Telium Development Suite
Training course

23/11/2010 – Kassovic – Support

TDS Training : **Contents**

- **Hardware and software architecture**
- **Software development steps and tools**
  - Install Ingedev, LLT from Ingedev, Simulation, Remote debugger, etc…
- **OS APIs**
  - Peripherals, Multitask, Flash/Ram, Services etc…
- **Security**
  - SST/SAT from Ingedev, Schemes, Secret area, etc…
- **Manager**
  - Mutli-application purpose, Graphic management, etc…
- **Miscellaneous**
  - Fonts, Message multi-language, etc…

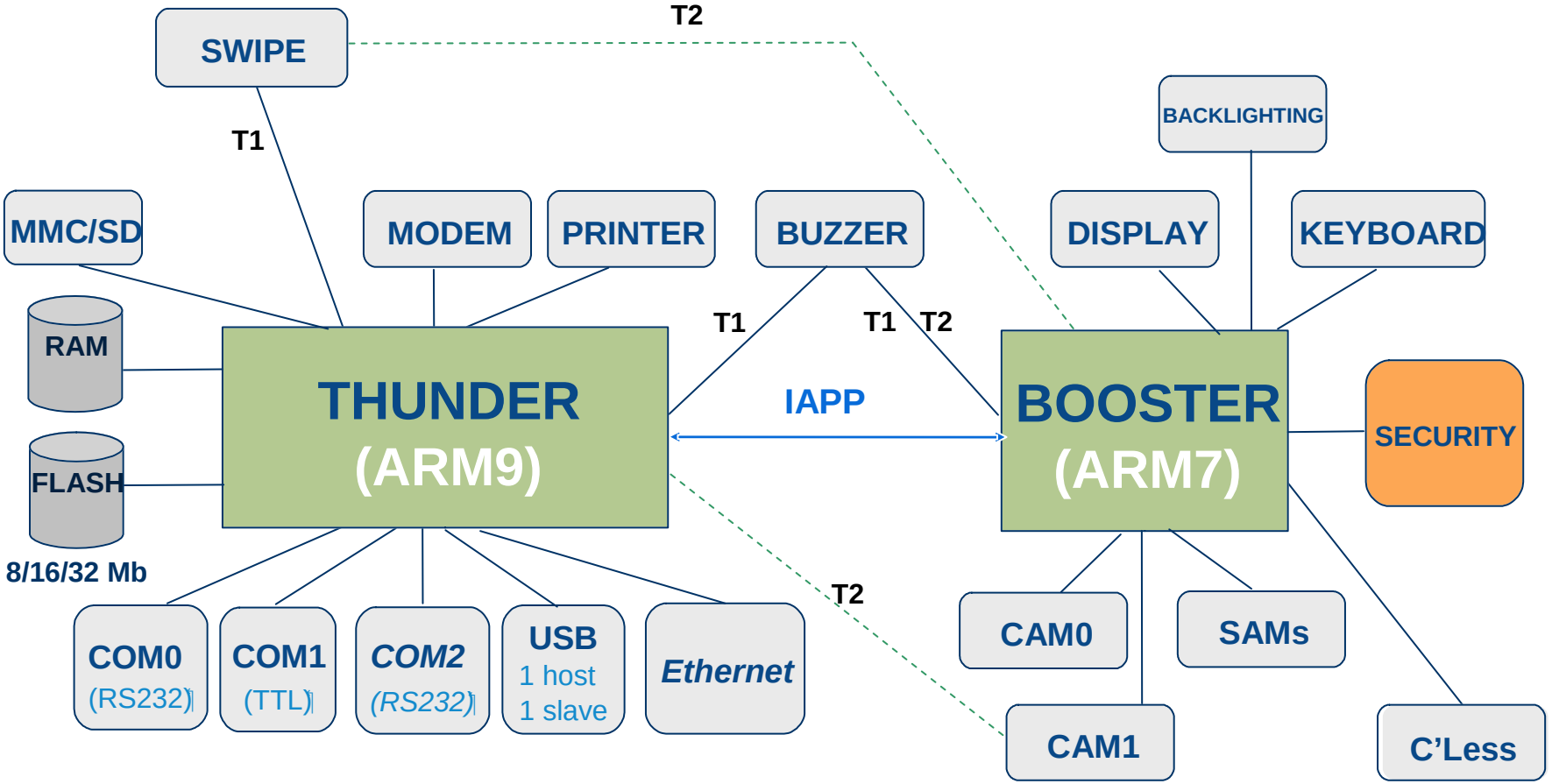
TDS Training : **Keywords**

- **DLL :** Library executed in the same context of the caller
- **Manager :** A particular application managing all others
- **Thunder :** Main processor running applications
- **Booster :** Crypto-processor running schemes
- **Scheme :** Small security application running in Booster
- **LLT :** Local Loading Tool
- **SST :** Software signature tool (Telium 1)
- **SAT :** Software authentication tool (Telium 2)
- **SKMT :** Initial key loading tool
- **Ingestate / TMS :** Terminal Management System

TDS Training : **Contents**

- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

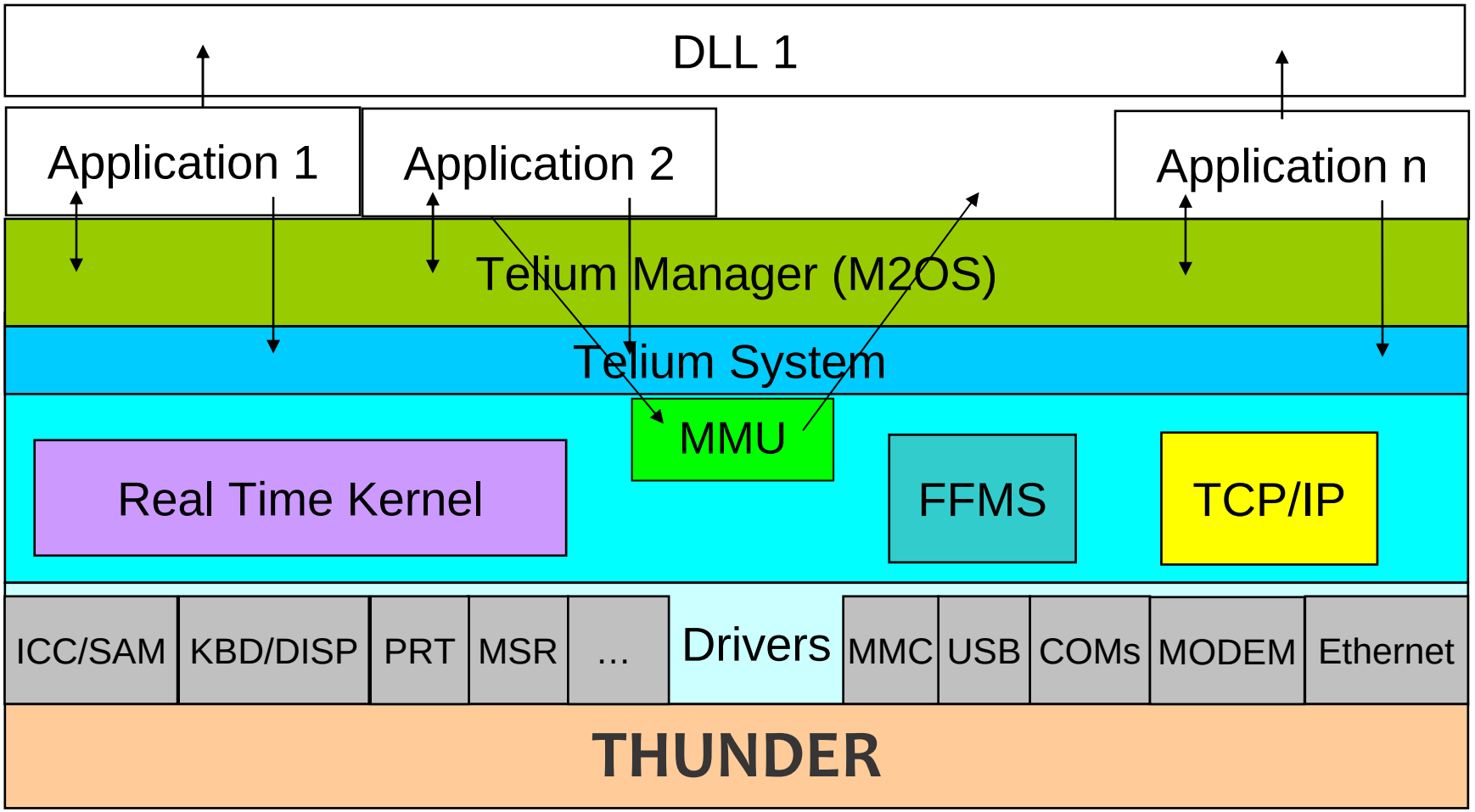
TELIUM : **Hardware architecture**



T1 = TELIUM 1

T2 = TELIUM 2

TELIUM: **Software architecture**



Telium product family : **Terminals 1/3**

Architecture based counter top terminal

EFT930S/SGE/SGEM
T1 (Thunder1-Booster1)

SMART2
T1 (Thunder1-Booster2)

ICT220/250
T2 (Thunder2-Booster3)

Architecture based mobile terminal

EFT930 (infra-red, GSM/GPRS, Bluetooth, Bio , WiFi)
T1 (Thunder1-Booster1)

EFT930CC (infra-red, GSM/GPRS, Bluetooth)
T1 (Thunder1-Booster2)

Telium product family : **Terminals 2/3**

Architecture based retail pin pad

ML30
(Multi-line peripheral)
T1 (Thunder1-Booster2)

IPP320/350 (New!)
(Multi-line peripheral)
T2 (Thunder2-Booster3)

Architecture based none programmable pin pad

PP30S
(Simple pinpad)
T1 (Booster2)

P30
(Graphic pinpad)
T1 (Booster2)

IPP220/250 (New!)
(Simple pinpad)
T2 (Booster3)

Telium product family : **Other products 3/3**

External contact-less reader

TeliumPass
C'less reader
T1 (Booster2)

Specific terminal

TWIN30
(health terminal)
T1 (Thunder1)

ISC350
(Signature-capture Video)
T2 (Thunder3-Booster3)

Architecture based unattended solution

CAD30 UCM
T1 (Thunder1)

CAD30 UCR
T1 (Booster2)

CAD30 UPP
T1 (Booster2)

TELIUM : File extensions (1/2)

Unsigned Binary files

- .BIN **Unsigned File (.MOT or .SKD for schemes)**
- **.TXT Descriptor File**

Signed files Telium 1

- .SGN Signed file
- .ADF **Application Descriptor File**
- .LDF **Library Descriptor File (DLL)**
- .PDF **Parameter Descriptor File**
- .DDF, .BDF **Driver Descriptor File (reserved INGENICO)**

Signed files Telium 2

- **.AGN Signed Application File**
- .LGN Signed Library File (DLL)
- .PGN Signed Parameter File
- .DGN Signed Driver File (reserved INGENICO)
- **Descriptor File embedded inside Signed File**

TELIUM : File extensions (2/2)

- **Catalog files (list of component loaded by LLT)**
  - .M30 **EFT30/SMART1**
  - .M31 **EFT930x, (including EFT930SGEM, EFT930W, EFT930B, EFT930G)**
  - .M32 **Twin30**
  - .M34 **UCM/CAD30 (vending machine)**
  - .M36 **ML30**
  - .M37 **SMART2**
  - **.M39** **EFT930 SGEM-C2 (color screen / contactless)**
  - **.M40** **ICT 220, ICT250(color screen)**
  - **.M43** **ICS 350 (touch screen)**
  - **.M46** **IPP320, IPP350**

- **Diagnostic file (available from LLT)**
  - .DIA Information regarding Thunder and Booster

TELIUM: File descriptor and values (set in Ingedev)

Field name	Size	Format	value
file type	1	hex	0 for parameter file, **1 for application**, 2 for DLL, 3 for driver
file name	15 max	Alpha	**File name with extension in UPPER CASE**
application type	4	hex	**Type of application**, unique in terminal. Range given by Ingenico for each VAR
application family	15 max	ALPHA UPPER	Generic name application this file belongs to **In case of DLL, this is the name to use for DLLLink()**
zip flag	1	0 = no / 1 = yes	**Compression indicator:** 0 for mock-up, 1 for a real terminal
crc flag	1	0 = no / 1 = yes	**CRC indicator:** 0 for mock-up, 1 for a real terminal
crc	4	hex	**CRC** of binary unzipped file computed by SST/SAT, Ignored for mock-up
unzipped file size	8 digits	hex	**Original (.SGN/.AGN)** file size computed by SST/SAT
zipped file size	8 digits	hex	**Zipped (.SGN/.AGN)** file size computed by SST/SAT, Ignored for mock-up
data size	8 digits	hex	Max **data size to be computed by ingedev** : search for **_bss_end** in MAP file **Data size = [(_bss_end value) – 0x200000] rounded to upper 4Kb**
code address	8 digits	Hex	**0 for an application,** **code base address for a DLL**
data address	8 digits	hex	**0x200000 for an application,** **data base address for a DLL**

Example: SAMPLE01.ADF

1,SAMPLE01.SGN,006E,MYSAMPLE,1,1,45E3,00003908,0000253E,00001000,0000000,00200000

TELIUM : Os + Manager pack

```
file type, file name, application type, application family, zip flag, crc flag, crc, unzipped size, zip size, data size, code @, data @

Parameter   OS                              0,  37770645.SGN,0000,  37770645,0,0,0000,00000788,       0,00000000,00000000,00000000
Parameter   MANAGER (Pack M2OS)              0,  37784400.SGN,0002,      M2OS,0,0,0000,00000684,       0, 0000000,       0,  000000
Application MANAGER (M2OS)                  1,  35920620.SGN,0002,      M2OS,1,1,DA85,0002B11C,0001D281,00012647,       0,  200000
Library     DLL     (WIFI)                  2,8136650207.SGN,001C,      WIFI,1,1,B122,0000B724,00008DB7,0000C000,E02F0000,E02FC000
Library     DLL     (Hardware Cnf)          2,  36420221.SGN,0014,     HWCNF,1,1,84C6,000116C0,0000BF83,000004BC,E0268000,E0288000
Library     DLL     (Extension)             2,  35940720.SGN,0001,    EXTENS,1,1,BD73,0001428C,0000E76F,0000BA0E,E007C000,E0092000
Library     DLL     (Graphic Library)       2,  35960440.SGN,0003,     LIBGR,1,1,1058,0000E580,00009676,00005864,E0062000,E0073000
Library     DLL     (M2OS Parameters)       2,  35970520.SGN,0004,     PARAM,1,1,C820,0000AE58,0000767F,00000AF6,E00A7000,E00B8000
Library     DLL     (Entry Keyboard)        2,  35980440.SGN,0005,    SAISIE,1,1,0F02,00006754,000046DF,00000F30,E0046000,E004E000
Library     DLL     (SV Cam/date Library)   2,  35990520.SGN,0006,        SV,1,1,68C9,000059F0,00004287,000007B3,E003C000,E0045000
Library     DLL     (PSC Protocol)          2,  36010340.SGN,0008,    DLLPSC,1,1,EF18,00001884,0000126A,000002DC,E0059000,E005B000
Library     DLL     (NULL Protocol)         2,  36020350.SGN,0009, PROTOCOLE,0,1,DE46,00000CAC,       0,00000030,E0200000,E0210000
Library     DLL     (Cryptographic Library) 2,  36060190.SGN,000A,    CRYPTO,0,1,19FA,000052B4,       0,00000018,E0227000,E022F000
Library     DLL     (EMV Selection)         2,  36080380.SGN,000C,     EMVSQ,1,1,DCB2,00005590,00003F10,00008334,E0021000,E002A000
Library     DLL     (IAM)                   2,  36090350.SGN,000D,       IAM,1,1,3783,00000C58,000008AE,00000424,E0053000,E0055000
Library     DLL     (PINPAD)                2,  36100360.SGN,000E,    PINPAD,1,1,750B,00001A14,0000109D,00000010,E0056000,E0058000
Library     DLL     (PPR30 Export)          2,  36110237.SGN,000F,     PPR30,1,1,CDD8,00009158,00005D70,00002000,E00B9000,E00C6000
Library     DLL     (PPS30)                 2,  36130205.SGN,0007,     PPS30,1,1,8E28,000043E4,00002A74,00000C00,E005C000,E0061000
Library     DLL     (GPRS)                  2,  36210480.SGN,0012,      GPRS,1,1,4124,00007A34,00005739,0000052C,E0250000,E0258000
Library     DLL     (UMS)                   2,  36160350.SGN,0013,       UMS,1,1,4740,00003008,000020CC,00008114,E025A000,E025F000
Driver      OS      (BOOTRAM)               3,  30140303.SGN,0000,   BOOTRAM,0,1,B5D3,00001D40,       0,       0,       0,       0
Driver      OS      (MODEM 930)             3,  40150264.SGN,0105,  40150264,1,1,3E2E,000752D4,0004E30B,0004E000,F1000000,F1E00000
Driver      OS      (TS SYSTEM)             3,8200360473.SGN,0001,8200360473,1,1,1B75,000B62E8,0007B44A,       0,       0,       0
Driver      OS      (VFS)                   3,  30210304.SGN,0103,       VFS,1,1,ED4E,0000AA4C,000088BD,    2000,F1500000,F1580000
Driver      OS      (Booster Full EMV2000)  3,  30468515.SGN,0002,  30468515,0,1,4329,0001323C,       0,       0,       0,       0
Driver      OS      (GPRS 930)              3,8200120136.SGN,0131,8200120136,1,1,2DE2,00006AC0,00004E90,00004000,F1650000,F1660000
Driver      OS      (BT Socle 930)          3,8200140129.SGN,0214,   BTSOCLE,1,1,F7F3,00024D54,00019913,   10000,F1680000,F16B0000
Driver      OS      (BT Portable 930)       3,8200150129.SGN,0215,BTPORTABLE,1,1,28A0,00025308,00019BBE,   10000,F1680000,F16B0000
Driver      OS      (UMS)                   3,8200210201.SGN,0133,      DUMS,1,1,9E23,00002884,00001C70,    1000,F1800000,F1880000
Driver      OS      (USBHID)                3,8200220111.SGN,0126,   DUSBHID,1,1,3C9C,000011A4,00000D03,    2000,F1620000,F1630000
Driver      OS      (USBCDC)                3,8200320302.SGN,8000,   DUSBCDC,1,1,7F79,00004918,00002AD8,    3000,F1900000,F1920000
Driver      OS      (TS WIFI)               3,8200540208.SGN,8006,8200540208,1,1,2914,0001A308,0001339A,00020000,F1C00000,F1C30000
```

TELIUM : Range of Application Name and Type

- **A range of numbers is given by Ingenico**
- **File name : range 822400 to 822499 build on 10 digits**
  - Application name (2 last digit represent software ID from 00 to 99)
  - 2 digit version
  - 2 digit release
- **Application TYPE : range 22400 (0x5780) to 22499 (0x57E3)**
- **Example for first application v01.01**
  - File Name = 8224000101
  - Application type = 0x5780 (22400 dec)
- **File name and application type are also apply for DLLs in the same range**
  - A DLL with same "application type" overwrite the existing one, but a DLL will never overwrite an application

TDS Training : **Contents**

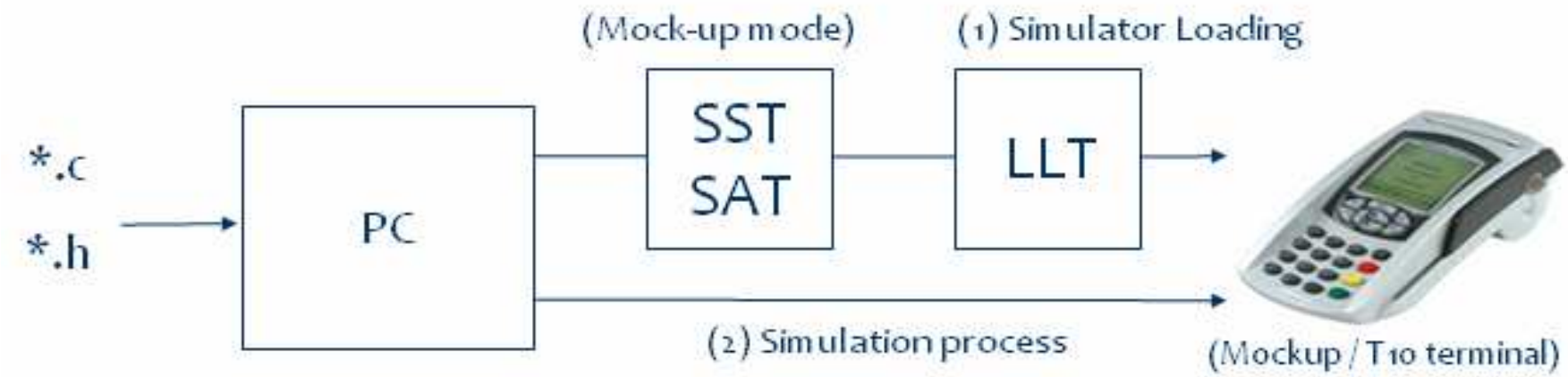
- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

TELIUM DEVELOPMENT : **Environment 1/3**

3 environments are available

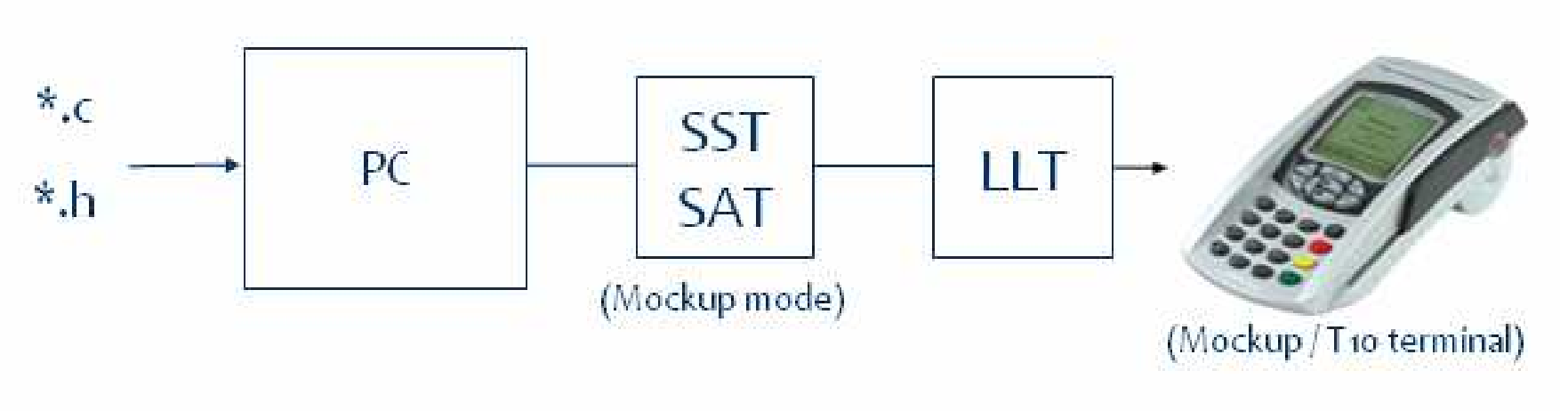
- **Simulation**
  - Require a simulator application to be loaded into the terminal
  - DO NOT need a downloading after each code modification
  - Some restrictions
  - Obsolete, no more evolution since 2009, replaced by Remote Debugger
- **Remote Debugger**
  - Require a downloading after each code modification
  - Allow debugging multitasking, DLL, etc…
  - Debug / Mockup / T10 mode mandatory
- **Release**
  - Final step to create "on the field" loadable application
  - Require final real signature tool

TELIUM DEVELOPMENT : **Simulation mode 1/3**



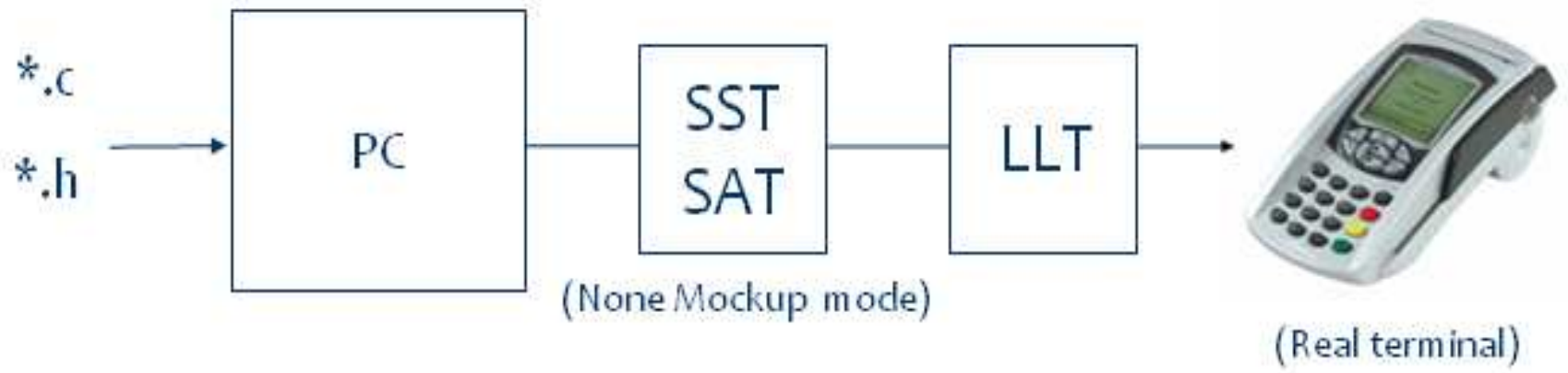
- Pc executes all standard C code
- Terminal executes all Telium APIs
- No downloading or signature process is required
- All the power of Ingedev debugging on PC
- Decrease development time
- Some restrictions (Task(s) & DLL(s) cannot be simulated)
- Obsolete, no more evolution since 2009

TELIUM DEVELOPMENT : **Remote debug mode (2/3)**



- All application is executed on terminal
- No signature process required (or only if scheme utilisation)
- Only for mock-up terminal
- Software downloading necessary (automatically done by Ingedev)

TELIUM DEVELOPMENT : **Release mode (3/3)**



- This is the final condition to run a real software
- Signature, downloading and secure terminal are required

TELIUM : **TIPS for easier development (1/2)**

ingenico

- **Do not provide these tips to merchant, for development only**
  - **Reset terminal**
    - Press «.» and «Yellow» keys at the same time
  - **Turn to LLT mode**
    - Reset then hole «arrow up» key or access loading menu throug F0211
  - **Delete all software, including parameters files**
    - Reset, then at boot hold F1 («*» is displayed)
      Then press quickly F4 («**» is displayed)
      Then press quickly F2 («***» is displayed)
      Then press quickly F3 («LLT» is displayed)

    (F1, F2, F3, F4, delete all software except parameters files)

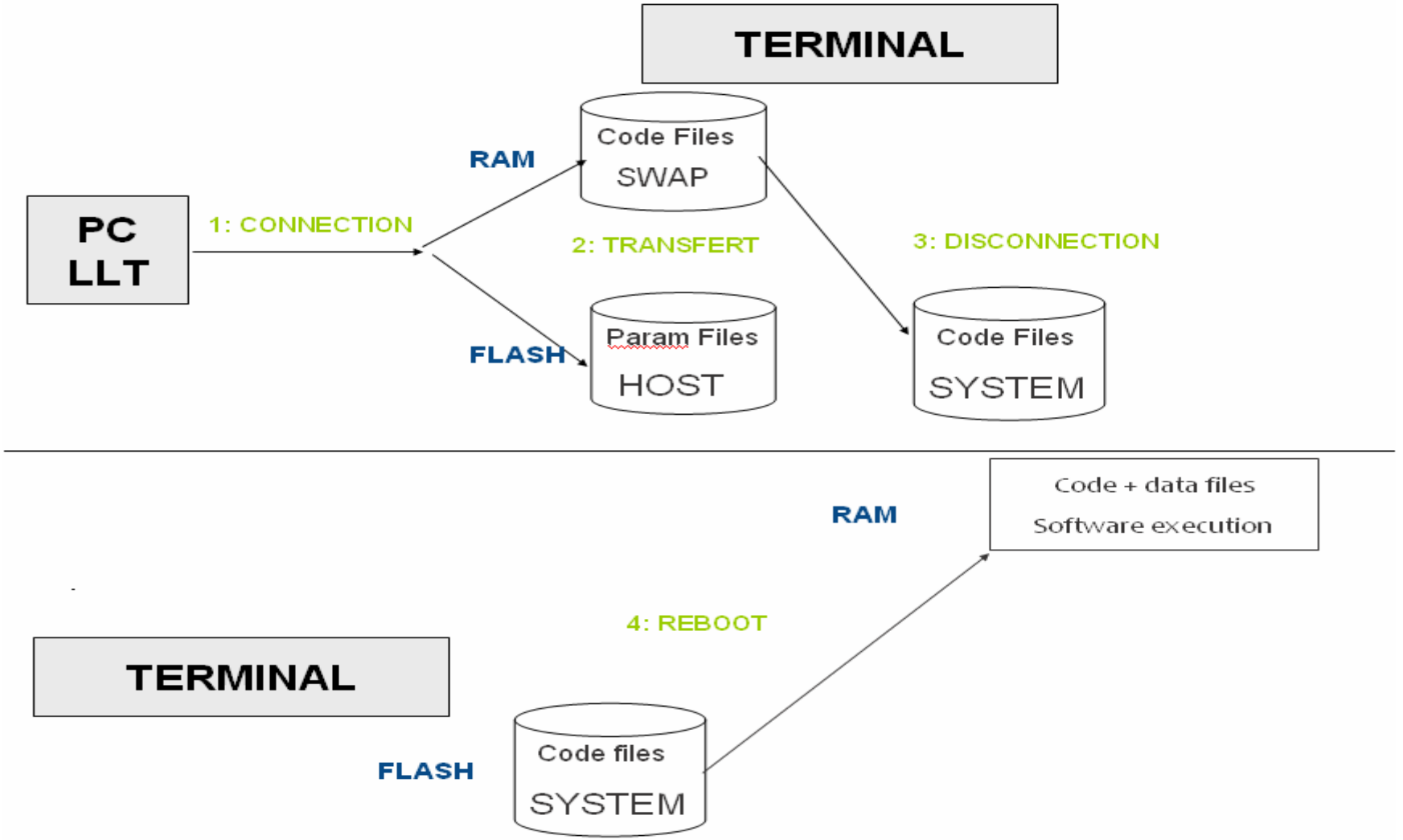
    (Note that LLT or Simulator will work only if trace is not activated on USB. If EFT resets :
    execute «reset» + «arrow up» to force LLT mode, download system.cfg with disable trace and restart
  - **Delete one application**
    - **Reset, then at boot when** «- - -» appears, press quickly «F2» (or «2» on ICT220/250)
      In menu, select «deletion», select one application, press «enter» and reset terminal again
      Enter code «26» for selection by number, code «07» for selection in a list

ingenico

TELIUM : **TIPS for easier development**

- **Do not provide these tips to merchant, for development only**
  - **To wake up an EFT930**
    - Press «paper feed» key
  - **To turn it off an EFT 930**
    - Execute a reset
      (If outside the base, it will turn off. If on the base, it will just reset and stay power on)
  - **Paper feed**
    - Press «yellow» key and hold for one second on ICT220/250
  - **On ML30 and 930M**
    - **Delete all : F1/F4/F2/F3 sequence = Green**/«.»/Up/Down
    - Delete menu : F2 (after « - - - ») = Up/Down

TELIUM : **Loading steps**



TELIUM TOOLS: **INGEDEV Install**

- **Install Ingedev_7.6.0.6.exe**
- **Select components to install**
  - Ingedev 7.6.0.6
  - SDK Telium 7.4
  - LLT 4.3.4
  - SST30 5.1.0 / SAT 1.5.1
- **Ingedev 7.6.0.6**
  - Follow the default installation
- **LLT 4.3.4 (embedded in Ingedev setup)**
  - Install USB drivers
  - Install manually PPPOE following user's guide
- **SST30 5.1.0 (embedded in Ingedev setup)**
  - Install card reader driver following user's guide
- **SAT 1.5.1 (embedded in Ingedev setup)**

TELIUM TOOLS : **LLT install (1/3)**

- Connect EFT to USB and select XP EFT driver (Telium Tools\LLT\Drivers)
- Install RAS Windows XP according to LLT user's guide
- Setup comXX parameters for selected port (115200 baud)

- Connect to terminal
- 2 Windows PC and Terminal
- File transfer using drag and drop
- Code Files are loaded in the SWAP temporary disk of terminal (or HOST for parameters files)
- Disconnect from Terminal and OS activates software

- Mater of care: On portable EFT930 series, there are RTS/CTS on Com0
  => if serial com port used, activate "hardware flow control"

TELIUM TOOLS : **Using LLT (2/3)**

- Three Activities :
  - Downloading: local loading of software and signed parameter files (stored in SYSTEM disk and never deleted… file read by application at each boot)
  - Diagnostic: upload of .DIA extension files for analysis (diagnostic, counters, user trace files…)
  - Maintenance: Local downloading of plaintext parameter files (stored in HOST disk deleted by application after reading)
- Filter Utilisation :
  - Filter Mxx (Catalogue of files) => Select Mxx, content will be downloaded
  - Filter *.* (List of files) => Select exclusively:
    - .adf, .sgn or .agn files for downloading an application
    - .par, .xxx for downloading parameter files
    - Do not transfer a .Mxx file in *.* filter as file itself will be downloaded such as a parameter file
- Always use .Mxx catalogue to prevent inconsistencies !!!

TELIUM TOOLS : **Using LLT (3/3)**

- System :
  - Mockup.Mxx for mock-up terminal
  - Prod.Mxx for real terminal
  - Mater of care: A mock-up cannot be turned into a real terminal
  - In case a mock-up is turned by error into real mode => EFT deactivated ":-(", read the document "How to reactivate a terminal.pdf" provided with the SDK pack
- Manager M20S :
  - Manager and it's DLLs
  - Export.Mxx (International) or Health.Mxx (French Sesam Vitale) or Internal.Mxx (French Banking)
- Your Applications

TELIUM TOOLS : **Simulator Install on EFT (1/3)**

- **Simulator becomes obsolete**
  - No more evolution since 2009 and NOW replaced by Remote Debugger
- **Download Simulator application**
  - Turn terminal on LLT mode
  - Open Ingedev, select a project under simulation configuration
  - Choose Terminal \ Download Telium Simulation App
- **Telium Simulator on screen**
  - Show Version, Appli type and Com port
  - By default Com port: 0 (Rs232)
- **Com port configuration**
  - After loading, reset and press repetitively at boot "0" for Com0 Serial Port use or "5" for USB use

TELIUM TOOLS : **Simulator Install on PC (2/3)**

- Create a simulation configuration containing the project to simulate
- Set a simulation parameters to run the project
- The simulation configuration can be executed in Run mode or Debug mode
- Use simulator according to Ingedev user's guide

TELIUM TOOLS : **Simulator restrictions (3/3)**

- Only 1 application can be simulated at a time
- Debugging introduces delays, so beware when you have real- time concern (rather use Trace Tool or Remote Debugger for time measure)
- Simulator uses 1 port (RS232 or USB) on EFT
- Forked tasks cannot be simulated
- DLL cannot be simulated

TELIUM TOOLS : **Remote Debugger Install on EFT (1/2)**

- LDBG located ..\SDK30\TDSxx\Component\Os\
  - 813304xxxx.ddf + 813304xxxx.sgn
  - Already loaded with the system
- SYSTEM_CFG located ..\SDK30\TDSxx\Component\Os\Debug
  - Using USB link, add to SYSTEM_CFG: LDBG_DEV=5 (at least on the first time)
- Download SYSTEM_CFG using LLT (Swap disk)
- After loading, the terminal shows a line describing the communication link and the communication parameters (see display). The terminal is ready to use Remote Debugger
- Recompile the project to debug under GNU_ARM_DEBUG and download it
- To disable LDBG
  - Reboot terminal and press F on LDBG setup screen and enter "0" on port value
  - To enable, do the same process and enter "5" on port value

TELIUM TOOLS : **Remote Debugger Install on PC (2/2)**

- Create a remote debugger configuration containing the project to debug
- Set a remote debugger parameters to debug the project
- The remote debugger can be used only in Debug Mode
- The real application code is debugged
- Use remote debugger according to Ingedev user's guide

TELIUM TOOLS : **Trace Install (1/2)**

- Trace folder is to be copied on PC disk
- Trace.ini file configuration required for PC com port setting
- Terminal cable on Comx or USB between terminal and PC
- Tracing any OS calls
- Tracing any application call through
  - trace(SAP, length, *data)
- Trace can be named
  - Define each trace SAP code in both your application and inside trace_tool.cfg
- Trace can be filtered
  - Each SAP consists in 2 bytes: first byte is family and second byte is event name
  - First byte is to be added in trace.ini file section “filter” set to 0 or 1 to respectively hide or see the trace by default. Can be changed later using “filter” menu in trace tool
- Trace is displayed
  - As SAP and time value ss.mm seconds, hexa value of selected area displayed, ASCII value of the same area

TELIUM TOOLS : **Using Trace (2/2)**

- **Port selection for trace :**
  - Edit file SYSTEM.CFG (\SDK30\TDSxx\Component\OS\Debug)
  - Select NOTRACE, COM0, COM1, COM2 or USB(COM5)
  - Load SYSTEM_CFG into SWAP file of EFT using LLT
  - Reset EFT manually (LLT disconnect is not enough to read new settings)
  - To disable trace, reload SYSTEM_CFG with NOTRACE setting
- **Matter of care :**
  - Each time terminal is resetting (because PC loose/reconnect the USB link), trace tool also loose the connection and do not reconnect automatically. So each time you reset the terminal, you have to disconnect/reconnect the trace tool too.

TELIUM TOOLS : **Eclipse Shortcuts**

ingenico

- Useful shortcuts :
  - Ctrl+Click : Open Declaration/Open Include
  - Ctrl+Space in editor : Content Assist
  - Ctrl+Shift+Space in editor, inside function parameters : Display function parameters
  - On a selected function Right click/Open call Hierarchy : Show where the function is called
  - Ctrl+Shift+R : Open any resource of the workspace (Navigate/Open Resource menu)
  - "Include Browser" view : Drag and drop a .c or .h file to this view to watch the include hierarchy
  - Ctrl+Shift+T : Open a C element of the workspace => function, struct, enum, etc... (Navigate/Open Element menu)
  - Ctrl+Shift+L : Display all shortcuts (Help/Key Assist menu)

TDS Training : **Contents**

- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

TELIUM : **OS APIs**

ingenico

Standard C and Peripheral Management

- **C standard routines :**
  - fopen(), fread(), fwrite(), fclose(), fprintf(), pprintf(),
  - putc(), fputc(), putchar(), puts(), fputs(),
  - getc(), ungetc(), getchar(), fgetc(), fgets()
  - ferror(), clearerr(), exit() …
- **TELIUM specific routine :**
  - ttestall(), status(), mask_event(), reset_buf()

ingenico

TELIUM OS APIs : **Peripheral synchronisation 1/4**

- **ttestall :** wait for selected input-output events
  - Place the caller in a standby status and wait for reactivation on a peripheral event

- unsigned int **ttestall** (unsigned in mask, unsigned int timeout)
  - mask in form of 32 bits word :
    - KEYBOARD, PRINTER, COM0, COM1, MODEM, CAM0, CAM1, SWIPE2, SWIPE31, COM5, COM6, CAM2, COM2, MOUSE, KEYUSB, SWIPE3, …
    - Or 0 if no peripheral expected (ttestall (0, 100) = wait only for time out 1s
    - timeout operates in units of 10 milliseconds. (ttestall (0, 0) = infinite wait)
- Returns mask or 0 for timeout
    - BIT 0: KETBOARD, BIT 1: PRINTER, BIT 2: COM0, … (see oem_public.h)
    - Latest bits of field (bit 32 and before) can be used for task synchronisation.

TELIUM OS APIs : **Peripheral synchronisation 2/4**

Events available for each peripheral:

– **KEYBOARD** :	KEYS_IN_BUFFER	At least one key has been pressed
– **PRINTER** :	PRINT_END	End of printing (buffer OS)
	PRINT_BUFFER_EMPTY	Buffer empty (buffer appli)
	PRINT_ERROR	Printing Error (no paper, paper jam)
– **SWIPEx** :	TRACK_READ	Swipe read => buffer available
– **CAMx** :	CAM_PRESENT	Card inserted
	CAM_ABSENT	Card absent
	CAM_REMOVED	Card removed
– **COMx** :	COM_SEND_END	End of transmission (buffer OS)
	COM_SEND_EMPTY	Send buffer empty (buffer appli)
	COM_REC_NOT_EMPTY	At least on character has been received
– **MODEM** :	Same as COM + DCD (lost of carrier)	
– **KEYUSB** :	COM_REC_NOT_EMPTY	At least one key has been pressed
– **MOUSEUSB** :	MOUSE_PRESENT	Click or position change

TELIUM OS APIs : **Peripheral synchronisation 3/4**

- **mask_event :** filter events
  - Used to received only selected events of one peripheral, hide others
- **Status :** test status on a peripheral
  - Which event? In case peripheral has several ones …
  - int status (FILE *stream, unsigned char *status)
    - Return EOF if task dialogue is not in progress
- **Reset_buf :** reset a peripheral file
  - void reset_buf (FILE *stream, unsigned char id_buffer)
  - _send_id = transmission buffer
  - _receive_id = reception buffer

TELIUM OS APIs : **Peripheral synchronisation 4/4**

Wait for multiple events

```
reset_buf (fModem, _receive_id);                    // Reset receiving buffer
mask_event (fModem, COM_REC_NOT_EMPTY);             // Mask event to COM_REC_NOT_EMPTY only, hide others
fprintf (fModem, "ATD,T00475810363\r");             // Sending dialing command
ret = ttestall (MODEM | KEYBOARD, 10*100)

// use a list of "if" with a mask on each event to prevent losing simultaneously event!!!
if (MODEM & ret){
    lng = fread(dial_buffer, 1, 3, fModem);         // Check AT OK
}

if (KEYBOARD & ret){
    keycode = getc(kbd);                            // Get key pressed
}

if (ret==0){
    // Case of timeout
}

// All events are catched in this case !!!
```

TELIUM OS APIs : **Display 1/2**

- **Peripheral name : *DISPLAY*** (standard stdout output device)
  - **Display functions**
    - fopen(), fclose() (enable, disable display)
    - putc(), puts(), printf() (write character, string and formatted string)
  - **Graphic display 128*64 or 128*128**
  - **Back lighting fully handled by OS (may be customized)**
  - **Flashing characters & cursor management can be set**
    - Blink(), gotoxy(), wherex(), wherey()
  - **Font size can be changed (width and height)**
    - font_size()
  - **.BMP logo display**
    - Defdisplaypattern()
  - **Only available on B&W terminal**
    - **Use Graphic library instead to keep compatibility between B&W and color terminal**
    - **More details on Manager section (see later)**

TELIUM OS APIs : **Display 2/2**

ingenico

Sample

```
FILE *hDsp=NULL;

hDsp = fopen("DISPLAY", "w"); CHECK(hDsp!=NULL, lblKO);      // Open "display" channel

font_size (6,8);                                             // **** Small font 8x21 ****
putchar('\x1B');                                             // Clear screen
gotoxy(0,0);   ret=printf("Small-1"); CHECK(ret>=0, lblKO);  // Display
gotoxy(1,2);   ret=printf("Small-2"); CHECK(ret>=0, lblKO);
gotoxy(2,4);   ret=printf("Small-3"); CHECK(ret>=0, lblKO);
gotoxy(3,6);   ret=printf("Small-4"); CHECK(ret>=0, lblKO);  //          with
gotoxy(4,8);   ret=printf("Small-5"); CHECK(ret>=0, lblKO);
gotoxy(5,10);  ret=printf("Small-6"); CHECK(ret>=0, lblKO);
gotoxy(6,12);  ret=printf("Small-7"); CHECK(ret>=0, lblKO);
gotoxy(7,14);  ret=printf("Small-8"); CHECK(ret>=0, lblKO);  //                  gotoxy
ttestall(0, 3*100);                                          // Delay 3s

font_size (8,16);                                            // **** Normal font 4x16 ****
putchar('\x1B');                                             // Clear screen
ret=printf("Normal-1        \n"                               // Dispaly
           "  Normal-2      \n"                               //         with
           "     Normal-3   \n"
           "        Normal-4\n"); CHECK(ret>=0, lblKO);       //                crlf
ttestall(0, 3*100);                                          // Delay 3s

font_size (13,32);                                           // **** Big font 2x8 ****
putchar('\x1B');                                             // Clear screen
blink(ON);                                                   // Blinking effect start
ret=printf("Big-%d   \n"                                     // Display
           "   Big-%d\n", lig1, lig2); CHECK(ret>=0, lblKO); //          with  format

fclose(hDsp);                                               // Close "display" channel
```


ingenico

TELIUM OS APIs : **Printer 1/2**

- **Peripheral name : *PRINTER***
  - **Printer functions**
    - fopen(), fclose() (enable, disable printer)
    - reset_buf(), pprintf() (reset buffer, print formatted string with check of printer flow)
  - **384 pixels width fast thermal printer (15lines/s)**
  - **Different printing modes (through ESCxx sequence)**
    - Single/double width, single/double height
    - Bold type / reverse mode printing
    - Normal mode (24 characters/line) / condensed (48 characters/line)
    - Inter-line spacing modification
  - **Paper-out detection**
  - **.BMP logo printing**
    - Defprinterpattern()
  - **2 specifics routines**
    - Enable/disable paper feed key (paper_feed())
    - Change the left and right margins (format())

TELIUM OS APIs : **Printer 2/2**

- **Sample**

```
FILE *hPrt=NULL;

hPrt=fopen("PRINTER", "w-"); CHECK(hPrt!=NULL, lblKO);                    // Open the "printer" channel

ret=pprintf("\x1b""E""Bold\n\n""\x1b""F"); CHECK(ret>=0, lblKO);          // Bold
ret=pprintf("\x1b""B1""Reverse\n\n""\x1b""B0"); CHECK(ret>=0, lblKO);     // Reverse
ret=pprintf("\x0E""Double Width\n\n""\x14"); CHECK(ret>=0, lblKO);        // Double Width
ret=pprintf("\x1b""H""Double Height\n\n""\x1b""F"); CHECK(ret>=0, lblKO); // Double Height
ret=pprintf("\x0F""48 columns\n\n""\x12"); CHECK(ret>=0, lblKO);          // 48 columns
ret=pprintf("Normal again\n\n\n\n\n\n\n"); CHECK(ret>=0, lblKO);          // Normal

ttestall(PRINTER, 2*100);                                                 // Wait until everything printed

fclose(hPrt);                                                             // Close "printer" channel
```

TELIUM OS APIs : **Keyboard 1/2**

- **Peripheral name : *KEYBOARD*** (standard stdin input device)
  - **Keyboard functions**
    - fopen(), fclose() (enable, disable keyboard)
    - reset_buf(), getchar() (reset keyboard FIFO, read character in keyboard FIFO)
  - **24/22 keys keyboard (SMART/EFT930)**
    - 10 numerical keys and "." key
    - 3 conventional keys : Validation, Correction, Cancel
    - 1 function key F and 4 user functions key F1 to F4
    - Navigation keys UP, DOWN, (C, OK on SMART)
    - 1 "paper feed" key (↑) not available at application level
  - **Special functions**
    - Automatic repeat entry mode (auto_repeat())
    - Place keyboard in alphanumeric mode (key_c_alpha())

TELIUM OS APIs : **Keyboard 2/2**

Sample

```
FILE *hKbd=NULL;

hKbd = fopen("KEYBOARD", "r"); CHECK(hKbd!=NULL, lblKO);          // Open "keyboard" channel
reset_buf(hKbd, _receive_id);                                     // Reset keyboard FIFO

idx = 0;
memset(buf, 0, 21+1);
while(idx <= 21)
{
    key=0;
    ret=ttestall(KEYBOARD, 5*100);                                // Wait for a key pressed until 5s
    if (ret & KEYBOARD)                                           // Key detected
        key = getchar();                                          // Retrieve key pressed

    buf[idx] = key;
    if(!buf[idx])
        break;                                                    // Exit loop if nothing is pressed during 5s
    if(buf[idx] == T_ANN)
        break;                                                    // Exit loop if red key is pressed
    if(buf[idx] == T_VAL)
        buf[idx]='V';                                             // Assign 'V' to green key
    if(buf[idx] == T_CORR)
        buf[idx]='C';                                             // Assign 'C' to yellow key
    if(buf[idx] == T_F)
        buf[idx]='F';                                             // Assign 'F' to function key
    if(buf[idx] == F1)
        buf[idx]='?';                                             // Assign '?' to F1
    gotoxy(3,0); ret=printf(buf); CHECK(ret>=0, lblKO);           // Display key pressed
    idx++;                                                        // One more key
}

fclose(hKbd);                                                     // Close "keyboard" channel
```

TELIUM OS APIs : **Magnetic stripe 1/2**

- **Peripherals name : *SWIPE2 SWIPE31 SWIPE3***
  - **Stripe functions**
    - fopen(), fclose() (enable, disable mag reader)
    - reset_buf(), fread(), is_isox() (reset buffer, read stripe in binary, read stripe in ASCII)
  - **ISO2 as standard and ISO2+1, ISO2+3 or ISO123 as options**
  - **All tracks are read at the same time, 2 directions reader**
  - **ISO checking routines (is_iso1, is_iso2 and is_iso3)**
    - Length, parity bit, LRC, delimiters, Luhn check, numerical check
  - **Possibility to read the track in bits map mode (fread)**

TELIUM OS APIs : **Magnetic stripe 2/2**

Sample

```
FILE *hMag13=NULL, *hMag2=NULL, *hMag3=NULL;

hMag13=fopen("SWIPE31", "r*"); CHECK(hMag13!=NULL, lblKO);      // Open "mag13" channel
hMag2=fopen("SWIPE2", "r*"); CHECK(hMag2!=NULL, lblKO);        // Open "mag2" channel
hMag3=fopen("SWIPE3", "r*"); CHECK(hMag3!=NULL, lblKO);        // Open "mag3" channel

sta = ttestall(SWIPE31 | SWIPE2 | SWIPE3, 0);                  // Wait for the first event ISO1
sta |= ttestall(sta ^ (SWIPE31 | SWIPE2 | SWIPE3), 10);        // Wait for the second event ISO2
sta |= ttestall(sta ^ (SWIPE31 | SWIPE2 | SWIPE3), 10);        // Wait for the third event ISO3

if(sta & SWIPE31)                                              // *** Retrieve and analyse ISO1
{
    memset(tmp, 0, 128);
    ret = is_iso1(hMag13, &len, tmp);                          // ISO1 in ascii format
    if (ret!=ISO_OK)
        IsoError (ret, trk1);
    else{
        // Save ISO1...
    }
}
if(sta & SWIPE2)                                               // *** Retrieve and analyse ISO2
{
    memset(tmp, 0, 128);
    ret = is_iso2(hMag2, &len, tmp);                           // ISO2 in ascii format
    if (ret!=ISO_OK)
        IsoError (ret, trk2);
    else{
        // Save ISO2...
    }
}

if(sta & SWIPE3)                                               // *** Retrieve and analyse ISO3
{
    memset(tmp, 0, 128);
    ret = is_iso3(hMag3, &len, tmp);                           // ISO3 in ascci format
    if (ret!=ISO_OK)
        IsoError (ret, trk3);
    else{
        // Save ISO3...
    }
}

fclose(hMag2);                                                 // Close "mag2" channel
fclose(hMag3);                                                 // Close "mag3" channel
fclose(hMag13);                                                // Close "mag13" channel
```

TELIUM OS APIs : **Chip Card 1/3**

- **Peripherals name : *CAM0 CAM1 SAM1-4***
  - CAM0 : Terminal main card reader (if pinpad connected => terminal becomes CAM2)
  - CAM1 : merchant optional card reader
  - SAM1 to SAM4 : Security module
  - **Adjustable parameters through format (Vcc, Clock) command**
  - **T0 and T1 protocols supported**
  - **ISO 7816 1/2/3 and EMV level 1 compliant**
  - **Synchronous card drivers (only on CAM0 from terminal)**
    - Siemens : S9/S10, SLE4436/5536, SLE4418/4428, SLE4432/4442
    - Gemplus : GPM256, GPM271, GPM416, GPM896

TELIUM OS APIs : **Chip Card 2/3**

Chip functions

- fopen(), fclose() (enable, disable chip reader)

– Power

- Power_on() (Iso7816 and return ATR)
- EMV_power_on() (EMV level 1 and return ATR)
- Power_down() (Power off the chip)

– Input/Output

- EMV_apdu() (Send command and receive answer)
  - » Returns OK or error code

    (card absent, invalid, mute, VCC/VPP problem, communication problem, card removed)
  - » This function can be used for ALL type of card
    - … ISO 7816 cards / Security Modules
    - … EMV cards

TELIUM OS APIs : **Chip Card 3/3**

Sample

```
FILE *hSmc=NULL;

hSmc=fopen("CAM0", "rw"); CHECK(hSmc!=NULL, lblKO);          // Open the main Cam channel


sta = ttestall(CAM0, 60*100);                                // Wait for a smart card inserted until 1mn
CHECK (sta!=0, lblKO);                                       // Timeout => exit


len=0;
ret=EMV_power_on(hSmc, &atr);                                // *** EMV power on and reset ***
if(ret == 5)                                                 // Power once more time  on VPP problem detected
    ret = EMV_power_on(hSmc, &atr);
CHECK (ret==0, lblKO);                                       // Power on error => exit
memcpy(rsp, atr.historic, atr.length);                       // Retrieve ATR and length
len=atr.length;

// ...
// Analyse ATR
// ...

                                                             // *** Select command ***
apduC.length = 5 + 14;                                       // APDU length
memcpy (snd,                                                 // APDU request data cla/ins/P1/P2/Le/data
        "\x00\xA4\x04\x00\x0E"
        "1PAY.SYS.DDF01",
        apduC.length);
apduC.data = snd;                                            // Sending buffer
apduR.data = rsp;                                            // Receiving buffer

ret = EMV_apdu(hSmc, &apduC, &apduR);                        // Call APDU communication routine
CHECK (ret==0, lblKO);                                       // Error exit

// ...
// Retrieve APDU response (rsp) data and analyse it
// ...

fclose(hSmc);                                                // Close "cam" channel

// ...
// Wait until card removed
// ...
```

TELIUM OS APIs : **Communication**

- **Matter of Care**
  - Peripherals name : ***COM0***, ***COM1***, ***COM2***
  - Peripherals name : ***MODEM***
  - Gprs Library : ***GSM***, ***TCP***, ***GPRS***
  - Wifi DLL : ***WIFI***

=> Rather use Link Layer APIs

- **The Link Layer**
  - The Link Layer component is designed to manage all the physical links and protocols available on Telium terminals.
  - This component is an application loaded in the terminal.

TELIUM Link Layer : **Overview 1/9**

- **OSI Model (layer 1 to 6)**
  - Physical : **Rs232**, **Modem**, **USB**, **Ethernet**, **Wifi**, **Bluetooth**, **GSM** and **GPRS**
  - Data Link : **SDLC**, **PPP**
  - Network : **IP**
  - Transport : **TCP**, (**X28** => can be put over all transport layers)
  - Session : identification across the network
  - Presentation : translation from application/network format
- **Mater of care**
  - Standard version (application code 3429xxxx.SGN) does not support IP protocols. Only perform TCP from Wifi or Gprs terminals.
  - TCP/IP version (application code 3628xxxx.SGN) supports PPP, TCP/IP and SSL protocols. It was submitted before under license.

TELIUM Link Layer : **Overview 2/9**

API

- LL_Configure (Create, modify or delete a Link Layer configuration)
- LL_Connect, LL_Disconnect, LL_Send, LL_Receive, LL_GetStatus, LL_GetLastError
- LL_clearSendBuffer, LL_clearReceiveBuffer
- LL_WaitEvent (Wait for the end of the send or for an incoming frame)
- LL_GetHandle (Retrieve handle on specific connection to get further information)

Principles

- The application must create a configuration, the Link Layer will give a handle on it. The application will use this handle to connect, disconnect, send, receive etc…
- The application can use the Link Layer to perform several communications at a time (modem + serial port for example).

Interface objects

- The Link Layer uses the TlvTree interface object to exchange the configuration parameters.
- AVL.lib + LinkLayerExeInterface.lib must be implemented to link properly

TELIUM Link Layer : **Com Port 3/9**

- **COM0/COM2 : RS232/V24 serial port**
  - 5 wires RS232 (GND, Rx, Tx, RTS, CTS), + 5v (for external power)
  - Up to 115Kbauds
- **COM1 : TTL serial port**
  - 5 wires RS232 (GND, Rx, Tx, RTS, CTS), + 5v (for external power)
  - Up to 115Kbauds
  - Dedicated to PP30 pinpad through PP30 DLL provided with manager
- **Protocol setting via => LL_Configure**
- **RTS/CTS controlled by hardware**
- **Buffers size : 1Kb sending / 2Kb receiving**

TELIUM Link Layer : **Com Port 4/9**

Sample

```
byte comNbr, datasize, parity, stopbits;
doubleword bps;
byte comNbr;
LL_HANDLE *ptComSession;
TLV_TREE_NODE hConfig=NULL, hPhysicalConfig=NULL;
int ret;

comNbr = LL_PHYSICAL_V_COM0;                                    // Use COM0
bps = LL_PHYSICAL_V_BAUDRATE_115200;                            // 115200 bauds
datasize = LL_PHYSICAL_V_8_BITS;                                // 8 bits
parity = LL_PHYSICAL_V_NO_PARITY;                               // No parity
stopbits = LL_PHYSICAL_V_1_STOP;                                // 1 stop bit

// *** LinkLayer parameters Root tag of the configuration tree
hConfig = TlvTree_New(LL_TAG_LINK_LAYER_CONFIG);

// *** TAG Physical layer parameters
hPhysicalConfig = TlvTree_AddChild(hConfig, LL_TAG_PHYSICAL_LAYER_CONFIG, NULL, 0);

// Com port number                                  TAG             VALUE       LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_LINK, comNbr, LL_PHYSICAL_L_LINK);
// Baud Rate                                        TAG               VALUE    LENGTH=4bytes
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_BAUDRATE, bps, LL_PHYSICAL_L_BAUDRATE);
// Data bits                                        TAG                     VALUE          LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_BITS_PER_BYTE, datasize, LL_PHYSICAL_L_BITS_PER_BYTE);
// Stop bits                                        TAG                   VALUE         LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_STOP_BITS, stopbits, LL_PHYSICAL_L_STOP_BITS);
// Parity                                           TAG               VALUE       LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_PARITY, parity, LL_PHYSICAL_L_PARITY);

// *** Link Layer configuration
ret = LL_Configure(ptComSession, hConfig);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConfigure);

// *** Release the configuration parameters tree
TlvTree_Release(hConfig);

// *** Link Layer connection
ret = LL_Connect(*ptComSession);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConnect);
```

TELIUM Link Layer : **Modem 5/9**

- **Software modem**
- **V22, V22bis, V32, V32bis** (V22 and V22bis only for SDLC)
- **MNP4/MNP5 compression and error correction**
- **Set of standard AT commands & register** (\r mandatory)
- **Incoming call management** (RING message and ATA command)
- **Line sharing detection**
- **Fast connect mode through specific settings**
- **Buffers size : 1Kb sending / 2Kb receiving**

TELIUM Link Layer : **Modem 6/9**

Sample

```
byte datasize, parity, stopbits;
doubleword bps;
char initString[50], *phone;
LL_HANDLE *ptComSession;
TLV_TREE_NODE hConfig=NULL, hPhysicalConfig=NULL, hDataLinkConfig=NULL;
int blindDial, ret;

bps = LL_PHYSICAL_V_BAUDRATE_1200;                          // Baud rate V22
datasize = LL_PHYSICAL_V_8_BITS;                            // 8 bits
parity = LL_PHYSICAL_V_NO_PARITY;                           // No parity
stopbits = LL_PHYSICAL_V_1_STOP;                            // 1 stop bit
blindDial = 1;                                              // Blind dialling
phone = "169012345";                                        // phone number
timeout = HDLC_TIMEOUT;                                     // Connection timeout

// *** LinkLayer parameters Root tag of the configuration tree
hConfig = TlvTree_New(LL_TAG_LINK_LAYER_CONFIG);

// *** TAG Physical layer parameters
hPhysicalConfig = TlvTree_AddChild(hConfig, LL_TAG_PHYSICAL_LAYER_CONFIG, NULL, 0);

// Modem                                    TAG                  VALUE              LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_LINK, LL_PHYSICAL_V_MODEM, LL_PHYSICAL_L_LINK);
// Baud Rate                                TAG                  VALUE       LENGTH=4bytes
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_BAUDRATE, bps, LL_PHYSICAL_L_BAUDRATE);
// Data bits                                TAG                    VALUE          LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_BITS_PER_BYTE, datasize, LL_PHYSICAL_L_BITS_PER_BYTE);
// Stop bits                                TAG                  VALUE       LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_STOP_BITS, stopbits, LL_PHYSICAL_L_STOP_BITS);
// Parity                                   TAG                  VALUE     LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_PARITY, parity, LL_PHYSICAL_L_PARITY);
// Modem initialisation string
sprintf(initString, "AT%sS144=16S145=1",blindDial ? "X1":"X4");
//                                          TAG                    VALUE
TlvTree_AddChildString(hPhysicalConfig, LL_MODEM_T_INIT_STRING, initString); // LENGTH (strlen initString)
// Phone number                             TAG                    VALUE
TlvTree_AddChildString(hPhysicalConfig, LL_MODEM_T_PHONE_NUMBER, phone); // LENGTH (strlen phone)
```

TELIUM Link Layer : **Modem 7/9**

```
// *** TAG Data link layer parameters
hDataLinkConfig = TlvTree_AddChild(hConfig, LL_TAG_DATA_LINK_LAYER_CONFIG, NULL, 0);

// Hdlc                                          TAG                          VALUE                  LENGTH=1byte
TlvTree_AddChildInteger(hDataLinkConfig, LL_DATA_LINK_T_PROTOCOL, LL_DATA_LINK_V_HDLC, LL_DATA_LINK_L_PROTOCOL);
// Connection timeout                            TAG                        VALUE       LENGTH=4bytes
TlvTree_AddChildInteger(hDataLinkConfig, LL_HDLC_T_CONNECT_TIMEOUT, timeout, LL_HDLC_L_CONNECT_TIMEOUT);
// Minimum tries for sending                     TAG                         VALUE      LENGTH=1bytes
TlvTree_AddChildInteger(hDataLinkConfig, LL_HDLC_T_MIN_RESEND_REQUESTS, 2, LL_HDLC_L_MIN_RESEND_REQUESTS);
// V80 mode                                      TAG                VALUE  LENGTH=1bytes
TlvTree_AddChildInteger(hDataLinkConfig, LL_HDLC_T_V80_MODE, 1, LL_HDLC_L_V80_MODE);

// *** Link Layer configuration
ret = LL_Configure(ptcomSession, hConfig);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConfigure);

// *** Release the configuration parameters tree
TlvTree_Release(hConfig);

// *** Link Layer connection
ret = LL_Connect(*ptcomSession);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConnect);
```

TELIUM Link Layer : **TCP/IP stack 8/9**

- **TCP/IP stack under NexGen Third party license**
  - One shot license, valid for the whole Telium family
  - Socket BSD 4.4 interface
  - FTP, FTPS, HTTP server, SMTP/POP3 protocols
  - SSL client & server (PKI management functions: X509 supports, PKCS12, CSR generation…)
  - DHCP or static Ethernet configuration
  - DNS resolver
- **Over modem, Ethernet, Wifi (WEP, WPA) , COM, USB, GPR S**
- **SSL Mastercard's PTS security program certification**

TELIUM Link Layer : **TCP/IP stack 9/9**

Sample

```
char *addr;
doubleword timeout, port;
LL_HANDLE *ptComSession;
TLV_TREE_NODE hConfig=NULL, hPhysicalConfig=NULL, hTransportConfig=NULL;
int ret;

addr = "192.168.1.2";                                          // IP address
port = 23;                                                     // Port number
timeout = TCPIP_TIMEOUT;                                       // Connection timeout

// *** LinkLayer parameters Root tag of the configuration tree
hConfig = TlvTree_New(LL_TAG_LINK_LAYER_CONFIG);

// *** TAG Physical layer parameters
hPhysicalConfig = TlvTree_AddChild(hConfig, LL_TAG_PHYSICAL_LAYER_CONFIG, NULL, 0);

// Ethernet                                    TAG                  VALUE                    LENGTH=1byte
TlvTree_AddChildInteger(hPhysicalConfig, LL_PHYSICAL_T_LINK, LL_PHYSICAL_V_ETHERNET, LL_PHYSICAL_L_LINK);

// *** TAG Transport and network layer parameters
hTransportConfig = TlvTree_AddChild(hConfig, LL_TAG_TRANSPORT_LAYER_CONFIG, NULL, 0);

// TCP/IP                                   TAG                        VALUE                      LENGTH=1byte
TlvTree_AddChildInteger(hTransportConfig, LL_TRANSPORT_T_PROTOCOL, LL_TRANSPORT_V_TCPIP, LL_TRANSPORT_L_PROTOCOL);
// Host Name                                TAG                 VALUE
TlvTree_AddChildString(hTransportConfig, LL_TCPIP_T_HOST_NAME, addr); // LENGTH (strlen addr)
// Port                                   TAG                VALUE    LENGTH=4bytes
TlvTree_AddChildInteger(hTransportConfig, LL_TCPIP_T_PORT, port, LL_TCPIP_L_PORT);
// Connection timeout                     TAG                                VALUE       LENGTH=4bytes
TlvTree_AddChildInteger(hTransportConfig, LL_TCPIP_T_CONNECT_TIMEOUT, timeout, LL_TCPIP_L_CONNECT_TIMEOUT);

// *** Link Layer configuration
ret = LL_Configure(ptComSession, hConfig);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConfigure);

// *** Release the configuration parameters tree
TlvTree_Release(hConfig);

// *** Link Layer connection
ret = LL_Connect(*ptComSession);
CHECK ((ret==LL_ERROR_OK) || (ret==LL_ERROR_ALREADY_CONNECTED), lblKOConnect);
```

TELIUM OS APIs : **Calendar**

Main characteristics

- Date in the following format: DAY / MONTH / YEAR
- Time in the following format: HOUR / MINUTES / SECONDS
- Leap years handled
- Back up by a lithium battery
- Summer and winter time are not supported
- Matter of care: requires **SV Library** (svlib_open();)

Calendar functions

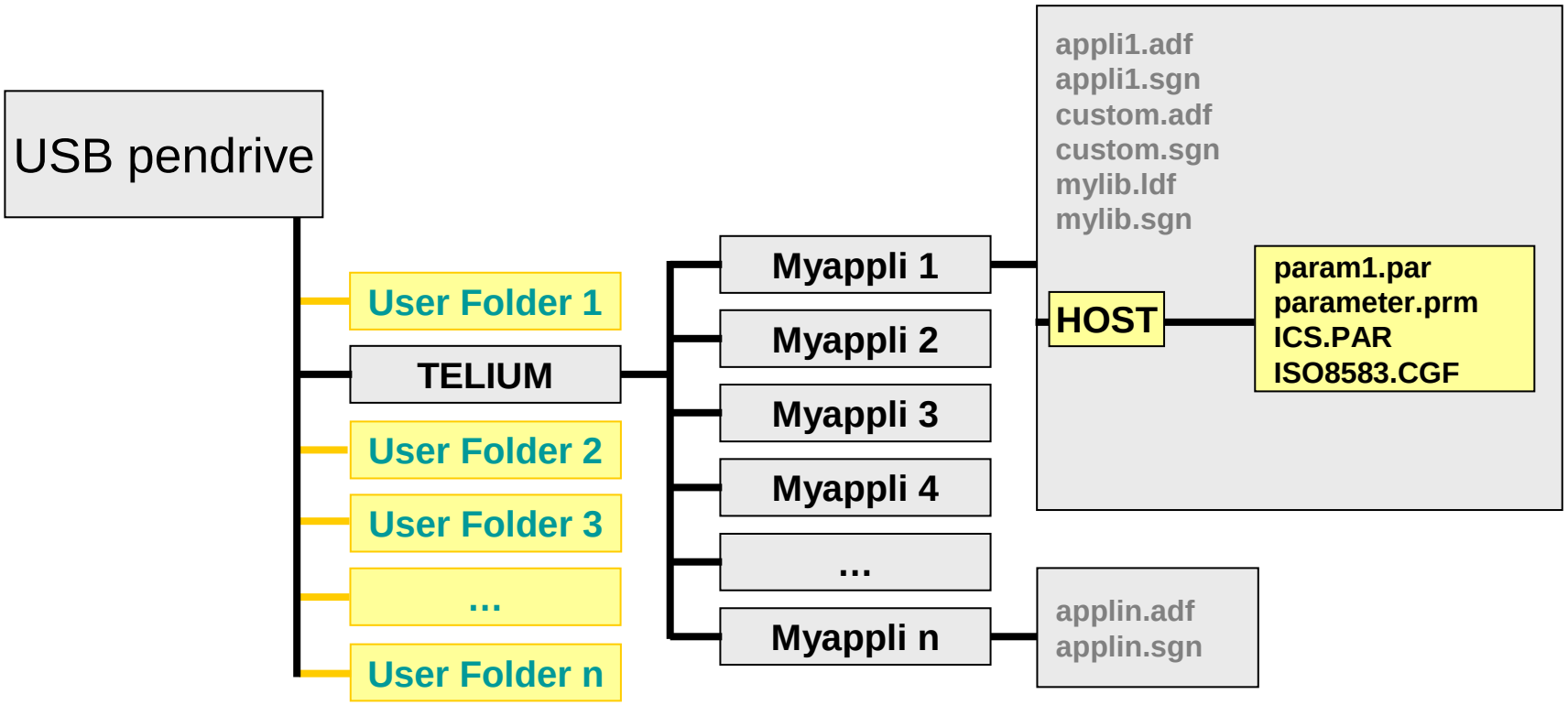
- int write_date (DATE *date)
- int read_date (DATE *date)
- Int ctrl_date (DATE *date)

TELIUM OS APIs : **USB ports 1/2**

- **1 USB host and 1 USB slave (mini size connectors on EFT 930)**
- **USB specific drivers :**
  - Serial line class 1.0 emulation
    - USB slave is COM5
    - USB host is COM6
  - USB to RS232 converter (some brand supported)
  - USB mouse (event generated on position and on click)
  - USB keyboard / Barcode reader (event generated on key pressed)
  - USB key
  - CBM fingerprint reader
  - TeliumPass contact'less card reader
- **Limitations : Fat 16/32 formatting (Max 8Gb)**
  - Max consumption for USB slave devices :
    - *300 mA for EFT30/EFT SMART*
    - *500 mA for EFT930S*
    - *100 mA for other EFT930*

TELIUM: **USB Key application downloading** 2/2

Create folders for application and parameter file loading with USB key :



USB downloading launched by F4 at boot

Caution : « TELIUM » and « HOST » must be in UPPER CASE

TELIUM OS APIs : **Virtual File System 1/4**

- **MMC/SD/µSD connector available for external flash memory**
  - Accessible through VFS driver (Virtual File System)
  - MMC/SD supported on EFT930x
  - µSD supported on SMART2
- **USB key also accessible as external flash memory extension**
  - Accessible through VFS driver (Virtual File System)
- **The VFS library provides a whole set of services allowing the applications to manage MMC card and USB key**
- **MMC and USB must be formatted at MS-DOS standard format**

TELIUM VFS : **Virtual File APIs 2/4**

ingenico

Directory management

- **mount()** : mount a file system
- **mkdir()** : create a directory
- **rmdir()** : delete a directory
- **opendir()** : open a flow on a directory to read files information
- **readdir()** : read files information on a directory (name, creation, size)
- **telldir()** : read directory position
- **rewinddir()** : reset the directory position
- **closedir()** : close a flow on a directory
- unmount() : unmount a file system

TELIUM VFS : **Virtual File APIs 3/4**

ingenico

- **Files management**
  - **open()** : open or create a file
  - **close()** : close a file
  - **unlink()** : remove a file from a directory
  - **read()** : read bytes from a file
  - **write()** : write byte to a file
  - **lseek()** : reposition the file pointer
  - **stat()** : get file status via its files name
  - **fstat()** : get file status via its file descriptor
  - **chmod()** : change file access mode via its file name
  - **fchmod()** : change file access mode via its file descriptor
  - **rename()** : change file name

ingenico

TELIUM VFS : **Virtual File APIs 4/4**

Sample

```
S_FS_PARAM_CREATE xCfg;
S_FS_FILE *pxFile;
char tcData[256];
struct msdosfs_args xFatArgs;
int iFile;
int iLen, iRet;

// *** Insert USB Key
iRet = WaitForUms(30, VFSType); CHECK(iRet==CR_ENTRY_OK, lblKO);          // Wait USB key

// *** Copy data into file VFS.TXT located in USB disk
memset(&xFatArgs, 0, sizeof(xFatArgs));
xFatArgs.fspec = (byte*) "/dev/umsd0a";                                   // USB mass storage device 0 partition a
xFatArgs.mask  = 0666;                                                    // rwx msdosfs perms
xFatArgs.flags = MSDOSFSMNT_LONGNAME;                                     // Constant
xFatArgs.magic = MSDOSFS_ARGSMAGIC;                                       // Constant
iRet = mount(MOUNT_MSDOS, (char*) "/USB", MNT_SYNCHRONOUS, &xFatArgs) ;
CHECK(iRet>=0, lblVfsKO);                                                 // Disk USB initialisation

iFile = open("/USB/VFS.TXT", O_RDWR|O_CREAT|O_TRUNC, 0666);               // Create if not exist and open the file VFS.TXT
CHECK(iFile>=0, lblVfsKO);

strcpy(tcData, "Hello, do you hear me?  \r"                               // Data to write
               "Hello, do you hear me?  \r"
               "Hello, do you hear me?  \r"
               "Hello, do you hear me?  \r"
               "Hello, do you hear me?  \r\n");

iLen = write(iFile, tcData, sizeof(tcData)); CHECK(iLen>=0, lblVfsKO);    // Write data into file

iRet = close(iFile); CHECK(iRet>=0, lblVfsKO);                            // Close file

iRet = unmount(tpcVolume[VFSType], MNT_FORCE); CHECK(iRet>=0, lblVfsKO);  // Disk USB deactivation
```

TELIUM OS APIs : **Other devices**

- **Buzzer control functions**
  - buzzer(), StartBuzzer() and StopBuzzer()
- **Backlight control function**
  - HWCNF_SetBacklightIntensity() and HWCNF_GetBacklightIntensity()

TELIUM OS APIs : **Main power cuts**

- **EFT30 is a stand-alone unit**
- **EFT 930x are battery saved but battery may be fully discharged**
- **The system proposes a mechanism to automatically backup and restore a memory area**
  - Put all your data to be saved in a RAM area "MyData"
  - At the beginning of your application execute :

    RegisterPowerFailure("E_MyArea", MyData, sizeof(MyData));
  - => in case of power cut, data are saved to flash
  - => when power comes back, data are restored from flash to RAM
- **Matter of care**
  - Backup is limited to 80kb for all applications and system => max 5kb per app.
  - On EFT930x, backup is done periodically (100 ms)
  - => don't use backup area for data often updated

TELIUM RTOS APIs : **Tasks 1/12**

- **Task_1 is the manager task (run by OS)**
  - Up to 20 tasks, stack by 2kb incremented up to 64kb
- **Task states**
  - Non operational (before the fork)
  - Operational (after the fork)
    - Current
    - Ready
    - Waiting
      - For Event or time-out (ttestall)
      - For Semaphore (P)
      - For MailBox (Receive)
  - Tasks shall finish by killing (itself or by application)
- **Functions**
  - **fork()** : launch a task
  - **kill()** : kill a task
  - **CurrentTask()** : get the current task identifier
  - **TaskState()** : return the state of the task (Current, Ready, Waiting, None operational)

TELIUM RTOS APIs : **Tasks 2/12**

Sample

```
static t_topstack *hTsk=NULL;


static word SecondTask(void)
{
    // *** Display on second line
    play('a');                                                  // Second task under animation

    // *** Task2 deletion
    kill(hTsk, "");                                             // Delete second task
    return 0;
}

void Task(void)
{
    byte dum1;
    int dum2=0;

    // *** Task2 creation
    MainTaskNbr = CurrentTask();
    hTsk=fork(SecondTask, &dum1, dum2); CHECK(hTsk!=NULL, lblKO);      // Fork second task

    // *** Display on first line
    play('A');                                                  // Main task under animation
}
```

TELIUM RTOS APIs : **Events 3/12**

- **Inter task signaling is performed through events**
- **An event can be set or cleared**
- **Each event is associated with a task : when an event is signaled it must be signaled to one task in particular**
- **There are 32 events available for each task including peripheral events, from 0 to 31 (ttestall)**
- **Between application and task or between 2 tasks**
- **Functions**
  - **SignalEvent()** : Signals the occurrence of an event to a task (event 0..31)
  - **ClearEvents()** : Clears the events specified in a list (mask bit 1..32)
  - **WaitsEvents() or ttestall()** : Waits for a list of events (mask bit 1..32)

TELIUM RTOS APIs : **Events 4/12**

Sample

```
static t_topstack *hTsk=NULL;
static word MainTaskNbr;


static word SecondTask(void)
{
    tStatus sta;

    // *** Signal an event to main task
    sta=SignalEvent(MainTaskNbr, 15); CHECK(sta==cOK, lblKO);          // Send event 15 (0..31) to main task. Second task is running

    // *** Display on second line
    play('a');                                                         // Second task under animation

    // *** Task2 deletion
    kill(hTsk, "");                                                    // Delete second task
    return 0;
}

void Task(void)
{
    byte dum1;
    int dum2=0;
    tStatus sta;
    tEventList eve;

    // *** Task2 creation
    MainTaskNbr = CurrentTask();
    hTsk=fork(SecondTask, &dum1, dum2); CHECK(hTsk!=NULL, lblKO);      // Fork second task

    // *** Wait an event from second task
    sta = WaitEvents (0x00008000, 0, &eve);                            // Wait event 15 (mask bit 1..32) from second task
    // ttestall (0x00008000, 0);                                       // Same effect with ttestall

    // *** Display on first line
    play('A');                                                         // Main task under animation
}
```

TELIUM RTOS APIs : **Semaphores 5/12**

- **A counting semaphore is an object composed of a**
  - Unit count (number of units available)
  - A count limit (max value on unit count)
  - A task queue (links the tasks waiting for units)
- **Functions**
  - **P()** : Taking a semaphore unit
  - **V()** : Releasing a semaphore unit
  - **TestP()** : checks if semaphore is available
  - Get/release semaphore number using (20 max)
    - **GetSemaphoreUser()**
    - **FreeSemaphoreUser()**

TELIUM RTOS APIs : **Semaphores 6/12**

Sample

```
static t_topstack *hTsk=NULL;
static word MainTaskNbr, semNbr;

static word SecondTask(void)
{
    tStatus sta;

    // *** Signal an event to main task
    sta = SignalEvent(MainTaskNbr, 15); CHECK(sta==cOK, lblKO);      // Send event 15 (0, 31) to main task. Second task is running

    // *** Display on second line
    play('a');                                                       // Second task under animation
    play('0');

    // *** Releasing a semaphore unit
    V(semNbr);                                                       // Semaphore release

    // *** Task2 deletion
    kill(hTsk, "");                                                  // Delete second task
    return 0;
}

void Task(void)
{
    byte dum1;
    int dum2=0;
    tStatus sta;
    tEventList eve;

    semNbr = GetSemaphoreUser();                                     // Get a semaphore number

    // *** Task2 creation
    MainTaskNbr = CurrentTask();
    hTsk = fork(SecondTask, &dum1, dum2); CHECK(hTsk!=NULL, lblKO);  // Fork second task

    // *** Wait an event from second task
    sta = WaitEvents(0x00008000, 0, &eve);                           // Wait event 15 (mask bit 1..32) from second task
    // ttestall (0x00008000, 0);                                     // Same effect with ttestall

    // *** Display on first line
    play('A');                                                       // Main task under animation

    // *** Taking a semaphore unit
    ret = P(semNbr, 0); CHECK(ret==0, lblKO);                        // Semaphore aquire infinite timeout

    // *** Main task restarts, second task is killed
    play('A');                                                       // Main task under animation

    FreeSemaphoreUser(semNbr);                                       // Release the semaphore number
}
```

TELIUM RTOS APIs : **Mailboxes 7/12**

- **Inter task communication can be performed through mailboxes**
- **A mailbox is an object consisting of a message queue, and a FIFO task queue.**
- **Maximum number of messages queuing in a mailbox is 60**
- **The "message" parameter is a double word (4 bytes)**
- **Functions**
  - **Send()** : send a message to another task.
  - **Receive()** : takes the first message on the mailbox, or waits until a message received.
  - **TestReceive()** : only checks if message received without taking it.
  - Get / release mailbox number using (30 max)
    - **GetMailboxUser()**
    - **FreeMailboxUser()**

TELIUM RTOS APIs : **MailBoxes 8/12**

Sample

```
static t_topstack *hTsk=NULL;
static word MainTaskNbr, semNbr, boxNbr;
static doubleword mail2=0;

static word SecondTask(void)
{
    tStatus sta;
    doubleword msg;

    // *** Signal an event to main task
    sta = SignalEvent(MainTaskNbr, 15); CHECK(sta==cOK, lblKO);      // Send event 15 (0, 31) to main task. Second task is running

    // *** Display on second line
    play('a');                                                       // Second task under animation
    play('0');

    // *** Send a message to a mail box
    msg = 2048;
    ret = Send (boxNbr, msg); CHECK(ret==0, lblKO);                  // Message sent to main task

    // *** Releasing a semaphore unit
    V(semNbr);                                                       // Semaphore release

    // *** Task2 deletion
    kill(hTsk, "");                                                  // Delete second task
    return 0;
}

void Task(void)
{
    byte dum1;
    int dum2=0;
    tStatus sta;
    tEventList eve;
    doubleword msg;

    semNbr = GetSemaphoreUser();                                     // Get a semaphore number
    boxNbr=GetMailboxUser();                                         // Get a mailbox number

    // *** Task2 creation
    MainTaskNbr = CurrentTask();
    hTsk = fork(SecondTask, &dum1, dum2); CHECK(hTsk!=NULL, lblKO);  // Fork second task

    // *** Wait an event from second task
    sta = WaitEvents(0x00008000, 0, &eve);                           // Wait event 15 (mask bit 1..32) from second task
    // ttestall (0x00008000, 0);                                     // Same effect with ttestall

    // *** Display on first line
    play('A');                                                       // Main task under animation

    // *** Taking a semaphore unit
    ret = P(semNbr, 0); CHECK(ret==0, lblKO);                        // Semaphore aquire infinite timeout

    // *** Main task restarts, second task is killed
    play('A');                                                       // Main task under animation

    // *** Wait a message from a mail box
    ret = TestReceive(boxNbr, &msg); CHECK(ret==0, lblKO);           // Message retrieves from second task
    if (msg == 2048)
        mail2=msg;

    FreeSemaphoreUser(semNbr);                                       // Release the semaphore number
    FreeMailboxUser(boxNbr);                                         // Release the mailbox number
}
```

TELIUM RTOS APIs : **Delays 9/12**

- **For cyclic or delayed execution of a function**
  - 10 ms resolution
  - Take care as it blocks the system, so function shall be short
  - No OS nor M2OS routines should be used
- **Dedicated to update counter**
- **Functions**
  - **StartLDelay()** : executes a function, once or on a cyclic basis
  - **StopDelay()** : cancel a previously launched function
  - Get / release delay number using (20 max)
    - **GetDelayUser()**
    - **FreeDelayUser()**

TELIUM RTOS APIs : **Delays 10/12**

Sample

```
static char buf[PRTW+1];
static word to;

static word Periodic (void)
{
    strcat(buf, ".");
    pprintf("%s\n", buf);
    to--;
    return 0;
}

void Delay(void)
{
    word dlyNbr=0;
    doubleword time, period;
    int ret;

    // ................

    memset(buf, 0, sizeof(buf));                                                    // Reset printing buffer
    // *** Repeat cyclically 10 times the function Periodic every 1s delay
    to = 10;                                                                        // TimeOut
    dlyNbr=GetDelayUser();                                                          // Get a delay number
    ret=StartLDelay(dlyNbr, 1*100, 1*100, Periodic); CHECK(ret==0, lblKO);          // Start periodic delay of 1 second
    while(to!=0)                                                                    // Wait until TimeOut expired
        ttestall(PRINTER, 10);
    StopDelay(dlyNbr, &time, &period);                                              // Stop periodic delay

    FreeDelayUser(dlyNbr);                                                          // Release the delay number

    // ................
}
```

TELIUM RTOS APIs : **Timers 11/12**

- **Based on the function get_tick_counter**
  - Return the number of elapsed ticks since start-up
  - One tick = 10ms
- **Functions**
  - **TimerStart()** : Start a timer number for a delay/100 sec
  - **TimerGet()** : Return the remaining time in x/100 sec
  - **TimerStop()** : Stop a timer number
    - No limitation regarding the number of timers
    - The sample is limited to 4 timers and can be easily increased

TELIUM RTOS APIs : **Timers 12/12**

Sample

```
void Timer(void)
{
    int ret, ret0;

    ret = TimerStart(0, 15*100); CHECK(ret>=0, lblKO);    // Timer0 starts to 15s
    while ((ret0=TimerGet(0)) > 0)                        // Check Timer0
    {
        CHECK(ret0>=0, lblKO);

        // *** Display Remaining Time
        gotoxy(0,0); ret = printf("Exp in %2d.%02d Sec", ret0/100, ret0%100);
        CHECK(ret>=0, lblKO);

        // *** Wait 200ms = ttestall(0, 20)
        ret = TimerStart(1, 20); CHECK(ret>=0, lblKO);    // Timer1 starts to 200ms
        while (TimerGet(1) > 0)                           // Check Timer1
        {
            CHECK(ret1>=0, lblKO);
            // ..........................
        }
    }

    gotoxy(0,0); ret = printf("Timer expired !!");        // Timer expired
    buzzer(10);
    ttestall(0, 1*100);

    TimerStop(0);                                         // Close Timer0
    TimerStop(1);                                         // Close Timer1

    // ....................................
}
```

TELIUM FFMS : **Files Management 1/6**

- FFMS uses NAND flash or RAM
- FFMS supports multi volume defined as disks
- Application Disks must be created, mounted and formatted, disk size is auto extended.
- 100000 formatting or defragmentation cycles
- Access to files via **/diskname/filename** in Flash or in RAM
- Opening a file:
  - r, a, r+ for Flash (reading, create or append, updating)
  - r, a, w, w+ for RAM (reading, create or append, writing, deleting)

TELIUM FFMS : **Files Management 2/6**

- **FFMS routines for disk management**
  - **FS_dskcreate()** : Create, initialise and format a disk
  - **FS_mount()** : initialise a disk with its name
  - **FS_unmount()** : deactivate a disk by its name
  - **FS_format()** : format a disk
  - **FS_dsksize()** : give the size of the disk in bytes
  - **FS_dskfree()** : number of free bytes on the disk
  - **FS_changemode()** : change the access mode (No access, Read only, R/W)
  - **FS_dskkill()** : disk suppression

TELIUM FFMS : **Files Management 3/6**

- **FFMS routines for directory management**
  - **FS_mkdir()** : create a directory on a disk
  - **FS_rmdir()** : delete a directory on a disk
  - **FS_opendir()** : open a flow on a directory to read files information
  - **FS_readdir()** : read files information (name, creation, size)
  - **FS_closedir()** : close a flow on a directory

TELIUM FFMS : **Files Management 4/6**

- **FFMS routines for files management**
  - **FS_open()** : open a flow on file
  - **FS_write()** : write data into a file
  - **FS_flush()** : flush buffered data to a file
  - **FS_read()** : read data from a file
  - **FS_close()** : close a flow on an existing file
  - **FS_unlink(), FS_exist(), FS_rename(), FS_eof(), FS_Length(), FS_seek(), FS_tell()**
- **FFMS routine for Garbage Collection**
  - **FS_GarbageCollection()** : Compact an area of flash. Calling this function avoid automatic garbage collection.

TELIUM FFMS : **Files Management 5/6**

Sample

```
S_FS_PARAM_CREATE cfg;
S_FS_FILE *fid;
unsigned long size;
unsigned int mode;
int ret;

// *** Disk creation ***
strcpy(cfg.Label, "TRAINING" );                    // Disk name
cfg.Mode = FS_WRITEONCE;                           // Disk on Flash
cfg.AccessMode = FS_WRTMOD;                        // r/w access
cfg.NbFichierMax = 16;                             // Max files number
cfg.IdentZone = FS_WO_ZONE_DATA;                   // Zone id
size= cfg.NbFichierMax*32768;                      // Disk size in bytes

ret = FS_dskcreate(&cfg, &size);                   // Create and format the disk
CHECK(ret==FS_OK, lblKODisk);

ret = FS_mount ("/TRAINING", &mode);               // Activate a disk
CHECK(ret==FS_OK, lblKODisk);

// *** File open ***
fid = FS_open("/TRAINING/MyFile.txt", "a");        // "a" mode creation or existing file
CHECK(pid!=NULL, lblKOFile);

// ........
// File management
// read, write, offset, EOF
// ........

// *** File close ***
ret = FS_close(fid);                               // Close file
CHECK(ret==FS_OK, lblKOFile);

// *** Disk suppression ***
ret = FS_unmount("/TRAINING");                     // Release ressources on disk
CHECK(ret==FS_OK, lblKODisk);

ret = FS_dskkill("/TRAINING");                     // Disk suppression
CHECK(ret==FS_OK, lblKODisk);
```

TELIUM FFMS : **Files Management 6/6**

Matter of care with FFMS file system

- Minimum writing record is 512 bytes, so if you write less, 512 will be written anyway (lost of room disk). Take care using flush command, prefer writing a whole file. A best way to work is create file in RAM and copy the whole file into flash when build !
- Maximum number of file for user is 1000 files
- One file is minimum 2 records of 512 = 1 kbyte minimum
- One file is maximum… = size of the user Flash

TELIUM FMG : **Easy Files Management 1/4**

- **The FMG library provides a whole set of services allowing the applications to manage and save files in the flash memory easily**
- **All services provides by the FMG library are based on the FFMS Library**

TELIUM FMG : **Easy Files Management APIs 2/4**

- **The FMG library enables applications to :**
  - Create / Delete a file by indicating only its name and its Path
  - Create / Delate a file by indicating only its type :
    - Batch
    - BlackList
    - Log
  - Manage files containing record with fixed / variable size
  - Add / Modify / Read / Delete a record in a file by its position :
    - Begin
    - End
    - Middle
  - Check file and record coherency
  - Get the FMG library status

TELIUM FMG : **Easy Files Management APIs 3/4**

The FMG routine for initialization

- **FMGInit()** : Initialize the file management module
  - This service must be called in the **after_reset** entry point.

The FMG routines for files

- **FMG_CreateFile()** : Create a file by indicating its name and path
- **FMG_DeleteFile()** : Delete a file by indicating its name and path
  - Disk creation and mounting must be done by the user (see FFMS)
- **FMG_CreateFileType()** : Create a file by indicating its type
- **FMG_DeleteFileType()** : Delete a file by indicating its type
  - Type => Batch, BlackList, Log
  - Disk creation and mounting is done by the system (FMG0161)
- **FMG_GetFileDescription()** : Description about files created by FMG

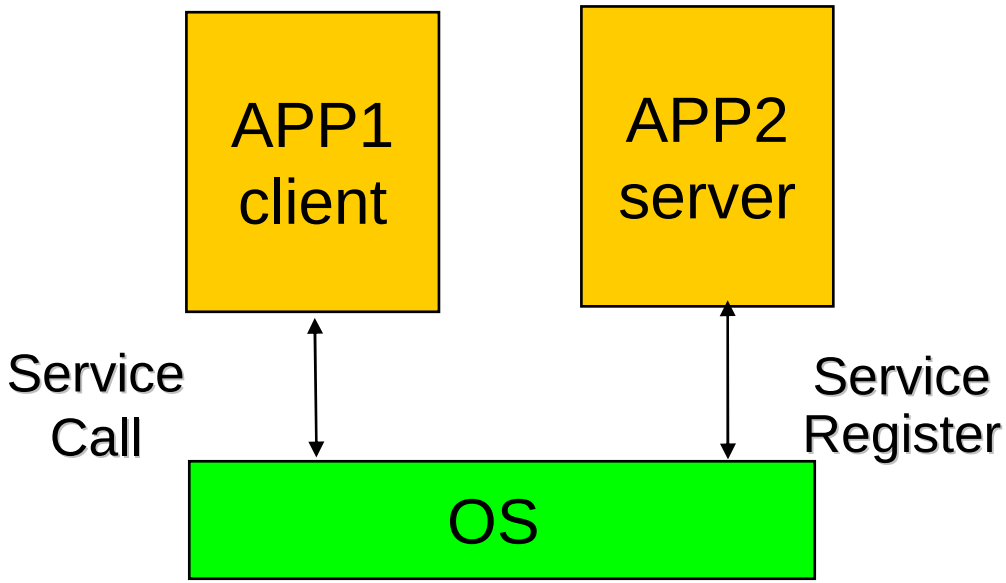
TELIUM FMG : **Easy Files Management APIs 4/4**

The FMG routines for records

- **FMG_AddRecord() :** Add a record in a file
- **FMG_ModifyRecord() :** Modify a record in a file
- **FMG_Read_Record() :** Read a record in a file
- **FMG_Delete_Record() :** Delete a record in a file

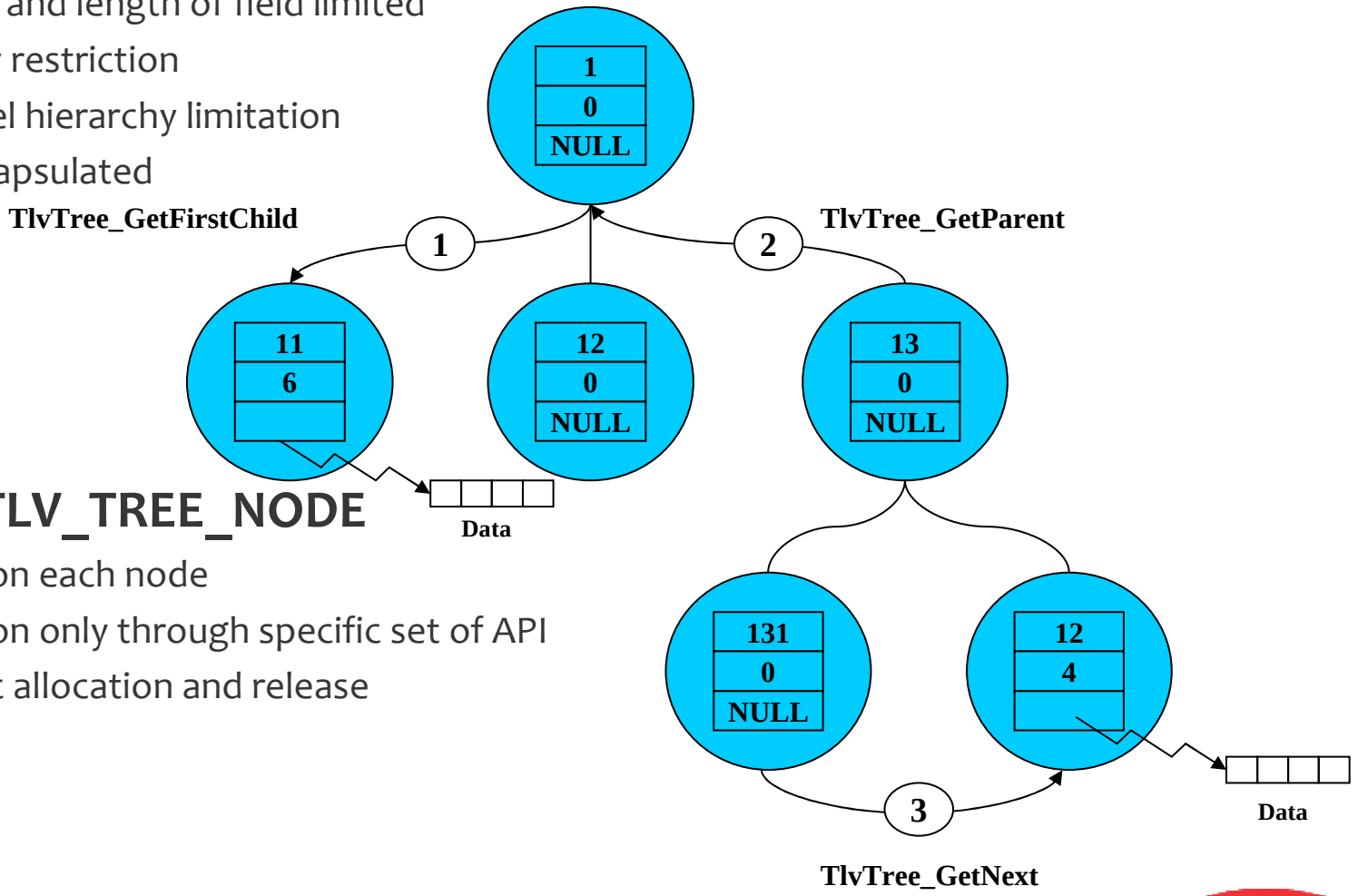
TELIUM : Inter Application Communication 1/6

- IAC interface provided at OS level
- Server Application registers services through ServiceRegister
- Client Application calls server application services through ServiceCall



TELIUM IAC : **TLV tree interface 2/6**

- **To replace DEL format by more efficient TLV tree format**
  - Number and length of field limited
  - Memory restriction
  - One level hierarchy limitation
  - Not encapsulated



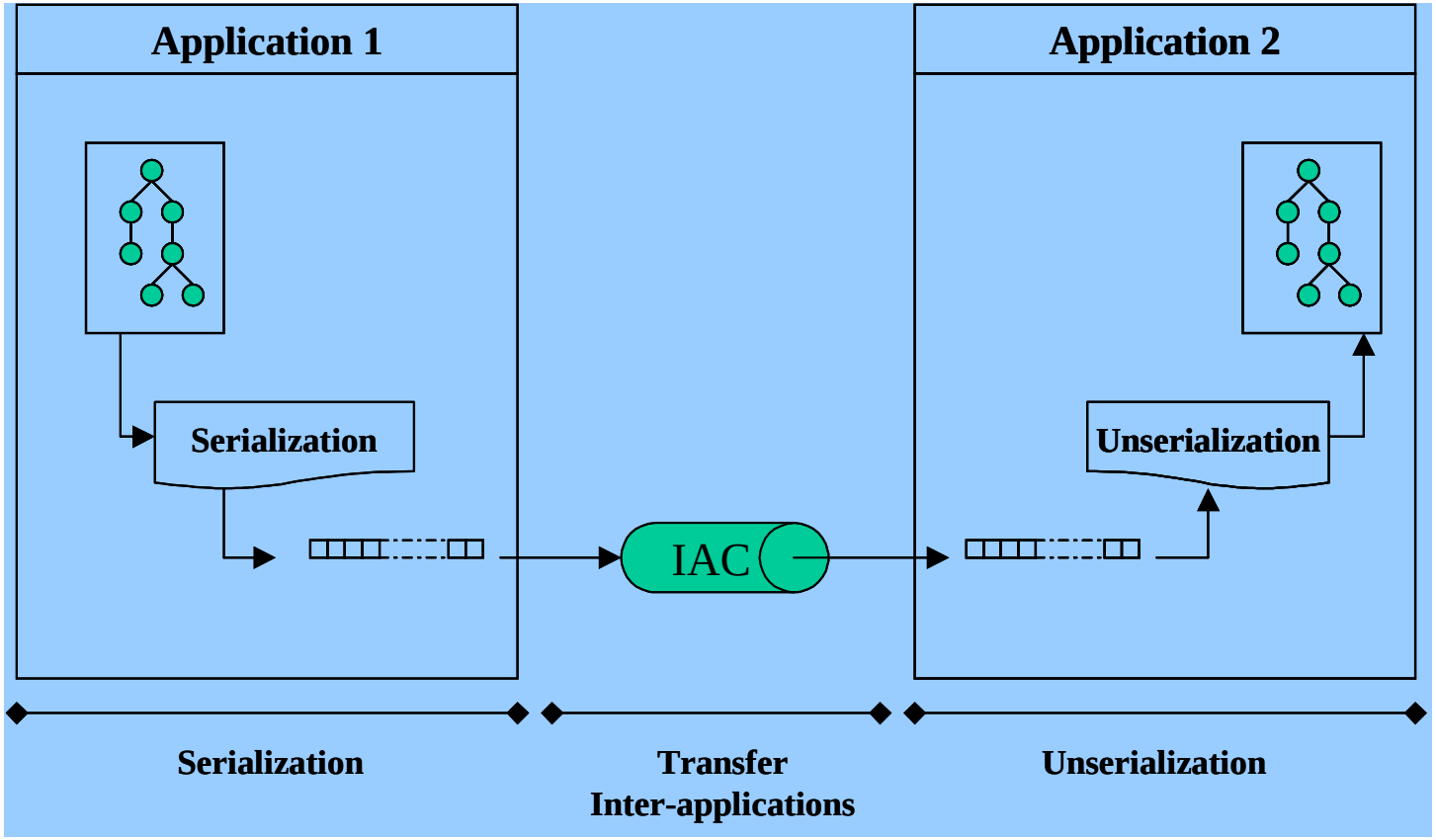
- **Object TLV_TREE_NODE**
  - Handle on each node
  - Utilisation only through specific set of API
  - Dynamic allocation and release

TELIUM IAC : **TLV tree interface 3/6**

- **A specific library to encapsulate and manage TLV Tree format**
  - AVL.lib must be implemented to link properly
- **Create / Kill tree**
  - TlvTree_New
  - TlvTree_AddChild / TlvTree_AddChildInteger / TlvTree_AddChildString
  - TlvTree_Copy
  - TlvTree_RemoveChild
  - TlvTree_Release
- **Browse tree / Read - Write elements**
  - TlvTree_Find
  - TlvTree_GetNext / TlvTree_Iterate
  - TlvTree_GetFirstChild / TlvTree_GetParent
  - TlvTree_GetTag / TlvTree_GetLength / TlvTree_GetData
  - TlvTree_SetTag / TlvTree_SetData / TlvTree_SetDataString
- **Serialisation / Unserialisation**
  - TlvTree_Serialize
  - TlvTree_Unserialize
  - TlvTree_GetSerializationSize

TELIUM IAC : **TLV tree interface 4/6**

- **Serialisation allows to pass a tree as parameter by value**
  - Convert TLV tree into a linear structure… (or others format in the future)
  - Pass this structure as parameter through IAC
  - Rebuild the same tree at reception side



TELIUM IAC : **TLV tree interface 5/6**

Sample Training Client

```
// Service to call
// ===============
NoAppli=0x162;                                                          // USER2 application to call
Service = WAIT_RESPONSE;                                                // Service to call
strcpy(snd, "Hello, do you hear me?");                                  // Data to send

// Build message to send under TLV tree format
// ============================================
hNode = TlvTree_New(0x1110);                                            // Create tree
TlvTree_AddChildString(hNode, 0x1111, snd);                             // Add message to the tree

nSize = TlvTree_GetSerializationSize(hNode, TLV_TREE_SERIALIZER_DEFAULT);   // Get serialized buffer size
CHECK(nSize>0, lblKO);
pucBuffer=(byte *)umalloc(nSize);                                       // Allocate the serialize buffer size
ret = TlvTree_Serialize(hNode, TLV_TREE_SERIALIZER_DEFAULT, pucBuffer, nSize);  // Serialize the tree in the allocated buffer
CHECK(ret>0, lblKO);

TlvTree_Release(hNode);                                                 // Serialization done release tree

// Call USER2 service to send message and receive response
// =======================================================
ret = ServiceGet(NoAppli, Service, &Priority); CHECK(ret==0, lblNoService);   // Check if User2 service WAIT_RESPONSE exists
ret = ServiceCall(NoAppli, Service, nSize, pucBuffer, &code); CHECK(ret==0, lblKO); // Call User2 service WAIT_RESPONSE
CHECK(code>=0, lblNoRsp);

// Parse message received from TLV tree format
// ============================================
ret = TlvTree_Unserialize(&hNode, TLV_TREE_SERIALIZER_DEFAULT, pucBuffer, nSize);   // UnSerialize the allocated buffer to create a tree
CHECK(ret==TLV_TREE_OK, lblKO);
memset (rsp, 0, sizeof(rsp));
strcpy (rsp,TlvTree_GetData(TlvTree_GetFirstChild(hNode)));             // Get message from the tree

// Release ressources
// ==================
TlvTree_Release(hNode);                                                 // Release tree
ufree (pucBuffer);                                                      // Release allocated buffer
```

TELIUM IAC : **TLV tree interface 6/6**

Sample User2 Server

```
int IacRegisterTraining(unsigned int size, void *data)
{

    // ........................................

    // Parse message received from TLV tree format
    // ===========================================
    ret = TlvTree_Unserialize(&hNode, TLV_TREE_SERIALIZER_DEFAULT, data, size); // UnSerialize the allocated buffer to create a tree
    CHECK(ret==TLV_TREE_OK, lblKO);
    memset (rsp, 0, sizeof(rsp));
    strcpy (rsp, TlvTree_GetData(TlvTree_GetFirstChild(hNode)));               // Get message from the tree

    // Build message to send under TLV tree format
    // ===========================================
    strcpy(snd, "Hi, how are you doing?");                                     // Data to send

    TlvTree_SetDataString(TlvTree_GetFirstChild(hNode), snd);                  // Add message to the tree

    ret = TlvTree_Serialize(hNode, TLV_TREE_SERIALIZER_DEFAULT, data, size);   // Serialize the tree in the allocated buffer
    CHECK(ret>0, lblKO);

    // Release ressources
    // ==================
    TlvTree_Release(hNode);                                                    // Release tree

    // ........................................

    return ret;
}
```

TELIUM : **Memory Management**

- **8 up to 32 MB of FLASH and DRAM**
- **MMU protection with firewalls**
- **MMU translates virtual to physical addresses (automatic relocation)**
- **Code always starts at 0, Data starts at 0x200000**
- **Stacks fully managed by OS (no memory allocation for stacks) Max size is 60KB**
- **Memory pages allocation**
  - 32 pages of 4 to 128 KB for one application => 4MB Max (modulo 4kb)
  - int *PageAlloc(unsigned int size) : allocate a secured memory page allocation
  - int PageFree(void *page) : release a secured memory page
  - int PageProtect(void *page, access_t access) : memory page access rights (r, rw, no)
- **Heap size**
  - 4Mb = Max size for one application (modulo 32bytes)
  - umalloc() and ufree() for unprotected utilisation (instead of malloc() and free())

TELIUM : **DLL Management**

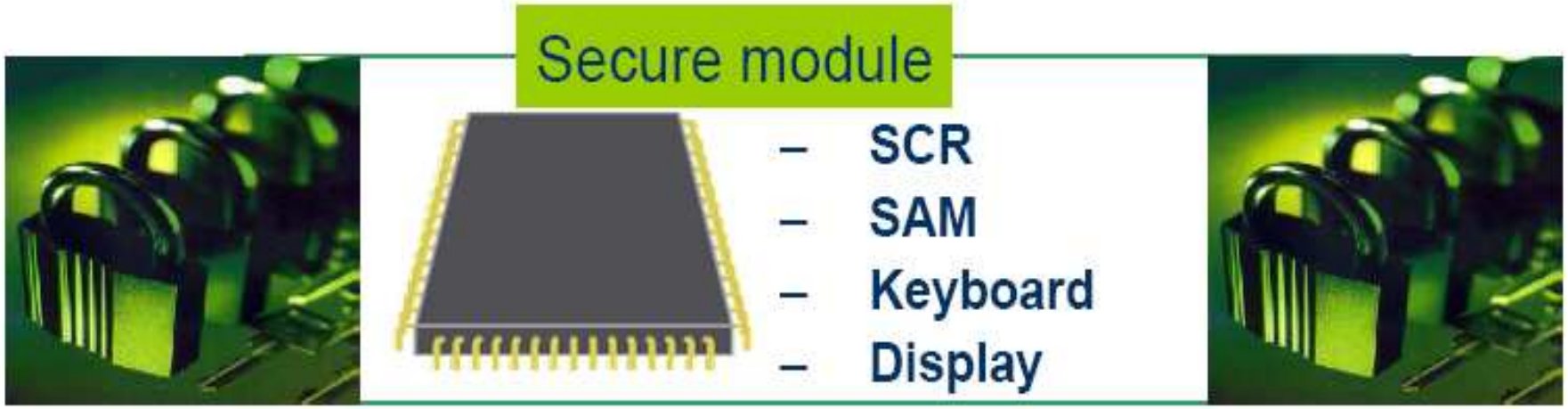
- **DLL is not an application, no service registration**
- **DLL is executed in the application caller context**
- **DLL is a separate project, link and binary**
- **DLL management OS routines**
  - ObjectLoad (OBJECT_TYPE_DLL, “MyDLL”);
  - GetProcAddress = Dlllink (“MyDLL”);
  - GetProcAddress (“Myfunc”);
  - DllUnlink(“MyDLL”);
- **Beware : a DLL has its own name and type is mapped**
  - Name and type are to be taken in the application name and type range given for each VAR to prevent identical names.
  - But the memory area is unique and shared by all VARs.
    - » VARs will have to verify that there is no overlapping if DLLs are used by both competing applications.

TDS Training : **Contents**

- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

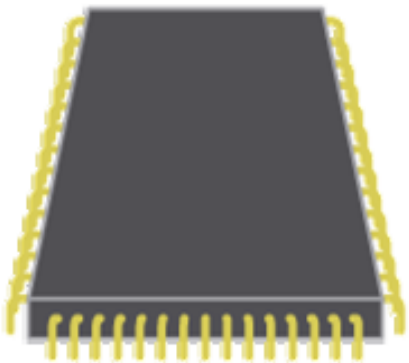
TELIUM : **Security**

- Tamper evident / Tamper detection / Tamper resistance device, PEDs/PCI compliance
- All secrets stored in the Crypto-processor and pin code never leaves the secure module
- Software or schemes are authenticated by the crypto processor before loading in the terminal



TELIUM SECURITY : **Tamper detection 1/2**

All the « Tamper components » are physically assembled in the secure module protected by tamper detectors :



- Opening of the terminal
- Opening of the secure module
- Removal of the silicon keyboard
- Drilling of the PCB
- Voltage out of specification
- Temperature out of specification

=> 3 security levels

- no security
- PEDs level
- ZKA level

TELIUM SECURITY : **Tamper detection 2/2**

- **Any tempering attempt :**
  - Clear the secret areas
  - Display a warning message (“**Alert irruption**” blinking)
  - Lock the keyboard
  - Reactivation of the terminal needed (Recovery tool)
  - :-) product is activated
  - :-(product is deactivated

TELIUM SECURITY : **Device versions**

- **Due to security enhancements, it exists different versions of Booster and Terminal :**
  - **Telium1 / Booster1 (Visa PED certified)**
    - Smart1, Eft930xx, Ppr30, Pps30, Pp30
  - **Telium1 / Booster2 (PCI PED certified v1.3)**
    - Smart2, Ml30, P30, PP30s, Eft930sgem-C2
  - **Telium2 / Booster3 (PCI PED certified v2.0)**
    - Ict220, Ict250

TELIUM SECURITY : **A secure device 1/4**

- **Terminal only accept software that have been certified**
- **Signature is made by SST on Telium1 products:**
  - All schemes are double signed (Manufacturer + Customer)
  - Application is signed by customer
  - RSA 1024 bits signature card and certificate
  - Schemes are authenticated at each utilization
  - Application is authenticated at first loading
- **Signature is made by SAT on Telium2 products:**
  - All schemes are double signed (Manufacturer + Customer)
  - Application is double signed (Customer + Online Manufacturer server access)
  - Description file embedded in signed file (and part of the signature)
  - RSA 2048 bits signature card and certificate
  - Schemes are authenticated at each utilization
  - Application is authenticated at first loading, at each terminal reboot and periodically

TELIUM SECURITY : **A secure device 2/4**

- **a VAR can only download software in his terminals according to the security profile.**
- **All terminals are delivered in pre-personalized security configuration, waiting for definitive local security profile.**
  - On Telium1 / Booster1 :
    - Terminals are loaded with a specific application waiting for a local **customization card** to be inserted.
  - On Telium1 / Booster2 :
    - The first loaded application signed with a local **signature card** will inject the security profile to the terminal.
  - On Telium2 / Booster3 :
    - Same as Booster2 but an additional **Certificate Key file** is loaded first or at the same time when the first application is loaded (this Certificate Key file is provided by Ingenico with SAT).

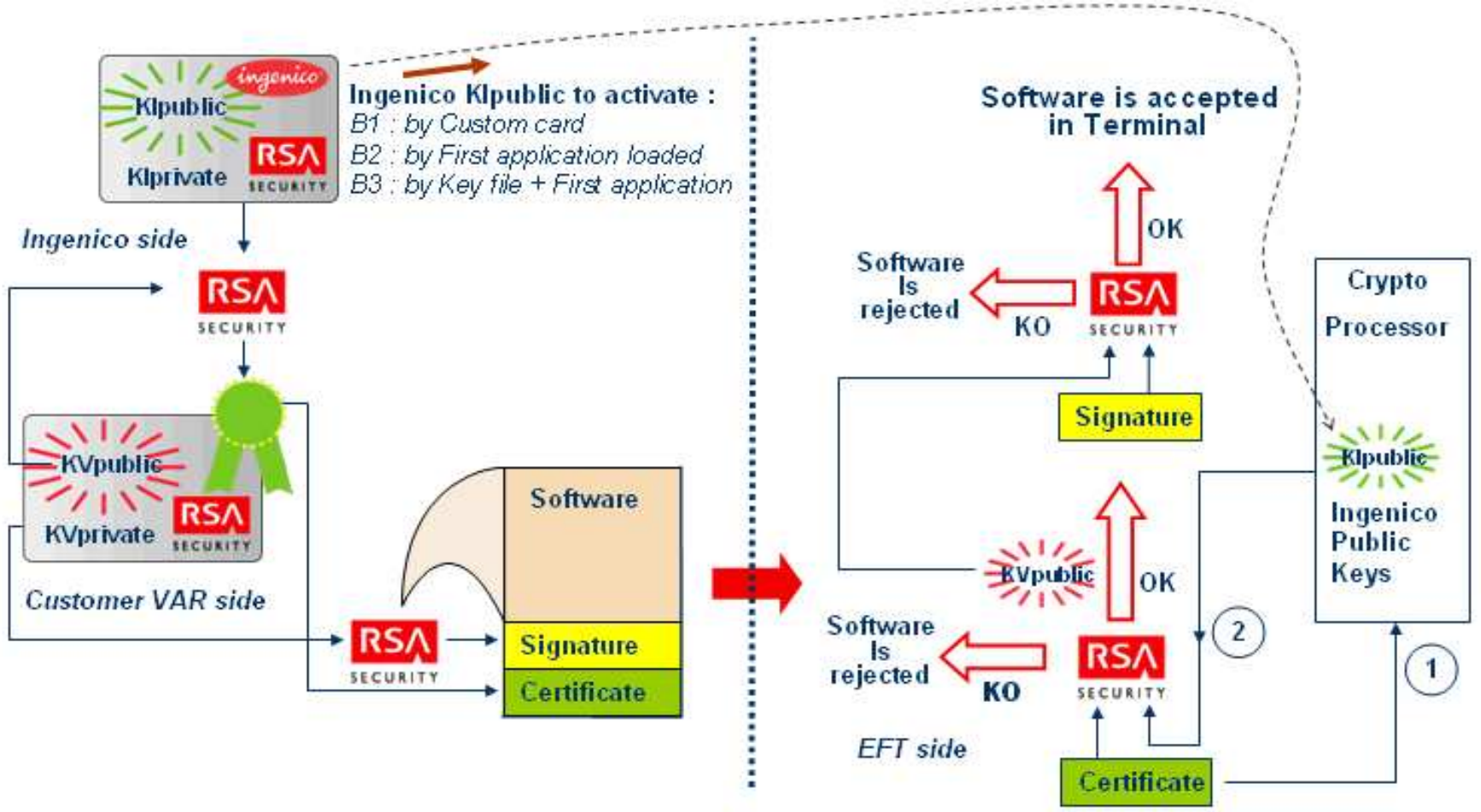
TELIUM SECURITY : **A secure device 3/4**

- **How to check security profile ?**
  - On Booster1 (Smart1, EFT930xx) :
    - Automatically displayed at boot time on latest system (Sxx)
    - Menu F|Manager|consultation|configuration|hardware = F0141

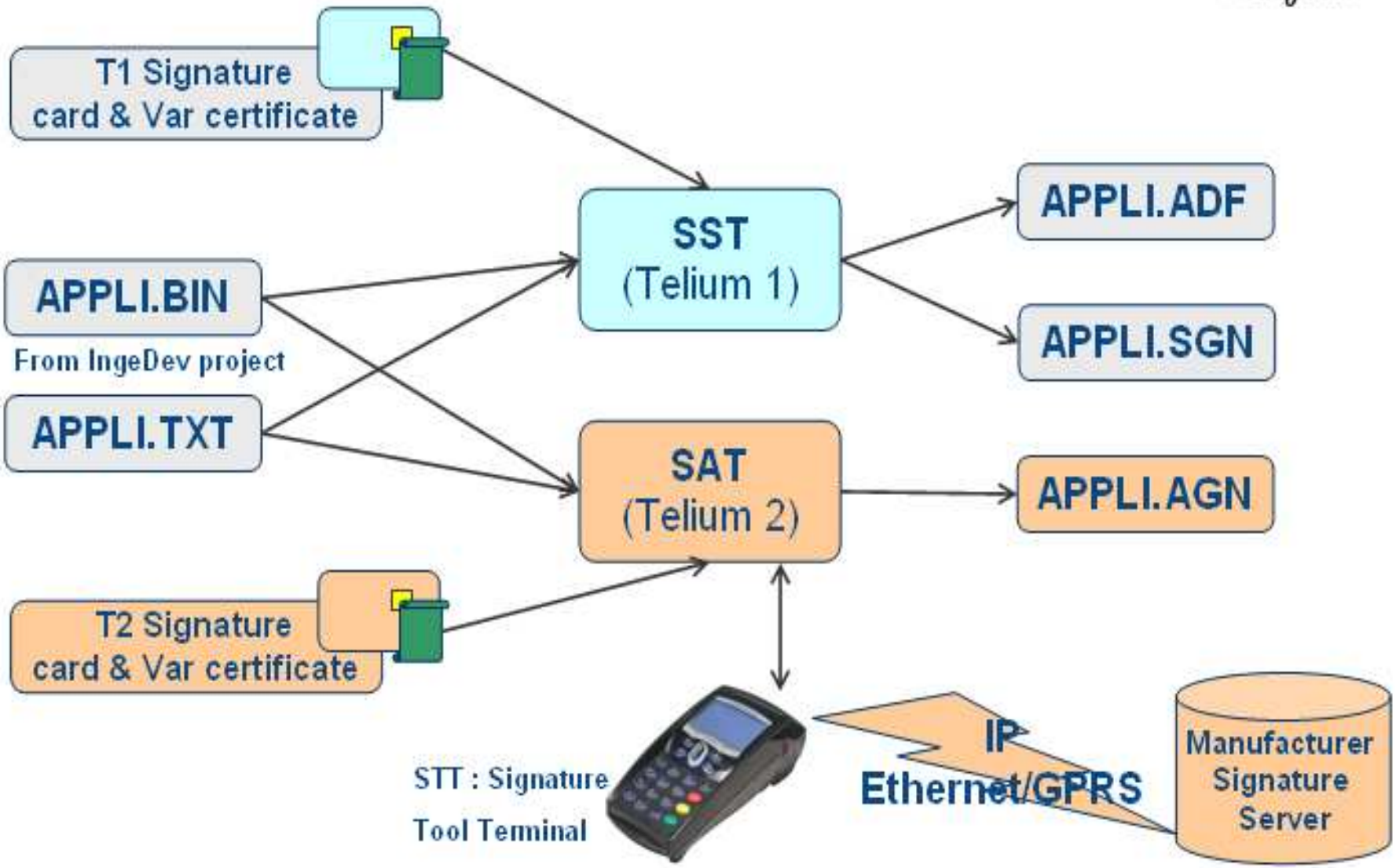
      Ticket printed, Activation infos : Sxx
  - On Booster2 / Booster3 (Smart2, Ml30, Eft930Sgem-c2, Ict220/250) :
    - Printout diagnostic ticket or get booster.dia file
    - Initial configuration is generic S98 / A98, always displayed whatever is the real security profile ! Ignore it and instead :
      - » Print out revocation table on terminal
      - » Make a dump (dump.exe or SAT/tool/dump) of application signed with your own card
      - » Compare revocation table of terminal and application
      - » Card signature and terminal are compliant if revocation table is the same

TELIUM SECURITY : **A secure device 4/4**

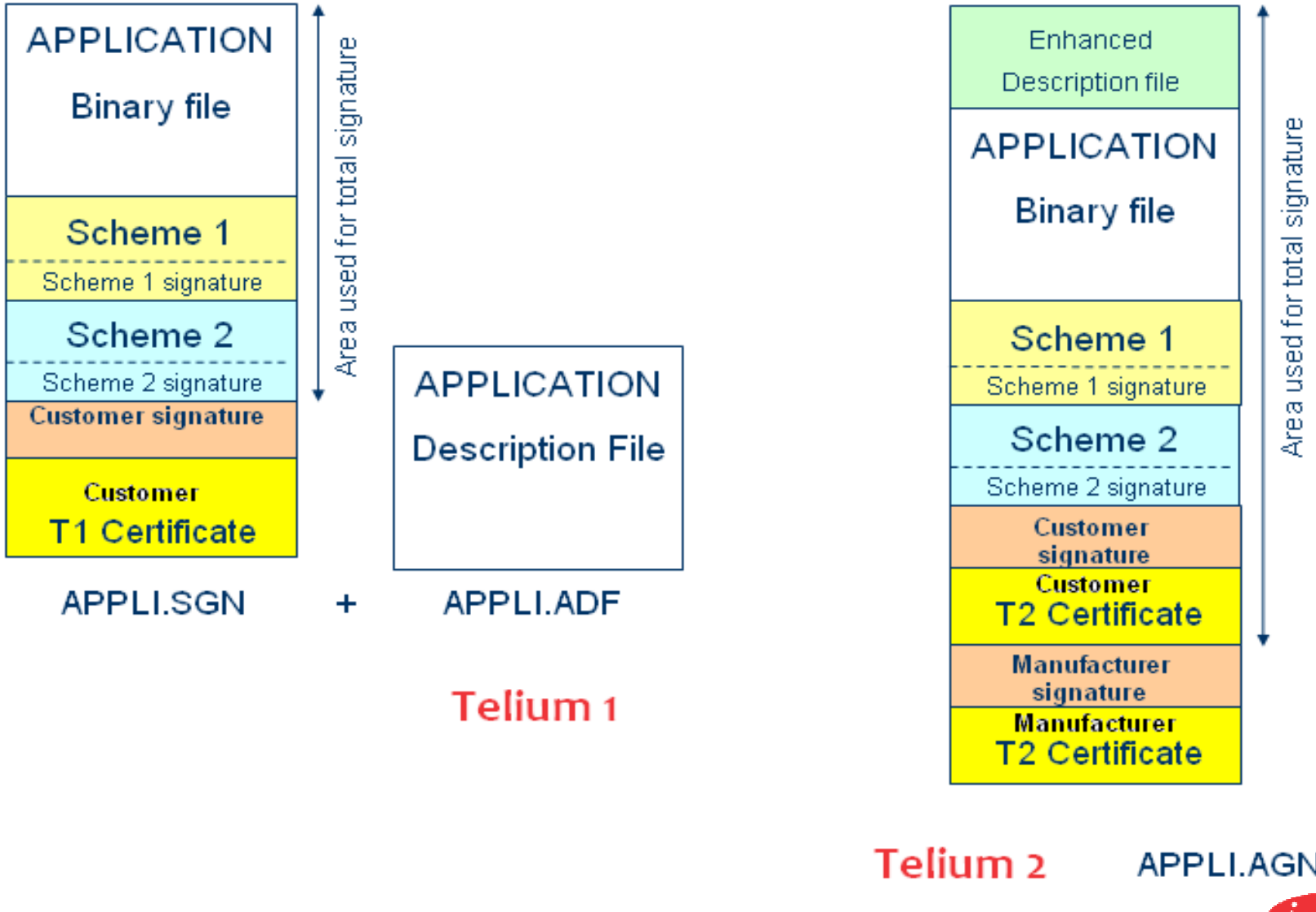
- **Crypto processor controls : Integrity & Authenticity**



TELIUM SECURITY : **SST/SAT Create certified loadable files (1/2)**



TELIUM SECURITY : **SST/SAT Create certified loadable files (2/2)**



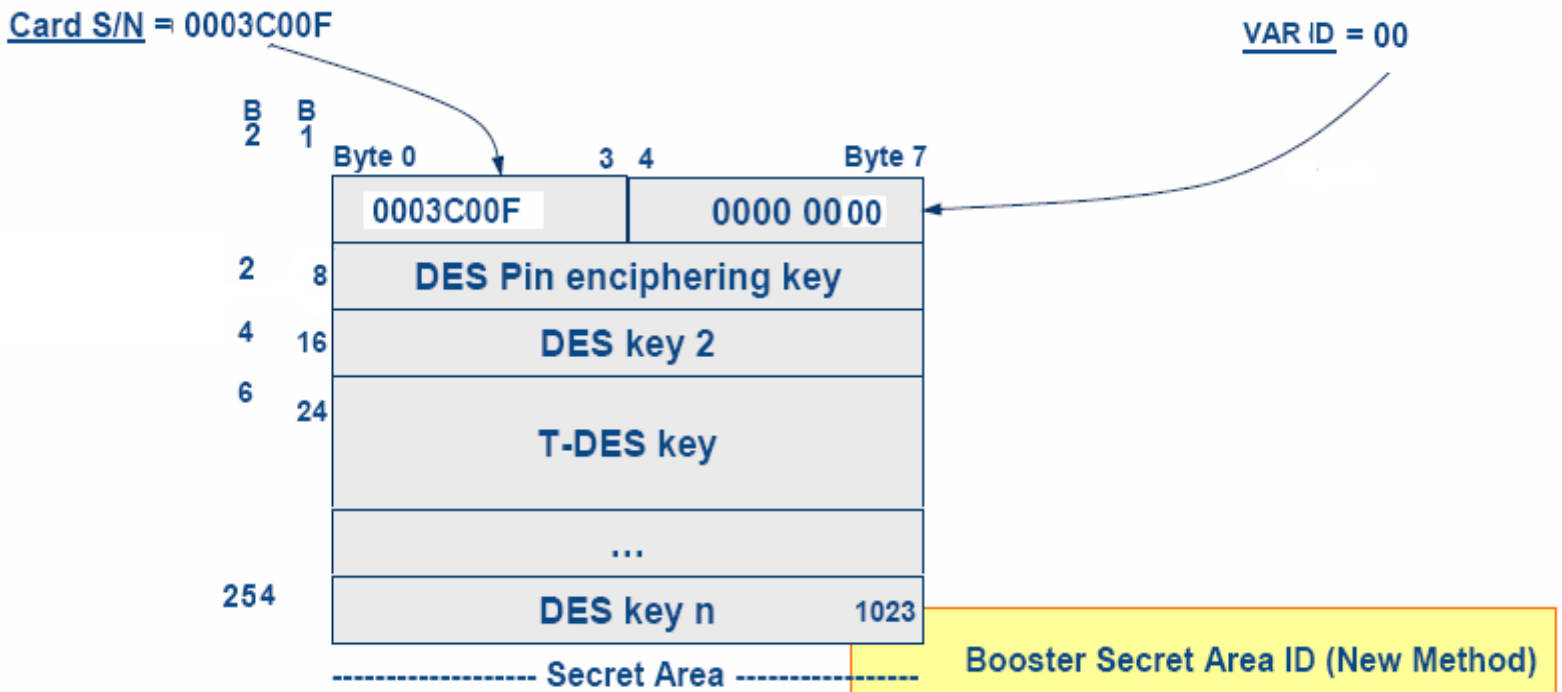
TELIUM SECURITY : **Schemes B1 vs B2/B3 TLV (1/2)**

Function	Scheme B1	Scheme B2/B3 TLV
Load keys	SchLoadData() (plaintext or enciphered)	*TlvLoadKey()* *(enciphered only)*
Load Data	*Function not available*	*TlvLoadData()*
Pin entry	SchGetPin() (Only under Simulation)	
Pin enciphering ISO9564	SchIso9564() DES or TDES	TlvIso9564()
Pin enciphering DUKPT	SchDukpt() / SchDukpTDes()	TlvDukpt() / TlvDukpTDes()
Encipher data (data or MAC calculation)	SchCipherDa() *SchCipherPR() for PPR30*	TlvCipherDa()
Random number generation	Sch_Random()	*Coming soon*
XOR calculation on keys	SchXOR()	*Coming soon*
Secret area deletion	SchFree() for Secret Area only	TlvFree() for Secret Area or Key deletion

TELIUM SECURITY : **Schemes B1 vs B2/B3 TLV (2/2)**

- All schemes must be signed by customer before utilization
  - Note that for development only, T10/Debug (mockup) already signed schemes are provided inside SDK.
- All necessary signed schemes are to be selected in SST/SAT before application signature
- To know which schemes to use
  - Make a list of all Security DLL functions used
  - Check for each function which schemes are required in **Security DLL user's guide**
- There is no SDK to develop your own scheme => Ask Ingenico for specifics requests

TELIUM SECURITY : **Secret area creation (1/4)**



Create Secret Area Sample : SEC_CreateSecretArea (C_SEC_CYPHERING, AREA ID, CARDS/N, VARID)

TELIUM SECURITY : **Secret area information (2/4)**

- **Card S/N :** written on card and provided with SST/SAT delivery
  - From SST certificate : 000012AB-00CD-89V03
  - From SAT certificate : 00010550-0033-0089.C13
- **Var ID :** extracted from Certificate
  - From SST certificate : 000012AB-00CD-89V03
    - Entered manually in SST / B1
    - Selected automatically in SST / B2
  - From SAT certificate : 00010550-0033-0089.C13
    - Selected automatically in SAT / B3
  - Can also be read with DumpSGN tool in SST/SAT
- **Secret Area ID :** example ID_SCR_MYBANK = 0089213A
  - 0089 = Var ID (unique filter to access secret area)
  - 1 & A = represent customer secret area name 0x1A
  - 2 & 3 = 0x23 padding values can't be changed

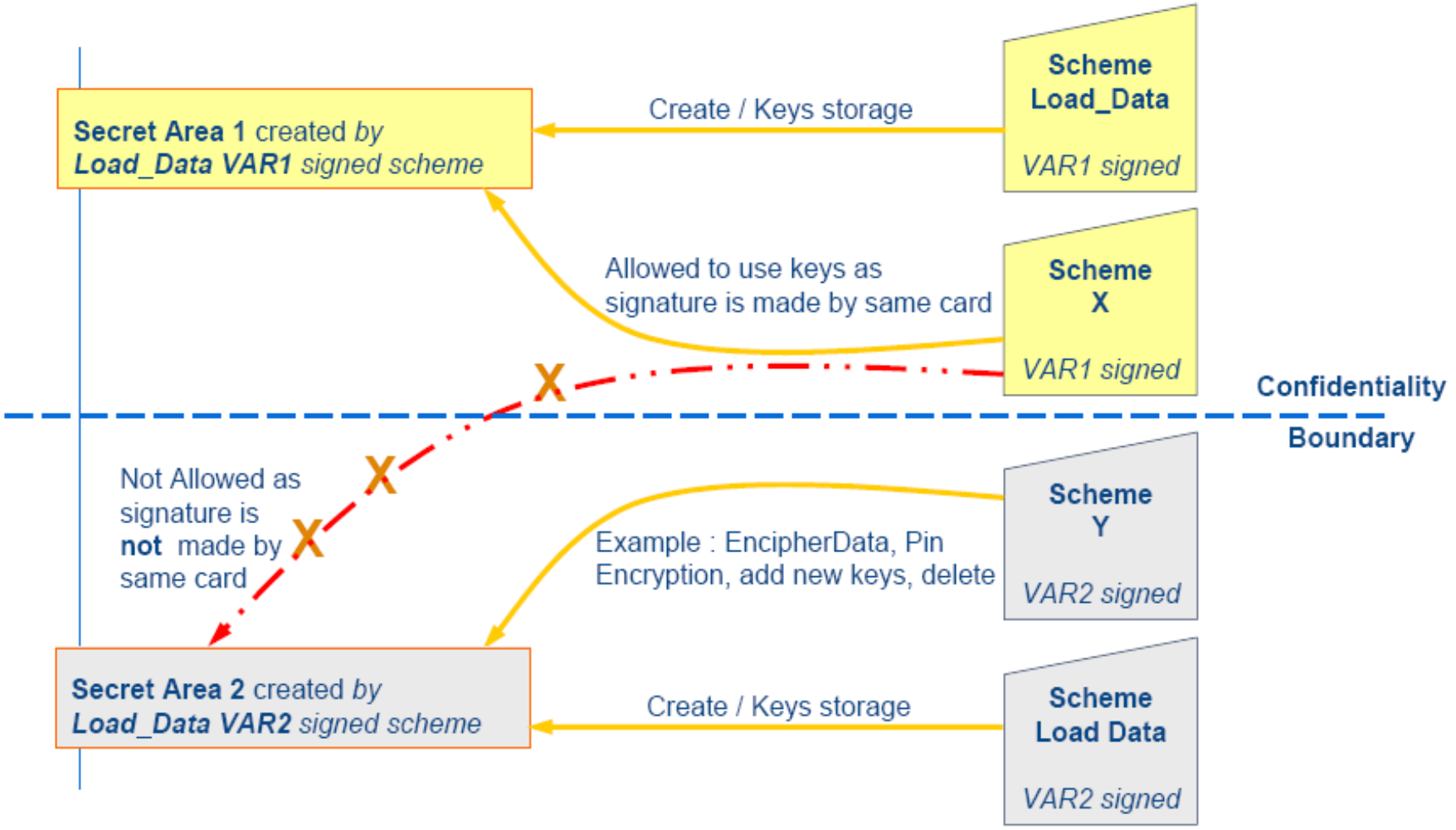
TELIUM SECURITY : **Secret area principle (3/4)**

- A secret area is like a "black box" containing keys used by their TLV identifier
- Key identification : T_SEC_DATAKEY_ID structure

int iSecretArea	SecretArea Id (same definition as New SecretArea ID described)
Char cAlgoType	Algorithm type (DES, TDES, DESDUKPT, TDESDUKPT..)
Char cNumber	**Key number**, from 2 to 255 Note : To keep compatibility with B1 format, relation between this number and B1 offset address in secret area should be : Offset = **KeyNumber** * 4 (with Pinkey at KeyNumber = 2)
Unsigned short usBankId	Bank Identifier

- Each key will be identified by its 4 bytes length tag : 0xBBBBTTNN
  - BBBB : usBankId
  - TT : cAlgoType
  - NN : cNumber
  - Key usage value : (Key Ciphering, Pin Ciphering, or Data Ciphering)

TELIUM SECURITY : **Secret area utilization (4/4)**



TELIUM SECURITY : **Which library ?**

- **Matter of Care**
  - Standard secure schemes : **Booster B1**
  - TLV standard secure schemes : **Booster B2**

=> Rather use DLL Security, making transparent B1,B2,B3,T1,T2

- **The DLL Security**
  - Specific schemes in SDK
    - Booster1 …\SDK30\TDS v.r\Component\Schemes\NotSigned\T1\Schxxx
    - Booster2 …\SDK30\TDS v.r\Component\Schemes\NotSigned\T1\Tlvxxx
    - Booster3 …\SDK30\TDS v.r\Component\Schemes\NotSigned\T2\Tlvxxx
  - Possibility to have B2/B3 Tlv and B1 schemes in the same application to have a portable application for both ICT220(B3) + SMART2(B2) + EFT930 (B1).
    - DLL Security allows to make transparent the development under all platforms
    - Caution : SST5.1 or SAT1.5 required
  - Easier to implement with high level access to security functions.
  - This component is a DLL (36560202.SGN) loaded in the terminal.

TELIUM SECURITY : **Dll Security 1/5**

- **Define the configuration for each secure part**
  - Pincode entries device => internal (terminal keyboard) or external (pinpad keyboard)
  - Card reader device => internal (terminal reader) or external (pinpad reader)
  - Cipher/decipher device => internal (terminal booster) or external (pinpad booster)
- **APIs**
  - **SEC_GetConfig() :** To get the configuration of the security component for the current application
  - **SEC_SetConfig() :** To configure the secure component according to the available boosters for every secure part
  - **SEClib_Open() :** To open and initialise the security component
  - **SEClib_Close() :** To close and to disable use of services of the security component

TELIUM SECURITY : **Dll Security 2/5**

ingenico

- **Sample** Pin entry on PinPad, Card reader on terminal and encryption on PinPad.

```
T_SEC_CONFIG tabConfParts[C_NB_PARTS];
int ret;


// Pincode device on external PinPad (USB)
// =======================================
tabConfParts[0].ptszBoosterPeripheral = IAPP_USB_NAME;
// Card reader device on terminal
// ==============================
tabConfParts[1].ptszBoosterPeripheral = IAPP_DEF_NAME;
// Cypher/decypher on external PinPad (USB)
// ========================================
tabConfParts[2].ptszBoosterPeripheral = IAPP_USB_NAME;

// Set the configuration of the security component
// ***********************************************
ret = SEC_SetConfig(C_NB_PARTS, tabConfParts); CHECK(ret==OK, lblKO);
```

TELIUM SECURITY : **Dll Security 3/5**

Provide high level access to security functions

- These functions are independent from the product (B1,B2 or B3) by calling the corresponding schemes (TLV or not)

APIs

- **SEC_CreateSecretArea() :** To create a secret area => SloadKeyTlv
- **SEC_FreeSecretArea() :** To free a secret data or the whole secret area => SchFree TlvFree SchutilTlv

- **SEC_LoadKey() :** To load a key in a secret area => SchLoadData TlvLoadKey SloadKeyTlv SchutilTlv

- **SEC_PinEntryInit() :** To initialise the parameters for the secure pincode entry => None
- **SEC_PinEntry() :** To manage the secure pincode entry (On line) => SchGetPin (Dll)

- **SEC_Iso9564() :** Pin ciphering (ANSI X9.8) => SchIso95640 TlvIso9564 SchutilTlv
- **SEC_DukptLoadKSN() :** To load the key serial number for DUKPT algorithm=> SchDukpt TlvDukpt SchutilTlv
- **SEC_DukptEncryptPin() :** To encrypt the pin code **=> SchDukpt TlvDukpt SchutilTlv**

- **SEC_SubmitPIN() :** Submit Pin to the CAM via the secure part (Off line) => None

- **SEC_ECBCipher() :** ciphering/deciphering data in Electronic Code Book => SchCipherPR TlvCipherDa SchutilTlv
- **SEC_CBCCipher() :** ciphering/deciphering data in Cipher Block Chaining => SchCipherPR TlvCipherDa SchutilTlv

- **SEC_ComputeMAC() :** To compute MAC with DES ciphering in CBC mode => SchCipherPR TlvCipherDa SchutilTlv
- **SEC_VerifyMAC() :** To verify MAC with DES ciphering in CBC mode => SchCipherPR TlvCipherDa SchutilTlv

- **SEC_KeyVerify() : To verify KVC for a key loaded in secret area => SloadKeyTlv SchutilTlv TlvKeyVerify SchCipherPR**

TELIUM SECURITY : **Dll Security 4/5**

ingenico

Sample1 Create secret area and load Root key

```
#define VAR_ID              0x00
#define CARD_NB             0x0003C009
#define AREA_ID             0x00002030
#define BANK_ID             BK_SAGEM
#define ROOT_KEY_LOC        1008    // (Key number 252)

const byte TheRootKey [TDES_KEY_SIZE+1] = "\x11\x11\x11\x11\x11\x11\x11\x11\x22\x22\x22\x22\x22\x22\x22\x22";

T_SEC_DATAKEY_ID stRootKey;
int iUsage;
int ret;

// Create Secret Area
// ******************
ret = SEC_CreateSecretArea (C_SEC_CIPHERING,            // Secure type
                            (SEG_ID) AREA_ID,
                            CARD_NB,
                            VAR_ID); CHECK(ret==OK, lblKO);

// Load Root Key
// *************
stRootKey.iSecretArea = AREA_ID;
stRootKey.cAlgoType   = TLV_TYPE_KTDES;             // This ROOT key is a TDES Key
stRootKey.usNumber    = ROOT_KEY_LOC;
stRootKey.uiBankId    = BANK_ID;
iUsage = CIPHERING_KEY;                             // Key to cipher PIN key or MAC key
ret = SEC_LoadKey (C_SEC_CIPHERING,
                   NULL, &stRootKey, (byte*) &TheRootKey, iUsage); CHECK(ret==OK, lblKO);
```


ingenico

TELIUM SECURITY : **Dll Security 5/5**

Sample2

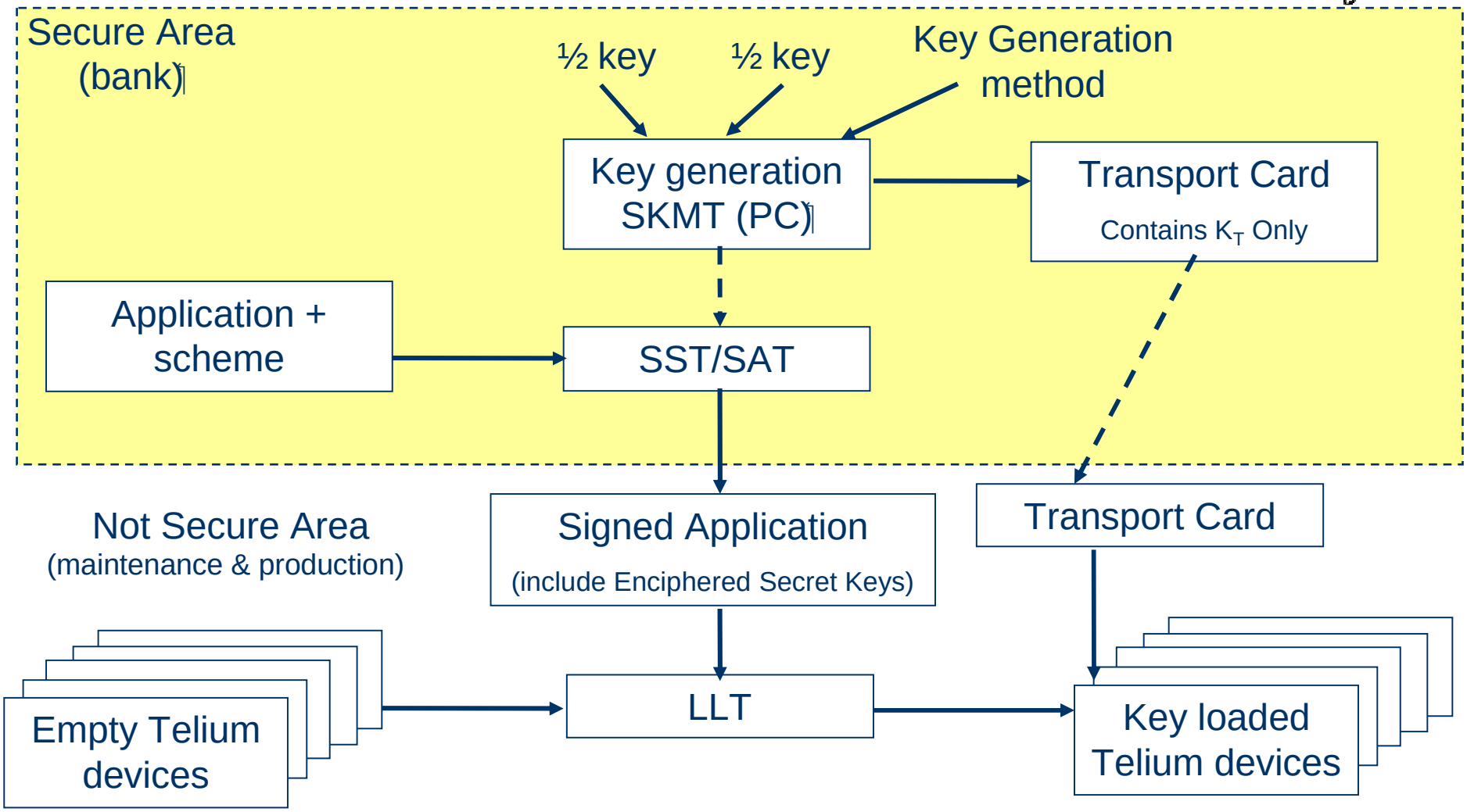
Load Pin key using Root key

```
#define VAR_ID              0x00
#define CARD_NB             0x0003C009
#define AREA_ID             0x00002030
#define BANK_ID             BK_SAGEM
#define ISO9564PIN_KEY_LOC  8   // (Key number 2))

const byte ThePinMasterKey [DES_KEY_SIZE] = "\x40\xB3\x9B\xA8\x55\xE0\x41\x16"; // unciphered is 6B218F24DE7DC66C

T_SEC_DATAKEY_ID stRootKey, stPinKey;
int iUsage;
int ret;

// Load Pin Key using Root Key
// ***************************
stRootKey.iSecretArea = AREA_ID;
stRootKey.cAlgoType   = TLV_TYPE_KTDES;              // This ROOT key is a TDES Key
stRootKey.usNumber    = ROOT_KEY_LOC;
stRootKey.uiBankId    = BANK_ID;

stPinKey.iSecretArea = AREA_ID;
stPinKey.cAlgoType   = TLV_TYPE_KDES;                // This PIN key is a DES Key
stPinKey.usNumber    = ISO9564PIN_KEY_LOC;
stPinKey.uiBankId    = BANK_ID;

iUsage = CIPHERING_PIN;                              // Key to cipher PIN entry
ret = SEC_LoadKey (C_SEC_PINCODE,
                   &stRootKey, &stPinKey, (byte*) &ThePinMasterKey, iUsage);
CHECK(ret==OK, lblKO);
```

TELIUM SECURITY : **SKMT2 initial key injection tool (1/4)**

- **Secure key management tool**
  - Injection of initial keys in secure area
- **Main features**
  - Secret keys entries : entry by 1/2 or 1/3 keys in secure room
  - Secure storage of these secret keys
  - Key loading in none secure room of batch of terminals
- **Secret keys are stored enciphered by transport key KT inside application**
- **KT is stored inside a Transport card, controlled by a pincode**

TELIUM SECURITY : **SKMT2 principle (2/4)**



TELIUM SECURITY : **SKMT2 features (3/4)**

- **Purpose**
  - Latest SKMT2 (>=6.0.1) allows to create secret area and inject keys in all terminals (T1 & T2)
- **SKMT2 features**
  - All B1/B2/B3 format compliant
  - All standard schemes compliance
  - All terminals T1/T2
  - If this tool does not comply with bank requirements
    - For Telium1/B1 : developer can develop its own injection key tool creating an EFT application using SchLoadKey (Plaintext format) scheme, creating its own SecretArea inside crypto processor.
    - For Telium1/B2 and Telium2 : Contact Ingenico, plaintext injection scheme available on request for customer injection key tool to be used in secure area only. Necessary to sign a commitment letter before delivery.
  - Two modes
    - SKMT2 is deliver in light mode by default, it is necessary to add an additional Database to activate full mode.
    - Light mode : only 1 cipher key can be loaded
    - Full mode : allowing others cipher keys to be loaded

TELIUM SECURITY : **SKMT2 loading keys (4/4)**

Where loading keys ?

- SKMT2 uses Security DLL and by the way the automatic configuration (on terminal or on Pinpad)
- If an external Pinpad is connected, all keys are injected inside Pinpad
- If no Pinpad plugged, all keys are injected inside terminal
- No need of SKMT in T10 Mockup mode, the scheme LoadKey allows to inject a plaintext key inside a Mockup terminal

TELIUM SECURITY : **SST Install 1/2**

- Launch GCR 410 card reader USB driver from SST CD root
- Copy your DataBase inside SST installation directory
- Launch SST, enter installation key

- *(Sign once all schemes with your own RSA card)*
- *SST batch mode - mock up : (see SST user's guide) : set pincode=0 in command file ,no DB nor RSA card)*
- *SST graphic mode - mock up : (see SST user's guide) : DB required, no card) add mockup = yes in file sst.ini*

- Keep RSA card + certificate + card pincode in a safe

(not on PC) for security (SST is major part of the security chain !!!)

TELIUM SECURITY : **SST Signature process 2/2**

- **Prepare first an appli.txt file** *(see.ADF format described in previous section)*
- **Insert RSA card and certificate from CD**
- **Select FIRST the necessary schemes**
- **Select "Sign application" from menu**
- **Select appli.txt file and appli.bin**
- **Enter RSA card pincode**
- **Select certificate when requested**
- **Enter extension number (2 digits xx) for .Mxx file**
- **Result is stored under xx\SST30\destination for application**

- *(Known issue : for a scheme … select "sign a scheme" (no need of .txt file) : take care to previously select the "signed scheme" folder on left window to get the .sgn file located in the relevant directory) … else they are in the current folder but can be moved anyway …*

TELIUM SECURITY : **SAT Install 1/2**

- Install SAT setup.exe (not yet embedded in Ingedev)
- Configure SAT/Tool management/Com port selection = COMx
- Prepare certificate ready to be selected
- Connect Signature Tool Terminal to Ethernet or GSM/GPRS
- Connect Signature Tool Terminal on PC/SAT through USB
- Launch application on terminal F/SGNSMO/START
- Insert signature card inside

TELIUM SECURITY : **SAT Signature process 2/2**

- Sign schemes first with SAT
- Use SAT to embed schemes inside the application
  - Select .bin application file
  - Select .txt application description file
  - Only one APPLI.AGN file is generated (instead of APPLI.SGN + APPLI.ADF) or …
    - AGN for application
    - PGN for parameter file
    - LGN for DLL
  - Equivalent "ADF file" content to be retrieved using: SAT/File/Dump/APPLI.AGN file
- Use Ingedev to embed schemes inside the application
  - Schemes manually embedded into Ingedev

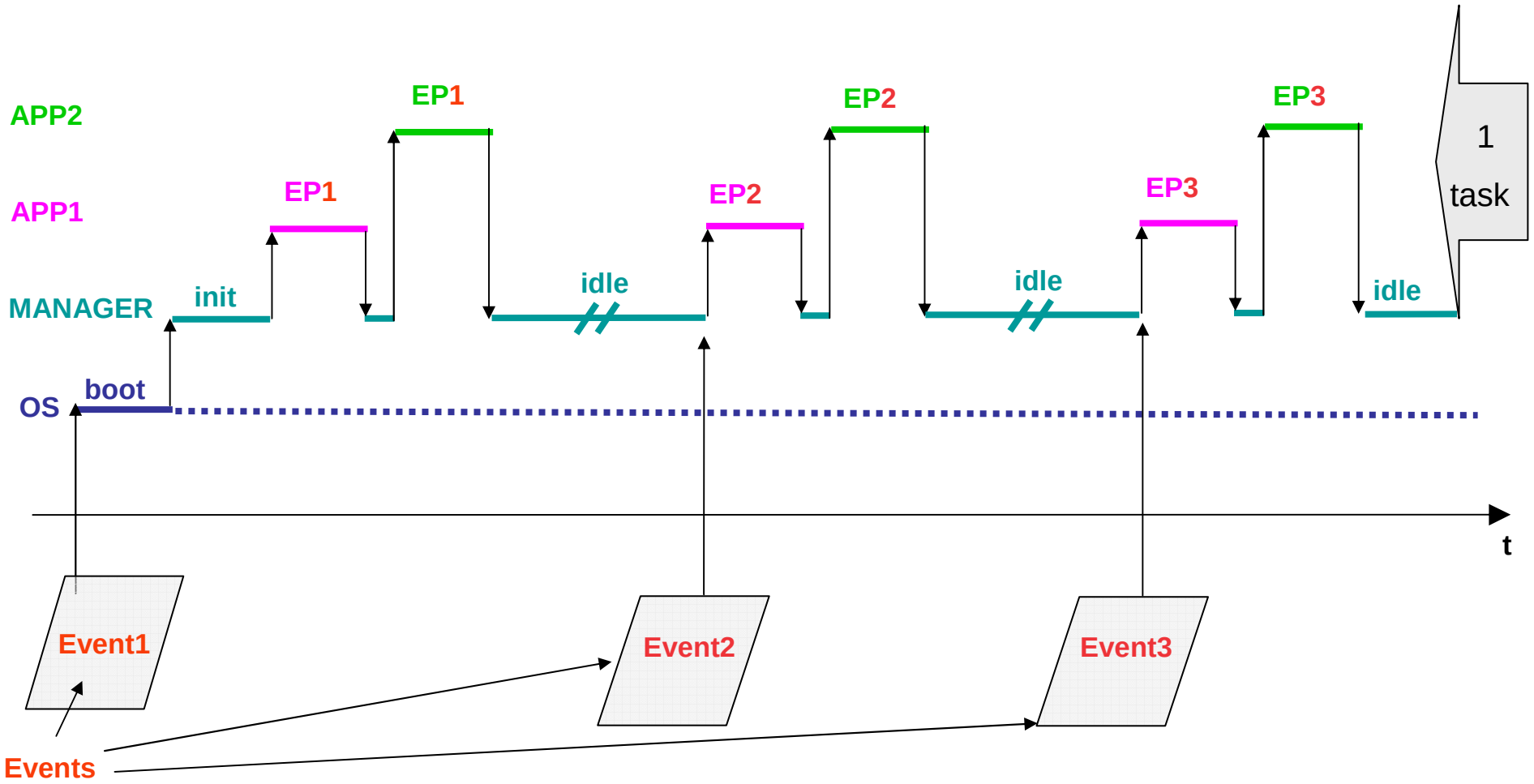
TDS Training : **Contents**

- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

TELIUM MANAGER : **Purpose**

- Software capabilities dedicated to POS terminals
  - Application supervision through Entry Points (EP)
  - Application selection mechanism for transaction processing
  - Management of common parameters (date/time, language, PinPad, Swipe, TMS, …)
  - Multi-language support (French and English by default, can be customised)
  - Maintenance functions (downloading, diagnostic, configuration)
  - Inter-application messaging (create your own event)
- Functions
  - Graphic library to draw and print text on display and printer (font management)
  - Navigation functions
  - Amount entry

TELIUM MANAGER : **Sequencer principle**



TELIUM MANAGER : **Application interface**

- **Entry points are application's functions lunched by the manager on events as :**
  - Reset
  - Key pressed
  - Clock (cyclic execution)
  - Magnetic card swiped
  - Smart card inserted
  - File received
  - It is also possible to create your own event using IAM mechanism…
- **Application must record services for all entry points at application boot**
- **2 types of entry point (EP)**
  - Polling EP : Manager polls all applications (broadcast)
    - Polling order is performed according to a defined priority
  - Selecting EP : Manager gives focus to one application
- **At the end of each entry point, application must return to Manager**
- **All EP have the same interface with 3 parameters :**
  - No : appli number, *p1 : data in (Manager to Appli), *p2 : data out (Appli to Manager)

TELIUM MANAGER : **Service Priority**

For all entry points (except Idle_Message)

Polling order is performed from highest to lower priority :

Priority value is
0 for highest, **FF** for lowest

Priority can be different for each entry point

If priority is equal, loading order in terminal determines polling order

For Idle_Message only

Message displayed is from function having highest priority :

Priority value is
FF for highest, **0** for lowest

!!! Caution : priority values are opposite of other entry points

If priority is equal => Conflict !
Manager displays its own message

TELIUM MANAGER : give_your_domain

- **int give_your_domain (… , void *p_in, S_INITPARAMOUT *p_out)**
  - First event received at terminal start-up (Polling according to priority)
  - No data coming from Manager
  - Data to return to Manager
    - Application_type
      - TYPE_EXPORT for all type of application outside France
      - TYPE_CARTE for french banking environement
      - TYPE_HEALTH for french health environment
    - mask
      - MSK_DATE | MSK_LANGUE | MSK_STANDARD | MSK_PINPAD | MSK_SWIPE | MSK_MDP…
        … Menu Function F031 PARAMETERS INITIALIZATION
        1 - Date and time
        2 - Language
        3 - Pabx
        4 - PinPad
        5 - Swipe
        6 - T.M.S
      - 0xFF All functions present

TELIUM MANAGER : **after_reset**

- **int after_reset (…, void *p_in, S_TRANSOUT *p_out)**
  - Activated on each terminal reset (Polling according to priority)
  - No data coming from Manager
  - Data to return to Manager
    - S_TRANSOUT
      - » If previous transaction was interrupted
  - Application reset processing
    - Check if cold reset (after downloading) or warm reset (no downloading)
    - Initialise data and create disks
    - Latest moment to call RegisterPowerFailure()
    - Load and Open all mandatory Dlls
      - » Note that these 2 last cases Dll and RegisterPowerFailure() can also be moved to entry() function, just after the service registering command.

TELIUM MANAGER : is_name

- **int is_name (…, void *p_in, S_ETATOUT *p_out)**
  - Activated when Manager wants to get application name (Polling/Selecting)
  - No data coming from Manager
  - Data to return to Manager
    - appname
      - » Application name to be displayed at operator side
    - no_appli
      - » Application number
  - This event is also returned many time by others events. Manager does not save the application name and number.

TELIUM MANAGER : is_state

- **int is_state (…, void *p_in, S_ETATOUT *p_out)**
  - Manager checks if application is initialised or not
  - Called every minute and at boot (Polling according to priority)
  - No data coming from Manager
  - Data to return to Manager
    - response
      - » REP_OK when application is initialised
      - » REP_KO when application is not initialised
  - At the end Manager checks if there is at least one initialised application in order to display the idle message otherwise the message initialise an application is displayed.
  - Is_name() is also called

TELIUM MANAGER : idle_message

- **int idle_message (…, void *p_in, void *p_out)**
  - Allows an application to display its own idle message
  - Activated when Manager goes back idle
    - Polling on higher priority, if conflict Manager display is own message
  - No data coming from Manager
  - No data return to Manager
  - This entry point is launched by a specific Manager fork task

TELIUM MANAGER : **more_function**

- **int more_function (…, void *p_in, void *p_out)**
  - Allows applications to execute their own functions via menus
  - Activated when pressing on **F key** to select the right application
    - Selecting, only call the application selected to reach the menu
  - No data coming from Manager
  - No data to return to Manager

TELIUM MANAGER : keyboard_event 1/2

- **int keyboard_event (…, void *p_in, void *p_out)**
  - Activated when key is pressed when terminal is in idle mode
    - Polling according to priority
  - Data coming from Manager
    - keycode
      - Key pressed
  - Data to return to Manager
    - keycode
      - same key : no process
      - 0 : key disabling
      - Another key : manager routing key

TELIUM MANAGER : keyboard_event 2/2

- Examples

```
// "F" Menu Function is hidden
// ===========================
case T_F:                          // Key "F" pressed
     p_out->keycode = 0;           // The key "F" is disable
                                   // no more access to Menu Functions  (routine Manager)
     break;

// "1" call "F" Menu Function
// ==========================
case N1:                           // Key "1" pressed
     p_out->keycode = T_F          // Return key "F"
                                   // key "1" will access the Menu Functions (routine Manager)
     break;


// Enable numeric keys to enter amount first
// =========================================
case N0: case N1: case N2: case N3: case N4:
case N5: case N6: case N7: case N8: case N9:
case T_VAL: case T_POINT: case F2: case F3:
     p_out->keycode = p_in->keycode;       // Enable numeric keys to enter amount first
     break;                                // then enter card or keyed (routine manager)

// Disable numeric keys to avoid enter amount first
// ================================================
case N0: case N1: case N2: case N3: case N4:
case N5: case N6: case N7: case N8: case N9:
case T_VAL: case T_POINT: case F2: case F3:
     p_out->keycode = 0;                  // Disable numeric keys (no action from manager)
     break;
```

TELIUM MANAGER : state consult mcall

- **int state (… , void *p_in, void *p_out)**
  - Allows applications to print their own state receipt (Polling according to priority)
    - application name, loadable file name, version, checksum, comments, etc…
  - Activated on F011 : Consultation ->State

- **int consult (… , void *p_in, void *p_out)**
  - Allows applications to print transactions total receipt (Polling according to priority)
    - Number of debit/credit transaction, Totals of debit/credit transaction, etc…
  - Activated on F012: Consultation -> Transaction

- **int mcall (… , void *p_in, void *p_out)**
  - Allows applications to print their own calls schedule receipt (Polling according to priority)
    - Batch release, Hotlist, Parameters, etc…
  - Activated on F0131: Consultation -> Call -> Planning of call

TELIUM MANAGER : is_time_function time_function

- **2 functions allow automatic execution of periodic functions**
  - **int is_time_function (…, void *p_in, S_ETATOUT *p_out)**
    - Manager asks if the peripherals should be closed at the next call of time_function()
    - Called every minute (Polling according to priority)
    - No data coming from Manager
    - Data to return to Manager
      - response
        - REP_OK Manager will close all peripherals at the next call of time_function()
        - REP_KO Manager will keep peripherals opened at the next call of time_function()
    - is_name() is also called
  - **int time_function (…, void *p_in, void *p_out)**
    - Called after is_time_function()
    - No data coming from Manager
    - No data to return to Manager
    - Dedicated to batch close at a specific time during the day

TELIUM MANAGER : is_change_init 1/2

- **int is_change_init (…, void *p_in, S_ETATOUT *p_out)**
  - Manager asks if parameters can be changed (Polling)
  - Activated each time Manager wants to change its own parameters
  - No data coming from Manager
  - Data to return to Manager
    - mask
      - Mask of bits representing one parameter each (**0** : accepting , **1** : refusing)
      - MSK_DATE | MSK_LANGUE
        - … Menu Function F031 PARAMETERS INITIALIZATION
          1 - Date and time
          2 - Language
          3 - Pabx
          4 - PinPad
          5 - Swipe
          6 - T.M.S
      - **0** accept all parameters changing
      - Manager prints which applications have refused the modification
  - is_name() is also called

TELIUM MANAGER : **modif_param 2/2**

- **int modif_param (…, S_MODIF_P *p_in, void *p_out)**
  - Manager reports parameters changing (Polling)
  - Activated each time Manager changed its own parameters
  - Data coming from Manager
    - mask
      - » Mask of bits representing one parameter each (**0** : not modified , **1** : modified)
  - No data to return to Manager

TELIUM MANAGER : is_evol_pg is_delete

- **int is_evol_pg (…, void *p_in, S_ETATOUT *p_out)**
  - Manager asks for a downloading session (Polling according to priority)
  - Activated each time Manager wants to start a downloading session
  - No data coming from Manager
  - Data to return to Manager
    - response
      - REP_OK Application authorizes the downloading process
      - REP_KO Application refuses the downloading process
  - Manager prints which applications have refused the downloading session
  - is_name() is also called
- **int is_delete (…, void *p_in, void *p_out)**
  - Manager asks for application deletion (Selecting)
  - Activated each time Manager wants to delete an application
  - No data coming from Manager
  - Data to return to Manager
    - response
      - DEL_YES Application authorizes the deletion process
      - DEL_NO Application refuses the deletion process
  - Manager will display a message regarding the application which has refused the deletion

TELIUM MANAGER : **file_received**

- **Int file_received (…, S_FILE *p_in, void *p_out)**
  - Manager reports parameters file received from LLT
  - Activated upon reception of a parameter file by the manager
  - File with extension .PAR and deleted by Manager after polling
  - Data coming from Manager
    - S_FILE contains diskname and filename to be used
    - Use FS_mount to mount the disk
    - Use FS_open to get data
  - No Data to return to Manager
  - !!! Warning The file should be loaded in HOST disk

TELIUM MANAGER : **message_received 1/3**

- **Int message_received (…, S_MESSAGE *pin, void *pout)**
  - Inter application messaging
  - Activated each time Manager received a message in its mailbox for this application
  - Data coming from Manager
    - Sender => Where the message is coming from
    - Message type => Selecting or polling message
    - Message length => Message size
    - Message value => Message contain
  - No data to return to Manager
  - Use Send_Message() to send back the message to the sender
  - Used through IAM mechanism

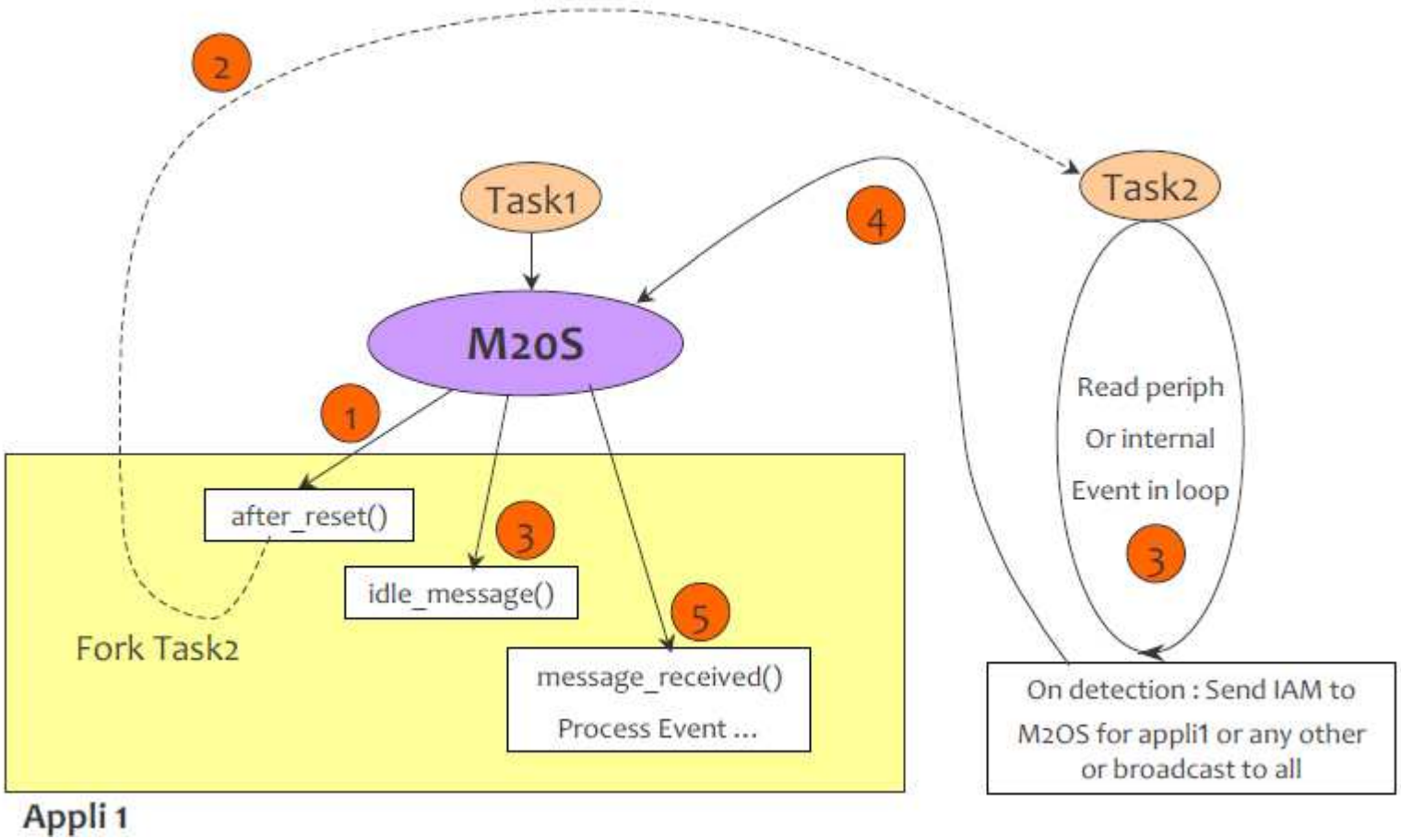
TELIUM MANAGER : message_received 2/3

- Sample

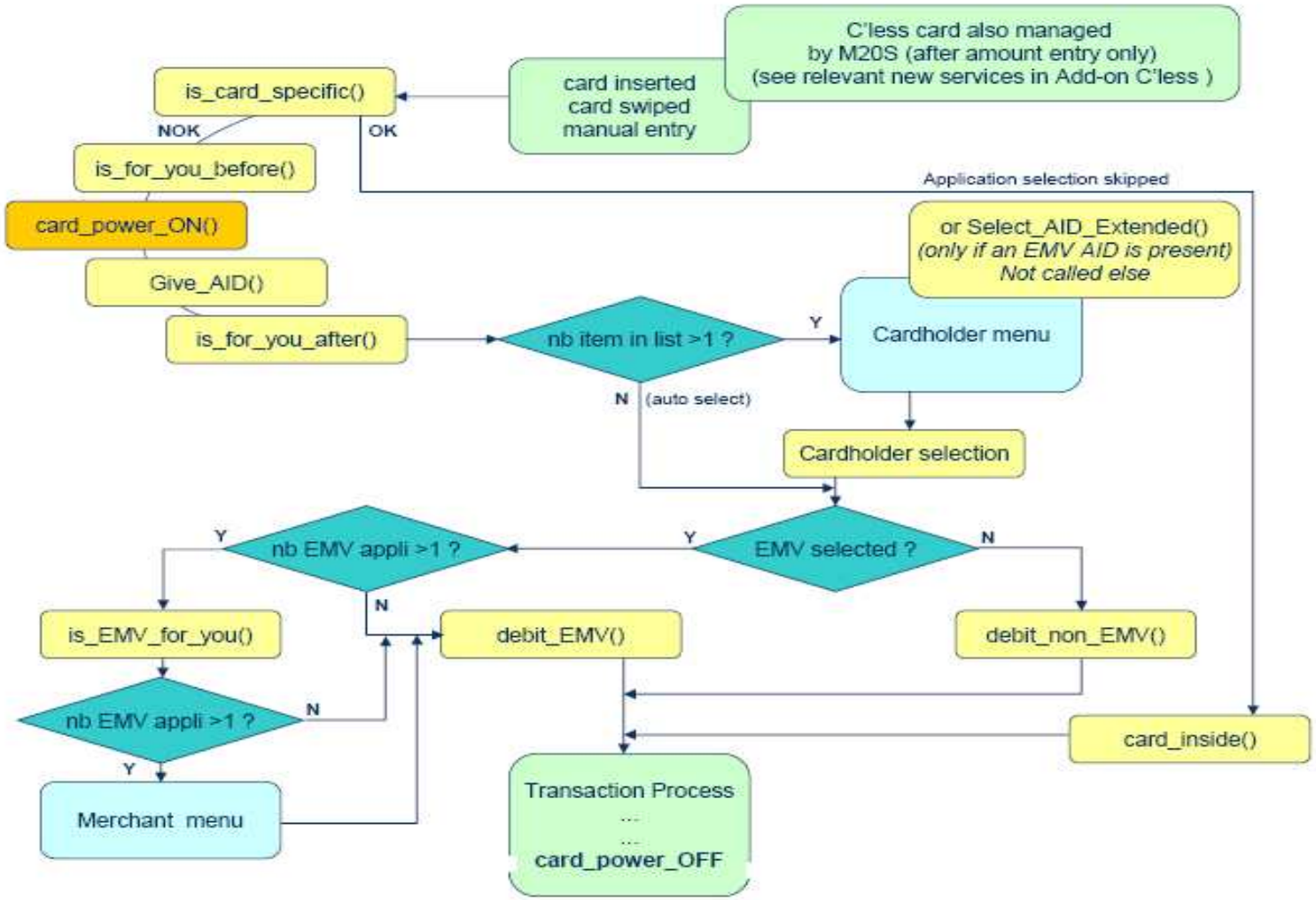
```
// Send message to USER2 through Manager mailbox
// =============================================
strcpy(snd, "Hello, do you hear me?");                          // Data to send

memset(message.value, 0x00, sizeof(message.value));
message.sender = TRAINING_APPLI_TYPE;                           // TRAINING is the sender
message.receiver = (TaskApplication*256) + USER2_APPLI_TYPE;    // USER2 is the receiver
message.type = 0;                                               // IAM type = 0
message.length = strlen(snd);                                   // Message length
memcpy(message.value, (unsigned char *) snd, message.length);   // Copy the message
Send_Message(&message);                                         // Send message to Manager mailbox
                                                                // Manager will call USER 2 with the message
```

TELIUM MANAGER : message_received 3/3

- **IAM from task to application**



TELIUM MANAGER : **Application Selection (1/2)**



TELIUM MANAGER : **Application Selection (2/2)**

- **is_card_specific()** => *to catch a card before any selection (loyalty, healthcare, specific …)*
  - **is_for_you_before()** => *voting for list #1 (Card power-on/reading/power off)*
    Only for application handling "old" cards not supporting software reset
  - **Card Power ON** => *After this step the card SHALL NOT be powered off before the end of the transaction*
  - **give_AID()** => *voting for list #2 (EMV Candidate list built by EMV kernel)*
  - **is_for_you_after()** => *voting for list #3 (Card software reset and reading only)*
    Event is given as entry data : Manual entry, non EMV chip card, Swiped card and Fallback process …
  - *Note that all responses can be conditional as the current list is given at each call*

  - **Cardholder Menu**, *built by M2OS concatenating lists #1, #2, #3 representing the mean of payment (or card issuer).*
    Can be customized using **Select_AID_Extended()** only if an EMV AID is present (not called else) and only for list#1 & list#3
    EMV list#2 section CANNOT be changed according to EMVCo certification..
    If list has only one item, then Menu is skipped and debit_???() is called directly.
    - **debit_non_EMV()** => *all non EMV cases, event also given as entry data*
    - **debit_EMV()** => *if selected AID is handled by a unique application*
    - **is_card_EMV_for_you()*** => *if selected AID is handled by several EMV applications (*optional)*
      - **debit_EMV()** *of selected AID if only one response*

      - **Cardholder Menu** *is to be managed for merchant, to select acquirer*
        - **debit_EMV()** *of selected AID /acquirer is called*

- **card_inside()** => *if selected at is_card_specific(), all previous application selection skipped*
- **Cards power off** *at the end of the transaction in all cases*

TELIUM MANAGER : **Get Manager Profile functions**

Menu Function F031 PARAMETERS INITIALIZATION

- Manager International Parameters
  - PSQ_Give_Date_Format
  - PSQ_Give_Language
  - PSQ_Is_PSTN, PSQ_Idf_telechgt
  - PSQ_No_Standard
  - PSQ_Is_pinpad, PSQ_Pinpad_Type
  - PSQ_is_isox
  - PSQ_Update_Language, _pabx, _phonenumber
- Manager French Bank Parameters
  - PSQ_No_terminal
  - PSQ_Est_Money
  - PSQ_Est_chainage
  - PSQ_Donner_noserie
  - PSQ_Est_caisse
  - PSQ_Donner_reference
- Manager French HealthCare Parameters
  - PSQ_get_no_lecteur
  - PSQ_get_vitesse_lecteur

TELIUM MANAGER : **Navigation and Data entry**

- Amount entry
  - **GetAmount()**
- Navigation and entry routines
  - **G_list_Entry() :** Manage Menu
  - **G_Numerical_Entry() :** Enter a numerical value
  - **G_Alphanumerical_Entry() :** Enter an alphanumerical value from a list
  - **G_extended_Entry() :** Enter an alphanumerical value in a grid or "GSM style"
- Each function above is always followed by **Get_Entry()** to get result value

TELIUM MANAGER : **Graphic management (1/5)**

- Graphic Library via a DLL
  - Compatible B/W and Color terminals
  - Colorized mode
- A graphic context of
  - 128*64 pixels

Color Library via a DLL

- Only Color terminals
- Full Color mode

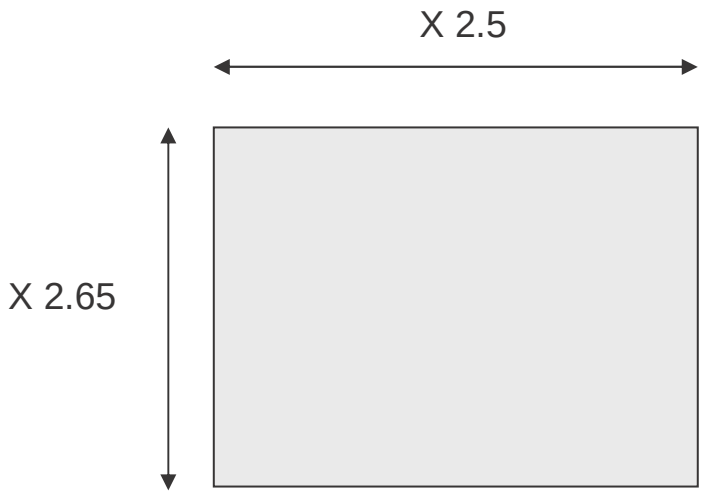
A graphic context of

- 320*240 pixels

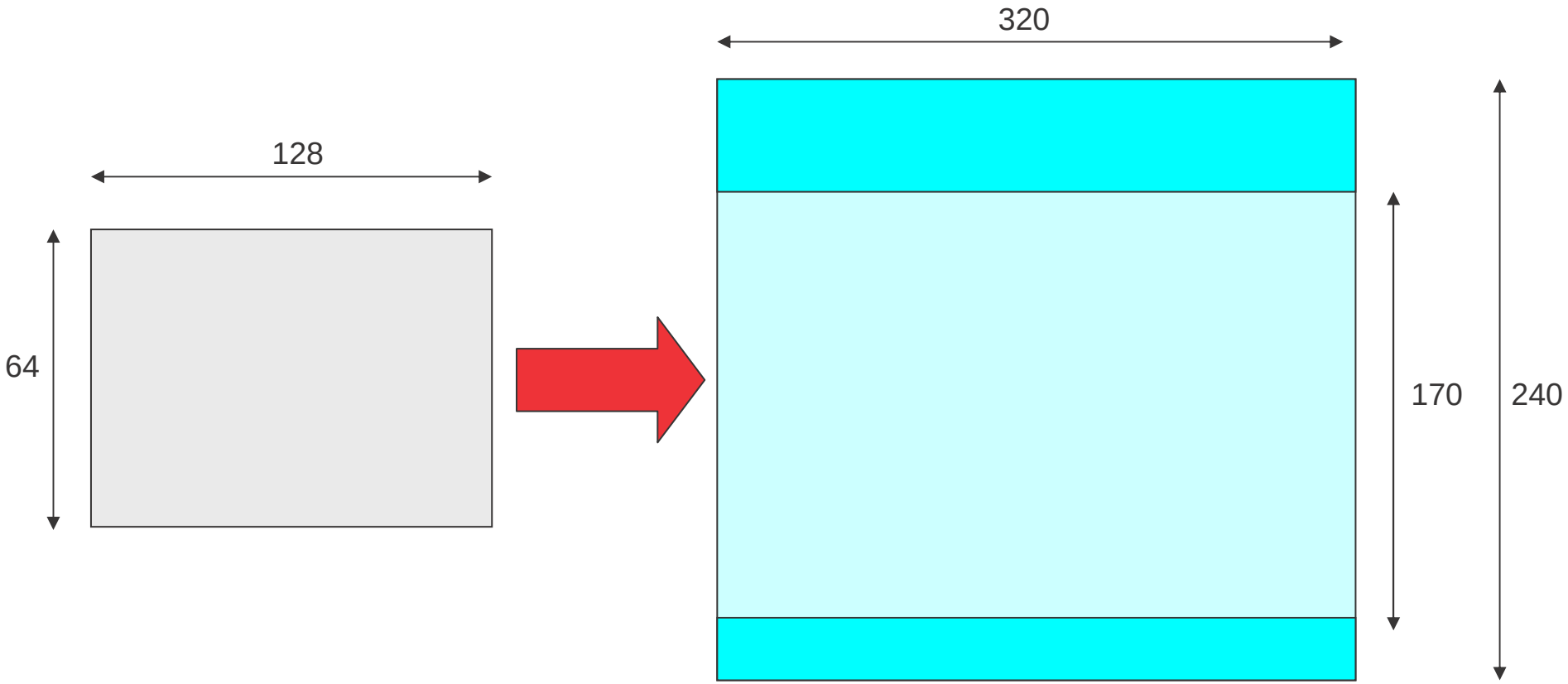
- Proportional and fixed size font
- Allows create drawings, mix texts and bit map, pixels management
- Bit map using routine, PC converter tool
- Reverse video attribute only on Graphic Library

TELIUM MANAGER : **Graphic management (2/5)**

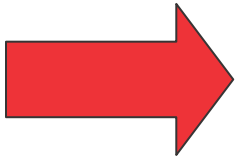
- What is Colorized Mode?
  - Emulation mode: monochrome application runs on a color terminal
  - header controlled by the manager
  - Text & background colors setting through Manager menu (Initialization/Hardware/Display/Color Setup)
  - Expansion factor performed by manager

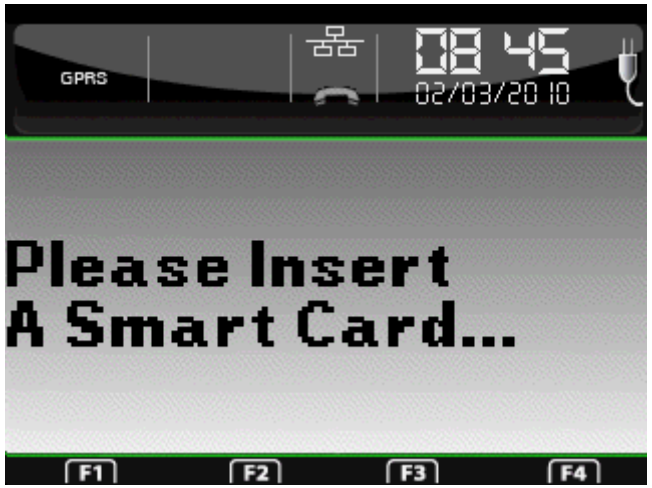


TELIUM MANAGER : **Graphic management (3/5)**



TELIUM MANAGER : **Graphic management (4/5)**





TELIUM MANAGER : **Graphic management (5/5)**

- What is Full Color Mode ?
  - Specific color graphic functions provided by color DLL
  - Monochrome and color compatibly must be performed at application level by using the right set of functions
  - Use IsColorDisplay() to test display type

TELIUM MANAGER : **Graphic & Color APIs (1/3)**

- 2 types of functions
  - **Function() :** refresh graphic memory AND display
  - **_Function() :** refresh only graphic memory (update display with PaintGraphics() or Paint())
- Pixels and bitmap management
  - Graphic Library / Color Library
  - **SetPixel()** **Pixel()** : set or reset a pixel in the graphic context
  - **GetPixel()** **GetPixelColor()** : get the pixel value
  - **DisplayBitmap()** **DrawBmp()** : insert a BMP file in the graphic context

TELIUM MANAGER : **Graphic & Color APIs (2/3)**

- Drawing routines
  - Graphic Library / Color Library
  - **DrawLine()** **Line()** : draw a line between 2 points
  - **DrawRect()** **Rect()** : draw a rectangle
  - **DrawCircle()** **Circle()** : draw a circle
  - **DrawEllipse()** **Ellipse()** : draw an ellipse
- Text routines
  - Graphic Library / Color Library
  - **DrawExtendedString8859()** **DrawText8859()** : set an ISO 8859 text in the graphic context
  - **DrawExtendedStringUnicode()** **DrawTextUnicode()** : set a UNICODE text in the graphic context

TELIUM MANAGER : **Graphic & Color APIs (3/3)**

- Graphic context management
  - **InitContextGraphics() :** create and initialise the graphic context
  - Graphic Library / Color Library
  - **GetScreenSize()** **GetColorScreenSize()** : get the height and width of the screen
  - **PaintGraphics()** **Paint()** : display the graphic context
  - **SaveScreen()** **SaveScreenExtended()** : save the whole graphic context
  - **RestoreScreen()** **RestoreScreenExtended()** : restore the whole graphic context
  - **clrscr()** **ClearScreen()** : clear the whole graphic context
  - **SetRegion()** **SetRegionColor()** : set or select a region
  - **GetRegion()** **GetRegionColor()** : get current region coordinate
  - **ClearRegion()** **ClearRegionColor()** : clear a selected region
  - **SetArea()** **Area()** : fill or clear a part of the graphic context
  - **CopyArea()** **RedrawUserArea()** : copy a part of graphic context somewhere
- Refer to "Graphic Library User Guide" & "Color Library User Guide"

TDS Training : **Contents**

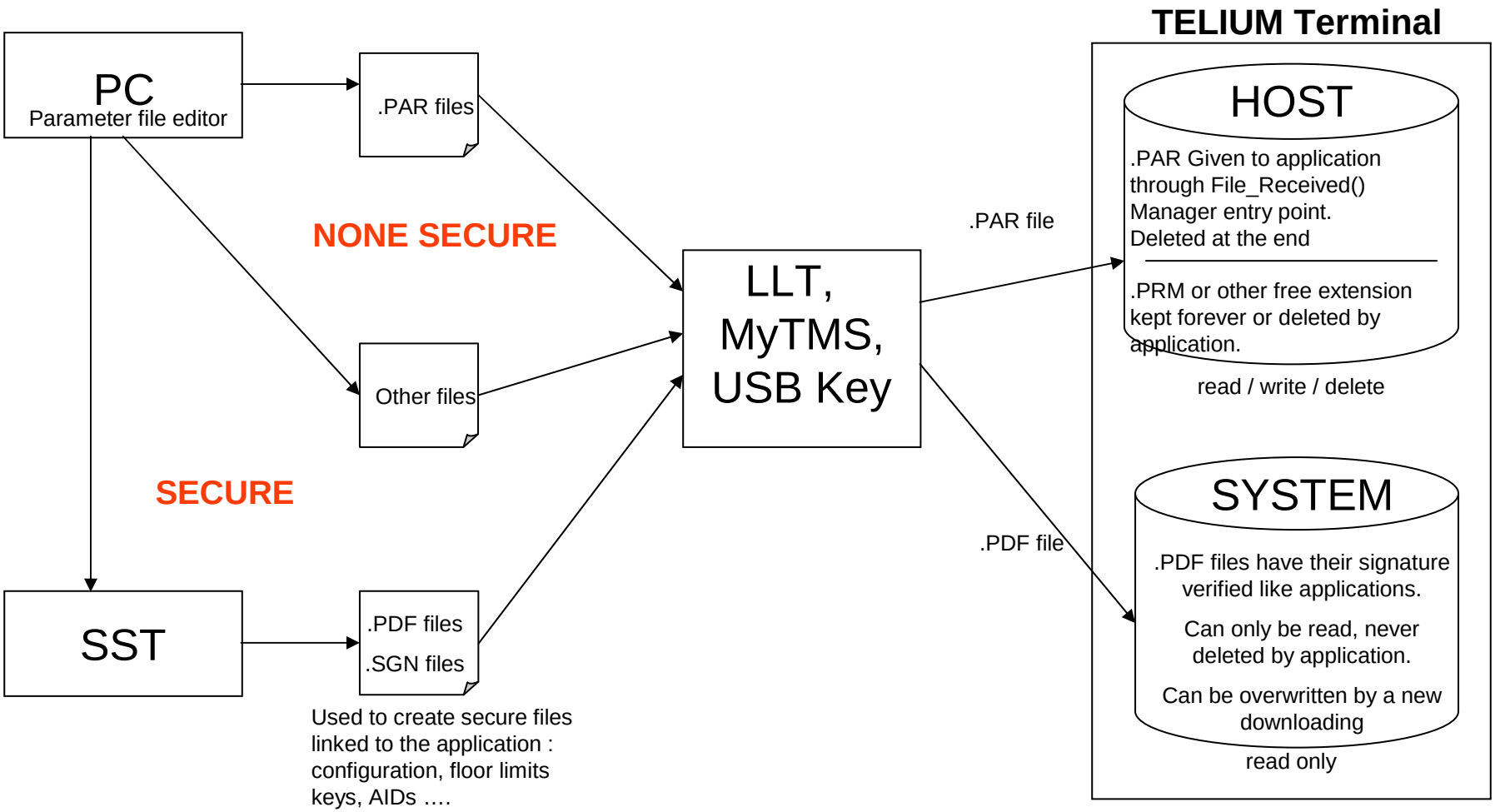
- Hardware and software architecture
- Software development steps and tools
- OS APIs
- Security
- Manager
- Miscellaneous

TELIUM : **Pinpad range**

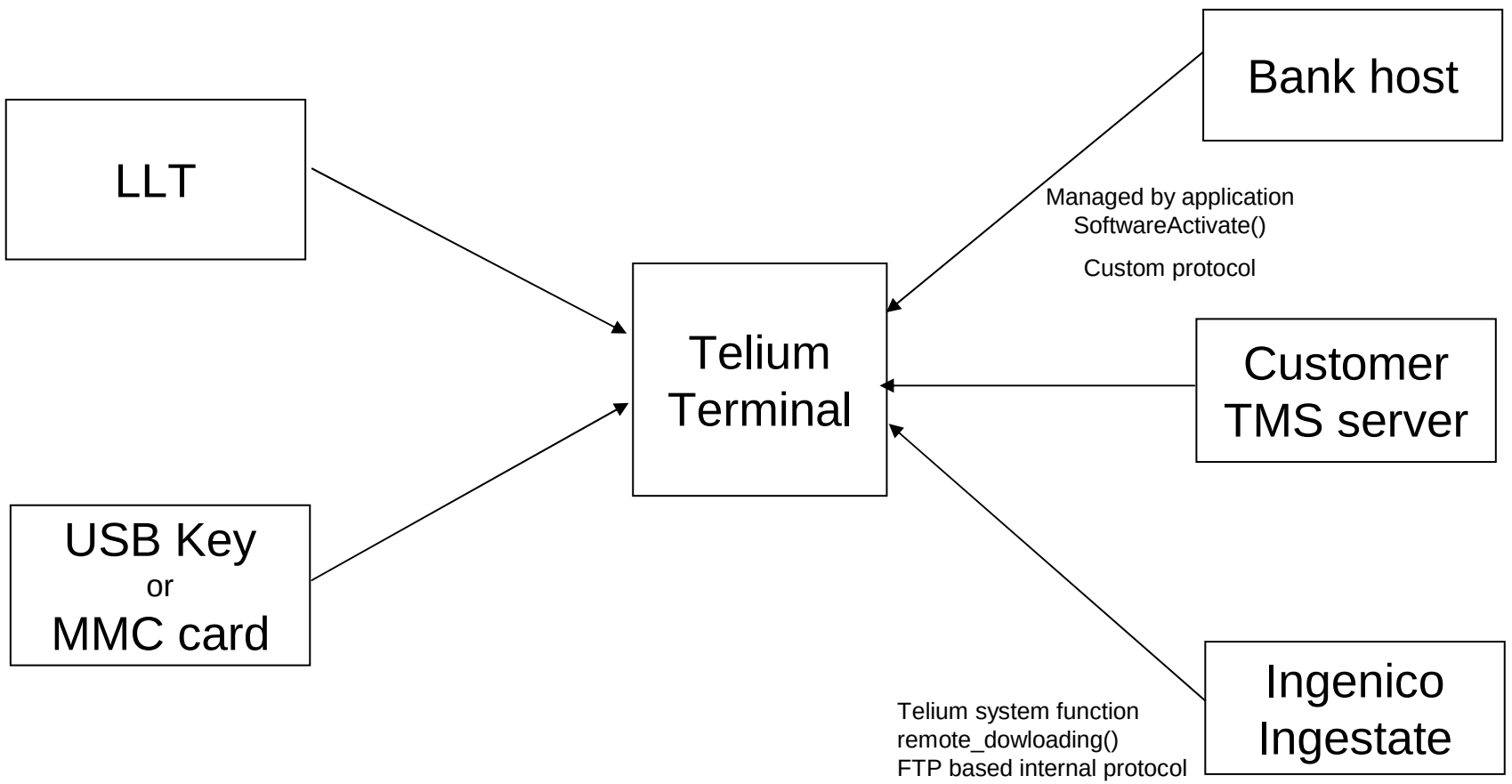
	PP30	PPC30	PPR30	P30	PP30S
crypto	DES	schemes	schemes	schemes	schemes
Chip reader	no	no	yes	yes	no
mag reader	no	no	no	yes	no
link	serial TTL	USB	USB	USB	USB
PC DLL	no	no	no	yes	no

- managed by the DLL security
- refer to "TELIUM DLL security user's guide“ and relevant “sample”

TELIUM : **Parameter files management**



TELIUM : **Terminal downloading**



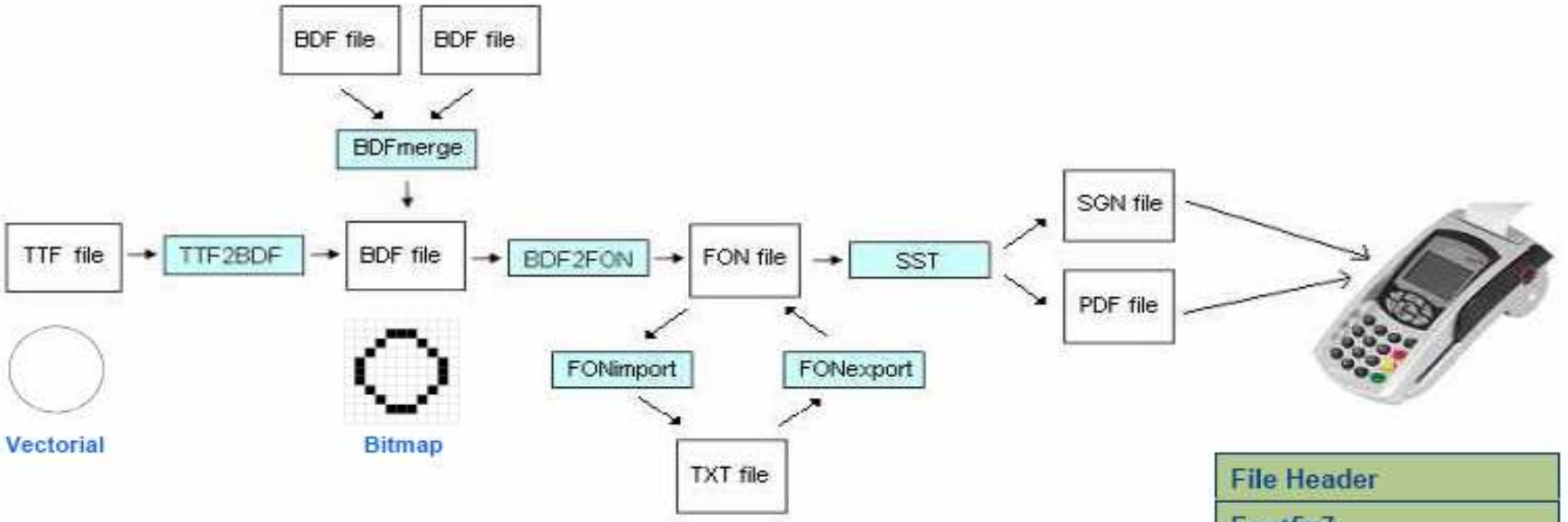
Local loading

Remote loading

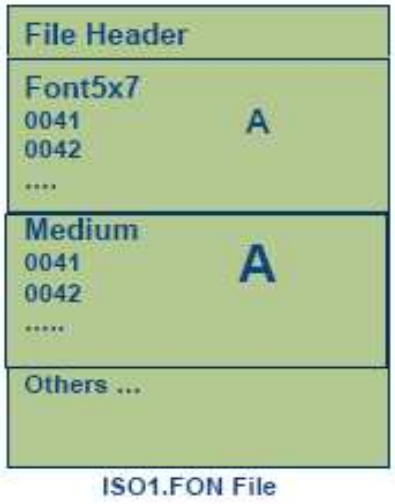
TELIUM : New Graphic and Font Library (1/5)

- Purpose :
  - SDK provides a set of tool for font generation
  - Replace defdisplaypattern()/defprinterpattern() (still supported)
  - Support format .TTF or .BDF conversion
  - Character set ISO8859 and Unicode compatible
  - Supported by Telium graphic library
  - See "FON tool user guide" and "TELIUM FontLib user guide"

TELIUM : **Create a font (2/5)**



A font file may content several size of the same font or several different fonts



> *See also « Font Tool user's guide » and Sample Fonts in SDK*

CAUTION : TTF fonts may be under license, use free fonts

TELIUM : **ISO 8859 Fonts coding (3/5)**

ingenico

- **ISO 8859 standard :**
  - ASCII 8 bits Extended coding with 128 additional characters per table.
  - Character depends on the selected table (or subset)
  - Define 15 tables : ISO8859-1, ISO8859-2… to adapt subsets to european languages
  - But doesn't cover all languages (Graphic languages)
  - Allows to write « left to right » or « right to left » (Arabic…)
  - Allows Character « position dependant » to change according to their location within a word : (Arabic style) (ISO 8859 mode ONLY … *for unicode it is necessary to specify each char.)*

TELIUM : **Unicode Fonts coding (4/5)**

ingenico

- **UNICODE standard :**

  - No ambiguity : a character number always represents the same character
  - A character is hexa 16 bits coded from 0x0000 to 0xFFFF
    (usually 32 bits but limited to 16 bits in TELIUM)
  - Universal : most of characters, symbols … are already defined.
  - Allows to write « left to right » or « right to left » (Thai …)

  - To convert text in UNICODE :
    - Enter text in Windows Notepad
    - Export to unicode with Notepad : 'save as' / select 'Unicode doc' format
    - Open same file with Hexadecimal editor
    - Copy / Paste inside C source code

ingenico

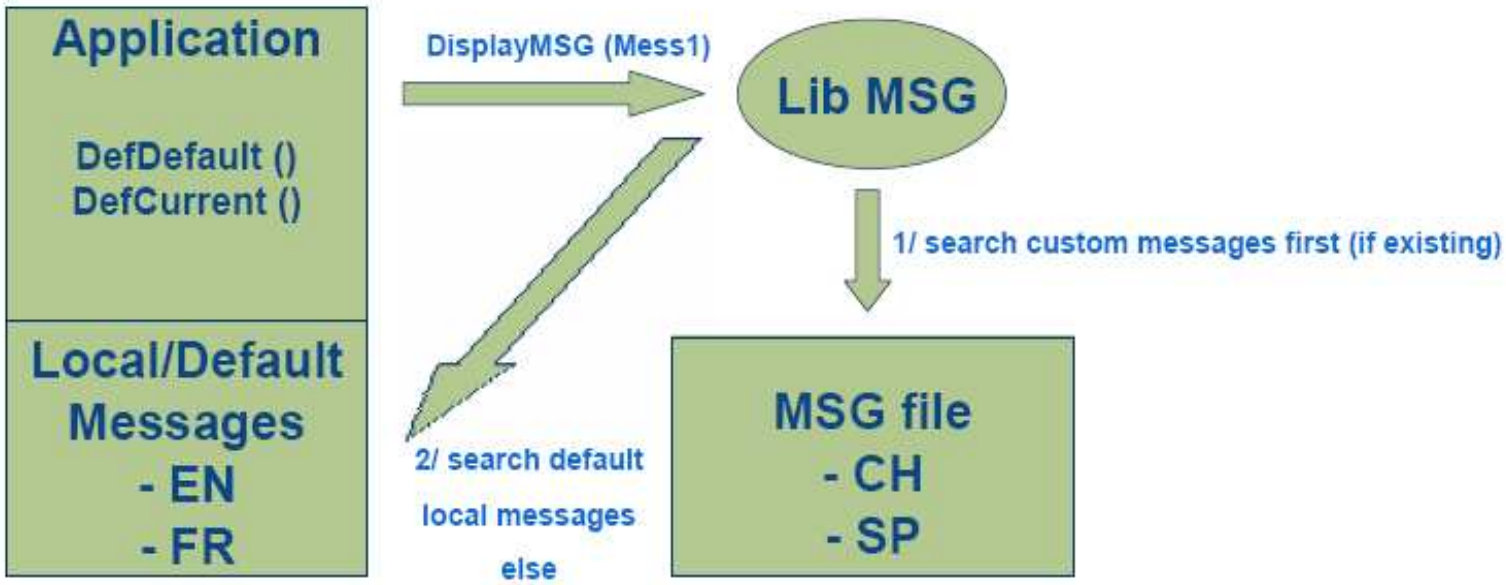
TELIUM : **Sample (5/5)**

In sample below, ISO5.SGN and ISO5.PDF shall be loaded in the terminal before using
« see also Fontlib User's guide »

```
MyFont=LoadFont("/SYSTEM/ISO5.SGN");        // Load FON file in RAM
DefCurrentFont(MyFont);                     // Set Default font

// Draw and Print « HELLO » from UNICODE format
DrawExtendedStringUnicode (0,0,"\x00\x54\x00\x58\x00\x45\x00\x54\x00", _OFF_, _dMEDIUM_, _FIXED_WIDTH_);
pprintfUNICODE("\x00\x54\x00\x58\x00\x45\x00\x54\x00\n\x00", _OFF_, _pBOLD_, _FIXED_WIDTH_);

// Draw and Print « HELLO » from ISO8859 format
DrawExtendedString8859 (0,0,"HELLO ", _OFF_, _dMEDIUM_, _FIXED_WIDTH_);
pprintf8859("\HELLO\n", _OFF_, _pBOLD_, _FIXED_WIDTH_);
```

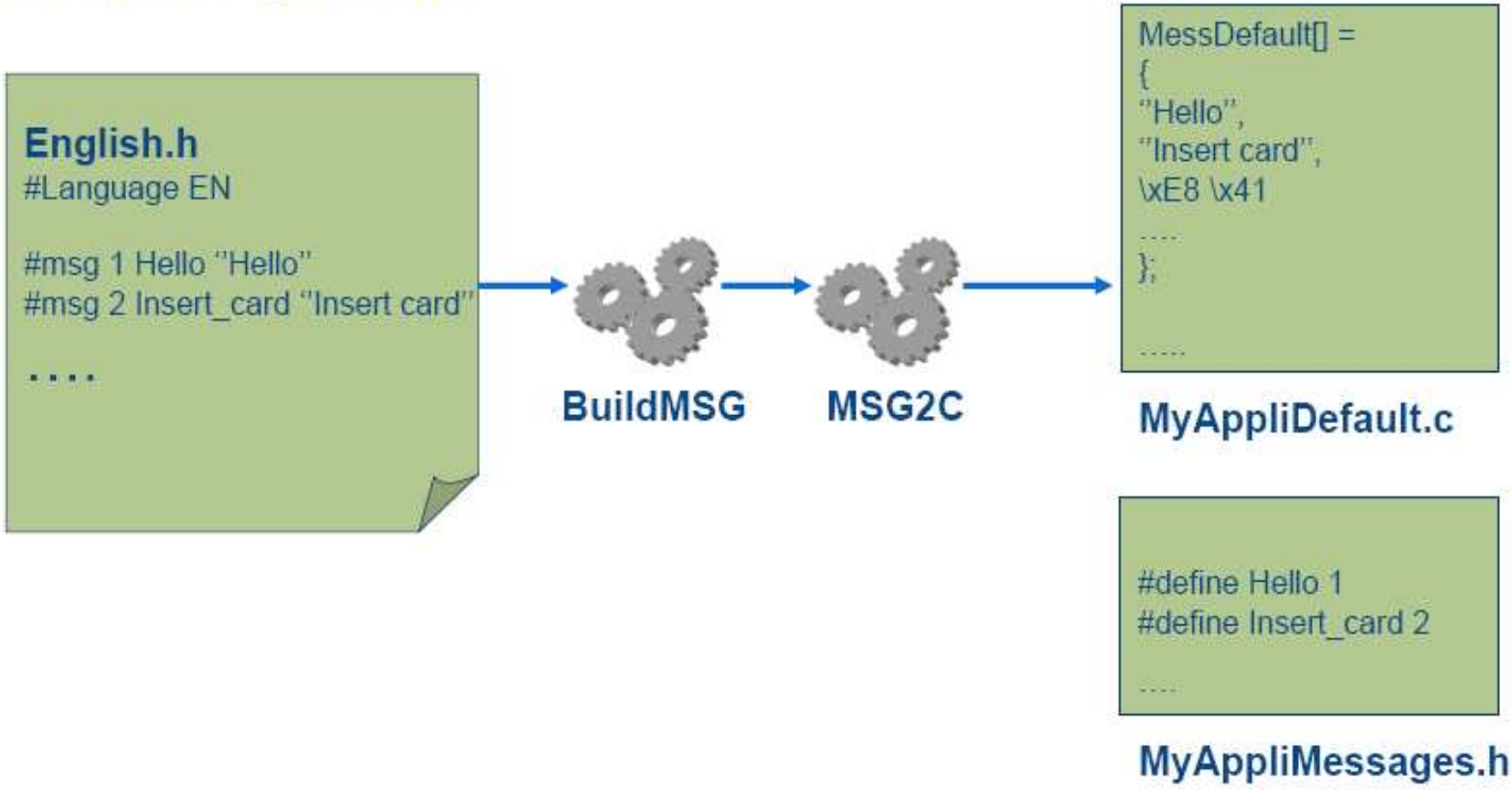
TELIUM : **Message Management (1/5)**

- Each application (or Manager) embed its own default messages (at compilation step)
- Each application (or Manager) can use customized messages
  (coming as a parameter file to be loaded in terminal)
- Manager customization consists in M2OS messages and Hardware Dll messages
- *(Pinpad DLL is not supported for system messages but can be customised at application level)*

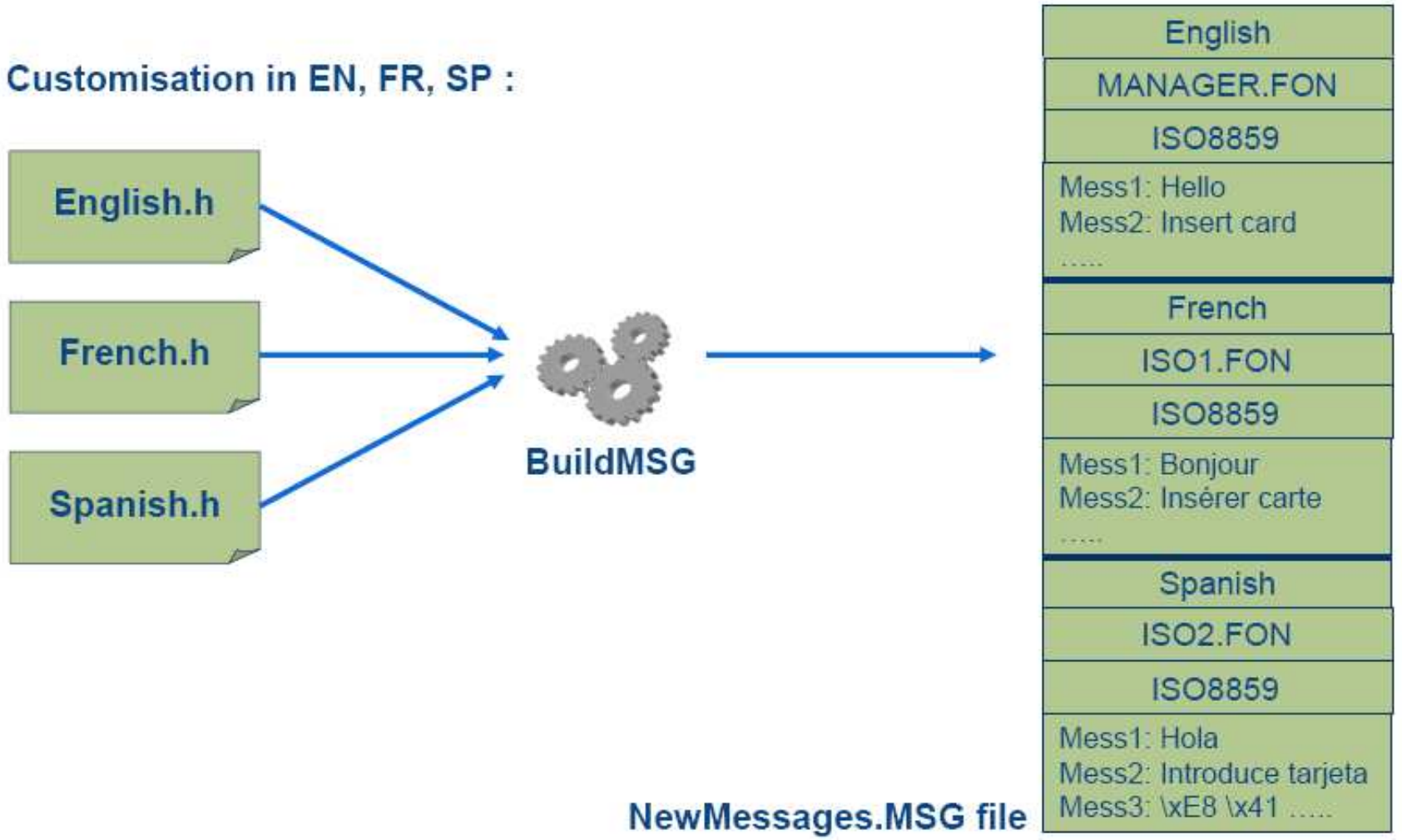


TELIUM : **Message Customisation (2/5)**

Default messages in EN :



TELIUM : **Message Customisation (3/5)**



TELIUM : **Using Messages files (4/5)**

- **To be loaded in Terminal :**
  - Application
  - Message file (NewMess.MSG) -> .PDF & .SGN)
  - Fonts Files (ISO1.FON, ISO2.FON, MANAGER.FON) -> .PDF & .SGN

  - Example of use : *(see also sample fonts in SDK)*

```
#include "MyAppliDefault.h"
#include "MyAppliMessages.h"

char * MyMSG;

MyMSG =LoadMSG("/SYSTEM/NewMessages.SGN");  // Load the MSG file in memory
DefCurrentMSG(MyMSG);                       // Register the current MSG file
DefDefaultMSG(MyAppliDefault);              // Register the default messages
DefCurrentLang("EN");                       // Register the current language
DefDefaultLang("FR");                       // Register the default language

// Draw and print Insert_Card message
DisplayMSG (0,0,Insert_Card, _OFF_, _dMEDIUM_, _FIXED _WIDTH_);
PrintMSG (Insert_Card, _OFF_, _pNORMAL_, _FIXED _WIDTH_);
```

TELIUM : **Manager customisation (5/5)**

ingenico

- **How to customise Manager language :**
  - Generate an EFT.MSG merging both English.h and HWCNFEnglish.h files
  - Two example of those file are given in manager folder from SDK for reference
  - Load signed file inside Terminal /Swap
  - New EFT file is automatically detected and new languages added to M2OS
  - If a message is missing in this file, Manager retrieve the default one

english.h

```
// #LANGUAGE EN
// #FONFILE /LOCAL/MANAGER.SGN
// #MSGCODING ISO8859
// #FILETYPE 1
// #APPLITYPE 2
#message 1 MESS2 "OTHERS"
#message 3 MESS4 "      LOAD\n AN APPLICATION"
#message 4 MESS5 "   INITIALIZE\n AN APPLICATION"
                        •
                        •
                        •
```

HWCNFEnglish.h

```
// #LANGUAGE EN
// #FONFILE /LOCAL/MANAGER.SGN
// #MSGCODING ISO8859
// #FILETYPE 2
// #APPLITYPE 14
#message 0 MODEM_SETUP "Modem Setup"
#message 1 MODEM_OUTPUT_LEVEL "Output Level"
#message 2 MODEM_DTMF_TIMING "DTMF"
                        •
                        •
                        •
```


ingenico

Other TDS packages

- SDK add-on :
  - Add-on CGUI : **How to use color environment (Html, CSS, JavaScript, Browser)** **New!**
  - Add-on PayPDA : **How to use IPA 280 terminal** **New!**
  - Add-on WebPos : **How to use IWP 250 cash register** **New!**
  - Add-on Morpho : **How to use EFT930M biometric terminal**
  - Add-on Telium Pass : **How to use Telium Pass device**
  - Add-on UCM : **How to use UCM/CAD device for vending machine**
  - Add-on TCPIP : **TCP/IP licence and documentation**
  - Add-on PC to Telium : **DLL for using P30 or C'less reader on PC**
- Easy Path :
  - Easy Path to EMV : **How to start an EMV application**
  - Easy Path to C'less MV : **How to use Paypass & Paywave C'less application**
- Host communication protocols :
  - ISO8583 / SPDH / APACS

TDS : Support

- For a quicker answer, try to precise:
  - On which platform you are working
  - Your SDK version
  - In case of crash, use LLT in diagnostic/maintenance mode to upload and send files:
    - APPTEXT.DIA (EFT30/SMART) or APPRESET.DIA (EFT930) (Thunder diagnostic file)
    - BOOSTER.DIA (Crypto processor diagnostic file)
    - RUNNING.LST (list of active software)
    - A sample / extract of your application / code
- For software updates, see your **confluence** account