S I S Specifications

API User’s Guide

Application Design

Guidelines

BCSS-SIS-TD-005Ter-02~~1a~~
20~~15~~/01~~12~~/1998~~7~~

TABLE OF CONTENTS

Document Revisions

Date	Version	Reviewed by	Updated by	Description
7/11/97	Draft		~~Y. Van Dooren~~	Creation
4/12/97	1	~~A. Tilmant~~	~~Y.Van Dooren~~	First release
7/01/98	2			Transaction execution: change to comment on certification.

Reference documents

	Document ID	Version	Date	Author	Title
R1	EUHCIF-3.DOC	1.1	7/11/1996	EU/G7 Healthcards - WG7	Interoperability of Healthcard Systems Part 3: Interoperability Specification
R2			02/06/1997	KSZ/BCSS	Annexe à l'arrêté royal relatif aux

					specifications des appareils de lecture de la carte d'identité sociale
R3	BCSS-SIS-TD-001	3	07/01/1998	KSZ/BCSS	SAM Commands Reference Manual
R4	BCSS-SIS-TD-002	2	12/11/1997	KSZ/BCSS	IIC Commands Reference Manual
R5	BCSS-SIS-TD-003	2	15/12/1997	KSZ/BCSS	Card Terminal Manager API (CTM_API) Reference Manual
R6	BCSS-SIS-TD-004	5	20/01/1998	KSZ/BCSS	Belgian Native Card Server API Reference Manual
R7	BCSS-SIS-TD-005	3	20/01/1998	KSZ/BCSS	Belgian Native Card Server Specification

Introduction

This document aims at providing guidelines on how to implement applications within the context of the specifications issued by KSZ/BCSS in order to interface the Social Identity Card (SIS). The specifications to which this document refers are those hereby previously defined as ***R3*** to ***R7*** inclusive.
The use of the CTM and BNCS APIs will be clarified for those responsible of writing application programs.

Figures 1 below reminds how the application fits into the global card point of utilisation specified components.

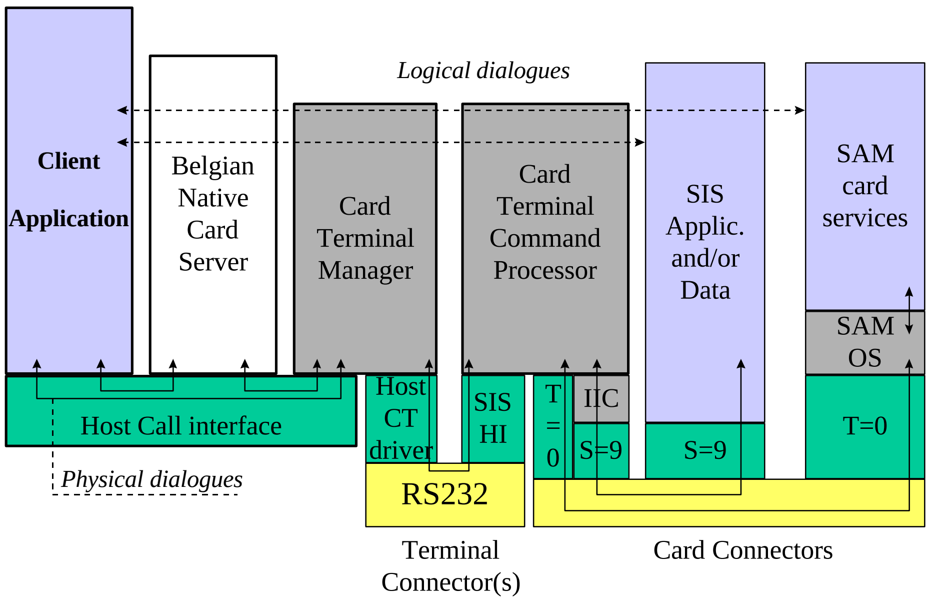


Figure 1 : Layered application interactions

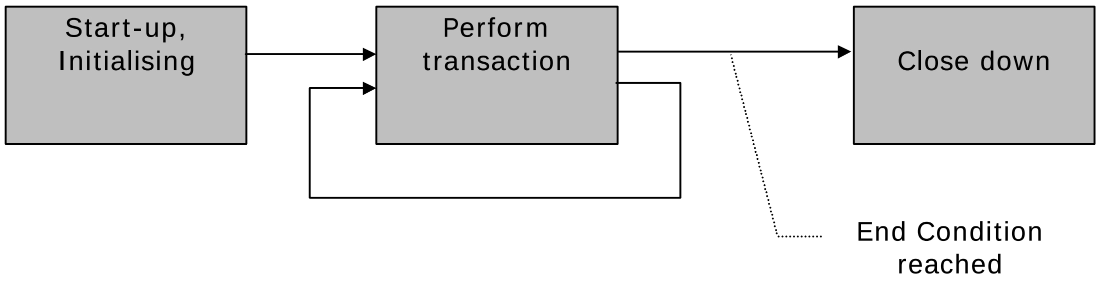
As seen on the above figure, the application direct partners are the CTM and BNCS software modules, although the services effectively solicited by the application are those provided by the SIS, SAM and terminal resources. The present document describes how these services are obtained using the BNCS and CTM APIs.

The main application API needs will be illustrated with a simple configuration as model, where one single application runs on a hosting system to which is attached a single terminal with two card readers.
The specificity of multi-applications and multi-terminals environments are described later, at the end of this document.

Warning: The next chapters describe the API calling sequences to apply. The sequences are illustrated through pseudo-C language program structures so as to match the C definitions of the API in their respective reference documents. However, for reasons of clarity, the proposed code does not fully comply with the C language programming rules.

Application processing flow

Applications using SIS services work like most transactional applications, i.e. adhere to the following processing flow:



The three steps above are discussed here for what concerns the BNCS and CTM API requirements.
The transaction execution step encompasses one or more of the following:

- Patient identification, including
    - Card recognition
    - Public data extraction
- Private data extraction
- Private data update
- Transaction certification

Transaction record format

Each transaction involving SIS card data capture and which related data must be transferred to a central system should produce a formatted transaction log record.
This log is built using data originating from the SIS, the terminal, the BNCS and the application.
The transaction record format is defined as follows for the Pharmacy applications:

Field Name	Number of bytes	Coding	Origin
Card Logical Number	10	Numeric ASCII	SIS
SSIN	11	Numeric ASCII	SIS
Card Holder Name	15	ASCII	SIS
Card Holder First Name	5	ASCII	SIS
Insuring Company Id	3	Numeric ASCII	SIS
Social Insurance Status	6	Numeric ASCII	SIS
Social Insurance Data Version	2	Numeric ASCII	SIS
Transaction Date	8	Numeric ASCII « CCYYMMDD »	Application
Prescription sequence number	6	Numeric ASCII	Application
Pharmacy Identifier	11	ASCII	Application
SFDF read certificate	32	ASCII	BNCS
Terminal manufacturer Id	*8*	*ASCII*	*CTM*

Terminal Serial Number	*8*	*ASCII*	*CTM*
Terminal KSZ/BCSS Registration Number	*4*	*ASCII*	*CTM*
BNCS provider Id	*8*	*ASCII*	*BNCS*
BNCS Software version Number	*4*	*ASCII*	*BNCS*
BNCS KSZ/BCSS Registration Number	*4*	*ASCII*	*BNCS*
Total	122 *or 158*	ASCII bytes	

Remark: The data represented in italics (tracing data) are optional.

Start-up - Initialising card and terminal services

Card services are required by different types of applications, each of them needing to initialise the dialogue with the card devices.

The classical way to set up the card service for an application is to use the following API call sequence. The error handling is not developed here.

```
/* attach application to CTM */
RetCode = CtmOpen(pAppHdl);
...
/* assign szResName with the logical name of the card terminal
managing the SIS */
szResName =Config.CT.ResourceName;
/* Attach application to card terminal */
RetCode = CtmResOpen(AppHdl, szResName, pCTHdl);
...
/* assign szResName with the logical name of the SIS_CP
managing the SIS */
szResName = Config.SIS_CP.ResourceName;
/* Attach application to SIS_CP */
RetCode = CtmResOpen(AppHdl, szResName, pCPHdl);
...
/* assign szResName with the logical name of the SIS card */
szResName = Config.SIS.ResourceName;
/* Attach application to SIS resource */
RetCode = CtmResOpen(AppHdl, szResName, pSISHdl);
...
/* assign szResName with the logical name of the SAM card */
szResName = Config.SAM.ResourceName;
/* attach application to SAM resource */
RetCode = CtmResOpen(AppHdl, szResName, pSAMHdl);
...

/* assign SzPrompt, EffectsFlags, TimeOut with data extracted
from application configuration data */
SzPrompt = Config.SAM.Open.SzPrompt;
EffectsFlag = Config.SAM.Open.EffectsFlag;
TimeOut = Config.SAM.Open.timeout;
*plenATR = length(ReceiveBuffer);
...
/* Open SAM */
RetCode = CtmCardOpen (SAMHdl, SzPrompt, EffectFlags, TimeOut,
             plenATR, pATR, pCardState, pStatus);
...
/* Check SAM ATR against expected ATR structure /*
```

```
...
/* Prepare command to verify SAM PIN */
if Config.SAM.Open.GetPINonHost
     GetFromKeyboard(PINtoSubmit);
     SecHandle = 0;
else
     Command = CT_PIN;
     Retcode = CmCtApplicationExchange (CTHdl, CPHdl,
                      CommandLen, pCommand, pRepLen,
                      pResponse, pStatus);
     /* PIN to submit stored by the SIS_CP */
     PINtoSubmit = "";
     SecHandle = CPHdl;
...
/* Prepare command to verify SAM PIN */
Command = C_VERIFY_PIN + PINtoSubmit;
...
/* SEND Command - The SecHandle is passed to the CTM to let it
decide whether to use the SIS_CP stored PIN (SecHandle =
CPHdl, or to use the PIN value sent within the command
(SecHandle = 0) */
RetCode = CtmCardCommand (SAMHdl, SecHandle, CommandLen,
                   pCommand, pRepLen, pResponse, pCardState,
                   pStatus);
...
/* initialisations done */
```

Transaction execution

Once resources have been opened, the application is ready to process transactions involving SIS cards.

Introduction

Each transaction contains a patient identification step.

This identification is achieved by

- reading the SIS card, or
- any other means specific to the application.

In any cases, the SIS data are required to proceed to the execution of the transaction. If the SIS is not accessed during the transaction, this means that SIS stored data can be retrieved from application storage (files) to perform the transaction.

Wheather~~n~~ the data used during a transaction derive from a Belgian SIS private data file or not, it should be accompanied by the SFDF read certificate obtained when effectively reading the private data from the card.~~, it is mandatory to terminate the transaction by making it signed by the BNCS module.~~

General Transaction Steps

1. Patient identification
2. If necessary, Get private data
3. If necessary, Update private data
4. Optionally, Sign Transaction

Patient Identification

Using a physical card

When using a physical card, it is mandatory to first identify it as a SIS or as any other recognised foreign card, and then to load or activate the appropriate card server.

Finally, the card data allowing to identify the patient can be retrieved by the application through the card server when it is activated. The following pseudo-code describes the identification steps:

```
/* Decide to use the physical card */
...
/* assign SzPrompt, EffectsFlags, TimeOut with data extracted
from application configuration data */
SzPrompt     = Config.SIS.Open.SzPrompt;
EffectsFlag  = Config.SIS.Open.EffectsFlag;
TimeOut      = Config.SIS.Open.timeout;
*plenATR     = length(ReceiveBuffer);
...
/* Open SIS */
RetCode = CtmCardOpen (SISHdl, SzPrompt, EffectFlags, TimeOut,
             plenATR, pATR, pCardState, pStatus);
...
/* Check SIS ATR against known ATR structure */
if NotBelgianSISDetected
     ...
     /* out of scope */
else
     LoadBNCS();
     ...
     /* attach application to BNCS */
     RetCode = BncsInitialise(SISHdl, SAMHdl,
                   Config.SIS.CfgFileName, pCSTracingData);
     /* Store BNCS tracing data for further use */
     ...
     /* assign DataSet, AuthMode */
     DataSet = C_ISDF_AND_PBDF;
     AuthMode = C_AUTHENTICATE;
     AccessKeys = C_NULL_STRING; /* 16 access keys */
     /* Read and get public data */
     RetCode = BncsReadData (SISHdl, DataSet, AuthMode,
                             ppAccessKeys);
     ...
     DataSet = C_ISDF;
     RetCode = BncsGetData(SISHdl, DataSet, pDataLen, ppData,
                           pReadCertDataLen, ppReadCertData);
     /* Extract data fields relevant to the application from
        the ISDF ASN.1 structure */
     ...
     /* For Pharmacy applications: To later build the
        transaction, store the following data :
        Card Logical Number,
        Social Security Identification Number,
     */
     ...
     DataSet = C_PBDF;
     RetCode = BncsGetData(SISHdl, DataSet, pDataLen, ppData
                           pReadCertDataLen, ppReadCertData);
     /* Extract data fields relevant to the application from
        the PBDF ASN.1 structure */
     ...
     /*  For  Pharmacy  applications:  To  later  build  the
        transaction, store the following data :
        Holder Name,
        Holder First Name,
     */

/* Use extracted data */

...

/* Patient identification done */
```

Using stored data

When using stored card data, the patient identification step is fully under application control. No calls to the BNCS or CTM APIs are required until the patient is recognised as a SIS cardholder, in which case the BNCS must be loaded and activated by a call to BncsInitialise() to prepare for the next steps.

```
/* Decide to use card stored data */
...
/* Identify the patient by means specific to the application
(customer number, other card, name ...) */
...
/* Retrieve the patient public data from storage */
...
if NotBelgianSISCardholder
     ...
     /* out of scope */
else
     LoadBNCS();
     ...
     /* attach application to BNCS */
     RetCode = BncsInitialise(SISHdl, SAMHdl,
                  Config.SIS.CfgFileName, pCSTracingData);
     /* Store BNCS tracing data for further use */
     ...

/* Use retrieved data */

...

/* Patient identification done */
```

Private data extraction

Using a physical card

The private data extraction phase mandatory follows the patient identification phase. The pseudo-code hereby describes the data extraction steps:

```
/* Physical card being accessed */
...
/* Need to obtain some private data */
```

```
    ...
/* assign DataSet, AuthMode */
DataSet = C_SFDF;
AuthMode = C_AUTHENTICATE;
for (i=0;i<16;i++)   /* 16 access keys */
      AccessKeys[i] = Config.SIS.Appl[AppNr].AccessKey[i];
/* Read and get private data */
RetCode = BncsReadData (SISHdl, DataSet, AuthMode,
                            ppAccessKeys);
...
RetCode = BncsGetData(SISHdl, DataSet, pDataLen, ppData,
                      pReadCertDataLen, pReadCertData);
/* Extract data fields relevant to the application from the
SFDF ASN.1 structure */
/* For Pharmacy applications: To later build the transaction,
store the following data :
      Insuring Company Number,
      Insuring Company Affiliate Identification,
      Social Insurance Data version,
      Social Insurance status,
      SFDF Read Certificate (ReadCertData)
*/
...
/* Use extracted private data */
...
```

Using stored data

To use stored data, all application relevant private data should be known to the application . No calls to the BNCS or CTM APIs are therefore required.

```
/* Continue using card stored data */
...
/* Retrieve stored private data */
...
/* Use retrieved private data */
...
```

Private data update

Updating a SIS card always require to have the physical card available. The update step mandatory follows patient identification and private data extraction steps. The following pseudo-code describes the update steps:

```
/* Physical card being accessed */
...
/* Patient identification performed */
...
/* Private data extraction performed */
...
/* Retrieve new SFDF file record in export format from a local
   or distant server into the “Data” field */
...
DataSet = C_SFDF;
DataFormat = C_EXPORT;
```

```
/* update private data */
RetCode = BncsWriteData (SISHdl, DataSet, DataFormat, DataLen,
                         pData);
...
RetCode = BncsGetWriteStatus(SISHdl, DataSet);
...
```

Transaction termination and certification

When terminating a transaction involving private data originating from a Belgian SIS card, the application must prepare a transaction record which contents may be sealed for integrity and authentication purposes. The certified transaction record contains the following data:

- Application data (including data read from the SIS card),
- Terminal and middleware identification data,
- The standard transaction signature provided by the BNCS.

Remark : The application requirements set forth to prepare this document do not require the final transaction certification step allowed by the BNCS_API : the call to BncsCheckAndSign() or BncsSignTransaction() are therefore optional.~~.~~

```
/* All transaction data are collected - terminate */
...
/* Retrieve stored SIS/BNCS data :
      Card Logical Number,
      Social Security Identification Number,
      Holder Name,
      Holder First Name,
      Insuring Company Number,
      Insuring Company Affiliate Identification,
      Social Insurance Data version,
      Social Insurance status,
      SFDF Read Certificate (ReadCertData)
/* Add application data */
      Transaction date
      Transaction sequence number
      Acquirer Id (Pharmacy ID for instance)
/*  Get  TracingData  (this  could  better  be  performed  once,
during the initialising phase) */
Retcode = CtmAutotest(CTHdl, pCTTracingDataLen,
                        pCTTracingData, pStatus);
...
/*  Format  transaction  (as  defined  in  this  document  for
pharmacy applications) */
...
/* Store transaction record to forward it later to the central
system */
...
/* Require card ejection if physical SIS was used */
if ActualSISUsed
      SzPrompt = Config.SIS.Close.SzPrompt;
      EffectsFlag = Config.SIS.Close.EffectsFlag;
      TimeOut = Config.SIS.Close.timeout;
      RetCode = CtmCardClose (SISHdl, SzPrompt, EffectFlags,
                        TimeOut, pCardState, pStatus);

...
/* Terminate the session with the BNCS */
RetCode = BncsTerminate(SISHdl);
...
UnoadBNCS(); /* optional */
...
/* Prepare for the next transaction */
```

Application close down

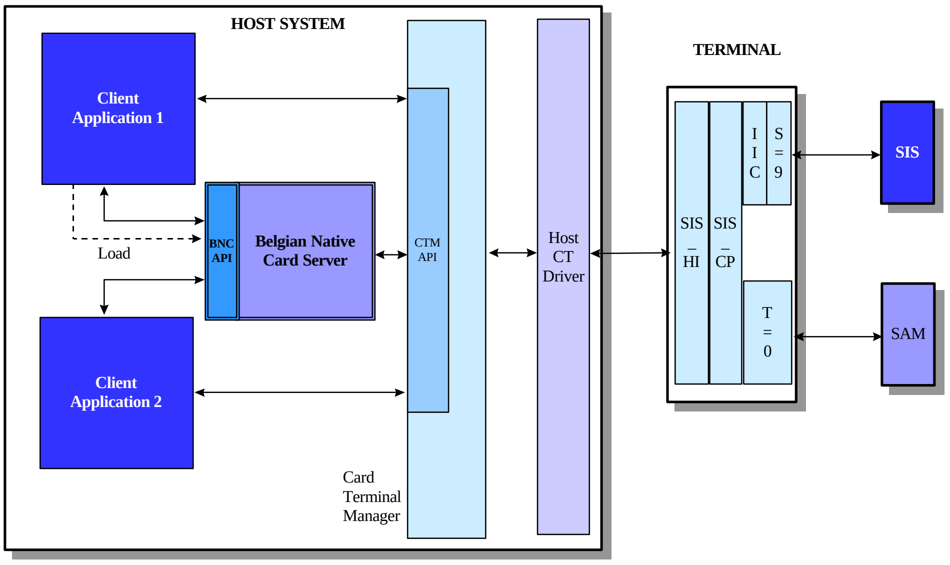
Upon application termination, a clean close down procedure is recommended, particularly to deactivate the SAM card.
The following API call sequence describes a classical close down process.

```
...
/* Eject SAM card */
SzPrompt = Config.SAM.Close.SzPrompt;
EffectsFlag = Config.SAM.Close.EffectsFlag;
TimeOut = Config.SAM.Close.timeout;
RetCode = CtmCardClose (SAMHdl, SzPrompt, EffectFlags,
                         TimeOut, pCardState, pStatus);
...
/* Free resources within CTM */
RetCode = CtmClose(AppHdl);
...
```

Multi-applications environment

Introduction

The figure below illustrates the case where two applications need accessing the same terminal and card devices as it could happen in systems offering multiple workstations sharing one single terminal.



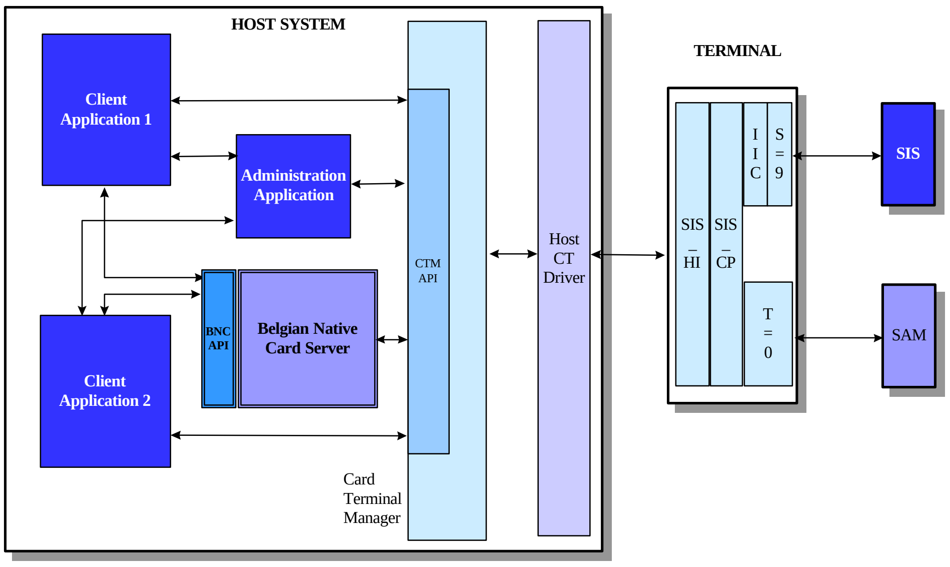
Such architectures require the application to manage concurrent accesses to the shared peripheral devices by using the CTM and BCNS API facilities.

Managing exclusivity

Applications must protect their SAM and SIS resources by locking them when necessary so as to manipulate consistent data during a whole transaction. The current chapter illustrates the use of exclusive control.

Application shared services

In multi-applications environment, some set up, close down and administration application services should be shared and therefore be only done once for the account of all applications by a common application performing shared utilities. The resulting architecture corresponding to the example is as follows:



The additional application can also be used to perform administration activities such as monitoring terminal and card status by using the CtmCt interface functions.
Such reconfiguration affects the API call sequences described above in the following ways:

Administrative application processing

Like any applications, the administration application has three main execution steps:
1. Application start up,
2. Loop executing transactions,
3. Application termination.

Start up

The administration application should be started before any client application. When starting, the administration application must perform the following:
1. Attach to the CTM,
2. Ask to open the SAM resource
3. If the SAM PIN has to be introduced from the terminal keyboard:
    4. Open the CT resource
    5. Open the SIS_CP resource,
6. Ask to open the SAM card,
7. Ask the operator to key in the SAM PIN,
8. Submit the PIN for validation to the SAM.

Loop executing transactions

When started, the administrative application offers interactive or automatic administration services to the operator, such as collecting terminal and cards status, inquiring trace messages, changing SAM PIN value, changing SAM etc. Setting exclusive control on the SAM card is not mandatory for those tasks since the client applications already protect their resources when necessary.

Termination

Upon termination, the administrative application can require to eject the SAM card, and always detaches from the CTM.

Client applications processing

Some functions performed by the administration application such as checking the SAM PIN are not performed anymore by the client applications, which in turn have to manage possible concurrent access to the SIS and SAM resources. This is performed by using the resource locking mechanism provided by the CtmSetExclusivity() function.

Client application start up

```
/* check Administrative application status */
...
/* attach application to CTM */
RetCode = CtmOpen(pAppHdl);
...
/* assign szResName with the logical name of the card terminal
*/
szResName =Config.CT.Open.ResourceName;
/* Attach application to card terminal */
RetCode = CtmResOpen(AppHdl, szResName, pCTHdl);
...
/* Get TracingData to add to further transactions */
Retcode = CtmAutotest(CTHdl, pCTTracingDataLen,
                        pCTTracingData, pStatus);
/* store TracingData */
...
/* assign szResName with the logical name of the SIS card */
szResName = Config.SIS.Open.ResourceName;
/* Attach application to SIS resource */
RetCode = CtmResOpen(AppHdl, szResName, pSISHdl);
...
/* assign szResName with the logical name of the SAM card */
szResName = Config.SAM.Open.ResourceName;
/* attach application to SAM resource */
RetCode = CtmResOpen(AppHdl, szResName, pSAMHdl);
...
/* initialisations done */
```

Patient Identification

Changes only concern the call sequence when using a physical SIS card:

```
/* Decide to use the physical card */
...
/* Set exclusive control on the SIS */
ExclScope = C_LOCK_CARD;
for (i=1;
      i < Config.SIS.Open.timeout/C_EXCL_LOOP_DELAY AND
      Retcode == CTM_RESOURCE_LOCKED;
      i++)
       Retcode = CtmSetExclusivity(SISHdl, ExclScope);
...
/* assign SzPrompt, EffectsFlags, TimeOut with data extracted
from application configuration data */
SzPrompt     = Config.SIS.Open.SzPrompt;
EffectsFlag  = Config.SIS.Open.EffectsFlag;
TimeOut      = Config.SIS.Open.timeout;
*plenATR     = length(ReceiveBuffer);
...
/* Open SIS */
RetCode = CtmCardOpen (SISHdl, SzPrompt, EffectFlags, TimeOut,
             plenATR, pATR, pCardState, pStatus);
...
/* Check SIS ATR against known ATR structure */
if NotBelgianSISDetected
      ...
      /* out of scope */
else
      LoadBNCS();
      ...
      /* attach application to BNCS */
      RetCode = BncsInitialise(SISHdl, SAMHdl,
                      Config.SIS.CfgFileName, pCSTracingData);
      /* Store BNCS tracing data for later use */
      ...
      /* Set exclusive control on the SAM */
      ExclScope = C_LOCK_CARD;
      for (i=1;
            i < Config.SIS.Open.timeout/C_EXCL_LOOP_DELAY
            AND Retcode == CTM_RESOURCE_LOCKED;
            i++)
           Retcode = CtmSetExclusivity(SAMHdl, ExclScope);
      ...
      /* assign DataSet, AuthMode */
      DataSet = C_ISDF_AND_PBDF;
      AuthMode = C_AUTHENTICATE;
      AccessKeys = C_NULL_STRING; /* 16 access keys */
      /* Read and get public data */
      RetCode = BncsReadData (SISHdl, DataSet, AuthMode,
                                  AccessKeys);
      ...
      DataSet = C_ISDF;
      RetCode = BncsGetData(SISHdl, DataSet, pDataLen, ppData);
```

```
     /* Extract and store data fields relevant to the
        application from the ISDF ASN.1 structure */
     ...
     DataSet = C_PBDF;
     RetCode = BncsGetData(SISHdl, DataSet, pDataLen, ppData);
     /* Extract and store data fields relevant to the
        application from the PBDF ASN.1 structure */
     ...
/* Use extracted data */

...

/* Patient identification done; SIS and SAM still owned by
current application through an exclusivity setting */
```

Private Data Extraction

No changes are necessary since a prior patient identification step is required and the SIS and SAM are therefore still owned by the current application through an exclusivity setting.

Private Data Update

No changes are necessary since a prior patient identification step is required and the SIS and SAM are therefore still owned by the current application through an exclusivity setting.

Transaction certification and termination

When starting this phase, the SIS is not required anymore. The lock owned by the current application on the SIS should therefore be released as soon as possible to allow other applications to read the SIS.
The lock on the SAM can also be released.

```
/* All transaction data were collected - terminate */
...
/* reset exclusivity over SAM */
ExclScope = C_UNLOCK_CARD;
Retcode = CtmSetExclusivity(SAMHdl, ExclScope);
...
/* Require card ejection if physical SIS was used, and reset
exlusivity */
if ActualSISUsed
     SzPrompt = Config.SIS.Close.SzPrompt;
     EffectsFlag = Config.SIS.Close.EffectsFlag;
     TimeOut = Config.SIS.Close.timeout;
     RetCode = CtmCardClose (SISHdl, SzPrompt, EffectFlags,
                      TimeOut, pCardState, pStatus);

     ...
     ExclScope = C_UNLOCK_CARD;
     Retcode = CtmSetExclusivity(SISHdl, ExclScope);
...
/* Retrieve stored SIS/BNCS data :
     Card Logical Number,
     Social Security Identification Number,
     Holder Name,
     Holder First Name,
     Insuring Company Number,
     Insuring Company Affiliate Identification,
     Social Insurance Data version,
     Social Insurance status,
     SFDF Read Certificate (ReadCertData)    */
/* Add application data
     Transaction date
     Transaction sequence number
     Acquirer Id (Pharmacy ID for instance) */
...
/* Retrieve TracingData from storage */
...
/*  Format  transaction  record  and  store  it  on  a  permanent
storage device */
...
/* Terminate the session with the BNCS */
RetCode = BncsTerminate(SISHdl);
...
UnoadBNCS(); /* optional */
...
/* Prepare for the next transaction */
```

Multi-terminals environments

Introduction

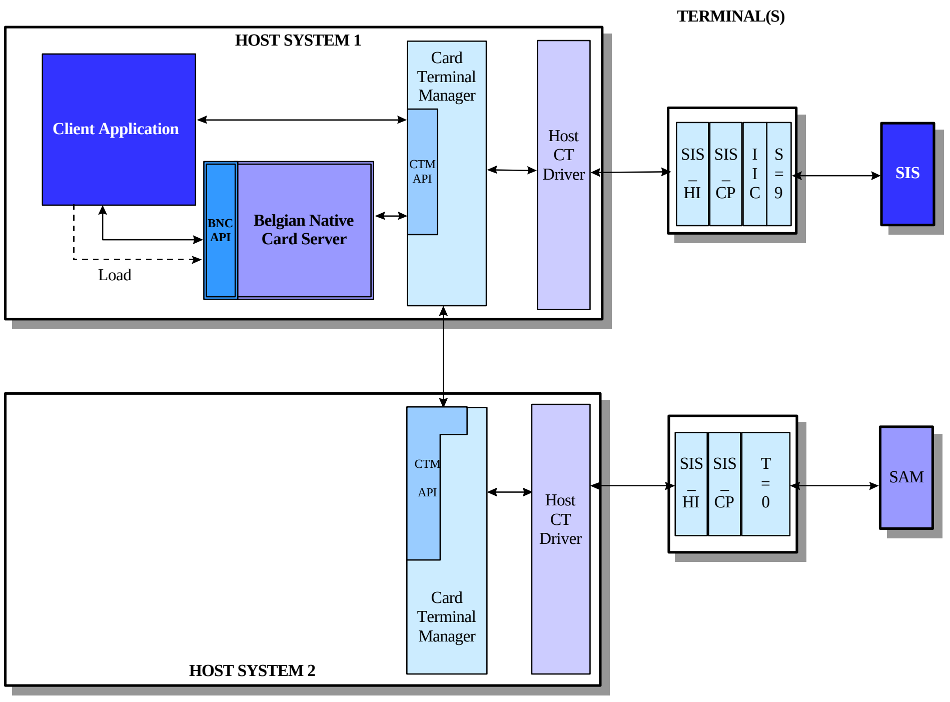


Figure 2 : Sample hardware and software environment

The configuration to which this chapter refers is illustrated by the above figure. This case study consists of two interconnected host computers (host systems 1 and 2), each managing the access to one terminal. The terminal connected to Host 1 allows reading and updating the SIS card, and the terminal connected to Host 2 allows interfacing with the SAM card. How are both hosts interconnected is not relevant to the application, provided that the used protocols implement CTM-API data exchanges.

Multi-terminals environments are transparent to the applications thanks to the CTM routing services. The application will however know more than one terminal (CT and SIS_CP) resources. In some cases, the application will also know more than one SIS or SAM resources. If necessary, the CTM could perform SAM resources multiplexing to dispatch the SAM services requests workload on more than one single SAM. Setting exclusivity on a SAM resource protects the application from disseminating co-ordinated requests on multiple SAMs.
It is also possible to manage multiple SIS readers, but the client application must address each one individually to allow the operator to inform the cardholder of the reader location.

No differences with the multi-applications environment are therefore expected on the application processing flow, except for the following aspects:

- The TracingData value depends on the used SIS terminal. It is consequently required to call CtmCtAutotest given the accurate SIS managing terminal resource name during each transaction execution step.
- The administration application normally runs on the host to which the terminal is attached and will only perform the corresponding initialisation steps.

It is important to notice that the applications must be aware, like the CTM of the hierarchical relationships between the card, terminal and SIC_CP resources. This knowledge is obtained by reading a configuration file or by querying the CTM through the non-standard CTM_API calls described in ***R5***.

Example of configuration file:

Resource		Parent Resource
Name	Type	Name
SIS1	SIS	CPT1
SIS2	SIS	CPT2
SAM	SAM	CPT1
CPT1	CP	TRM1
CPT2	CP	TRM2
TRM1	TRM	
TRM2	TRM	

Example of CTM_API query:

```
retcode = CtmGetResParent (pResName, pRestype, pParentRes);
```

store topology record in dynamic table

Start-up - Initialising cards and terminal services

The following API call sequence applies to a multi-terminal environment with more than one SIS and only one SAM. The error handling is not developed here.

```
/* attach application to CTM */
RetCode = CtmOpen(pAppHdl);
...
```

```
/* Get the hierarchical link SIS - Card Terminal */
n = 0;
do
      szResName = Config.SIS[n].ResourceName;
      retcode = CtmGetResParent (szResName, pRestype,
                   Config.SIS[n].CPParentName);
      ...
      retcode = CtmGetResParent (pCPResName[n], pRestype,
                   Config.SIS[n].CTParentName);
      ...
while (retcode == 0 && n < Config.SISNumber);

/build list of disctinct card terminal in Config.CT */
...
for (i=0;i < Config.CTNumber;i++)
      /* Attach application to card terminal */
      RetCode = CtmResOpen(AppHdl, Config.CT[i].ResName,
                          pCTHdl[i]);
      ...
for (i=0;i < Config.SIS.Number;i++)
      /* Attach application to SIS */
      RetCode = CtmResOpen(AppHdl, Config.SIS[i].ResName,
                          pSISHdl[i]);
      ...
...
/* assign szResName with the logical name of the SAM card */
szResName = Config.SAM.ResourceName;
/* attach application to SAM resource */
RetCode = CtmResOpen(AppHdl, szResName, pSAMHdl);
...

/* assign SzPrompt, EffectsFlags, TimeOut with data extracted
from application configuration data */
SzPrompt = Config.SAM.Open.SzPrompt;
EffectsFlag = Config.SAM.Open.EffectsFlag;
TimeOut = Config.SAM.Open.timeout;
*plenATR = length(ReceiveBuffer);
...
/* Open SAM */
RetCode = CtmCardOpen (SAMHdl, SzPrompt, EffectFlags, TimeOut,
             plenATR, pATR, pCardState, pStatus);
...
/* Check SAM ATR against expected ATR structure /*
...
/* Prepare command to verify SAM PIN */
if Config.SAM.Open.GetPINonHost
      GetFromKeyboard(PINtoSubmit);
      SecHandle = 0;
else
      /* Get SAM environment */
      retcode = CtmGetResParent (Config.SAM.ResouceName,
                       pRestype, Config.SAM.CPParentName);
      retcode = CtmGetResParent (Config.SAM.CPParentName,
                       pRestype, Config.SAM.CTParentName);
      /* Open SAM CP */
      RetCode = CtmResOpen(AppHdl, Config.SAM.CPParentName,
                     pSAMCPHdl);
      /* Open SAM CT */
      RetCode = CtmResOpen(AppHdl, Config.SAM.CTParentName,
                     pSAMCTHdl);
      /* Obtain PIN from terminal keyboard */
      Command = CT_PIN;
      Retcode = CmCtApplicationExchange (SAMCTHdl, SAMCPHdl,
                       CommandLen, pCommand, pRepLen,
                       pResponse, pStatus);
      /* entered PIN stored by the CP */
      PINtoSubmit = “”;
      SecHandle = SAMCPHdl;
...
/* Prepare command to verify SAM PIN */
Command = C_VERIFY_PIN + PINtoSubmit;
...
/* SEND Command - The SecHandle is passed to the CTM to let it
decide whether to use the SIS_CP stored PIN (SecHandle =
CPHdl, or to use the PIN value sent within the command
(SecHandle = 0) */
RetCode = CtmCardCommand (SAMHdl, SecHandle, CommandLen,
                   pCommand, pRepLen, pResponse, pCardState,
                   pStatus);
...
/* initialisations done */
```