



beyond
payment

Telium Software development rules

ICO-OPE-00156-EN-V2.05

Contents

1. Document Information	4
1.1. Evolution follow-up	4
1.2. Objective	4
2. TELIUM design rules	5
2.1. TELIUM application running principle	5
2.1.1. Application boot	5
2.1.2. Application running	5
2.2. Mandatory events to handle	7
2.2.1. GIVE_YOUR_DOMAIN	7
2.2.2. IS_STATE	7
2.2.3. STATE	7
2.2.4. IS_DELETE	8
2.2.5. IS_EVOL_PG	8
2.2.6. GIVE_YOUR_SPECIFIC_CONTEXT	8
2.3. Strongly recommended events to handle	9
2.3.1. AFTER_RESET	9
2.4. PLATFORM software requirements	9
2.5. Focusing on HIGH LEVEL APIs	10
2.6. Generic Components	10
2.7. SDK30.h File Reference	10
3. Multi Application Rules	11
3.1. Telium applications behaviour	11
3.2. Returning the handle to the TELIUM Manager	11
3.3. Key capture	11
3.4. Application name - Application type – VarId	11

3.5. Name of flash disk managed by the application	11
3.6. Name of parameters files	11
3.6.1. Names	11
3.6.2. Versions	12
3.7. Name of SSL Profiles	12
3.8. Management of HOST drive	12
3.9. Screen Management	12
3.10. Register event priority	12
3.11. Crypto processor	13
3.12. Peripheral access	13
3.13. Peripheral management	13
3.13.1. Modem and incoming call Management	13
3.13.2. APNs management for GPRS	13
3.13.3. Contactless management	13
3.14. Power management	14
3.15. RAM management - Flash disk management	14
3.16. Date and Time shall only be handle by the Manager	14

4. Communication 15

4.1. Use of the LinkLayer component	15
4.2. No direct access to the communication peripheral drivers	15
4.3. Communication peripheral sharing	15
4.4. Security requirements	15

1. Document Information

1.1. Evolution follow-up

Revision	Type of modification	Author	Date
1.00	Initial Release	JR. POLIGNY	2007/07/02
1.01	Review	R. MARCEL	2008/02/14
2.03	New Release (extended rules)	A. ROUXEL / JL FRIMOUR	2012/07/09
2.04	Precisions about APIs : 2.7. SDK30.h file reference + some corrections	A. ROUXEL	2012/09/18
2.05	Precisions about Power Management for Bluetooth [®] terminals	A. ROUXEL	2013/01/13

1.2. Objective

The purpose of this document is to define the development rules to strictly apply to develop applications that will be Telium compliant (TELIUM PLATFORM compliant, Multi-application context compliant and INGENICO group solutions compliant).

Non application of these rules may lead to:

- problems of coexistence with other applications (ex : INGENICO group generic applications)
- dysfunction of certain TELIUM PLATFORM services

Readers and developers shall be aware that compliancy with these best practices will grant easy migration to a new platform.

2. TELIUM design rules

2.1. TELIUM application running principle

2.1.1. Application boot

A Telium application starts to run when the TELIUM MANAGER asks the TELIUM SYSTEM to authenticate and to load this software. At the end of this operation the TELIUM SYSTEM calls the main entry point of the application (**`void entry(void)`**).

During this step the TELIUM application has to:

- Open its TELIUM DLL that will be used during future processing
- Record the list of the different services that are supported and treated thanks to "`ServiceRegister()`".
- Return back to its caller.

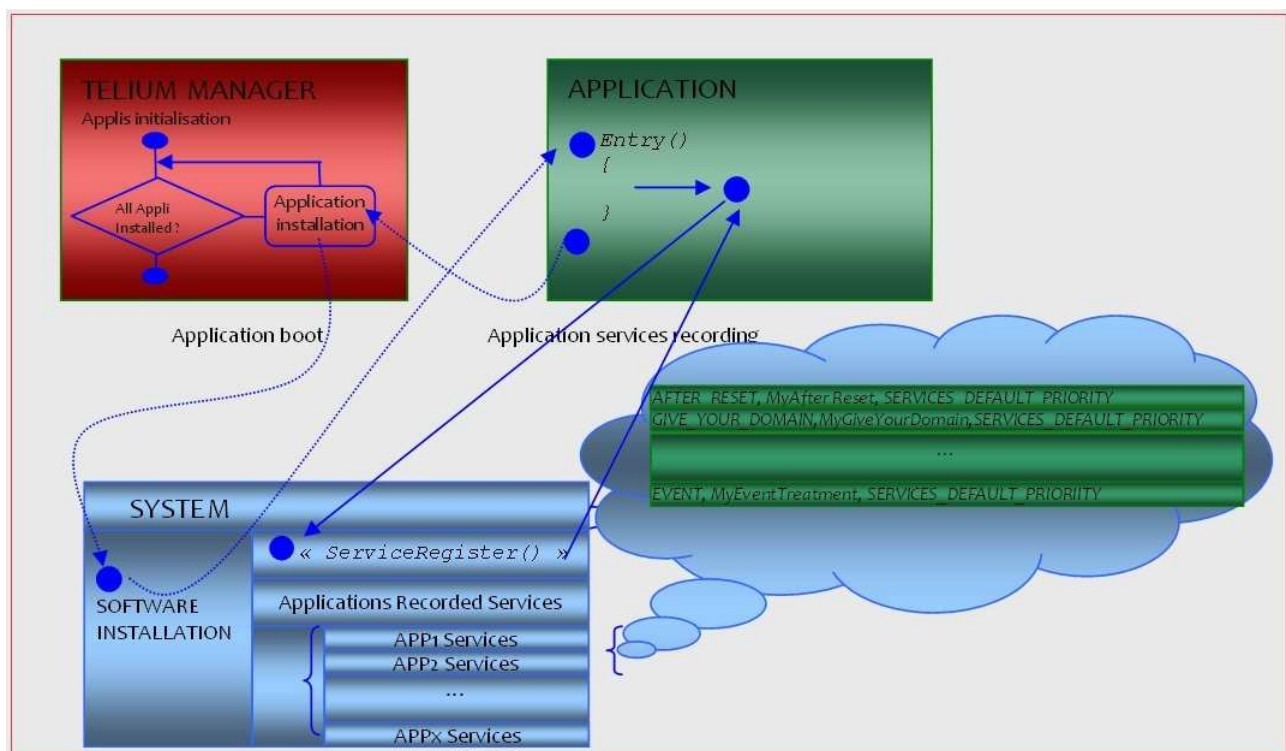


Figure 1 : Telium application boot step

No other processing must be done during "`entry()`" call, in particular:

- Access to device or high services that are not currently at this step fully installed and available.

2.1.2. Application running

A compliant TELIUM application never runs by itself: the application calls are event-driven. When the application manager (TELIUM MANAGER) is notified of an event, it performs a call-back to the application thanks to its recorded services list.

During these call-backs the application can use all available devices or services (except those restricted by security aspects).

Before returning back to the caller, the application must restore devices and services used in their initial state (set as before using).

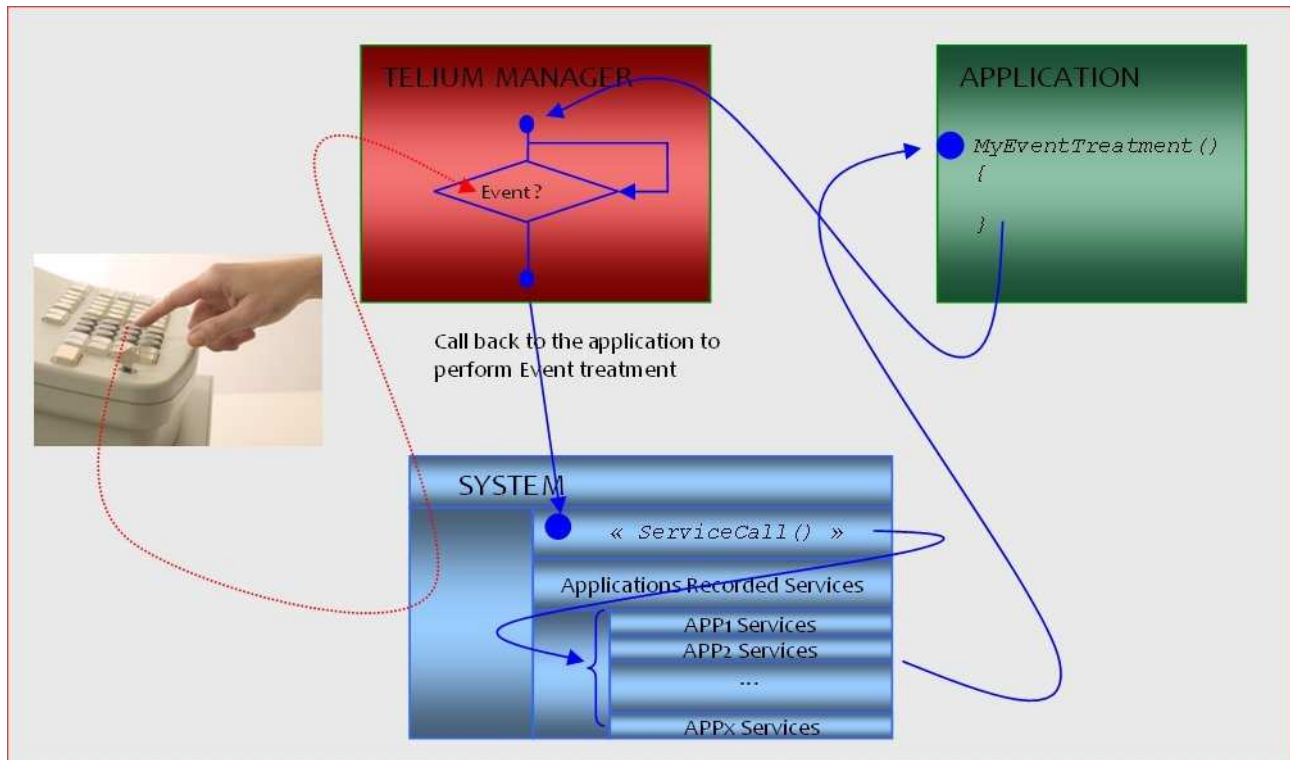


Figure 2 : Telium application running

The TELIUM PLATFORM offers to applications a “fork” capability. This feature must be used with care:

- “fork” has not to be considered as giving real time capability to application: application cannot know the sequencing of the different forked tasks in its context. Acting on “Priority” parameter will not guaranty that “forked” task with lower priority will be blocked.
- TELIUM application must only act on its created tasks in its context: “kill” ...
- “forked” treatments must be done in a way to not interfere with TELIUM MANAGER/ application calls-backs mechanism.
- “forked” treatments must not lock some devices or services (in particular Human Machine Interface).

2.2. Mandatory events to handle

To build a compliant TELIUM application, the registering and handling of some events is mandatory. This following table summarizes this list of events.

EVENTs	Description
GIVE_YOUR_DOMAIN	Provides working domain to TELIUM Manager
IS_STATE	Provides status to TM application (whether it is initialized or not)
STATE	Provides information printout to TM application
IS_DELETE	Provides deletion strategy to TM application
IS_EVOL_PG	Provides loading or downloading strategy to TM application
GIVE_YOUR_SPECIFIC_CONTEXT	Provides its compliance with advanced GUI to TM

Tableau 1 – Mandatory events list

2.2.1. GIVE_YOUR_DOMAIN

An application must register GIVE_YOUR_DOMAIN entry point based on the following prototype:
`int give_your_domain(NO_SEGMENT no, void *in, S_INITPARAMOUT *out);`

The application must return its working environment to the TELIUM Manager. According to the environment, the TELIUM MANAGER will select common parameters set and adapt its internal processing (some messages, keyboard management...).

2.2.2. IS_STATE

An application must register IS_STATE entry point based on the following prototype:
`int is_state(NO_SEGMENT no, void *in, S_ETATOUT *out);`

The application must return its state (initialized or not) to the TELIUM MANAGER.

🔔 In the case of a single “not initialized” application in the terminal, the message 'Initialize an application' will be displayed, even if no action is required by the application.

2.2.3. STATE

An application must register STATE entry point based on the following prototype:
`int state(NO_SEGMENT no, void *in, void *out);`

The application must return its software information print out to the TELIUM Manager. The TELIUM MANAGER makes this callback when it prints the global software configuration receipt of the terminal.

The application has to print:

- Its name, including version
- Its checksum
- The size of flash disk used by the application
- The size of ram used by the application

🔔 To obtain a global software configuration receipt with a homogeneous rendering, the print fields formatting must be organized as other applications and as TELIUM MANAGER tickets part.

2.2.4. IS_DELETE

An application must register IS_DELETE entry point based on the following prototype:

```
int is_delete (NO_SEGMENT no, void *in, S_DELETE *out);
```

The application must return if its deletion is authorized or not and must have destroyed all its secret stored data if needed.


So the application has the capabilities to create its own customized deletion menu in blocking the generic TELIUM software deletion end user menu.

2.2.5. IS_EVOL_PG

An application must register IS_EVOL_PG entry point based on the following prototype:

```
int is_evol_pg(NO_SEGMENT no, void *in, S_ETATOUT *out);
```


When the TELIUM MANAGER is ready to download the terminal (in local or in remote mode), it asks to each application if it's authorized to process it.

 Example: an application can refuse a software update if transactions are still stored in terminal.

2.2.6. GIVE_YOUR_SPECIFIC_CONTEXT

When the application is compliant with an advanced GUI, it must register GIVE_YOUR_SPECIFIC_CONTEXT entry point based on the following prototype:

```
int give_your_specific_context(NO_SEGMENT no, void *in, S_SPECIFIC_CONTEXT *out);
```

 Example: an application using GOAL interface must handle this EVENT and return the expected information.

2.3. Strongly recommended events to handle

To build a compliant TELIUM application, the events listed in the following table must be handled with care.

EVENTs	Description
AFTER_RESET	Allow application to process processing of actions to be carried out after reset.

Tableau 2 – Events to manage with care

2.3.1. AFTER_RESET

An application must register AFTER_RESET entry point based on the following prototype:

```
int after_reset (NO_SEGMENT no, void *in, S_TRANSOUT *out);
```

After TELIUM MNAGER has installed all the applications, it notifies all applications about completion of this step. During this call back, the application can carry out the actions required after a reset or an initial loading.

Typical treatments that can be performed are:

- Disk creation and formatting
- Ram backup management and application files coherence and recovery
- Cold or warm reset treatments.
- ...

As application notification is performed by TELIUM MANAGER in a random order:

- No services provided by applications can be called during this treatment.

🔔 To address this use case, the application must send to itself a message to notify that a treatment as to be performed.

- Example: a message to ask the application to connect to payment host.

2.4. PLATFORM software requirements

Just after start-up and before performing generic event treatments, a TELIUM application must check that its running conditions are achieved:

- Minimum SYSTEM version package present
- Minimum MANAGER version package present
- Presence and minimum version of all mandatory DLL and services application needed to run the application
- Presence of some critical data or parameters files needed to run the application
- ...

In case of failure, explicit reports of logs must be provided to final user to help solving such type of issues.

2.5. Focusing on HIGH LEVEL APIs

According to PLATFORM evolution and to backward compatibility constraints, existing functionalities are maintained until it's decided to push them in an end of life cycle. So given functionalities can be provided at different level in the PLATFORM, recent implementation being more abstracted and target focusing than historical one.

Best practices on choosing functionality APIs are based on the following rules:

- The application must use the more abstracted and High Level APIs present in PLATFORM for a given functionality.
- The “end of life” cycle entry must be anticipated: when a new functionality level of abstraction is created, its usage must be introduced as early as possible taking opportunity of correction or application evolutions.
- Direct usage of SYSTEM functions is not recommended.

2.6. Generic Components

TELIUM PLATFORM provides some generic components. A compliant TELIUM application must first use provided features of these generic components.

The main TELIUM generic components are:

- TELIUM Manager
- Services DLL provided by TELIUM MANAGER
- Link Layer – This component is designed to manage all the physical links and protocols available on TELIUM terminals
- Security DLL - Component to manage secure schemes for TELIUM applications.
⚠ Its usage is mandatory.
- GOAL (Graphic Object Advanced Library) - This component provides to TELIUM application an advanced graphical user interface. (GUI).
⚠ Its usage is mandatory when creating a new application
⚠ Migration of existing application is strongly advised if application has to be maintained for a long time.
⚠ **Old graphic interfaces (LIBC, Libgr, CGUI, Color DII) shall no longer be used.**
- Easy Path to EMV
- Easy Path to C'less (new architecture)
- Communication Components ISO 8583, SPDH, APACS

2.7. SDK30.h File Reference

In order to have all API definitions and to keep compliant, TELIUM application must include the file *SDK30.h* provided in TELIUM SDK.

As all non-documented Ingenico APIs (= non present in Telium SDK or add-ons include files) may be modified or deleted without notice, you must not use them in applications.

3. Multi-Application Rules

This chapter defines the rules to be applied on the TELIUM platform when an application will co-exist with other applications on the same terminal.

3.1. Telium applications behaviour

Any TELIUM application must not impact on the operation of the other TELIUM applications.

3.2. Returning the handle to the TELIUM Manager

When a TELIUM application has nothing to do, it must give back the handle to the TELIUM Manager. Example: All key captures must be protected by time-out.

3.3. Key capture

TELIUM-compliant application shall not capture keys in idle state. If only one application is loaded on the terminal, key capture functionality can be enabled using application menu.

Technical proposal

TELIUM application can check if other application(s) has registered KEYBOARD_EVENT Manager entry point.

3.4. Application name - Application type – VarId

The TELIUM application name and type must be compliant with range value provided by Ingenico.

3.5. Name of flash disk managed by the application

To guarantee a unique name, the name of the flash disk managed by the TELIUM application must be prefixed with the application type.

Example: application Type = 0x6F05: disk name = "6F05param"

3.6. Name of parameters files

3.6.1. Names

To guaranty a unique name, the name of parameters files managed by the application must be prefixed with the application type.

Example: application Type = 0x6F05: disk name = "6F05key.par"

This rule is applicable for parameter files used via the host drive and also for static parameters files loaded at the same time as the TELIUM application in the system drive.

3.6.2. Versions

To manage the version of the configuration file loaded in the system drive, the version and release must be coded as the last four digits of the name.

Example: 6Fo5Param0100.cnf

3.7. Name of SSL Profiles

To guarantee a unique name, the name of each profile managed by the TELIUM application must be prefixed with the application type.

Example: application Type = 0x6Fo5: SSLprofile name = "6Fo5_xxxx" where xxxx is understandable name which can be displayed or printed out.

3.8. Management of HOST drive

The HOST drive is a shared drive, and by its nature is not secure. It is also used to manage specific features like screen saver and parameters files (.PAR extension).

It is totally forbidden to use the HOST drive for downloading files and then to use the SoftwareActivate function with the HOST drive.

A TELIUM application must not use /HOST drive to store software when managing download thanks to "SoftwareActivate": the application must create a specific disk volume to store the downloaded files (in flash or in RAM).

3.9. Screen Management

TELIUM application must give back the initial state of the display to the TELIUM Manager.

Example:

- Display of the header/Footer
- Back-light management
- Leds management

3.10. Register event priority

All events registered from the TELIUM application to the TELIUM Manager must have priority from 150 to 255.

3.11. Crypto processor

Any secret area created by a TELIUM application must respect the VarId naming rules provided by Ingenico.


When TELIUM application is deleted by TELIUM Manager (IS_DELETE event), keys stored in created secret area in the crypto processor have to be suppressed too.

No secret area and no schemes shall be running when the TELIUM application gives the handle back to the manager.

3.12. Peripheral access

For each peripheral, specific primitives are provided. These primitives are related to the type of peripheral. This can include, for instance, special functions related to data capture on the keyboard or programming of the printer ...

Before using, peripherals must be opened using the fopen command, the peripherals are closed using the fclose command.

 State of peripherals has to be managed. Indeed, using fopen for an already opened peripheral will cause a reboot of the terminal. Using fclose for an already closed peripheral will cause a reboot too.

3.13. Peripheral management

When the application gives the handle back to the Telium Manager, all peripherals must be released.

3.13.1. Modem and incoming call Management

Each application will register an event to indicate to the Manager that the application can manage an incoming call. The application can also indicate to the Manager its priority.

When the event occurs, the Manager wakes up the application with the highest level priority.

Note: this function is not implemented. It is only a technical proposal.

3.13.2. APNs management for GPRS

To improve the efficiency in using GPRS, it is better not to disconnect the APN. Therefore, when a transmission is required, each application must check the status of the GPRS connect and APN last used. If the APN is correct the connection can proceed immediately, otherwise the application (payment or auxiliary) must disconnected the attached APN and attach the required APN.

3.13.3. Contactless management

The contactless interface is usually dedicated to quick transactions. So, the application must take care of performance. For example, MasterCard PayPass enforce to spend less than 100 ms in the terminal, including the time required to send the commands to the card (up to 20 ms). So, only 80 ms is dedicated to the kernel and the application processing!

Because the contactless interface is very power consuming, an application shall not indefinitely wait for a contactless card at idle state:

- On battery powered terminals, the life of the battery will be dramatically decreased.
- And most of the others terminal models will become hot. In some cases, it may reduce the lifetime of the product.

Precautions must be taken when using contactless on some terminals:

- A contactless card can be accidentally be read when the user wants to swipe or insert a card (for example when swiping the card, or if the card is moved over the screen from top to bottom to be inserted into the contact chip reader).
- The contactless interface generates a powerful electromagnetic field. It may cause phantom magnetic card reads or difficulties to read a card.

3.14. Power management

The wireless terminal turns to Power Safe mode automatically after a standby delay if the terminal is set out of the cradle.

This action is not performed if one of these peripheral are opened:

- MODEM / CLESS / MMC / PRINTER / IAPP / CAM (powered)/COMx(via Bluetooth)

3.15. RAM management - Flash disk management

With embedded systems, special attention shall be given to Ram usage And Disk size. Here are some recommendations:

- To use a minimum RAM as needed by the application
- For nominal functions, to use static RAM as allocated ram instead of umalloc() or dllmalloc(). Note that the GNU compiler supports dynamic arrays: “unsigned char Array[size];”, where ‘size’ is a variable. The GNU compiler allocates the memory on the stack.
- TLV Tree shall be considered instead of DEL (Data Element List) mechanism that requires an intensive use of RAM with very limited features.
- To Use Flash disk as needed by the application. It is not necessary to reserve more Flash than is necessary by the application.

3.16. Date and Time shall only be handle by the Manager

A TELIUM application is not allowed to change date and time parameters.

If an application has to manage these data, it shall create its own date/time based on the current system date/time.

4. Communication

4.1. Use of the LinkLayer component

Each TELIUM application shall use the services of the LinkLayer component to perform outgoing communications. The LinkLayer component guarantees the independency and the compatibility with all the Telium hardware and SDKs.

4.2. No direct access to the communication peripheral drivers

It is highly recommended not to directly access the system or to request services provides by the peripheral drivers.

For the additional GPRS services (USSD, SMS, roaming...), TELIUM application shall use services of the GPRS library and ExtraGPRS library.

4.3. Communication peripheral sharing

A TELIUM application shall not monopolize communication channel.

An application shall consider that another application can access to a communication channel. It shall configure the channel every time before accessing it.

4.4. Security requirements

Each TELIUM application (mainly the payment ones) shall respect the minimum requirements of the security approvals (Mastercard PTS and PCI Open-protocol). Please refer the online documentation of the SDK [SDK_Telium.chm].

It shall also apply the best practises described in documents: Package IP Security Guidance [ICO-PE-045-GU-EN] and the Package SSL Security Guidance [ICO-PE-046-GU-EN].