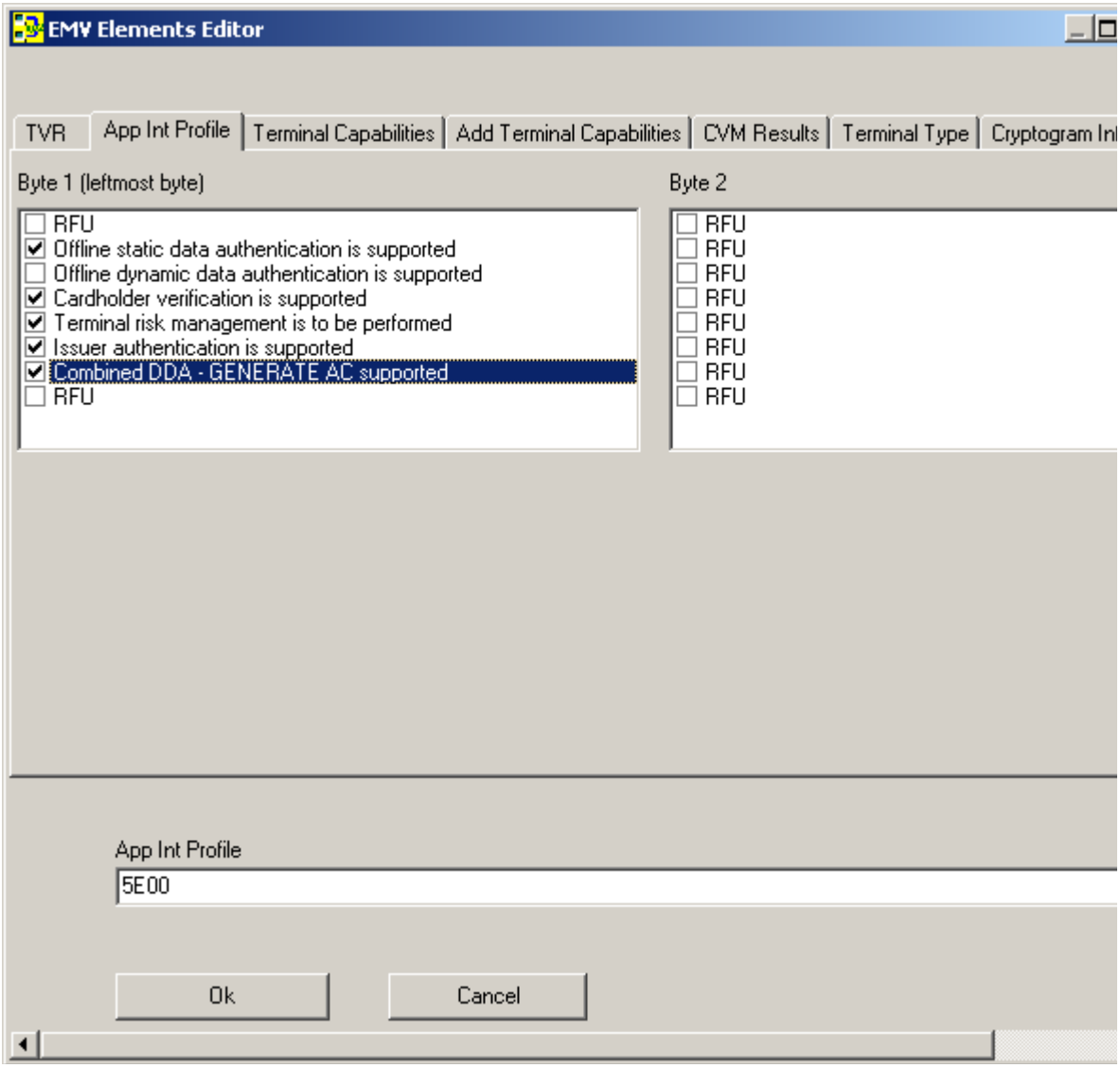

Figure 7 Sample View on CEPTEST's EMV Elements Editor

To get the editor, click [Edit], [Edit Window...] and the button 'EMV Editor...'.

4.3.6 Generating a Value Pool File for Multiple Elements

For this type of value pool, where several elements need to appear together in the same message (or container), values are entered in the form:

tag1=value1;tag2=value2;...

Binary and Compressed numeric values are entered in HEX. Encrypted values are entered clear text, encription follows during generation of the XML document.

4.3.7 Sample Value Pool File with Multiple Elements

The data below is on one line per transaction (without the bullet). Spaces between items are optional. We need at least 10 different transactions for our test. We decide to have 6 ICC transactions, 3 magstripe and 1 manual data entry transaction.

- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0001; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000001D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0002; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000002D000; ep2:Unpred=36961209; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0003; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000003D000; ep2:Unpred=48257665; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0004; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000004D000; ep2:Unpred=24768809; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0005; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000005D000; ep2:Unpred=15433785; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0006; ep2:CryptInfo=80; ep2:Track2EqDat=0000000000000006; ep2:Unpred=38723514; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000008000

The magstripe transactions are:

- ep2:Track2Dat=0000000000000007D000; ep2:EncPIN=123456; ep2:CVMRes=420302; ep2:POSEntry=02; ep2:TVR=0000048000
- ep2:Track2Dat=0000000000000008D000; ep2:EncPIN=444555; ep2:CVMRes=420302; ep2:POSEntry=02; ep2:TVR=0000048000
- ep2:Track2Dat=0000000000000009D000; ep2:EncPIN=887766; ep2:CVMRes=420302; ep2:POSEntry=02; ep2:TVR=0000048000

The manual data entry transaction is:

- ep2:AppExpDate=020728; ep2:AppPAN=0000000000000010; ep2:CVC2=222; ep2:CVMRes=5E0002; ep2:POSEntry=01; ep2:TVR=0000008000

4.3.8 Sample Value Pool for Off-line Transactions

Our Off-line transactions must be somewhat different, because it is restricted to ICC transactions and of we have a different value for Cryptogram Information Data. Therefore we make a separate Value Pool File:

- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0001; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000001D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0002; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000002D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0003; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000003D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0004; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000004D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0005; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000005D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0006; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000006D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0007; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000007D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0008; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000008D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0009; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000009D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000
- ep2:AC=0000000000000000; ep2:AppEffDate=010502; ep2:AppIntProf=7C00; ep2:AppPANSeq=1; ep2:ATC=0010; ep2:CryptInfo=40; ep2:Track2EqDat=0000000000000010D000; ep2:Unpred=02413982; ep2:CVMRes=410302; ep2:POSEntry=05; ep2:TVR=0000000000

Note that there is no strict requirement to have the exact number of cards and terminals. Off-line transactions are recorded only and will be submitted later. But there is no risk to request authorisation twice at the same time from the same terminal or card. In our example, we have chosen to present 10 transactions.

5 Creating Messages

When value pool files are ready, we are prepared to create messages. We need the following messages:

- Authorisation Request for On-line authorisations
- Authorisation Response
- Authorisation Request for Off-line authorisations
- Reversal Notification
- Reversal Acknowledge
- Data Submission File Notification from Terminal
- Data Submission File Acknowledge
- Data Submission File Notification from PMS

5.1 Authorisation Request for On-line Authorisations

- **<ep2:authreq>**
  The message container should include the value pool with our data elements for on-line transactions:

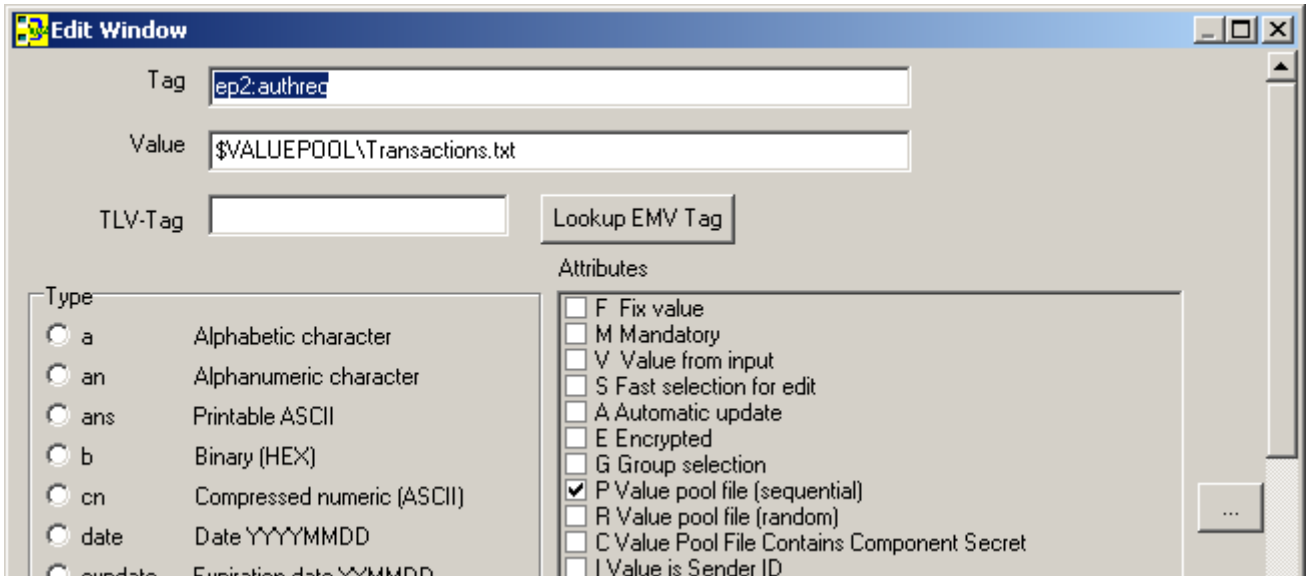

Figure 8 Message Container <ep2:authreq>

Tip: If you have a two way alias for $VALUEPOOL, you may click on the button [...], and select the file. While storing the file, CEPTEST will replace the path information by the alias. The element in the source window looks like:

<ep2:authreq>$VALUEPOOL\Transactions.txt, P3

- **<ep2:AcqID>**

  While adding elements, we recommend to check the EMV tag of the element, if this element belongs to the Data Object List of the Acquirer Transaction Certificate. Enter first the XML-tag, then click on the button 'Lookup EMV Tag'. If CEPTEST knows this tag, it places it into the field 'TLV-Tag'.

  The acquirer ID is typically also the public key owner id. Therefore we set the 'O'-attribute (Value is public key owner ID). This overrides an eventual different value set in the menu [Parameters], [Message Header...].

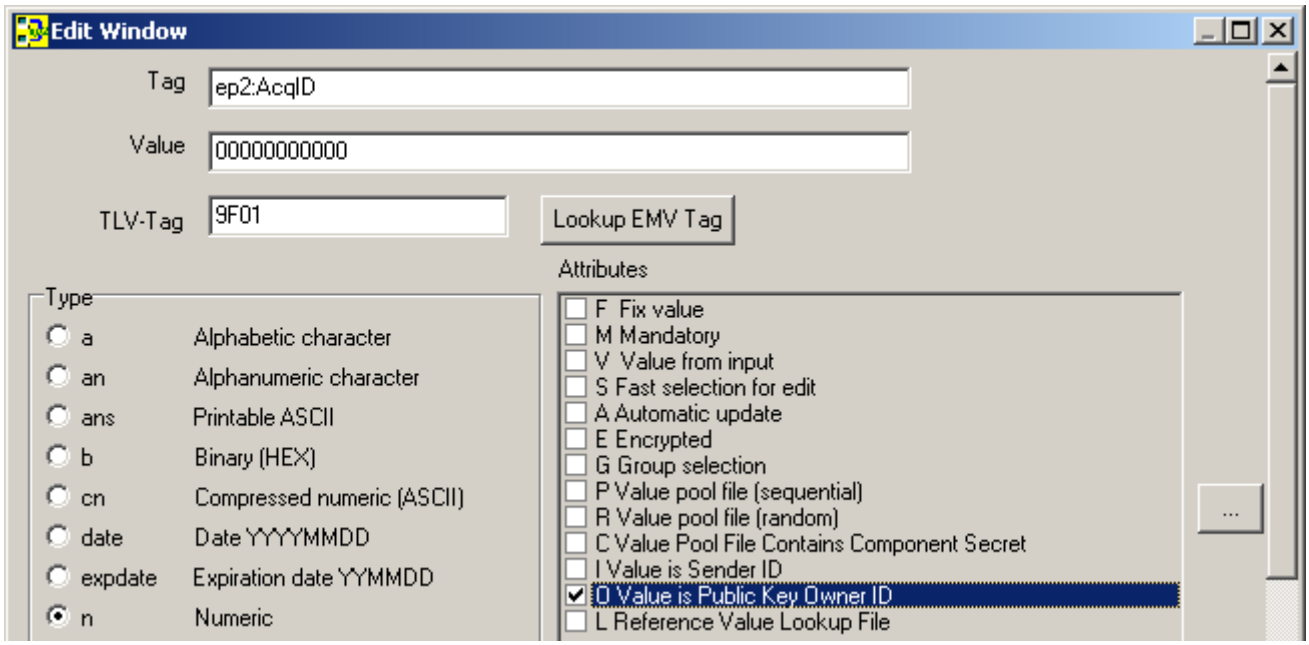

Figure 9 Element <ep2:AcqID>

The element in the source window looks like:

<ep2:AcqID>00000000000</ep2:AcqID>: n, 9F01

- **<ep2:AmtAuth>**

  Since we decided to have a randomly selected value pool for this element, we use the entries of Figure 10.

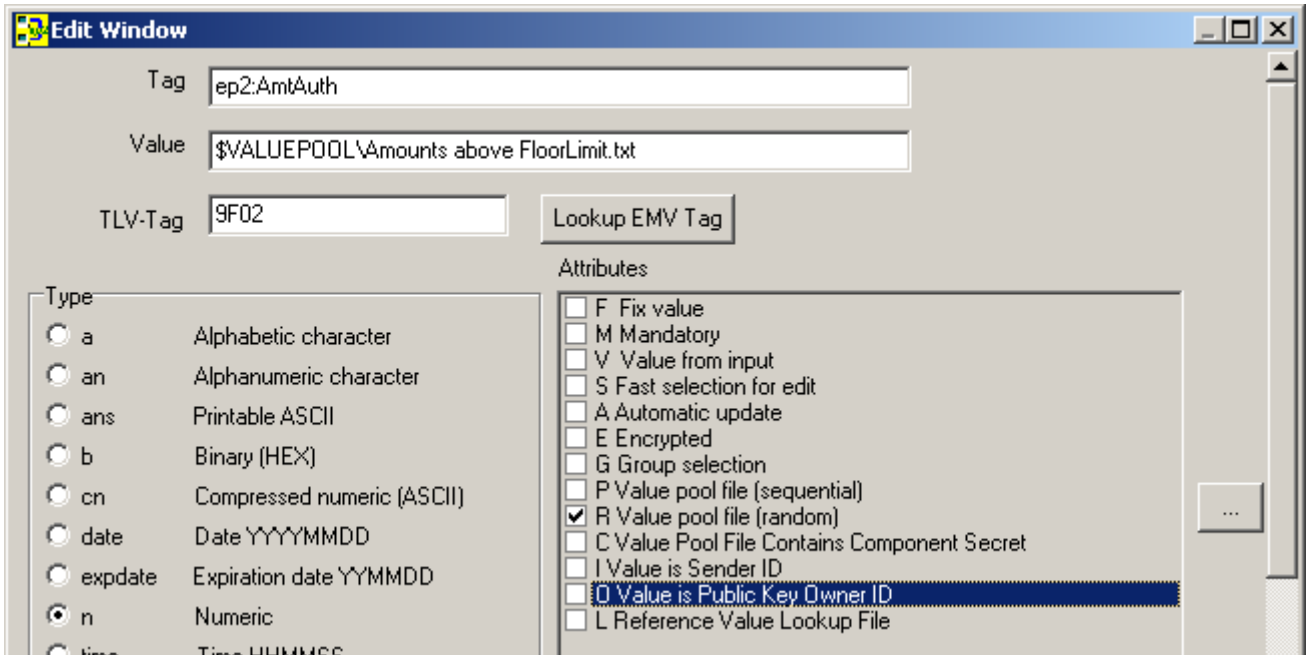

Figure 10 Element <ep2:AmtAuth>

The element in the source window looks like:

<ep2:AmtAuth>$VALUEPOOL\Amounts above FloorLimit.txt</ep2:AmtAuth>: n, R, 9F02

- **<ep2:AC>, <ep2:AppExpDate>, <ep2:AppIntProf>, <ep2:AppPANSeq>, <ep2:ATC>, <ep2:CryptInfo>, <ep2:CVMRes>, <ep2:POSEntry>, <ep2:Unpred>, <ep2:TVR>**

  These elements have in common, that they belong to the value pool file included in the container <ep2:authreq> above. Nevertheless, we need to add them here, with empty values. This is used by CEPTEST to retrieve type, encryption and TLV-tag information.

  <ep2:AC></ep2:AC>: b, 9F26
  <ep2:AppExpDate></ep2:AppExpDate>: expdate, 5F24
  <ep2:AppIntProf></ep2:AppIntProf>: b, 82
  <ep2:AppPANSeq></ep2:AppPANSeq>: n
  <ep2:ATC></ep2:ATC>: b, 9F36
  <ep2:CryptInfo></ep2:CryptInfo>: b, 9F27
  <ep2:CVMRes></ep2:CVMRes>: b, 9F34
  <ep2:POSEntry></ep2:POSEntry>: n, 9F39
  <ep2:Unpred></ep2:Unpred>: b, 9F37
  <ep2:TVR></ep2:TVR>: b, 95

- **<ep2:AppPAN>, <ep2:Track2EqDat>, <ep2:Track2Dat>**

  These elements are encrypted, in addition to the above,. Therefore we need to set the 'E' attribute:

  <ep2:AppPAN></ep2:AppPAN>: cn, E, 5A
  <ep2:Track2EqDat></ep2:Track2EqDat>: cn, E, 57
  <ep2:Track2Dat></ep2:Track2EqDat>: cn, E

- **<ep2:AID>, <ep2:TrxTypeExt>, <ep2:TrxCurrC>**

These elements have fix values in our example. Therefore they are entered with values:

<ep2:AID>A0 00 00 00 04 10 10</ep2:AID>: b, 9F06
<ep2:TrxTypeExt>0</ep2:TrxTypeExt>: n
<ep2:TrxCurrC>756</ep2:TrxCurrC>: n, 9F2A

- **<ep2:TrmID>**

  Terminal ID and Component Secret are in an own value pool file.

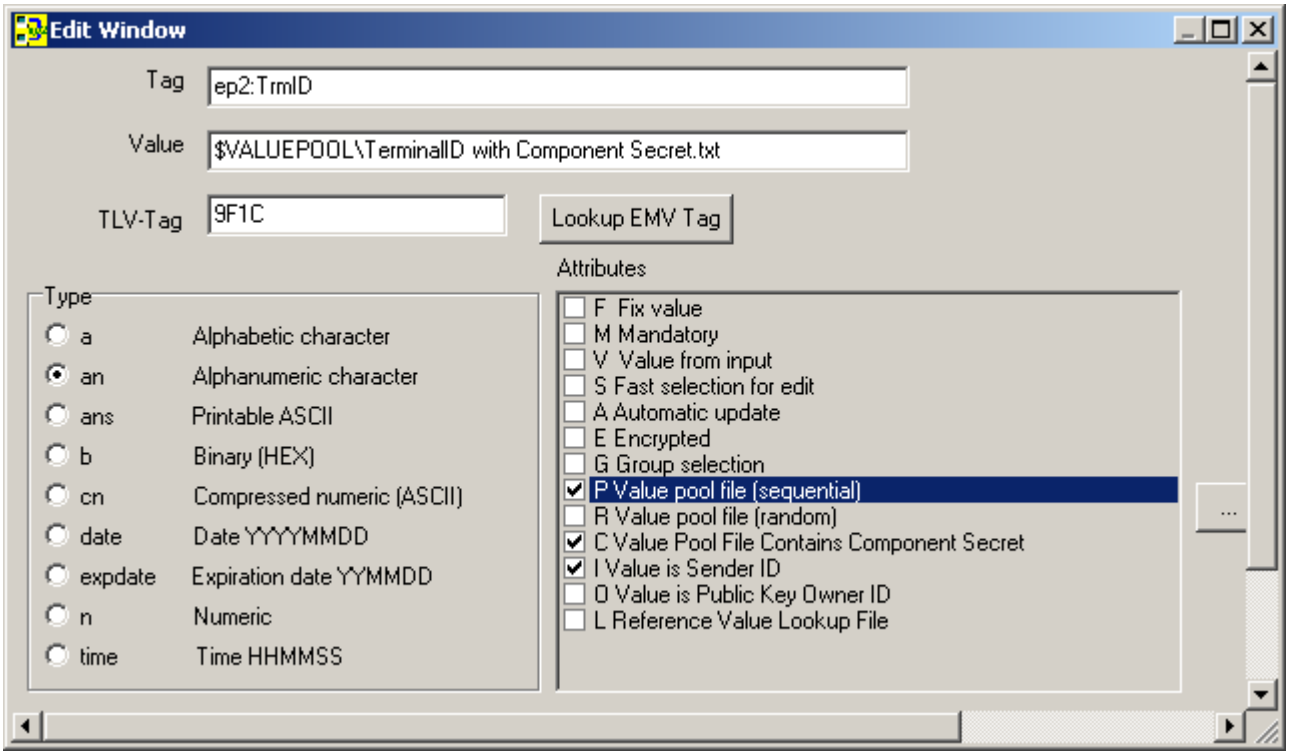

Figure 11 Element <ep2:TrmID>

Since Terminal ID is also sender ID, we set the 'I' (Value is Sender ID) attribute. The 'C' attribute forces CEPTEST to use the component secret from the value pool file instead of the one configured in the menu [Parameters], [Message Header...].

The Source Window looks like:

<ep2:TrmID>$VALUEPOOL\TerminalID with Component Secret.txt</ep2:TrmID>: an, P8CI, 9F1C

- **<ep2:TrxDate>, <ep2:TrxSeqCnt>, <ep2:TrxTime>**

  These elements shall be set by CEPTEST to current values. Therefore we set the 'A' (Automatic Update) attribute.

  <ep2:TrxDate>20060302</ep2:TrxDate>: date, A, 9A
  <ep2:TrxSeqCnt>22786</ep2:TrxSeqCnt>: n, A, 9F41
  <ep2:TrxTime>104119</ep2:TrxTime>: time, A, 9F21

5.1.1 Sample Source Window

Our final authorisation request message for on-line authorisations is:

```
<ep2:authreq>$VALUEPOOL\Transactions.txt, P3
  <ep2:AcqID>00000000000</ep2:AcqID>: n, 9F01
  <ep2:AmtAuth>$VALUEPOOL\Amounts above FloorLimit.txt</ep2:AmtAuth>: n,
R, 9F02
  <ep2:AC></ep2:AC>: b, 9F26
  <ep2:AppExpDate></ep2:AppExpDate>: expdate, 5F24
  <ep2:AID>A0 00 00 00 04 10 10</ep2:AID>: b, 9F06
  <ep2:AppIntProf></ep2:AppIntProf>: b, 82
  <ep2:AppPAN></ep2:AppPAN>: cn, E, 5A
  <ep2:AppPANSeq></ep2:AppPANSeq>: n
  <ep2:ATC></ep2:ATC>: b, 9F36
  <ep2:CryptInfo></ep2:CryptInfo>: b, 9F27
  <ep2:CVMRes></ep2:CVMRes>: b, 9F34
  <ep2:POSEntry></ep2:POSEntry>: n, 9F39
  <ep2:TrmID>$VALUEPOOL\TerminalID with Component Secret.txt</ep2:TrmID>:
an, P8CI, 9F1C
  <ep2:TVR></ep2:TVR>: b, 95
  <ep2:Track2Dat></ep2:Track2Dat>: cn, E
  <ep2:Track2EqDat></ep2:Track2EqDat>: cn, E, 57
  <ep2:TrxCurrC>756</ep2:TrxCurrC>: n, 9F2A
  <ep2:TrxDate>20060302</ep2:TrxDate>: date, A, 9A
  <ep2:TrxSeqCnt>22786</ep2:TrxSeqCnt>: n, A, 9F41
  <ep2:TrxTime>104119</ep2:TrxTime>: time, A, 9F21
  <ep2:TrxTypeExt>0</ep2:TrxTypeExt>: n
  <ep2:Unpred></ep2:Unpred>: b, 9F37
```

5.2 Authorisation Response

In this message we need to decide, which elements we want:

- To be present in all responses
- To contain specific values

We mark all elements, which must be present (messages without this element are considered as errors), with the ‘M’ (Mandatory) attribute. All elements with known expected value are marked with an ‘F’ (Fix value) attribute.

Values of elements, which are not marked with an ‘F’ attribute are not used by CEPTEST and may be set or left blank.

We recommend to enter all optional elements in the message, which may be present. This allows us later to write these elements into log files.

5.2.1 Sample Source Window

```
<ep2:authrsp>
  <ep2:AcqID>00000000000</ep2:AcqID>: n, FM
  <ep2:AmtAuth>0</ep2:AmtAuth>: n, M
  <ep2:AuthC></ep2:AuthC>: an
  <ep2:AuthRespC>00</ep2:AuthRespC>: an, FM
  <ep2:AuthReslt>1</ep2:AuthReslt>: n, FM
  <ep2:IssAuthDat></ep2:IssAuthDat>: b
```

<ep2:TrmID>TRM00000</ep2:TrmID>: an, M
<ep2:TrxSeqCnt></ep2:TrxSeqCnt>: n, M
<ep2:IssScrpt></ep2:IssScrpt>: b

5.3 Authorisation Requests for Off-line Authorisations

This message is similar to the Authorisation Request for On-Line Authorisations except:

- Value Pool Files for the message container <ep2:authreq> and the element <ep2:AmtAuth> are different.
- Since only ICC transactions are used, the element <ep2:Track2Dat> is not needed
- <ep2:TVR> contains the known value ’0000000000’

5.3.1 Sample Source Window

<ep2:authreq>$VALUEPOOL\Transactions off line.txt, P2
<ep2:AcqID>00000000000</ep2:AcqID>: n, 9F01
<ep2:AmtAuth>$VALUEPOOL\Amounts below FloorLimit.txt</ep2:AmtAuth>: n, R, 9F02
<ep2:AC></ep2:AC>: b, 9F26
<ep2:AppExpDate></ep2:AppExpDate>: expdate, 5F24
<ep2:AID>A0 00 00 00 04 10 10</ep2:AID>: b, 9F06
<ep2:AppIntProf></ep2:AppIntProf>: b, 82
<ep2:AppPAN></ep2:AppPAN>: cn, E, 5A
<ep2:AppPANSeq></ep2:AppPANSeq>: n
<ep2:ATC></ep2:ATC>: b, 9F36
<ep2:CryptInfo></ep2:CryptInfo>: b, 9F27
<ep2:CVMRes></ep2:CVMRes>: b, 9F34
<ep2:POSEntry></ep2:POSEntry>: n, 9F39
<ep2:TrmID>$VALUEPOOL\TerminalID with Component Secret.txt</ep2:TrmID>: an, P7CI, 9F1C
<ep2:TVR>0000000000</ep2:TVR>: b, 95
<ep2:Track2EqDat></ep2:Track2EqDat>: cn, E, 57
<ep2:TrxCurrC>756</ep2:TrxCurrC>: n, 9F2A
<ep2:TrxDate>20060302</ep2:TrxDate>: date, A, 9A
<ep2:TrxSeqCnt>22816</ep2:TrxSeqCnt>: n, A, 9F41
<ep2:TrxTime>121312</ep2:TrxTime>: time, A, 9F21
<ep2:TrxTypeExt>0</ep2:TrxTypeExt>: n
<ep2:Unpred></ep2:Unpred>: b, 9F37

5.4 Reversal Notification

- **<ep2:AcqID>, <ep2:AID>, <ep2:AuthRespC>, <ep2:TrmID>**

These elements shall contain the same values as the previous authorisation request/response conversation. Therefore they are marked with the 'V' (Value from Input) attribute.

- **<ep2:OrignDE>**
  All elements of this container need to have the 'V' (Value from Input) attribute, including <ep2:TrxSeqCnt>.
- **<ep2:TrxSeqCnt>**
  This element appears twice: Once in the message container and once in the container <ep2:OrignDE>. While the value in the <ep2:OrignDE> container is the one of the authorisation request and therefore taken from there, the value in the message container is a new value, which shall be updated by CEPTEST and therefore marked with the 'A' (Automatic Update) attribute.

5.4.1 Sample Source Window

```
<ep2:revntf>
  <ep2:AcqID>00000000000</ep2:AcqID>: n, V
  <ep2:AID>A0 00 00 00 04 10 10</ep2:AID>: b, V
  <ep2:AuthRespC>00</ep2:AuthRespC>: an, V
  <ep2:TrmID>TRM00000</ep2:TrmID>: an, V
  <ep2:TrxDate>20060302</ep2:TrxDate>: date, A
  <ep2:TrxSeqCnt>18980</ep2:TrxSeqCnt>: n, A
  <ep2:TrxTime>123045</ep2:TrxTime>: time, A
  <ep2:OrignDE>
    <ep2:AID>A0 00 00 00 04 10 10</ep2:AID>: b, V
    <ep2:TrmID>TRM00000</ep2:TrmID>: an, V
    <ep2:TrxSeqCnt>18975</ep2:TrxSeqCnt>: n, V
    <ep2:TrxTypeExt>0</ep2:TrxTypeExt>: n, V
  <ep2:RevReason>0</ep2:RevReason>: n
```

5.5 Reversal Acknowledge

This is a simple response message, with mandatory and fix value elements.

5.5.1 Sample Source Window

```
<ep2:revack>
  <ep2:AcqID>00000000000</ep2:AcqID>: n, FM
  <ep2:TrmID>TRM00000</ep2:TrmID>: an, M
  <ep2:AuthReslt>0</ep2:AuthReslt>: n, FM
```

5.6 Data Submission File Notification from Terminal

- **<ep2:TrmID>**

We take this value from the previous authorisation conversation and mark it as 'Sender ID'.

- **<ep2:DSFDet>**
  We mark this element with a 'D' (Value is Recorded Data File) attribute. If found, CEPTEST searches in the directory of recorded data for a file with terminal ID for name and the extension XML (e.g. TERM0000.xml). If found, the content of this file is entered into this container.

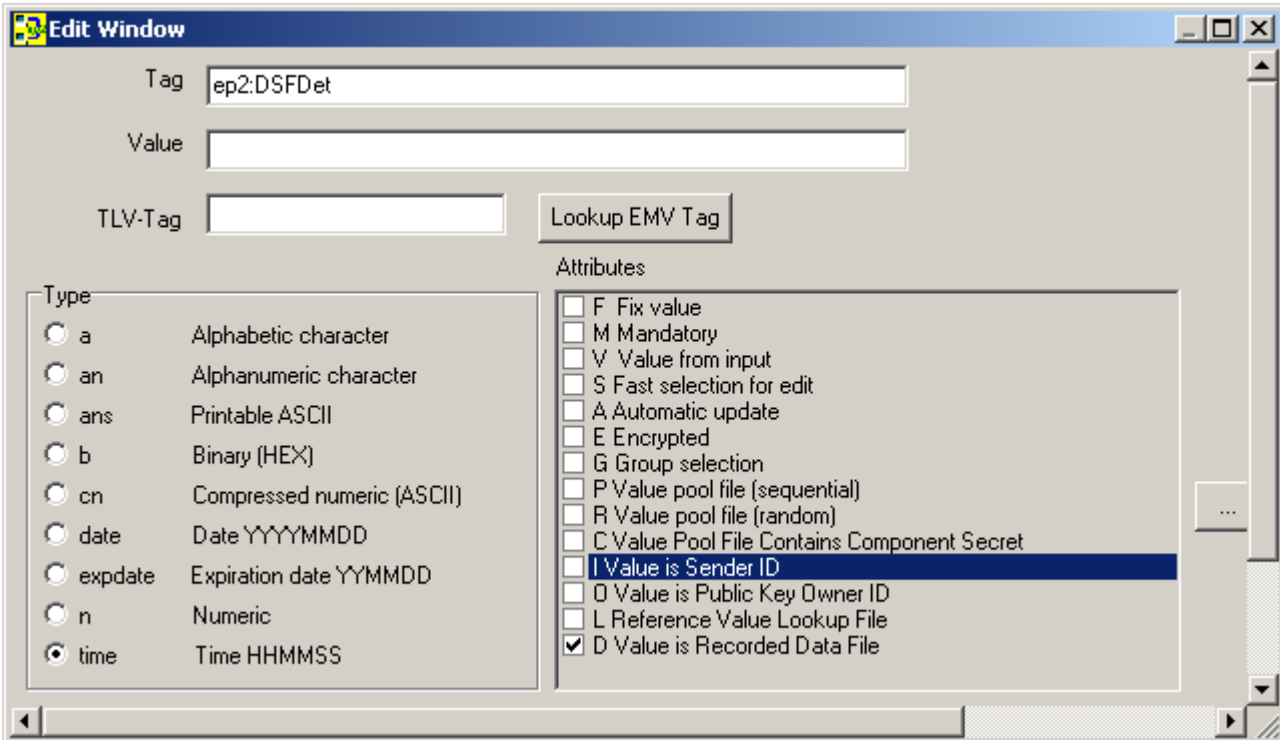

Figure 12 Element <ep2:DSFDet>

5.6.1 Sample Source Window

```
<ep2:dsfntf>
  <ep2:AcqID>00000000000</ep2:AcqID>: n
  <ep2:TrmID></ep2:TrmID>: an, VI
  <ep2:DSFDet>, D
```

5.7 Data Submission File Acknowledge

5.7.1 Sample Source Window

```
<ep2:dsfack>
  <ep2:AcqID>00000000000</ep2:AcqID>: n, FM
```

<ep2:TrmID></ep2:TrmID>: an, M

5.8 Data Submission File Notification from PMS

The difference compared to a submission file from a terminal is, that a PMS file contains transactions from several terminals. If a data submission file notification message contains a PMS ID, CEPTEST assumes, that all files in the recording directory with the extension .xml are recorded terminal files with the name of the file as terminal ID. So the directory, which contains the recorded files should be kept clean of other files which could be misinterpreted.

5.8.1 Sample Source Window

<ep2:dsfntf>
<ep2:AcqID>00000000000</ep2:AcqID>: n
<ep2:PMSID>PMS12345678</ep2:PMSID>: an, I
<ep2:DSFDet>, D

6 Generating an Automated Sequence

When all messages are ready, we can generate an automated sequence. Select for this the menu [Sequences], [Sequence...].

6.1 Sequence Messages

Click on the button 'Select Messages...' for a file dialogue. Select files by double clicking on a file, which transfers the file name into your sequence and moves the cursor to the next cell. So double click on the following messages:

- Authorisation Request for On-line authorisations (Client message)
- Authorisation Response (Server message)
- Reversal Notification (Client message)
- Reversal Acknowledge (Server message)
- Authorisation Request for On-line authorisations (Client message)
- Authorisation Response (Server message)
- Authorisation Request for Off-line authorisations (Client message)
- Click on the button 'Move to Next Cell', because there is no server message
- Data Submission File Notification from Terminal (Client message)
- Data Submission File Acknowledge (Server message)

- Authorisation Request for On-line authorisations (Client message)
- Authorisation Response (Server message)
- Authorisation Request for Off-line authorisations (Client message)
- Click on the button 'Move to Next Cell', because there is no server message
- Data Submission File Notification from PMS (Client message)
- Data Submission File Acknowledge (Server message)

Close the file selection window. At this point, your sequence should look similar to figure 13.

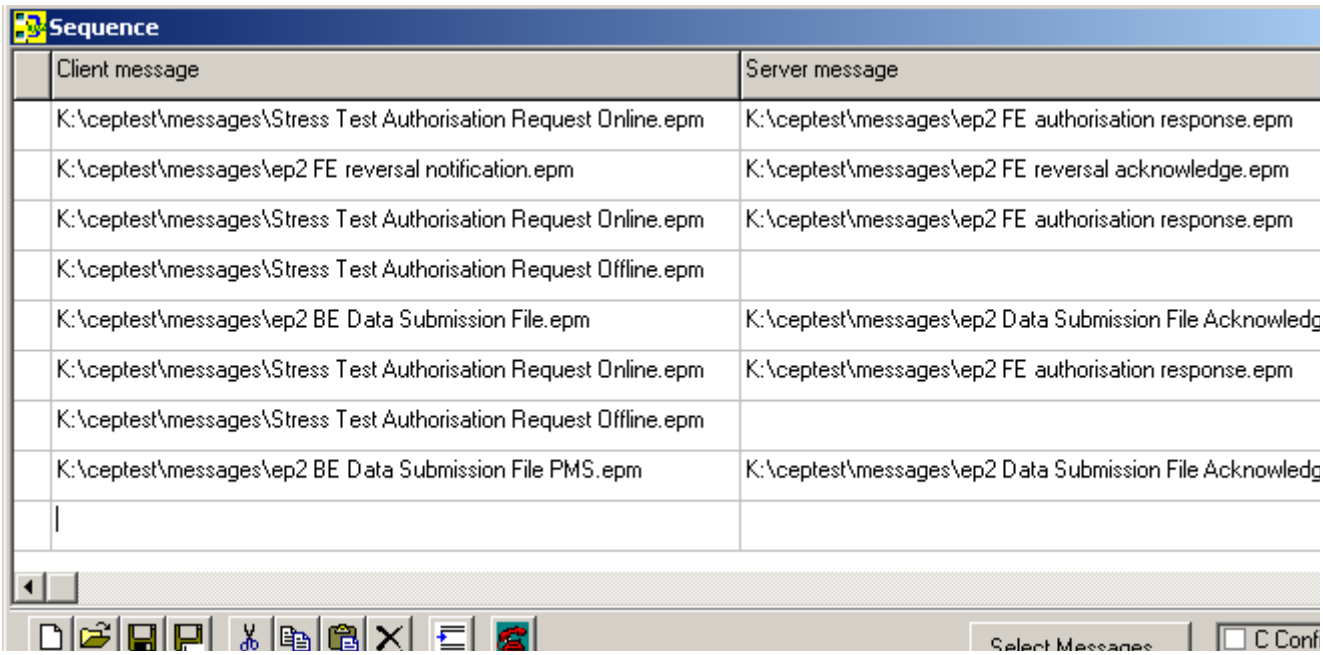

Figure 13 Sequence Messages

Tip: A right click with the mouse on the grid pop ups a menu with the basic editing functions. You can also edit directly into the grid.

6.2 Attributes

We must now select the right attributes for each message row. For this, click anywhere on a row. Check the attributes, which shall be valid for the row:

- Set the 'D' attribute on every row. This causes CEPTEST to disconnect the TCP/IP connection after the server responded. Since all of the rows define a complete session, the connection shall be disconnected at the end.
- For the 2 off-line authorisations, set the 'F' attribute.
- Set the 'T' attribute on each row, which refers to the previous one: The reversal notification message shall contain many elements, which were used in the authorisa-

tion: Transaction Sequence Counter, Terminal ID... These elements were marked with a 'V' (value from input) attribute in the message. Setting the 'T' attribute forces CEPTEST to look for actual values of elements marked with the 'V' attribute in either the last received message, or the last sent message.

We also need to set the 'T' attribute for the off-line authorisations and data submissions by terminal. This because we want these messages to contain the same terminal ID as the on-line authorisation.

6.3 Client Threads

The number in the column 'Client Threads' defines the number of parallel open messages sent to the server. We decided to have 10 parallel requests, so we put that number in the first line.

An entry in the 'Client Threads' column remains valid until a new entry is found. Therefore we leave this column empty until the last column, where we have data submission by PMS. Since a PMS submits all recorded data from all terminal, we have only 1 client thread there.

Note that there is a difference if the column 'Client Threads' is empty or if the same number as in the previous row is repeated: If the number is there, new threads are generated, which have nothing to do with the previous ones. So if a 'T' (Transfer input values) attribute is set in a row, where there is a 'Client Threads' entry, this attribute has no effect, because any values from a previous message are lost when a new thread is created.

The resulting sequence can be seen in figure 14. This sequence can be saved now.

6.4 IP Address/Port

If different port are used for authorisation and submission, these columns need to be filled with the port address information. In our example this is not used.

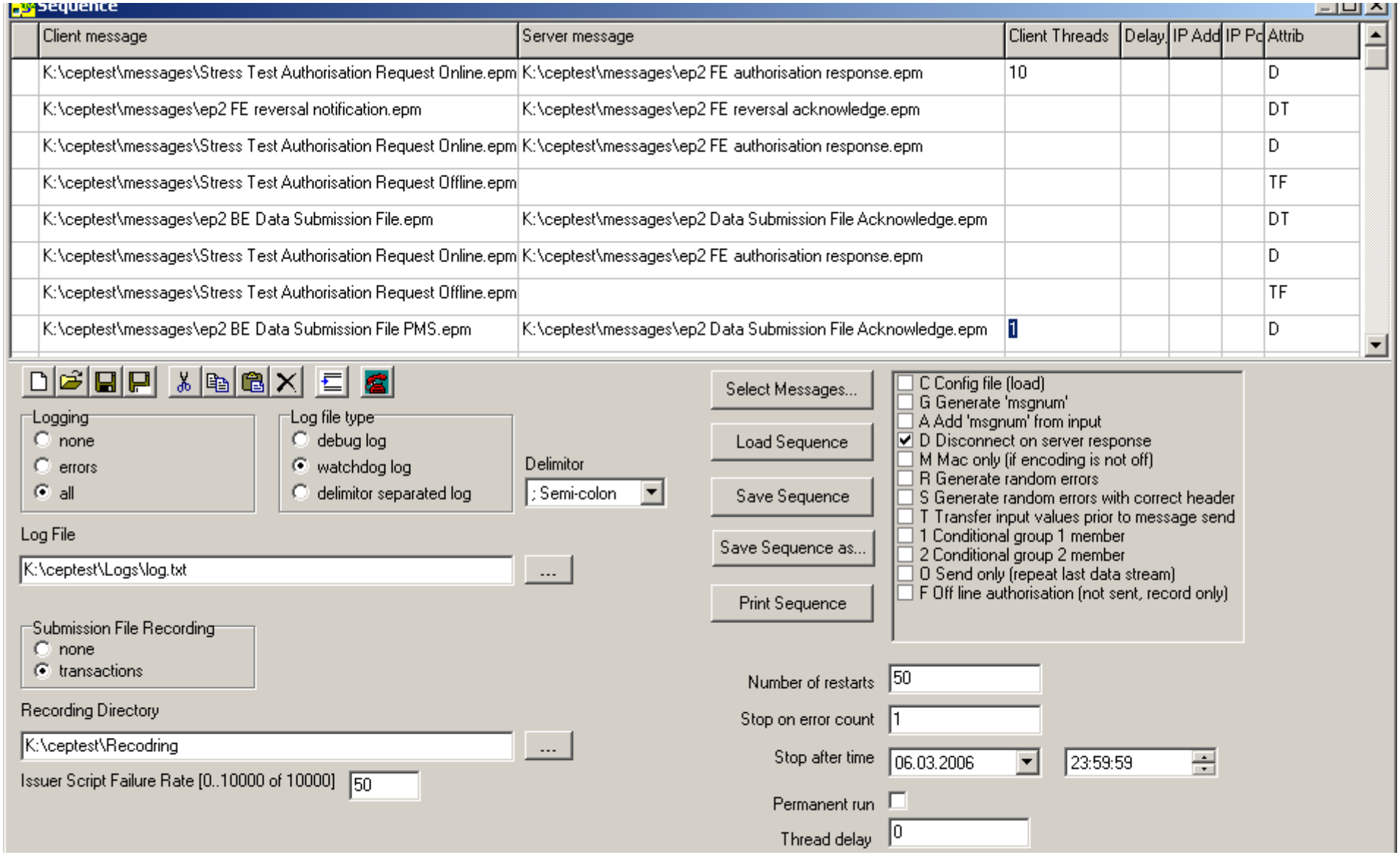

Figure 14 Sequence Setup

6.5 Parameters for Sequences

These parameters are part of the configuration, not the sequence itself, which means, that they are stored with the configuration file and not the sequence file.

6.5.1 Logging

The first question is, what we want to log. The options are:

- Nothing
- Only errors
- All messages

We decide to log all messages.

We have then 3 different types of logs to choose from:

- Debug log, where each log entry contains all windows on the main screen
- Watchdog log, where we get one line of data per client/server message pair
- Delimitor separated log, where entries are prepared for import into databases or spreadsheets.

We decide for the watchdog log.

Then we have to define the log file. CEPTEST takes the name and adds a counter to it, which counts the files. Every file contains 100 entries. So if your name is log.txt, you get files log.0.txt, log.1.txt...

6.5.2 Content of Log Files

The content of the log is defined in a file ‘logfile.ini’. This file is in the same directory as CEPTEST.exe.

Logfile.ini contains 1 line per entry of the form: title + ; + XML-tag or special value + ; + field length.

Example:

Amount;ep2:AmtAuth;10

This sets a title "Amount" in the title line. On the log entry, the value of the element with a tag ep2:AmtAuth is written. The size of the column shall be 10 characters. Note that the size should be large enough to accommodate the largest value of the element or the title (if the title is larger than the largest value).

There is a number of special values, which can be used instead of the XML-tag:

- $DATE: Current date
- $TIME: Current time
- $EXECTIME: Time between client and server message
- $SENDMESSAGE: Tag of the message sent by CEPTEST
- $RECEIVEDMESSAGE: Tag of the message, received by CEPTEST
- $ERROR: If an error occurs, error text is written into the log. If no error occurs, no entry is made

- $SENDCOUNTER: The time [ms], where the request for a TCP/IP connection to the server is invoked, measured from a 0-base which is the same for all client threads (server version). This time shows, how many parallel requests where requested by CEPTEST in what time.

For our example, we use the following logfile.ini:

```
Date;$DATE;10
Time;$TIME;08
Exectime;$EXECTIME;08
Send Time;$SENDCOUNTER;12
Send Message;$SENDMESSAGE;12
Received Msg;$RECEIVEDMESSAGE;12
Amount;ep2:AmtAuth;10
Currency;ep2:TrxCurrC;08
Track 2 Equivalent Data; ep2:Track2EqDat; 40
Track 2 Data; ep2:Track2Dat; 40
PAN;ep2:AppPAN; 40;
Exp Date;ep2:AppExpDate;10;
Terminal ID;ep2:TrmID;11
TrxDate;ep2:TrxDate;8
TrxTime;ep2:TrxTime;8
Sequence Counter;ep2:TrxSeqCnt;16
Result;  ep2:AuthReslt;7
Response Code;ep2:AuthRespC;14
Error;$ERROR;5
```

6.5.3 Submission File Recording

Since we want to record submission files, we need to select the radio button ’transactions’.

We have to define a directory, where files shall be written into. It is a good idea, to define a separate directory for this purpose only.

Tip: All files, which are used for submission, are then transferred into a subdirectory ’archive’ of the recording directory.

While recording authorisation requests/responses CEPTEST generates <Issuer Script Results> for <Issuer Script Commands> found in authorisation responses. These results are normally generated for successful commands. However, if you want to have a probability for failures in your results, you may define a failure rate (number of failures per 10000 commands). In our example, we have selected 50 failures per 10000 commands.

Tip: The file ’submission.ini’ contains all elements, which shall be recorded. If you have additional elements, which appear either in the authorisation request or in the authorisation response message, you may enter its xml-tag. If such an element belongs to the acquirer transaction certificate data object list, you must also add the EMV-tag of the element.

6.5.4 Other Parameters

The remaining parameters are:

- The number of restarts of our sequence (50)
- After how many errors, the sequence shall stop (1)

- A time, after which the sequence is stopped
- If the check box 'Permanent run' is checked, all of the above is invalid. The sequence runs until the user stops is.
- Thread delay: A random delay [ms] each threads waits before sending its message to the server. This is because CEPTEST is able to send out all requests within few milliseconds. If you want a larger distribution of the requests, you may add a delay here.

7 Run the Sequence

Now everything is set, we are ready to run our sequence.

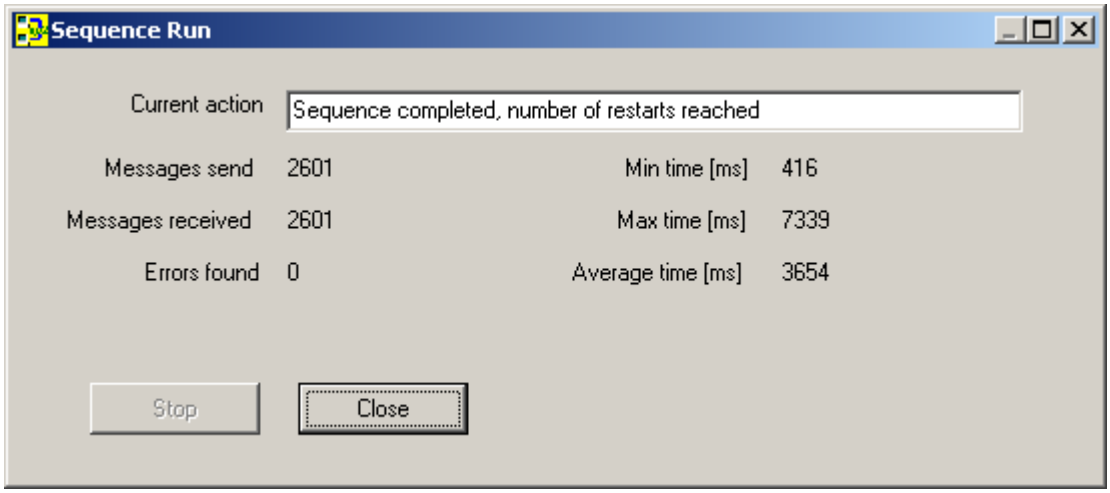

Figure 15 Sequence Status Window

A glance into the log with only parts of the columns is given in table 2. It shows, how different message use different data for authorisation. Note that the column Send Time is based on a arbitrary 0 level. But e.g. the first 10 requests are sent 1406..1409 ms after this 0 base, thus within 3 ms.

Execution Time (Response time of the server) and send time of Off-line authorisations are 0, since no message is sent.

Another point is the amount column, where values are randomly selected, as requested.

Date	Time	Exec-time	Send Time	Send Message	Received Msg	Amount	Track 2 Equivalent Data	Track 2 Data	PAN	Exp Date	Terminal ID
2006.03.06	13:32:44	3844	1409	ep2:authreq	ep2:authrsp	2000	398C4324BA7B95E7B5483C96C812DC62				TERM0008
2006.03.06	13:32:44	3745	1406	ep2:authreq	ep2:authrsp	200000	1155FB132EB3EDCEACC8BF41C9C12E1E				TERM0009
2006.03.06	13:32:44	3744	1407	ep2:authreq	ep2:authrsp	4300	50A88075775AF33E0F72ED15423E39F5				TERM0010
2006.03.06	13:32:44	3750	1408	ep2:authreq	ep2:authrsp	5600	0CF679CCBB66F381F1B27DEB523E47AD				TERM0001
2006.03.06	13:32:44	3777	1408	ep2:authreq	ep2:authrsp	1600		2B0A87B933E46C1794401721A1AC3A94			TERM0002
2006.03.06	13:32:44	3797	1408	ep2:authreq	ep2:authrsp	2000		FCEAEDDA66B666CBB8F5FB8ED37B8AE9			TERM0003
2006.03.06	13:32:44	3801	1408	ep2:authreq	ep2:authrsp	4300		2C2975E75D3FFC787E83714CC434D6CD			TERM0004
2006.03.06	13:32:45	3809	1409	ep2:authreq	ep2:authrsp	4300			7A66F1E4F2FCC5A8D8CB853C6F463B22	020728	TERM0005
2006.03.06	13:32:45	3828	1409	ep2:authreq	ep2:authrsp	4100	0F6C88C8781503D416A7C02D4BABB2D1				TERM0006
2006.03.06	13:32:45	3830	1409	ep2:authreq	ep2:authrsp	1000	9888402B27F9F68D8217B4B1C451E92B				TERM0007
2006.03.06	13:32:51	3815	1918	ep2:revntf	ep2:revack						TERM0008
2006.03.06	13:32:51	3816	1919	ep2:revntf	ep2:revack						TERM0009
2006.03.06	13:32:51	3824	1919	ep2:revntf	ep2:revack						TERM0010
2006.03.06	13:32:51	3733	1915	ep2:revntf	ep2:revack						TERM0001
2006.03.06	13:32:51	3735	1917	ep2:revntf	ep2:revack						TERM0002
2006.03.06	13:32:51	3756	1917	ep2:revntf	ep2:revack						TERM0003
2006.03.06	13:32:51	3758	1917	ep2:revntf	ep2:revack						TERM0004
2006.03.06	13:32:51	3779	1918	ep2:revntf	ep2:revack						TERM0005
2006.03.06	13:32:51	3781	1918	ep2:revntf	ep2:revack						TERM0006
2006.03.06	13:32:51	3783	1918	ep2:revntf	ep2:revack						TERM0007
2006.03.06	13:32:57	3898	1419	ep2:authreq	ep2:authrsp	1600	398C4324BA7B95E7B5483C96C812DC62				TERM0008

Table 2 Sequence Log

Date	Time	Exec-time	Send Time	Send Message	Received Msg	Amount	Track 2 Equivalent Data	Track 2 Data	PAN	Exp Date	Terminal ID
2006.03.06	13:32:57	3898	1420	ep2:authreq	ep2:authrsp	100	1155FB132EB3EDCEACC8BF41C9C12E1E				TERM0009
2006.03.06	13:32:57	3900	1420	ep2:authreq	ep2:authrsp	2000	50A88075775AF33E0F72ED15423E39F5				TERM0010
2006.03.06	13:32:57	3775	1416	ep2:authreq	ep2:authrsp	100	0CF679CCBB66F381F1B27DEB523E47AD				TERM0001
2006.03.06	13:32:57	3815	1418	ep2:authreq	ep2:authrsp	4200		2B0A87B933E46C1794401721A1AC3A94			TERM0002
2006.03.06	13:32:57	3833	1418	ep2:authreq	ep2:authrsp	4300		FCEAEDDA66B666CBB8F5FB8ED37B8AE9			TERM0003
2006.03.06	13:32:57	3835	1419	ep2:authreq	ep2:authrsp	4000		2C2975E75D3FFC787E83714CC434D6CD			TERM0004
2006.03.06	13:32:57	3804	1418	ep2:authreq	ep2:authrsp	1600			7A66F1E4F2FCC5A8D8CB853C6F463B22	020728	TERM0005
2006.03.06	13:32:57	3863	1419	ep2:authreq	ep2:authrsp	1000	0F6C88C8781503D416A7C02D4BABB2D1				TERM0006
2006.03.06	13:32:57	3869	1419	ep2:authreq	ep2:authrsp	4100	9888402B27F9F68D8217B4B1C451E92B				TERM0007
2006.03.06	13:32:59	0	0	ep2:authreq		10	398C4324BA7B95E7B5483C96C812DC62				TERM0008
2006.03.06	13:32:59	0	0	ep2:authreq		20	1155FB132EB3EDCEACC8BF41C9C12E1E				TERM0009
2006.03.06	13:32:59	0	0	ep2:authreq		100	50A88075775AF33E0F72ED15423E39F5				TERM0010
2006.03.06	13:32:59	0	0	ep2:authreq		120	0CF679CCBB66F381C9302663A4BFF0EA				TERM0001
2006.03.06	13:32:59	0	0	ep2:authreq		500	2B0A87B933E46C1794401721A1AC3A94				TERM0002
2006.03.06	13:32:59	0	0	ep2:authreq		10	FCEAEDDA66B666CBB8F5FB8ED37B8AE9				TERM0003
2006.03.06	13:32:59	0	0	ep2:authreq		20	2C2975E75D3FFC787E83714CC434D6CD				TERM0004
2006.03.06	13:32:59	0	0	ep2:authreq		10	7A66F1E4F2FCC5A85FD09D526D16B548				TERM0005
2006.03.06	13:32:59	0	0	ep2:authreq		20	0F6C88C8781503D416A7C02D4BABB2D1				TERM0006
2006.03.06	13:32:59	0	0	ep2:authreq		100	9888402B27F9F68D8217B4B1C451E92B				TERM0007

Table 2 Sequence Log

Date	Time	Exec-time	Send Time	Send Message	Received Msg	Amount	Track 2 Equivalent Data	Track 2 Data	PAN	Exp Date	Terminal ID
2006.03.06	13:33:11	4085	7202	ep2:dsfntf	ep2:dsfack						TERM0008
2006.03.06	13:33:11	4086	7202	ep2:dsfntf	ep2:dsfack						TERM0009
2006.03.06	13:33:11	4090	7202	ep2:dsfntf	ep2:dsfack						TERM0010
2006.03.06	13:33:11	4016	7198	ep2:dsfntf	ep2:dsfack						TERM0001
2006.03.06	13:33:11	4027	7200	ep2:dsfntf	ep2:dsfack						TERM0002
2006.03.06	13:33:11	4041	7200	ep2:dsfntf	ep2:dsfack						TERM0003
2006.03.06	13:33:11	4063	7201	ep2:dsfntf	ep2:dsfack						TERM0004
2006.03.06	13:33:11	4018	7200	ep2:dsfntf	ep2:dsfack						TERM0005
2006.03.06	13:33:11	4065	7201	ep2:dsfntf	ep2:dsfack						TERM0006
2006.03.06	13:33:11	4073	7201	ep2:dsfntf	ep2:dsfack						TERM0007
2006.03.06	13:33:17	3821	1420	ep2:authreq	ep2:authrsp	2000	398C4324BA7B95E7B5483C96C812DC62				TERM0008
2006.03.06	13:33:17	3827	1420	ep2:authreq	ep2:authrsp	2000	1155FB132EB3EDCEACC8BF41C9C12E1E				TERM0009
2006.03.06	13:33:17	3839	1420	ep2:authreq	ep2:authrsp	4100	50A88075775AF33E0F72ED15423E39F5				TERM0010
2006.03.06	13:33:17	3857	1421	ep2:authreq	ep2:authrsp	100	0CF679CCBB66F381F1B27DEB523E47AD				TERM0001
2006.03.06	13:33:17	3872	1421	ep2:authreq	ep2:authrsp	6620		2B0A87B933E46C1794401721A1AC3A94			TERM0002
2006.03.06	13:33:17	3896	1421	ep2:authreq	ep2:authrsp	100		FCEAEDDA66B666CBB8F5FB8ED37B8AE9			TERM0003
2006.03.06	13:33:17	3773	1418	ep2:authreq	ep2:authrsp	2345		2C2975E75D3FFC787E83714CC434D6CD			TERM0004
2006.03.06	13:33:17	3860	1421	ep2:authreq	ep2:authrsp	100			7A66F1E4F2FCC5A8D8CB853C6F463B22	020728	TERM0005
2006.03.06	13:33:17	3792	1419	ep2:authreq	ep2:authrsp	6620	0F6C88C8781503D416A7C02D4BABB2D1				TERM0006
2006.03.06	13:33:18	3815	1420	ep2:authreq	ep2:authrsp	2000	9888402B27F9F68D8217B4B1C451E92B				TERM0007
2006.03.06	13:33:20	0	0	ep2:authreq		600	398C4324BA7B95E7B5483C96C812DC62				TERM0008
2006.03.06	13:33:20	0	0	ep2:authreq		40	1155FB132EB3EDCEACC8BF41C9C12E1E				TERM0009

Table 2 Sequence Log

Date	Time	Exec-time	Send Time	Send Message	Received Msg	Amount	Track 2 Equivalent Data	Track 2 Data	PAN	Exp Date	Terminal ID
2006.03.06	13:33:20	0	0	ep2:authreq		100	50A88075775AF33E0 F72ED15423E39F5				TERM0010
2006.03.06	13:33:20	0	0	ep2:authreq		200	0CF679CCBB66F381C 9302663A4BFF0EA				TERM0001
2006.03.06	13:33:20	0	0	ep2:authreq		400	2B0A87B933E46C179 4401721A1AC3A94				TERM0002
2006.03.06	13:33:20	0	0	ep2:authreq		20	FCEAEDDA66B666CB B8F5FB8ED37B8AE9				TERM0003
2006.03.06	13:33:20	0	0	ep2:authreq		20	2C2975E75D3FFC787 E83714CC434D6CD				TERM0004
2006.03.06	13:33:20	0	0	ep2:authreq		600	7A66F1E4F2FCC5A85 FD09D526D16B548				TERM0005
2006.03.06	13:33:20	0	0	ep2:authreq		600	0F6C88C8781503D41 6A7C02D4BABB2D1				TERM0006
2006.03.06	13:33:20	0	0	ep2:authreq		200	9888402B27F9F68D8 217B4B1C451E92B				TERM0007
2006.03.06	13:33:27	424	6234	ep2:dsfntf	ep2:dsfack						TERM0001

Table 2 Sequence Log