GLOBALPLATFORM®
THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

Why The Mobile Industry is Evolving Towards Security

GlobalPlatform's GPD/STIP Solution for Mobile Security

August 2007

secretariat@globalplatform.org | *www.globalplatform.org* |

Contents

I. Introduction

The mobile phone industry is evolving. Mobile phones are now holding critical information, such as end user personal information (requiring privacy) and operator and OEM information (requiring security) during processing and storage.

On the contrary, mobile phones are becoming more open and complex, utilizing high-level operating systems largely available on the market today. They offer customers the ability to download, on demand, applications with heterogeneous security levels.

Mobile phones are now holding critical end-user information requiring both privacy and security.

However, a broad range of applications will only be deployed or enhanced if mobile phones fulfill market requirements for interoperability, flexibility, reactivity and provability of the relevant security level. In this environment, contactless transport ticketing, mobile payment involving screen, keyboard and/or contactless communication, digital rights management (DRM) of high-value multimedia content and broadcast service protection with conditional access system (CAS) can all flourish.

Some of these requirements seem to contradict one another and hence the fulfilment of all of them simultaneously seems impossible. The purpose of this White Paper is to present a practical and well-tested solution to solve this paradox utilizing GPD/STIP technology specified by the GlobalPlatform consortium (www.globalplatform.org).

The following figure (Figure 1) illustrates the driving need for including security in mobile phones. The illustration shows the need for operational cost reduction and protection of operator assets, as well as the need for new value-added secure services deployment.

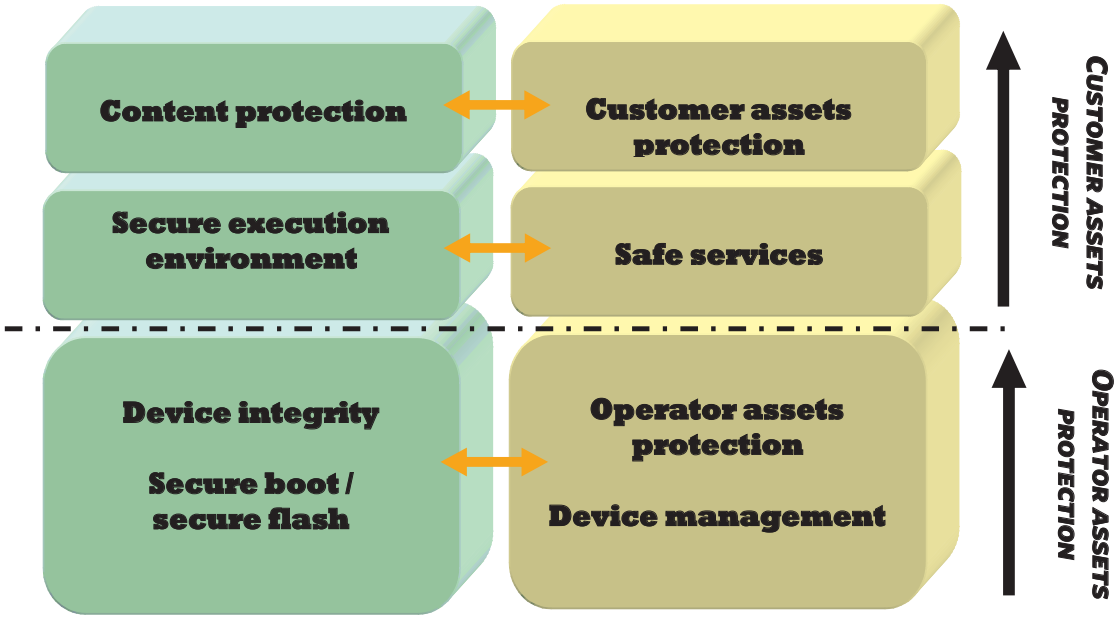

Figure 1 : Business Drives the Mobile Security Levels

II. Examples of Use Business Cases

A. Contactless Payment Use Case

In this example let us assume the mobile phone contains a contactless payment application whereby the customer can see the transaction activity (e.g. the amount) on the phone display and validate it if needed. The phone software platform is handling the communication between the different parties involved in the payment transaction (e.g. secure element, mobile, customer, server, etc.).

Here below are detailed two scenarios:

- The mobile phone is set to emulate a contactless card and it is assumed to securely conduct the exchange between contactless Point of Sale and the phone's payment application.
    - o There is a need to secure the access to the payment application, whether it is in a SIM, a removable secure element or on the phone's own memory. This requires a software/hardware platform providing a high level of security and access controls which will, however, remain open to execution and download of new applications.
    - o When the payment application is running, it must itself be controlled and protected.
    - o The downloading of the payment application and its personalization data onto the mobile phone must be done by a specific application in a highly secured way.
- The mobile phone is used as an electronic purse and it is assumed to securely store and access electronic money.
    - o There is a need to secure the access to the memory where the electronic money is stored. This requires, as in the first scenario, a software/hardware platform providing a high level of security and access controls which will, however, remain open to execution and download of new applications.
    - o When the payment application is running, it must itself be controlled and protected, especially when dealing with user payment authorization (via keyboard).
    - o The downloading of electronic money onto the phone must be done by a specific application in a highly secured way.

In the mobile environment, when the payment application is running, it must be controlled and protected.

What can be deduced from this example?

- The payment application must be highly secured at the time of download and installation. Note this can be a specific application for a given kind of payment acceptance device, so several similar applications might be downloaded on the mobile phone at different times, according to the contracts the customer accepts with service providers.
- When the payment application is running, the underlying platform must secure the relevant means of communication. If several applications are running, there must be a strict protection and isolation of all payment application components (User Interface, secure element, wallet application, contactless front-end), such that it cannot be sniffed nor tampered with by other applications.

B. DRM (Digital Rights Management) Use Case

Digital rights management is used to secure the content against any attempt to copy and reuse it by the customer or any third party.

One such example would be a customer renting a movie to be downloaded and played on a mobile phone with rights to play the movie three times within a month.

- The first phase is the purchase transaction , which includes a payment operation between the mobile phone and some provider's server.
- The second phase is the download and storage of the DRM protected content and rights.
- The third phase is the rights verification, content decryption and rendering. It requires several operations:
  - Access to the DRM file to check if there are still rights to play and update of the DRM file by decrementing the play count.
  - If the play is authorized, then the content is decrypted and sent to the video/audio subsystem for rendering to the customer.

The rights verification, content decryption and content rendering are security sensitive operations:

Digital rights management is used to secure the content against any attempt to copy and reuse it by the customer or any third party.

- Rights verification must be done in a trusted environment by authorized application.
- Decryption of the content must be done by an authorized application to maintain confidentiality during the decryption.
- The rendering must be done by an authorized application to avoid decrypted content leakage and unauthorized access by a man in the middle application.

III. The GPD/STIP Solution and its Applicability to Mobile Phones

GlobalPlatform's GPD/STIP virtual platform has been designed to facilitate the environment expressed above, and to do so at the software level. It is an interoperable and secure open virtual platform answering the related requirements that we will develop later in this White Paper.

GPD/STIP is an interoperable and secure open virtual platform.

Current mobile phones are far more than just phones. An increasing number of mobile phones embed open environments and operating systems, allowing the downloading of many new applications. Security-sensitive applications, such as payment applications, could be vulnerable in such an environment as they are difficult to protect efficiently. On the other hand, a well-designed virtual platform in terms of security and interoperability, such as GPD/STIP, allows security-sensitive applications from various providers to be managed, along with differing levels of rights. However, it is not intended and cannot protect itself from attacks from the underlying platform it is built upon.

There are two cases allowing a relevant use of the GPD/STIP virtual platform that address the business needs without compromising the basic security of the virtual platform.

- **"Separate" secure execution environment for customer oriented mobile phone**

  This approach is currently implemented, and tested, on various hardware for mobile phones from different chipset providers (Figure 2). This approach suggests having two distinct execution environments, one hosting non-sensitive applications (called the common execution environment) and one dedicated to security-sensitive operation execution (called the secure execution environment). When an application in the common execution environment requires a security-sensitive operation, it delegates its execution to the secure execution environment. This secure execution environment can autonomously execute sensitive applications on its own.

In this environment, it is as if the device would have two hardware processors, however, the GPD/STIP platform and applications reside and are executed only inside the secure processor.

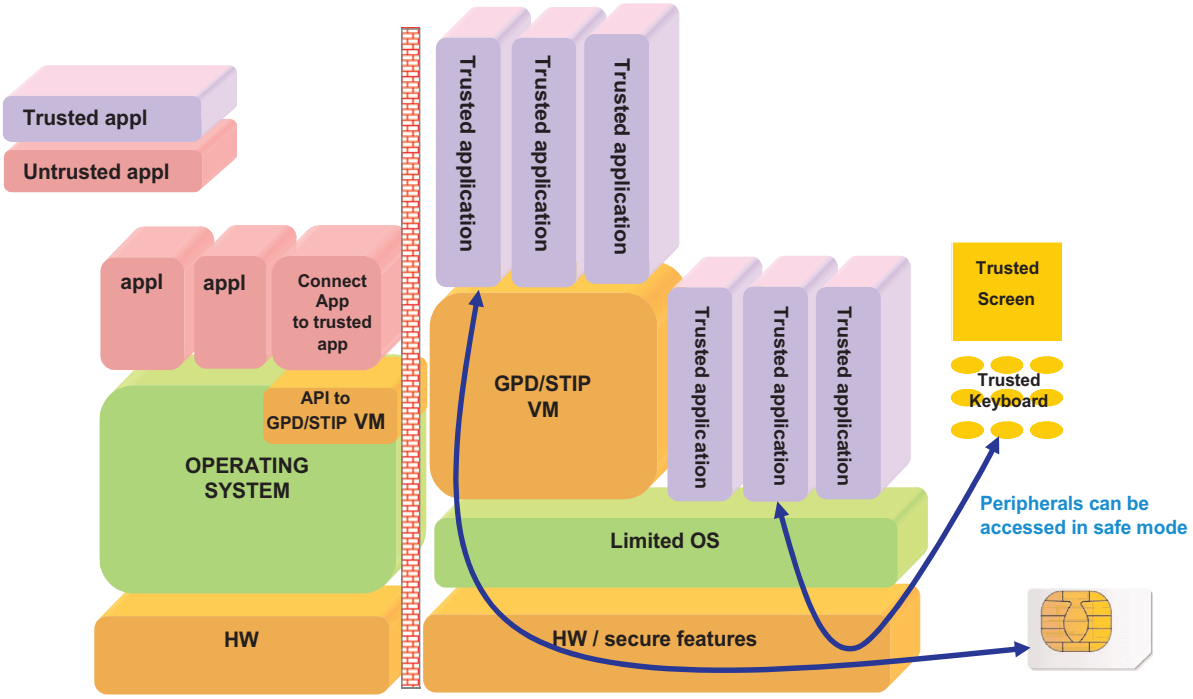

Figure 2 : Separate secure execution environment

- **"Dedicated" secure execution environment for mobile phones**

  This approach is currently being developed and tested by some device manufacturers. It suggests having mobile phones with an exclusive and secure GPD/STIP environment. Such phones are not oriented to be open for downloading any kind of unsecured applications rather they are dedicated to traditional mobile phone modem-based functions, along with payment functionality.

Each approach delivers benefits for their respective markets and makes use of the GPD/STIP specifications to solve security and interoperability related issues.

IV. The Requirements and their Paradoxes

We will now examine the precise requirements that have to be fulfilled by a virtual software platform.

Interoperability:

For network operators wishing to distribute, manage and control the quality and conformity of various applications on mobile devices and the devices themselves provided by various manufacturers, a common interoperable virtual software platform shared by all these devices is extremely compelling. The ability to write once and execute anywhere corresponds to a real business need. To satisfy this need, a comprehensive API is required to address all the various peripherals (integrated or remote) that a mobile device can use.

Security:

Providing robust security for security-oriented applications is a profound and difficult task. When faced with the need to host, and even simultaneously run, different applications with different levels of trust, security remains a fundamental requirement. What is highly desired is the ability to perform "fine grain" control of the right to access any resource of any peripherals by each application. This requirement not only impacts the execution mechanisms and API of the software platform, but also necessitates a broader view on the interactions between various actors involved in the making, downloading and use of an application, and how these can be translated in technical terms.

The required virtual software platform must satisfy the needs for interoperability, security, flexibility and reactivity

Flexibility:

From one particular device to another, the kind of peripherals available and their characteristic features may change. The complexity of this environment makes it challenging to define a common software platform with a fixed API. One solution is to limit the API to handle only the common features found on all platforms. A more flexible solution would allow different peripherals with specific characteristics to be utilized by different security-oriented applications. The challenge here is to do so without compromising the need for interoperability and security.

Reactivity:

When several input peripherals are present (smart card readers, Internet connection, keyboard, phone call), it is necessary to concurrently manage these inputs. One approach is to use explicit multithreading in the programming of the applications. However, this can be cumbersome, and may conflict with security requirements that necessitate a rigorous certification of the application behavior.

It would seem impossible to simultaneously fulfill all these requirements. As seen above, there are trade-offs between interoperability and flexibility, just as there are trade-offs between security and flexibility, and with reactivity and security.

To overcome this paradox, a global solution is required – a solution that takes into account all requirements simultaneously and is based on the long experience of security-related industries.

V. GlobalPlatform GPD/STIP Solution

The GPD/STIP solution from GlobalPlatform includes several features which are briefly detailed here below.

- **Use of a portable base language**

  The basis for application interoperability on different hardware platforms is to use a common programming language with completely defined semantics. The GPD/STIP solution relies on the use of completely typed object-oriented languages such as Java. The GPD/STIP technology has further defined a subset of the most current base API for this language in order to provide a common subset usable on all platforms, regardless of the language version implemented on each platform.

- **Notion of service control interface and service control manager**

  The central difficulty is to provide a flexible API that allows various configurations of peripherals while not compromising, and even reinforcing, security and interoperability. The solution lies in the following approach: each possible resource of a platform (peripherals, storage, etc) is considered as a service, which is implemented by a software library if present on the platform. The GPD/STIP API does not provide any direct access to these libraries. Instead, an application can only access to a standardized service control interface for each kind of service. To access and use a service, the application must request the opening of a communication channel between the service control it handles and the real service hidden behind.

  In addition, to obtain a service control object of a given type, the application must first ask a service control manager to provide a service control object of this type. The application cannot by itself create a service control object. There are advantages to this approach:

  - When a particular type of service is not present on a given platform, there is no need for this platform to implement the related service control or service library. The GPD/STIP platform only implements the service control interface declaration but not its implementation. The same API with a maximal set of service control interfaces can be implemented on each platform but at no particular expense if the service type is not present at the hardware level. Interoperability is thus possible without sacrificing flexibility.

  - Applications do not deal with specific service APIs but with standardized service control APIs. This is also important for interoperability, as libraries implementing services and service controls can be specific to each platform but the interface can be common for all platforms.

  - Security (the right to use a particular resource) and safety (that the application still behave correctly if a particular service is not present, unavailable or unauthorized) are smoothly managed by the software platform, as no resource can be accessed without two specific requests from the application. Those requests are to obtain a service control instance of a given type through a request to the service control manager, and to open through this service control a communication channel with a specific service.

    This kind of approach offers the major security advantage of having access to services and their features controlled by the software platform immediately when the request is posted, or even at the time the application is installed on the platform. This contrasts with current security methods where each resource has to call a central security manager to check the right of the caller when accessed. With the GPD/STIP approach, the real service implementations do not have to protect themselves in this way as they are already protected beforehand from any unauthorized access.

In conclusion, the GPD/STIP simultaneously satisfies the need for flexibility, security and interoperability, and does so in a way that reinforces each without creating conflict or contradiction.

- **Concurrent behavior via event-based model**

The base language API of the GPD/STIP approach does not contain any explicit thread API. The Java thread API does not provide any warranty on the underlying behavior of the multithread system, which is implementation dependent, and is not adequate if one wants to have applications verified and officially certified (often mandatory in the security-sensitive application world) independent of the particular platform on which it may run.

GPD/STIP satisfies the need for flexibility and security without creating conflict or contradiction.

The GPD/STIP approach relies on the systematic use of an event-based programming style. The underlying machinery to request events is simple, completely specified and implementation independent. This enhances the programming of highly reactive applications while reinforcing safety, verifiability, and interoperability of applications with various hardware platforms. Moreover, the GPD/STIP approach provides a specific API to mimic sequential programming to help programmers for such sequential parts of applications.

- **DASM**

The Device Application Security Management (DASM) specifications are a new extension of the GPD/STIP specifications.

The first DASM document, the Concepts and Description Documents Specification, was released in April 2007. It provides a comprehensive description of all the actors and roles involved in the creation, delivery and control of an application. It defines various kinds of messages exchanged in XML format between the actors and also defines the syntax of XML documents that must accompany the application when downloading it on the platform.

This is used to determine the Capabilities the application requires from the platform, and the Protection Domain in which it will be placed on the platform from which its Rights will be fixed. If there is a discrepancy between the Rights and Capabilities requested by the application, the application is removed. Also, the knowledge of the Protection Domain allows the platform to check at run time the right of the application to access particular resources.

Further DASM documents, the Provisioning Specification and the Key and Certificate Management Specification, will be published later in 2007. Together, the DASM specifications will provide a complete specification of the provisioning process for GPD/STIP applications on mobile devices.

VI. Appendices

A. History of GPD/STIP

The STIP Technology effort started in 2000 with the incorporation of the STIP Consortium.

STIP stands for 'Small Terminal Interoperable Platform'.

The initial intent was to specify an interoperable open software platform for all types of small terminals integrating smart card readers, supporting multiple applications and running security sensitive applications. This included vending machines, ATMs, EFT-POS payment terminals as well as GSM mobile phones or other kind of mobile devices used to run security sensitive applications.

Nevertheless, the STIP Consortium member companies were initially all related to the EFT-POS payment terminals. The idea behind this specialization was to focus on providing a way to integrate various applications with different levels of security and rights, using the same open software platform, but in a controlled environment already assumed as secure, as required by the payment device industry.

Another point of interest is that EFT-POS payment terminals may have a great variety of peripherals to take into account in the open software platform, while in practical cases many of these peripherals are not necessarily present. This required a high degree of flexibility from the interoperable open software platform to adapt to all these particular cases. Also, the presence of numerous I/O peripherals required a high degree of reactivity from the applications and hence the underlying software platform.

In 2002, the main features of the STIP technology answering all the challenges cited above were fixed and well tested. The STIP Consortium provided a comprehensive and flexible open platform specification for EFT-POS payment terminals based on a generic fundamental technology.

After this first achievement, several new member companies from the mobile phone and mobile device industry joined the STIP Consortium. Then came the idea to reuse all the well tested and validated fundamental STIP Technology used in the STIP EFT-POS payment terminal specifications for another set of specifications dedicated to mobile devices and, more specifically, mobile phones.

In fact, it appeared quickly that for the mobile phone devices, the specifications would be mainly a subset of the specifications for the EFT-POS payment terminals, plus some very specific extensions.

In 2003, a new set of specifications derived from what was then called an EFT-POS profile and a Mobile profile, were proposed.

At this time, the STIP Consortium started negotiations to transfer its IP assets to the GlobalPlatform Consortium, to establish a larger, more visible entity to bring solutions to the mobile phone industry. The combined resources of both organizations provided a stable, long term environment to host, promote and maintain the mobile profile (and also the EFT-POS profile).

This transfer was achieved in 2003, and the STIP specifications became the GPD/STIP specifications (GPD: GlobalPlatform Device).

Since this transfer of assets, a representative of the Mobile Phone industry has taken the head of the GlobalPlatform Device Committee, and several links with other consortiums such as OMA and OMTP have been established.

B. GPD/STIP API for Mobiles

Here below is the list of resource/peripheral service control Interfaces available for the GPD/STIP mobile profile as of GPD/STIP version 2.2.0

Package	Description
stip.devicecontrol.beeper	Provides access to the device's beeper service.
stip.devicecontrol.chv	Provides access to Card Holder Verification services.
stip.devicecontrol.comm	Base package providing access to device's communication services.
stip.devicecontrol.comm.serial	Provides access to the device's serial port services.
stip.devicecontrol.comm.serial.modem	Provides access to the device's modem services.
stip.devicecontrol.contactless	Provides access to contactless services.
stip.devicecontrol.contactless.initiator	Provides access to contactless devices in Initiator mode.
stip.devicecontrol.contactless.target	Provides access to contactless devices in Target mode.
stip.devicecontrol.crypto	Provides access to cryptographic services.
stip.devicecontrol.crypto.certificate	Provides access to cryptography operations related to digital certificates.
stip.devicecontrol.crypto.certificate.x509	Provides access to cryptography operations related to X509 digital certificates.
stip.devicecontrol.date	Provides access to the date service.
stip.devicecontrol.file	Provides access to the device's file service.
stip.devicecontrol.led	Provides access to LEDs on the device.
stip.devicecontrol.mastervolume	Provides access to the master volume management service.
stip.devicecontrol.media	Provides access to the media service.
stip.devicecontrol.net	Provides access to network services based on socket interfaces and the HTTP protocol.
stip.devicecontrol.net.datagramsocket	Provides access to the datagram socket service.
stip.devicecontrol.net.http	Provides access to the HTTP service.
stip.devicecontrol.net.httpserver	Provides access to the HTTP server service.
stip.devicecontrol.net.serversocket	Provides access to the server socket service.
stip.devicecontrol.net.socket	Provides access to the socket service.
stip.devicecontrol.power	Provides access to the power management service.
stip.devicecontrol.simpleui	Provides access to the device's Simple User Interface service.
stip.devicecontrol.smartcardslot	Provides access to a smart card reader slot.
stip.devicecontrol.speech	Provides access to speech recognition services.
stip.devicecontrol.timer	Provides access to the device's timer service.
stip.devicecontrol.ui	Provides a high level abstraction of the device's user interface elements, including the display, the keyboard, and the printer.
stip.devicecontrol.ui.browser	Allows the application to interface with the device's display and keyboard on an abstract level.
stip.devicecontrol.ui.printer	Allows the application to interface with the device's printer on an abstract level.
stip.devicecontrol.vibrator	Provides access to the device's vibrator service.

C. References

- The GPD/STIP Specifications, Technical overview, Test-Plan and User's guide
  http://www.globalplatform.org/specificationview.asp?id=device
- The GlobalPlatform Site
  http://www.globalplatform.org/

D. Glossary

- **API**: Application Programming Interface. Defines the library interfaces that an application can use on a given software platform
- **Core API**: the part of the API defining the application programming language capabilities (independently of other resource/peripheral API's).
- **DRM**: Digital Rights Management
- **EFT-POS Terminal**: Electronic Funds Transfer Point of Sales Terminal
- **Java**: an object oriented programming language. There are several versions of Java core API, the one defined by GPD/STIP being common to all versions (except the special version defined for Java Card)
- **GlobalPlatform**: the global leader in smart card infrastructure development and its proven, technical specifications for cards, devices and systems are known as the standard for smart card infrastructure. GlobalPlatform is a member driven association with cross-industry representation from all world continents.
- **Interoperable Software Platform**: a virtual software platform common to different devices from different manufacturers, allowing thus the same application code to run on these various devices without change.
- **OEM**: Original Equipment Manufacturer
- **OS**: Operating System
- **STIP**: Small Terminal Interoperable Platform
- **Virtual Software Platform**: the API viewed by an application, independently of the various hardware specificity of the hardware embedding platform. This term is reserved usually for API's that are independent of the underlying OS, as it is the case for Java.