PEPPOL Deliverable D1.1
Requirements for Use of Signatures in Public Procurement Processes

Part 2: E-tendering Pilot Specifications

Version 1.2

PEPPOL WP1 2009-04-30

Borderless eProcurement

Let's make it happen!

Table of contents

1 Summary and Structure of Document

1.1 Scope and Structure of Deliverable D1.1

This document is a part of the multi-part deliverable D1.1 “Requirements for Use of Signatures in the Procurement Processes” issued by the PEPPOL[1] (Pan-European Public Procurement On-Line) project. PEPPOL is a three-year (May 2008 – May 2011) large scale pilot under the CIP (Competitiveness and Innovation Programme) initiative of the European Commission.

D1.1 consists of the following documents:

Part 1: Background and Scope

Part 2: E-tendering Pilot Specifications

Part 3: Signature Policies

Part 4: Architecture and Trust Models

Part 5: XKMS v2 Interface Specification

Part 6: OASIS DSS Interface Specification

Part 7: eID and eSignature Quality Classification

The D1.1 deliverable is the first version of **functional specifications** for cross-border interoperability of e-signatures in Europe. The specifications are specifically targeted at cross-border public procurement, the topic of PEPPOL. However, if the resulting solution is successful it is believed that it will be applicable also to other application areas in need of e-signature interoperability.

Signature interoperability in PEPPOL focuses on verification of e-signatures and their associated eIDs. Interoperability of signing solutions is not handled as it is assumed that all actors are capable of signing documents within their corporate infrastructure.

The specifications guide the implementation, testing, and piloting of e-signature interoperability solutions to be done by PEPPOL. The specifications are publicly available and comments from any interested party are most welcome. Note that since the specifications of D1.1 by necessity will evolve as a result of further work in PEPPOL, any party using or referring to the specifications must ensure that the latest version is used; contact the PEPPOL project for information.

1.2 Scope of WP1 Pilots

1.2.1 Tendering Pilots

Since tendering is identified as an important area for e-signature interoperability by the EU Public Procurement Directives (see part 1 of D1.1), WP1 regards it as crucial that PEPPOL answers the requirements by running pilots on tendering. E-tendering in the time frame of PEPPOL is regarded as a largely manual process that will not use the PEPPOL transport infrastructure[2], hence tendering is not covered by other pilots apart perhaps for use of Virtual Company Dossier (VCD) and catalogues. There is however nothing that prohibits e-tendering across the PEPPOL infrastructure.

[1] http://www.peppol.eu

[2] See http://www.peppolinfrastructure.com

This document thus outlines plans for e-tendering pilots run by WP1. The details on which parts of the functional specifications of D1.1 to pilot and what to leave out of scope are yet to be defined. Considerable evolution of the content of this document must be expected as pilot scenarios are gradually developed and tested in an iterative process.

1.2.2 Post-Award e-Signature Pilots

It is highly recommended to run trials based on the specifications of D1.1 on signed orders, order confirmations, and invoices; preferably also VCDs and catalogues. However it makes no sense having WP1 set up separate pilots for this purpose. If done, this must be specified as integral parts of the pilots of PEPPOL WP2-WP5.

1.3 Structure of This Document

In the following, pilot assumptions are outlined in chapter 2. Product platforms to be used are briefly described in chapter 3 and the pilot phases are described in chapter 4. Appendix 1 provides information on generic card support for signing based in particular on input from Germany and Italy.

1.4 Version, List of Contributors

Version 1.0	2009/02/11	Complete version for internal quality assurance.
Version 1.1	2009/02/27	Submitted to PEPPOL project management, approved with comments at project management meeting 2009/03/27.
Version 1.2	2009/04/29	For publication, updated according to comments.

The following organizations, in alphabetical order, have contributed to Deliverable D1.1.

- **bremen online services, Germany,** http://www.bos-bremen.de
- **CNIPA, Italy** http://www.cnipa.it
- **DGME, French Ministry of Finance** http://www.references.modernisation.gouv.fr/
- **DNV, Norway** http://www.dnv.com

The following persons (alphabetical ordering for each participating organization) have contributed to the work:

Jörg Apitzsch	bos	Uwe Trostheide	bos	Dr. Daniele Tatti	CNIPA
Markus Ernst (co-editor)	bos	Jens Wothe	bos	Mario Terranova	CNIPA
Mark Horstmann	bos	Martine Schiavo	DGME	Anette Andresen	DNV
André Jens	bos	Stefano Arbia	CNIPA	Dr. Leif Buene	DNV
Dr. Jan Pelz	bos	Giovanni Manca	CNIPA	Jon Ølnes (editor)	DNV
Marco von der Pütten	bos	Adriano Rossi	CNIPA		

2 General Pilot Assumptions

It is assumed that all actors can sign inside their corporate infrastructure. The e-tendering platform must offer download functionality for forms and other material to be filled in, signed, and returned by an economic operator. Signing in a user interface of the e-tendering platform is optional; be aware that it is unrealistic to create such an interface in a way that supports more than a limited set of cards and other e-signature devices. Appendix 1 provides information on generic card support for signing.

Encryption is not directly handled by this scenario but it is possible for an e-tendering platform to offer an advanced “upload and encrypt” function instead of merely “upload”.

This document describes use of validation services for pilots. Interfaces to such services are described in parts 5 (XKMS) and 6 (OASIS DSS) of D1.1. It is not decided if pilots using the OASIS DSS interface will be run.

A validation service may be a technical service or a Validation Authority (VA) that takes on liability and acts as a trust anchor of its own (see D1.1 part 4). Validation services may be chained in order to serve requests that cannot be handled by the “local” validation service.

The following assumptions apply for the pilot:

- If the validation service can handle the CA in question itself, it does not have to chain the request.
- The validation service does not have to be “national”; a commercial validation service handling a broader range of CAs is another alternative.
- The validation service may be a “real authority” in the sense that it is run by an actor that takes on own liability for the results, i.e. with respect to the recipient taking on liability for results even if these come from another validation service or from a CA.
- It may also be a software solution (trusted for the technical verification) where liability resides with the individual CAs.
- The recipient is free to handle signatures and eIDs entirely on his own. Typically, “local” (national) eIDs will be handled locally while the PEPPOL solution will be used for foreign eIDs. However the trade-off of local handling versus use of a validation service is solely at the discretion of the recipient.

3 Product Platforms for Pilots

3.1 Property Rights, Licensing, Declaration of Background

To show the ability of the PEPPOL verification infrastructure two approaches are used, which are described in the following. Both approaches are based on existing software and services.

Note that use of existing product for PEPPOL pilots does not imply a transfer of property rights to software or other aspects of the products and services. A provider of software or services should declare existing components as “background” that is made available for PEPPOL subject to certain conditions.

Software components that are developed (part financed) in the PEPPOL project should be made available as free software according to the GNU GPL license[3]; however when such components are additions to existing software products, this does not imply any change of licensing conditions for the previously existing software components (the background).

3.2 Governikus Signer – Software Based Verification

The Governikus Signer is software to apply electronic signatures to any kind of document. The Governikus Signer exists as a stand-alone solution (Governikus Signer) and an integrable function library (Governikus Integrated Signer) for business process applications. Both include a verification component that allows verification of signatures and signature certificates.

The Governikus Signer is a stand-alone solution to apply electronic signatures to documents and to verify signed documents and certificates. The user is supported by a graphical-user-interface. The verification results are summarized in a validation protocol. The Governikus Signer is a Java application, i.e. common operating systems such as MS Windows and SUSE Linux are supported.

The Governikus Signer can be enabled to be called with all functions directly from a third application. That means a functional extension for business applications without complex changes. The integration is realised via various interfaces. All functions (sign, verify, encrypt, decrypt) can be accessed either locally (via JAVA API, command line) or by means of a dynamic web call (with JNLP/JSP). A SOAP based interface is provided, too. All functions are requested by pre-defined parameters, so that the existing business process application does not need to be adapted

All kinds of document formats (Word, Excel, PDF, XML data,...) can be signed directly and this way be protected from manipulation.

The Governikus Signer can sign and verify single documents, several documents or the content of complete directories.

The Governikus Signer is realised by bremen online services GmbH & Co. KG. It is not planned to implement special features for the PEPPOL project. The Governikus Signer shall be connected to the PEPPOL Infrastructure via a configured XKMS responder.

[3] http://www.gnu.org/licenses/licenses.html#GPL

3.3 Validation Authority – Service Based Verification

DNV has been running a Validation Authority (VA – see[4] http://va.dnv.com) service for e-signatures and eIDs. It has been decided to discontinue the VA as a DNV service and transfer the service to BBS (http://www.bbs-nordic.com) as the responsible party. The service approach is still of interest to PEPPOL; however, provision of VA services to the PEPPOL pilots is at present not decided.

The VA service allows existing locally issued credentials to be used on the international stage and offers to:

- process certificates issued by Certificate Authorities (CAs) that its customers ask it to work with, in principle any CA,
- assess the quality and trustworthiness of the issuing CA and its certificates,
- verify the signatures of various formats on the transaction or document,
- assess the quality of the signatures.

Another feature of the DNV VA is an objective quality classification of CAs enabling Relying Parties (RP) to decide what level of trust to put in eIDs issued by a particular CA. This quality classification system is extended and built upon by PEPPOL, see part 7 of D1.1.

The role of the VA is illustrated in the figure below.

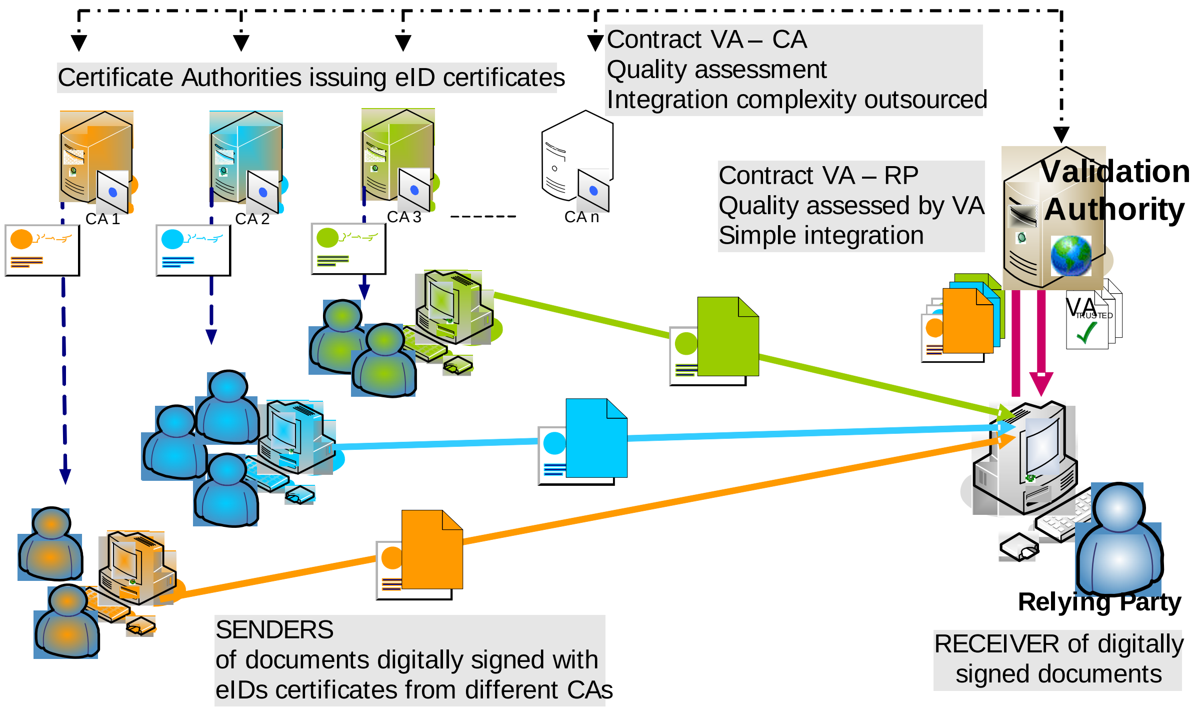
Certificate Authorities issuing eID certificates
Contract VA – CA
Quality assessment
Integration complexity outsourced
CA 1
CA 2
CA 3
CA n
Contract VA – RP
Quality assessed by VA
Simple integration
Validation Authority
VA
TRUSTED
Relying Party
RECEIVER of digitally signed documents
SENDERS
of documents digitally signed with eIDs certificates from different CAs

Figure 1: Overview of the DNV Validation Authority Service

The VA service is based on agreements: One agreement between the VA and the RP (the customer), and agreements between the VA and the individual CAs. This transforms the legal situation from national law (of the different CAs) to contractual law. The complexity of validating certificates and

[4] This link is likely to be discontinued before summer 2009.

signatures from several different CAs is not removed but rather outsourced to the VA, and only one agreement is needed with each CA. The VA takes on a uniform liability for signatures and eIDs of a certain quality, regulated by the agreement between the RP and the VA.

The VA service is based on Web Services using defined XML structures and the SOAP protocol. The protocol used by the VA service is currently a non-standardized protocol made available at http://va.dnv.com For PEPPOL, the service interface shall be changed to comply with the OASIS DSS and XKMS specifications of parts 5 and 6 of D1.1, and the quality classification system shall be changed to comply with the specifications in D1.1 part 7.

A possible scenario for provision of VA services to PEPPOL is that Difi (the Norwegian Agency for Public Management and eGovernment, project co-ordinator for PEPPOL) launches a call for tender for VA services using D1.1 as basis for the requirements specifications. The actor that is awarded the contract will then offer VA services to eGovernment services in Norway as well as to the PEPPOL pilots. Note that no formal decision has been taken with respect to such a call for tender.

3.4 PEPPOL XKMS Responder

For the pilot it is necessary to implement a PEPPOL XKMS responder. This infrastructure component has following features:

- The PEPPOL XKMS Responder can validate certificates against configured CAs.
- It can use the PPRS (see below) to pass XKMS requests towards other PEPPOL XKMS Responders. Therefore it is crucial that the component can handle XKMS v2 requests and responses.

3.5 TSL Distribution Service, PEPPOL Public Registry Service

As stated by the Commission’s action plan on e-signatures and e-identification (see D1.1 part 1), and specified by the expert group advising actions related to the EU Service Directive, information on TSPs (Trusted Service Providers) and their services shall be distributed in the form of Trust-service Status Lists (ETSI TS 102 231).

Within the time frame of the PEPPOL pilots, there is a risk that TSL distribution services will not be available, meaning that PEPPOL may have to implement a preliminary TSL distribution service. TSLs will firstly be published in a human readable format. Later TSLs will also be published in a machine processable format in XML (possibly also ASN.1) format. In both phases the authenticity of the list will be assured by means of an electronic signature from the TSL issuer. Public keys of TSL issuers will be published on PEPPOL web sites.

The TSL specifications are primarily targeted at distribution of information about CAs, while PEPPOL additionally needs information about validation services. For chaining of validation requests, the originating validation service must know which other validation service to call, i.e. a validation service that handles the particular CA in question.

It is not recommended to include a list of CAs handled in the VA entry in the TSL, Rather the TSL entry should include information about protocol and access point, and a link where the VA publishes information about the CAs covered.

Alternatively, information on CA coverage may be maintained in a PEPPOL Public Registry Service (PPRS); a VA (or even a relying party directly) may use the PPRS to find a VA covering a specific CA. The PPRS may also store all information needed for TSL issuing, and TSL issuing may be linked to the PPRS. This is further described in D1.1 part 4.

4 Phases of the Pilot

4.1 Lab-Test of Verification, XKMS

The following figure describes the flow related to signatures for a tendering pilot with verification at receiving side:

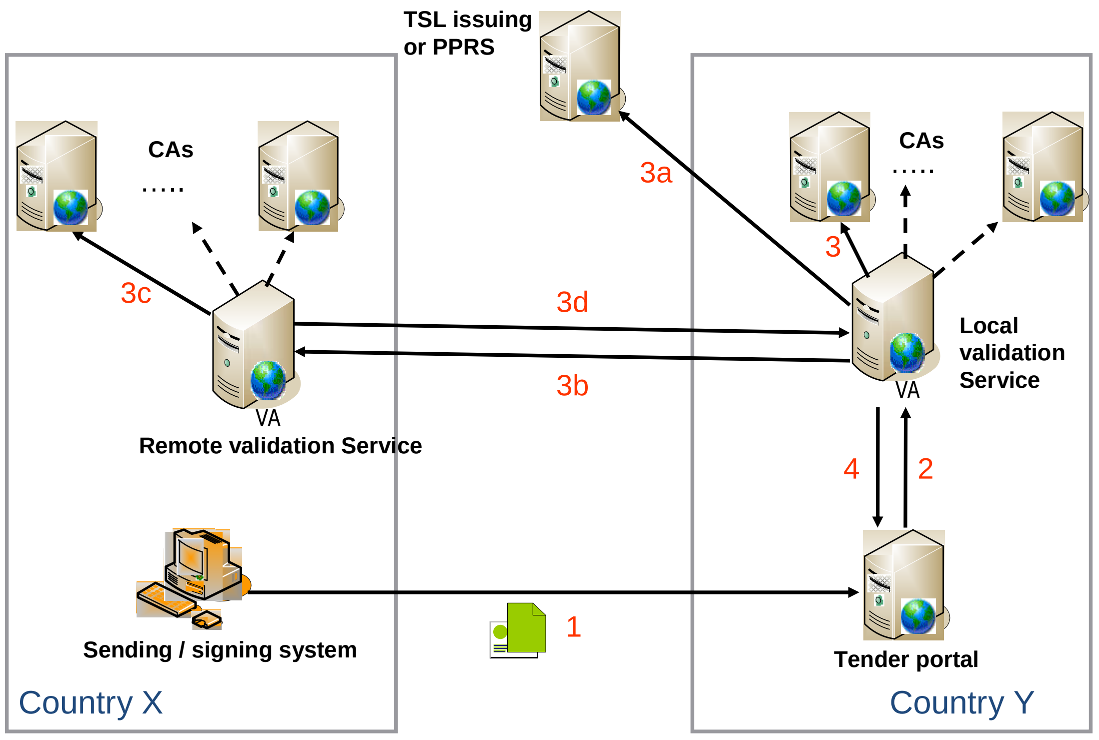

Figure 2: Overview of verification process.

Lab tests will in the first run be done by bos (bremen online service) to test the XKMS interface (D1.1 part 5). There are two test cases:

1. Using recipient's XKMS responder accessing sender's CA. Steps 3a-3d listed in Figure 2 are not needed in this case as the local XKMS responder handles all requests locally.
2. Recipients XKMS responder mediates request to an XKMS responder able to handle the request (presumably close to the sender). Recipient's responder re-signs result to establish trust.

The first version of this pilot (XKMS only) is a lab-test. The validation service shall be fed by a human readable TSL used to configure responders.

The process as illustrated in Figure 2 consists of the following steps:

1. Sender applies signature(s); https upload of signed documents to tender portal;

2. Validation service trusted by recipient is used to validate certificate(s) by an XKMS interface (alternatively OASIS DSS for the entire, signed document);
3. If the CA that has issued a certificate is known to the local validation service, the validation service checks certificate validity against this CA, else the following steps are performed:
   a) Information obtained from TSL(s) (or the PPRS) is consulted to discover a validation service that handles the CA;
   b) XKMS request is forwarded to remote validation service;
   c) Remote validation service checks certificate validity against CA;
   d) Remote validation service returns XKMS response to local validation service.
4. Local validation service forms the final XKMS (or OASIS DSS) response, re-signs and sends the validation response to the recipient.

In Figure 2, validation services are shown as “national”. This is just one possible scenario that may be relevant e.g. if validation services are run by national agencies. In general, the word “country” in Figure 2 can be substituted by “domain”, and validation services should be allowed to cover a range of CAs according to customer and business requirements.

4.2 Beta Testing, Scale Up Testing, Roll Out, XKMS

Beta testing must rely on close co-operation with PEPPOL WP8 providing the PEPPOL infrastructure. It is suggested to test in two pilot member states, using a selection of CAs (e.g. one per country).

Documents will be “mock up” tenders and the roles of economic operator and awarding authority will be played by PEPPOL personnel. Tests will use development/test/staging environments of real e-tendering platforms. An eProcurement system supplier should be selected *at latest by summer 2009* for integration of interface in the platform. When candidate member states are identified, the suppliers of the systems in use in those countries should be selected.

In a system scale up phase, more member states should be added as well as more CAs. However the main items added are:

- Scaling up trust relationships (VA instances),
- Using machine readable TSLs for routing validation requests,
- Establishing a PEPPOL Public Registry Service (if deemed appropriate).

Finally, the solution is rolled out by integration in the production versions of the e-tendering platforms, involving eProcurement software manufacturers and real contracting authorities to be involved. Then, real tenders and real economic operators shall be brought into play.

Software hardening is still possible in this phase.

4.3 Implementation and Testing of OASIS DSS

Implementation and testing of an OASIS DSS interface (D1.1. part 6) is for further study since there is at present no actor in the PEPPOL consortium that plans to provide this interface. If Difi (Norway) runs a call for tender for VA services (see 3.3 above), this call could include a request for provision of an OASIS DSS interface.

Note that chaining of validation requests (test case 2 in 4.1) will be equal for DSS and XKMS. OASIS DSS requests will not be chained as the OASIS DSS responder will take care of signature verification itself. Request chaining will be used only for validation of eIDs from “unknown” CAs, and for this purpose the OASIS DSS responder will use XKMS.

4.4 Interactive Demonstrator to DNV VA

An interactive web-interface to the DNV VA should be provided to demonstrate interoperability. The program code for the interface exists but the setup is not in place. Given the state of the VA services this demonstrator cannot be committed but it would add value.

At present in the order of 45 CAs are configured in DNV’s staging system, and documents signed using any of these CAs can be validated. A number of different signature formats are supported, as well as multiple signatures on one document.

5 Figures

6 Appendix 1: Card Support

6.1 Generic Card API

Creating a Generic Card API to support European procurement processes means to meet many challenges. In chapter 6.2 proven and new standards in context with smart cards and smart card readers will be introduced. Subsequently, the relevant aspects and demands will be discussed. To conclude the topic in chapter 6.3, suggestions are made for development of a new software component that meets as many demands as possible.

It has to be decided if such work shall be carried out in PEPPOL or if a liaison with the STORK pilot project[5] shall be done on the topic of signing and card support.

6.2 Standards and Specifications

6.2.1 Operating Systems

Deciding to build a software module to support different smart cards and smart card readers means being able to support all common operating systems as well as current versions of Windows (Windows 2000 et seq.), the most important linux distributions (SuSE,RedHat) and Mac OS X. This way as many users as possible will be enabled to employ the component without extensive adaptations of their IT environment. That is why platform independent software components will be recommended.

6.2.2 Smart card readers

Protocols

Different protocols are employed for smart card and card reader communication. The most important asynchronous protocols, such as T=0, T=1, as well as synchronous protocols Sx (x=8,9, etc.) have to be supported. This is due to the fact that use and circulation of the protocols follows the specific rules of the federal states. In Germany, for example, the standard is T=1, while France uses T=0.

Drivers

There are various drivers to control smart card readers, that all have different options to support the handling of different smart cards and/or processes. The most important varieties are PC/SC 1.0 (final), PC/SC 2.0 (draft), CT-API with MKT (final) and SICCT (not yet final). PC/SC drivers only support T=0 and T=1 and version 1.0 does not deliver an opportunity to use pin pad and display of a card reader. The disadvantage of version 2.0 mainly is the draft state. Nevertheless PC/SC is widely spread, because it is supported by a large variety of operating systems and it is a common base for further driver standards that serve developers to provide specific extensions beyond the PC/SC standard. Even PKCS#11 card modules often use this interface to control card readers.

The amount of messages is limited, because with PC/SC 2.0 it can only be chosen from a defined set of messages for different languages. Most card readers are locally plugged in via USB or a serial bus.

In the German banking sector and in public health the employed standard is CT-API (e.g. interfaces for C++) in combination with the MKT standard exclusively. The MKT standard defines the contents for transport while CT-API delivers a simple communication to the card reader via one port.

[5] http://www.eid-stork.eu

Synchronous as well as asynchronous protocols can be handled with CT-API and MKT specifications. The use of pin pad and display is possible, whereas standard messages (in most cases in only one language) and even freely definable messages can be chosen. The character set is limited to ASCII.

The only disadvantage of CT-API is that the connection to smart card and reader is put up for each single use. But this problem can be solved technically as well as the message alert in context with inserting and removing smart cards in and from card readers. Concerning events, the interface acts passively, i.e. it does not send alerts automatically in case of changes. That is why the software component has to be able to poll the state of the card reader and then generate suitable events.

In the German banking sector, only class 3 card readers (with pin pad and display) are authorized for payment processes using the money card. In a network, there are only two ways to run a card reader with CT-API. The first one was developed by Sagem Monetel and employs a special, network-compatible c-library. The second option was built by Cherry and uses a port-emulation.

At the moment, SICCT is specified as a new standard that is to be employed in the context of the new German Health Care Card. With difference to other driver standards, card reader companies do not offer a SICCT driver up to now, i.e. every software manufacturer has to develop an own driver for his components/applications. SICCT card readers are mainly connected via network – a fact that makes its employment with terminal servers or Citrix systems much easier. All other standards largely use local buses. SICCT is basically a further stage of the MKT standard and supports all protocols, display and pin pad. Moreover, functions can be added that serve to interrogate card reader properties, e.g. concerning pin pad and display. SICCT offers the same possibilities as CT-API for messages and the character set of messages can be additionally specified. Besides the opportunity to poll the state of the card and card reader, it also defines an active software message notification by the card reader.

Standard	Protocol	Pin pad/ display	messages	port	connection	employment
PC/SC 1.0	T=0, T=1	-	-	-	local	general
PC/SC 2.0	T=0, T=1	X	Fix via ID, in different languages	-	local	general
CT-API/MKT	T=0, T=1, S=8, S=9, S=12	X	Some standard messages / others variable, ASCII	port	local/LAN	Health Care Banking
SICCT	T=0, T=1, S=8, S=9, S=12	X	Some standard messages / others variable, different character sets possible	IP	local/LAN	Health Care

Figure 3: Overview of card reader drivers

6.2.3 Card

Protocols

In Germany, for the digital signature normally smart cards with the asynchronous protocol T=1 are used. Sometimes also T=0 is employed. The Health insurance card is a plain memory card with a synchronous protocol.

Communication

Communication with memory cards is regulated by the standard ISO14443. ISO 7816 defines different APDU (Application Protocol Data Units), more or less a set of orders that is commonly supported by all cards. While the orders to read and write are almost completely corresponding, amount, form and content of orders for cryptographic operations are varying. This makes a generic support of different cards in different applications hard and conflicts with the aim of a widespread use of smart cards.

The largest differences concern pin management. The commands for pin management as verify, change or unblock are specified by ISO 7816. The possibilities and commands for requesting the current value of the Reset Retry Counter are extremely different according to the various card operating systems. Cards and card readers offer different options to handle pin entering. In view of usability aspects, these options should be applied. The amount of cancelled cards can also be increased this way.

The specifications CWA 14890-1/CWA-14890-2 describe the e-sign application processes of different cards. As this is a border crossing specification, it is a good base for the employment of smart cards with secure messaging.

COS

Similar to Italy, Germany has a wide range of smart cards with different operating systems and different versions, such as TCOS 2.0r3/3.0, MICARDO 2.0/3.0/3.1, StarCOS 2.2/2.3/2.4/3.0/3.1, GPK, ACOS, Abacos, SECCOS 5.0 or as Java™ cards. The biggest differences concern pin management, card activation and directory selection. The execution of cryptographic functions also varies.

Smart Card Data

Essentially, smart cards contain directories/application, pins, files (data, data sets, certificates, CVC) keys and access rules – varying on the different cards. While the processes for access rules are mainly identical on all cards, the contents always depend on the cards' operating systems. Up to now, no general solution exists.

Information

There are different standards for information concerning the smart card structure. At the moment, they are all more or less supported, e.g. PKCS#15, EF.DIR, DF.CIA, ISO 7816-15, EF.SSD and ISO 24227 (CIF files). Most widely spread is PKCS#15 and on German cards template structures following EF.SSD are to be found. Except the CIF files defined in ISO 24227, the information is always located on the card. The documentation of CIF files (scheme file) does not suffice at the moment. The use and value of the containing information to support smart cards generically cannot finally be assessed. The internal structure, the particularly missing data sorting and lack of content references, which is possible with XML, make the use of CIF files even harder. The way a card deals with errors and different APDU sequences, especially in the context of secure messaging, at the expense of performance. The idea to use APDU sequences to identify smart cards had to be rejected, because it does not allow a high-performance smart card recognition, as pieces of information cannot be sorted into a decision tree.

standard	**Diffusion rate**	**on smart card**	**content**	**EFs**	**code**
PKCS#15	high	X	directories, files, key, pins	one DF per card with several files	ASN.1
EF.DIR	high	X	directories	exactly 1 per DF	ASN.1
DF.CIA	low	X	directories, files	one DF per card	ASN.1
ISO 7816-15	low	X	unknown	unknown	ASN.1
EF.SSD	low	X	Templates for cryptographic operations (APDU sequences)	exactly 1 per card	ASN.1
ISO 24727 (CIF)	very low	-	processes for card recognition, directories, files, key, pins access rules, protocols (references to APDU sequences) for cryptographic operations	one node per card in the CardInfo file (incl. redundancies)	XML

Figure 4: Overview of card information structures

Application

The following standards all were and are aiming at a widespread employment of smart cards in software application using different operating systems – that is independent of its application purpose.

PKCS#11/CSP

The two common standards for signature cards are PKCS#11 and CSP modules. PKCS#11 is platform independent while CSP (and the CAPI interface) is Windows™ specific. As for the German case, other operating systems rarely employ PKCS#11-Module for German smart cards. Both standards base on SOA interfaces.

The support rate of pin pad and display depends on the implementation of PKCS#11 and CSP modules. Messages cannot be influenced by the application, because the interface does not offer sufficient opportunities. In this context, not only the selection of the content, but also of the language is possible. Besides, only two pins are supported, and this will become a problem with newer cards.

ISO 24727 (Service Layer/IFDLayer)

The new ISO 24727 standard defines a SOA with several layers in order to control cards and card readers generically. This way it integrates a configuration of many pieces of information on the card. Nevertheless, messages, language and times cannot be influenced – a fact that conflicts with the employment within European contexts. The referencing of protocols for the keys of a card also is not sufficient, so that the implementations of newer cards with newer processes (APDU sequences) have to be extended and maybe (expensive) certified. That is the reason why it can be regarded more as a theoretic standard, of which the distribution has to be doubted at the present moment.

6.3 Aspects of a Solution

6.3.1 Participants/Organisations

The following people and organisations take part in the development of the solution:

- People who specify the smart card reader standards

- card reader manufacturers
- people who specify the card standards
- chip manufacturers
- manufacturers of smart card operation systems
- smart card manufacturers
- TrustCenters/Certification Authorities
- card publishers
- software development companies
- institutions (legal or commercial)
- users (very important).

The list is heterogenic and varies due to the country and makes high demands to the organisation and development of the solution.

6.3.2 Compatibility

Operating Systems and Smart Card Readers

The use of card readers with different operating systems generally makes a driver installation necessary. Depending on the operating system, the levels of difficulty for the installation vary.

Card Readers and Smart Card

According to practical experiences, the compatibility of card readers and smart card cannot always be guaranteed, although only evaluated cards and readers are employed. In some cases, the protocol and/or the timing between card and reader cannot be defined correctly. If this is the case, the card data e.g. cannot be read or cryptographic operations do not take place. In most cases a driver and sometimes a firmware update becomes necessary.

Cards and Applications

The basic demand is that each card of a user can be employed for an application, no matter which card reader and TrustCenter is embedded or what Member State he lives in.

6.3.3 Demands

The substantial demand to the solution is an easy handling for all participants.

Software Accessibility

As the software is to be employed in different countries, the language must be selectable for every single signing process. Besides, it would be of advantage if the timeouts for pin entry will also be selectable for every process. Basic precondition for appropriate user guidance is a high quality of message alerts to the user, either on the screen or on the card reader's display (if a reader with display is used). Laymen should be able to use and understand the software easily.

Hardware

Smart cards and card readers should be affordable and widespread and its employment should not be limited to certain card and reader standards. Furthermore, it should be possible to use different cards and card readers simultaneously and the employed hardware should support asynchronous and synchronous protocols (T=0, T=1 and Sx).

Software API

A software API for cards and card readers must be based on software that is easily installable and usable. The API should be equipped with a simple, structured interface that meets all demands. Besides, multiple signatures and different document standards such as XML and PDF are to be supported. Diverse cryptographic algorithms like hashing, signing, de- and encrypting, authenticating and padding must be possible. To avoid problems and competing access to cards and readers, there should only be one API implementation on each computer. That also means that the API has to support as many card reader standards, card readers and smart cards via a generic interface as possible.

Acceptance

To push forward a solution, the acceptance of as many participants as possible is needed. The user wants an application that is easy to understand and to be handled. Participating companies focus on the expenses spent for development and certification of the components. So, the solution has to be as easy as possible and as complicated as necessary. The same goes to the amount and complexity of the considered standards.

6.3.4 Conclusion

Currently, there is no standard that covers all demands for a coherent user guidance and general user acceptance. PKCS#11 is not suitable, because the solution has to be language independent. But the software should be able to be employed according to CSP or PKCS#11. That means that the internal object model has to concentrate basically on PKCS #11 and, even more, on smart card objects.

The complexity of a general card support solution with many demands and standards implies a European way that is not part of PEPPOL.

A general idea, a simple guess, about a solution for further thinking can be found in 6.5 below.

6.4 Italian Card Support Layer

6.4.1 Purpose

Although all the smart cards used for authentication or signature are declared compliant to the ISO/IEC 7816 family of standards, there are differences in the operating systems and in the internal data structures (file systems) that cannot assure the correct functionalities.

The problems to be solved were: how can I use everywhere my smart card to sign a document? How can I access on line services from a generic terminal?

In Italy, starting from 2002, two possible different solutions have been selected to solve the problem:

- homogenous implementation and interpretation of the card operating systems, its organization and internal data structures;
- specific recognition of the card by the application that uses the smart card for the specific purpose.

The first solution is cleaner. All the smart cards are totally compliant to ISO standards. The implementation of security objects and position of data into the smart card are the same for all the manufacturers.

This solution was discussed in 2002 with all the main manufacturers with the purpose of supporting different smart cards for the access (using SSL/TLS) of e-government services.

In 2003 all ten manufacturers subscribed a MoU (Memorandum of Understanding) to assure the implementation of operating systems at APDU (Application Protocol Data Unit – the specific binary commands of the smart card operating system) level; this compliant to a functional specification annexed to the MoU. This specification provides the standardization of the APDUs giving a shared interpretation of the ISO/IEC 7816 family of standards. In the MoU was shared also the internal data structure of the smart card or as commonly described in Italian official documents – the file system.

Today only five manufacturers assure the compliance with this MoU but these mechanisms are used by about 18 million smart cards for authentication and about 2.5 million for digital signature (the signature ruled by the 5.1 article of the EU E-signature Directive).

On the other hand, the support to the above described MoU is not enough to implement digital signatures. This is due to the particular requirements of smart cards as secure signature creation devices (SSCD). The security of these smart cards is assured by the Common Criteria certification. Unfortunately, every card manufacturer implements a different file system. For this reason it is not possible to use a unique library for digital signature.

Starting from this problem, a Unified Management Software (Unique Layer) was developed in Italy to solve the problem at the application level.

The availability of a common set of APDUs has sometimes simplified the implementation of the software but this specific aspect is not critical for the characterization of the solution.

This software works now (august 2008) with about ten different smart cards of five different manufacturers and the development model allows support for new operating systems. The next section gives a detailed architectural description of this Unique Layer. Because of the diffusion of Windows operating systems, the development of the software must manage two different philosophies. However this aspect does not influence significantly the architecture of the software development.

6.4.2 Unique Layer

As described in the previous paragraph, this solution for interoperability doesn’t put constraints to the operating system of the smart cards.

The interoperability is assured by software layers as showed in Figure 5.

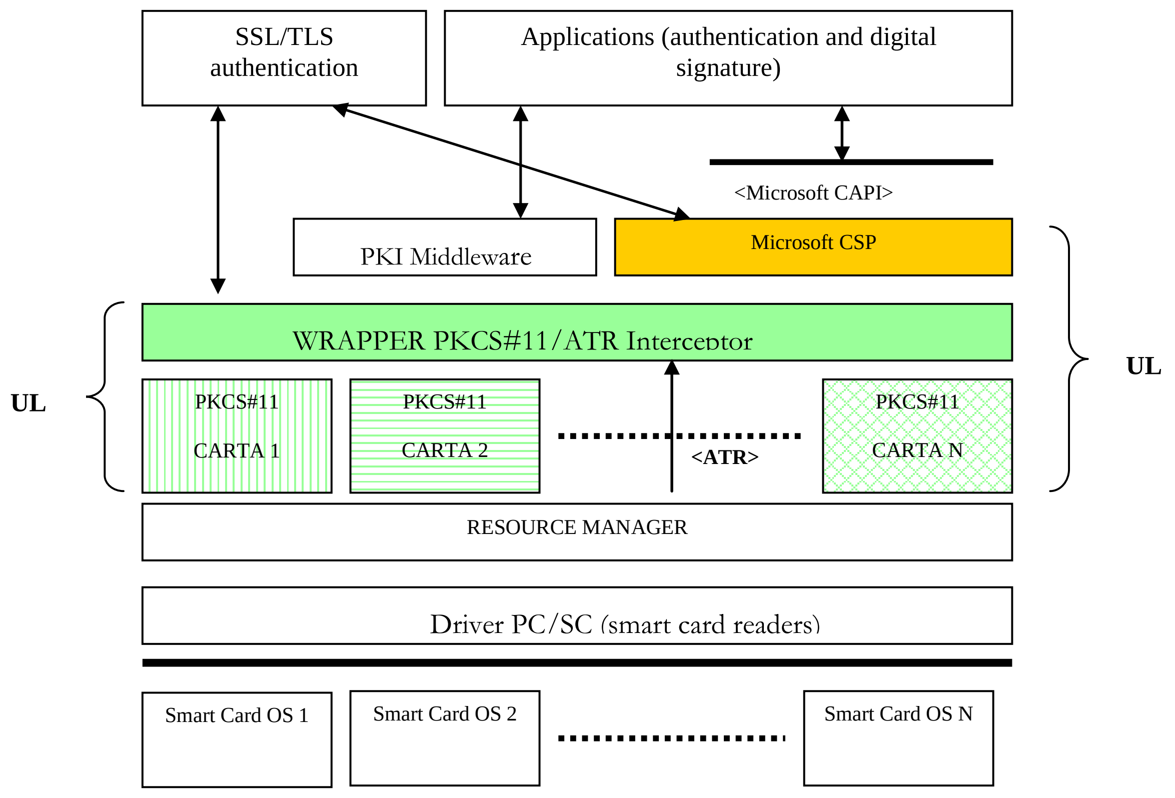

Figure 5: Italian Card API – software layers

This approach allows the use of smart cards produced by different manufacturers without possible constraints due to the specific APDU implementation. This architecture is extremely useful to manage the digital signature applications. In fact the different file systems can be managed with the dedicated library provided by the manufacturer.

In this scenario every manufacturer must provide with the smart card a cryptographic library conformant to the PKCS#11 public specifications. Obviously this software library will support the specific implementation of the APDUs and the particular data structures for authentication and signature. This software library must be integrated in the Unified Management Software (Unique Layer).

In Italy the file system for authentication (SSL/TLS) is standardized and it is theoretically possible to use the same unique library for different smart cards; however these libraries must be integrated in the Unique Layer.

When the smart card is inserted in the reader the Resource Manager of Windows (or an equivalent agent in some other operating system) makes available the information of answer to reset (ATR).

This information is a string structured in a standard format. This allows the recognition of the right smart card and the activation of the card specific PKCS#11 library. This library must be “installed” and “configured” in the Unique Layer.

The start of the specific library can be made by a particular tool named Wrapper. The wrapper also ensures a homogeneous presentation of the software calls offered by every different PKCS#11 implementation.

After this operation it will be possible to activate the authentication processes that use SSL/TLS v3 with browsers that can use the PKCS#11 software. The same holds for the software that implements digital signatures.

To allow use of browsers like Internet Explorer it is necessary to have a different implementation layer specific for the Microsoft environment. This implementation must use the Crypto Service Provider (CSP). By use of this layer it is possible to develop applications that use the Cryptographic API of Microsoft (C-API) for authentication using SSL/TLS or for digital signature.

Generally the smart card manufacturers provide a CSP library along with the PKCS#11 library. Note that the CSP libraries must be used also with Outlook and Outlook Express and in general with all applications that run in a Microsoft environment.

The last release of Unique Layer uses the CSP library directly without use of a card specific PKCS#11 library (see Figure 6). Here it is possible to map CSP calls with the equivalent calls of the PKCS#11 specification.

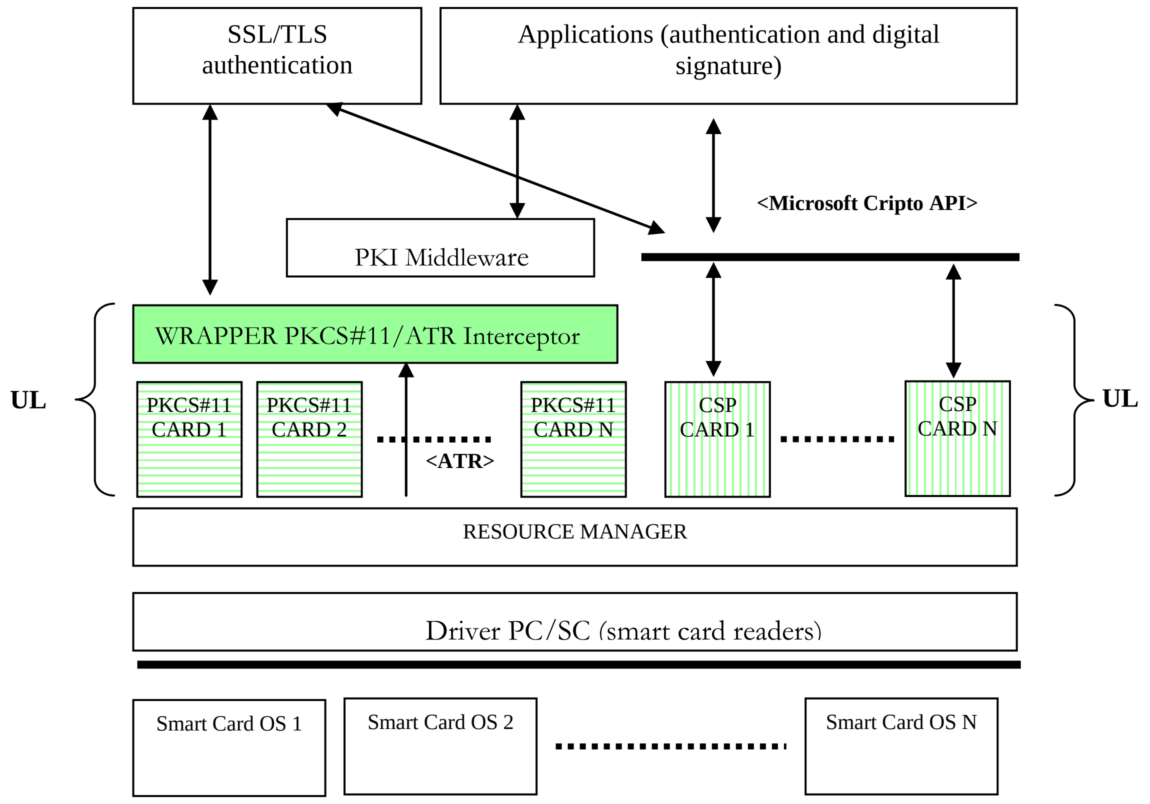

Figure 6: Unique Layer with CSP library

In this implementation the software installed on the terminal (PC) directly uses the ICC (Integrated Circuit Card) Resource Manager to interface the smart card operating system.

Practically, as showed in the figures, the mechanism of selection of the right library (PKCS#11 or CSP) is the same. The interceptor/wrapper recognizes the smart card and allows selection of the right software component. Obviously the semantics of the ATR must be defined. This information is standard in Italy. In fact using the substring of ATR named historical bytes it is possible to recognize without ambiguity the specific smart card. This standardization covers both Italian Electronic Identity Card and National Services Card allowing the use of either card to access on line services provided by public administration. The security of the access is assured by the SSL/TLS protocols.

6.4.3 Italian state of the art

The success of this solution is remarkable. A reasonable independence by the manufacturers is allowed. It is also possible to extend the support to new smart cards only by upgrading the Unique Layer and the associated software components.

In the PEPPOL project the mobility of the signer is not critical but these solutions are described to make available a more detailed analysis.

At the moment, the Italian market supports about ten smart cards for digital signatures and about six smart cards for SSL/TLS authentication. The PC operating systems supported are Windows XP and Vista, the more distributed Linux kernels and in some particular applications also MAC OS 10.X (also using the Token interface).

Italy is also developing open source software (based on the Open SC project) to support a subset of the smart cards available in the Italian market.

Increasing knowledge in this technology has enabled during the last two years the possibility to make available PKCS#11 or CSP software libraries that support directly multiple smart cards. This solution is light weight and efficient but works without particular problems only if the developer has detailed knowledge about the internals (the operating system) of the smart card.

For this reason in Italy the solution is to make available a dedicated software library for authentication (SSL/TLS support per services and electronic identity card) and a unique layered architecture for digital signature. Probably the influence on the European market of the standards related to the European Citizen Card (CEN/TS 15480) will drive the Italian choices in the next twenty four months.

6.5 Generic Solution for Card Support

A simple guess about a generic solution for card support is described in the following chapters as indicated in 6.1, including a more detailed conclusion.

Currently, there is no standard that covers all demands for a coherent user guidance and general user acceptance. PKCS#11 is not suitable, because the solution has to be language independent. But the software should be able to be employed according to CSP or PKCS#11. That means that the internal object model has to concentrate basically on PKCS #11 and, even more, on smart card objects.

As the SOA definitions of PKCS#11 and CSP imply the mapping of object models, which burdens the performance, on the one hand, and switching between SOA and oo-layers while implementing the software on the other hand, SOA also is no suitable standard.

The definition of the solution should be described in UML, because it is independent from computer languages. As some standards' intentions are auspicious, there are at least approaches for a software based solution for a generic smart card support. Not only for smart cards, but also for card readers a generic solution to control them has to be developed. This solution should be recommended to build on existing object models of readers according to driver standards and not following a certain interface standard.

Naming objects and interfaces should be natural and not define unnecessary obsolete terms (e.g. the controversial tag and object 'DifferentialIdentity' at CIF-Files of ISO 24727). Besides, the solution has to meet the demands of all participants concerning cost intensity. E.g. it must be guaranteed that identification of smart card, structures and workflows of smart cards and readers are largely configurable following efforts of a client-server model.

For user guidance the application should get access to a suitable event system. Additionally, the solution should support options to influence timeouts for pin entries, language, message alerts and character sets. This becomes necessary especially when the systems are distributed with individual implementations on different computers that communicate with one another and mutually provide card readers. In this case the language depends on the single user and not on the computer.

Generic card support layer

The draft has to consider theoretical and practical aspects. Considering the costs for implementations and evaluation the solution should be as configurable as possible and the participants have to add the respective pieces of information concerning smart cards and card readers. An implementation additionally needs further data that are not considered here.

The basic architecture is similar to the Italian solution, except that PKCS#11 is not implied, configuration data is used and different driver standards are supported as object models with an optional service layer.

Figure 7 shows a scheme of the basic model differing between a high level and a low level API - model and service layer. Figure 8 roughly defines the object model.

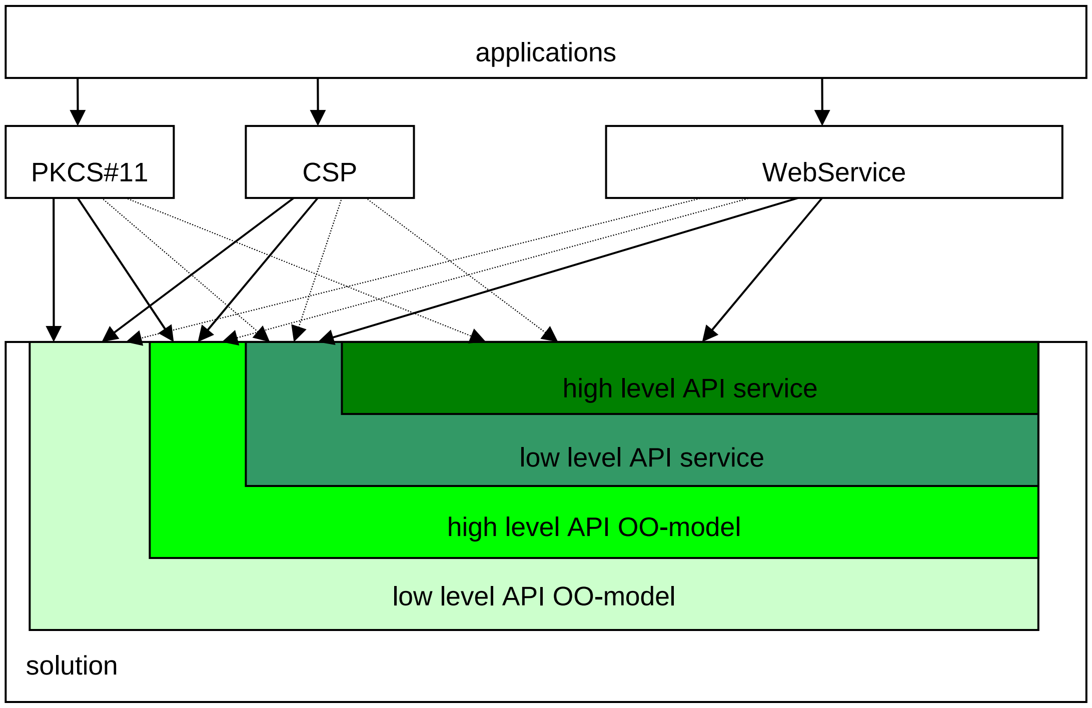

Figure 7: Basic model differing between a high level and a low level API

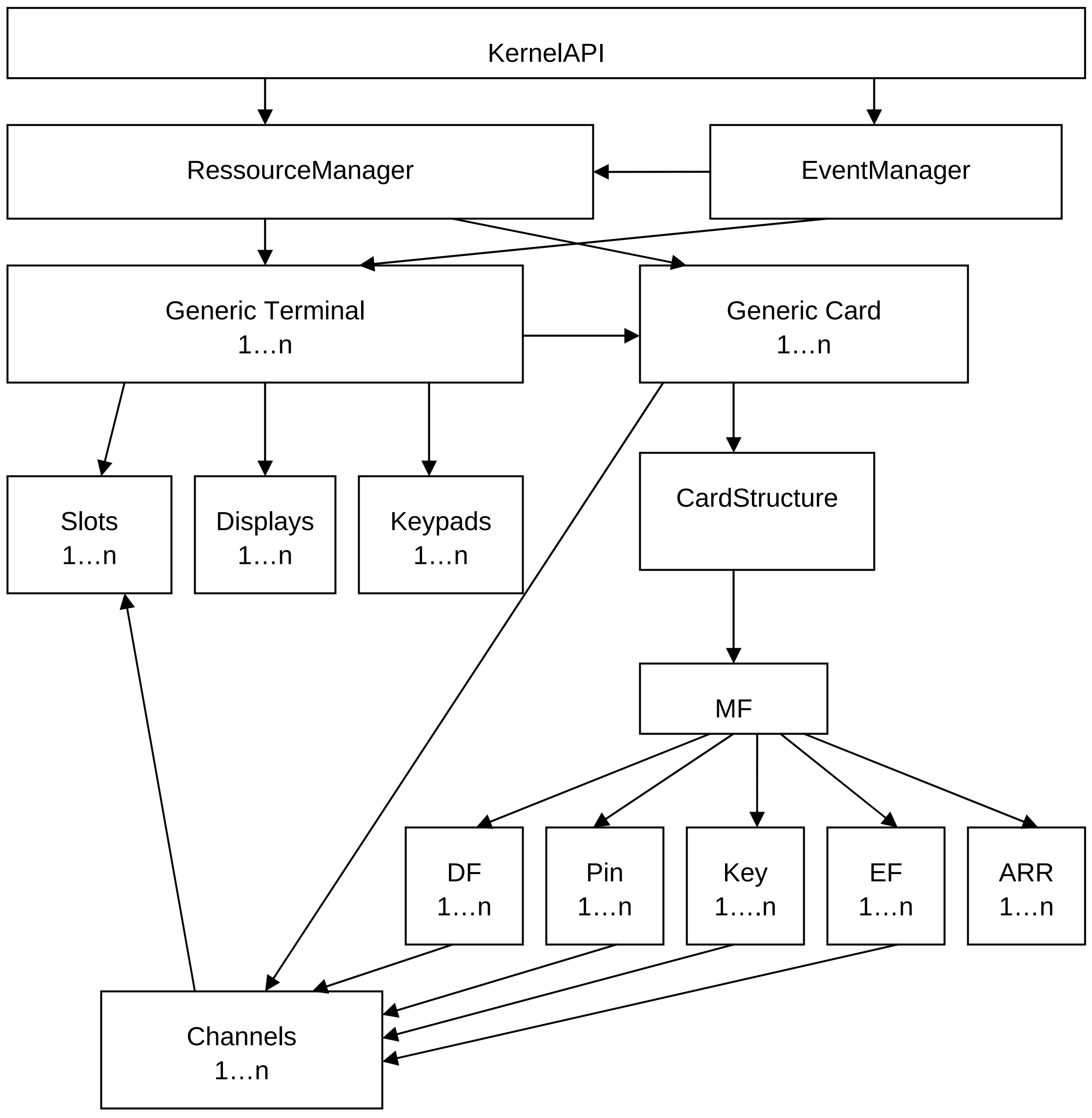

Figure 8: Generic solution object model

A generic terminal offers different wrappers for the various driver standards. The generic terminal has to provide interfaces that contain the highest possible amount of functions of different driver standards.

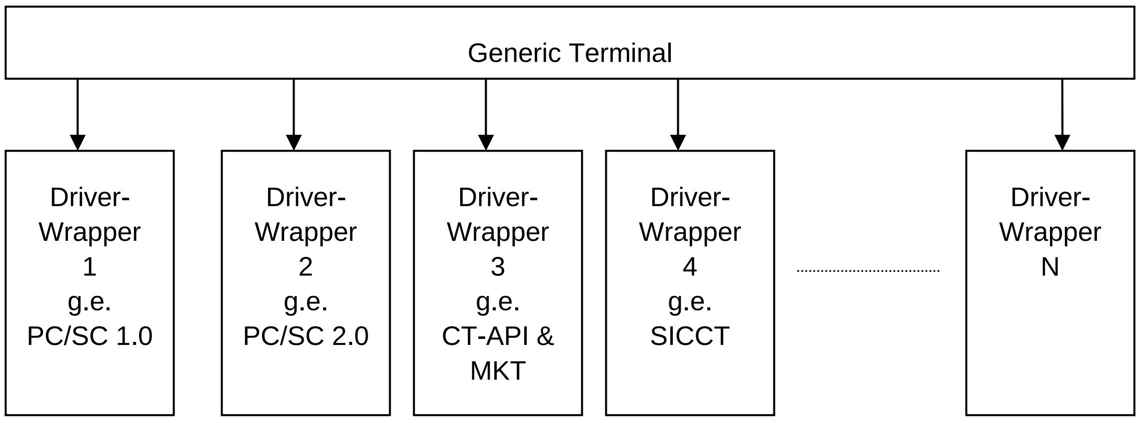
Generic Terminal
Driver-Wrapper 1 g.e. PC/SC 1.0
Driver-Wrapper 2 g.e. PC/SC 2.0
Driver-Wrapper 3 g.e. CT-API & MKT
Driver-Wrapper 4 g.e. SICCT
Driver-Wrapper N

Figure 9: Generic terminal

Restricted to identification and one key, a generic terminal for a smart card looks like this:

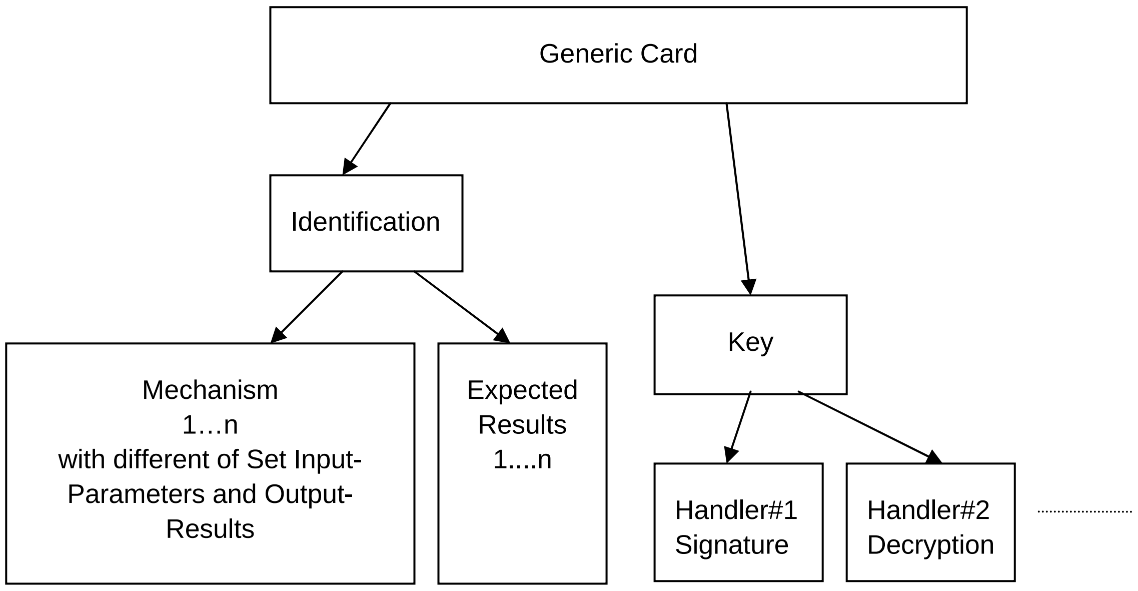

Figure 10: Generic card

To recognize a card, a certain set of mechanisms using configurable parameters and results should be established. The use of parameterized mechanisms should provide a possibility to build a high-performance decision tree for card identification in internal processes and a simple recognizable way for card configuration providers. Also results could be cached automatically to avoid e.g. that different data are read out more than once and to allow references to configuration data. A Handler represents a single execution e.g. of a signature as workflow for one or more smart cards. It is assumed that configuration data with static information to be used with known mechanisms and APDU sequences are to be provided for all objects, handlers, wrapper, etc. As this would mean an immense configuration effort, references should be used in order to avoid redundancies. Configuration files and contents should be assigned individually, if possible, to separate participant groups. The administration or integration of configuration files to one file should generally be possible, because of the technical division of contents.

The handlers mentioned above correspond to defined workflows that consist of processing instructions and commands for one certain card. Here, the amount of information necessary for card reader and card reader standards consciousness is not specified in detail.