Journal of Web Engineering, Vol. 1, No.2 (2003) 147-162
© Rinton Press

AN XML-BASED PLATFORM FOR E-GOVERNMENT SERVICES DEPLOYMENT

ANASTASIOS IOANNIDIS
University of Athens,Greece
aioann@di.uoa.gr

MANOS SPANOUDAKIS
University of Athens,Greece
grad0370@di.uoa.gr

GIANNIS PRIGGOURIS
University of Athens,Greece
iprigg@di.uoa.gr

STATHES HADJIEFTHYMIADES
Hellenic Open University, Greece
shadj@di.uoa.gr

LAZAROS MERAKOS
University of Athens,Greece
merakos@di.uoa.gr

Received March 13, 2003
Revised April 9, 2003

Transaction services that enable the on-line acquisition of information, the submission of forms and tele-voting, are currently viewed as the future of E-Government. Deploying such services requires platform independent access and communications security as a basis. This paper presents a platform for supporting E-Government services in a highly distributed public network environment, based on XML protocols. We discuss the technical details of the developed software that implements these XML protocols. The platform relies on the well-established IPsec and SSL/TLS technologies for ensuring the security of the critical, e-government related data, exchanged over the public network.

Key words: XML, E-Government, E-Voting, Transaction Services, IPsec, VPN, Authentication, Security, SSL/TLS, GSM/WAP, J2EE
Communicated by: M Gaedke & N Guimaraes

1 Introduction

Transaction services that enable the on-line acquisition/purchase of products or services or the submission of data (e.g., forms, voting, requests for the issue of certificates, etc.), are perceived as the future of the electronic government (E-Gov) concept. Forms are of extreme importance to many administrative processes. Tele-voting is employed for the realization of opinion polls and provides a rather unique opportunity for the establishment of a form of direct democracy. The concept of direct democracy suggests that all citizens decide, through voting, on their problems.

The main objective of the EURO-CITI project (realized in the context of the EU IST Programme) is to assess the potential of on-line democracy by developing and demonstrating fully-fledged pilots on transaction services such as tele-voting, electronic submission of forms (allowing citizens to submit official requests) and tele-consulting (allowing citizens to request clarifications and assistance in various subjects). The development of these services relies on open, well-established technologies such as XML, WWW and IPsec.

We discuss the EURO-CITI distributed architecture that allows the end users (citizens) to access services through home or public PCs, kiosks and GSM/WAP handsets and employing security mechanisms like smart cards and digital certificates. The same architecture caters for the joint realization of e-government transaction services between different European sites (termed EURO-CITI nodes, E-C nodes). Such sites are interconnected by means of virtual public networks (VPN) deployed over the public Internet. The E-Government specific protocol that is used between E-C nodes is based on the XML standard.

The rest of the document is structured as follows. In Section 2, the EURO-CITI platform requirements from the perspective of the involved entities are presented. In Section 3, the general network requirements of the EURO-CITI services are introduced. The security mechanisms utilized for connections between EURO-CITI servers are also described and details are provided for a server-to-server connection concerning the tele-voting service. The issues of user authentication and service access are addressed here, with emphasis on security. Section 4 provides more details for the XML protocols introduced in the EURO-CITI platform. In Section 5 we discuss the various scenarios for service operation within EURO-CITI networks. The exchange of messages between EURO-CITI servers is presented. In Section 6, a description of the internal architecture of the ESM kernel is provided. We conclude this paper in Section 7, summarizing the platform architecture and supported functionality.

2 EURO-CITI Platform Requirements

The EURO-CITI platform is designed to support requirements for a number of different entities that interact with the platform. These entities include the end users, the LAs (Local Authorities) and the system administrators. Determination of these requirements was made through the use of questionnaires, which were distributed and answered by the different entities.

The results of the questionnaires indicated that LAs wish to provide an as wide as possible support for the tele-voting, tele-consulting and consultation services to end users. Some services were already available by other means, while others, such as large scale votings, were unavailable but there was a desire to integrate them into existing processes. Especially for tele-voting, the application area consisted of voting for issues that were not of a sensitive nature, and did not impose strict security requirements such as those for national elections. The EURO-CITI platform was therefore designed with a security framework that allowed a high degree of security at the communication channels, but treating the servers at the LAs as trusted entities.

The questionnaires also indicated that end users desired to have a wide range of access mechanisms at their disposal. This included access from kiosks, home PCs and even WAP-capable phones for a small minority, but also included traditional methods of service access such as mail and personal contact. The integration of the new facilities imposed the need for process reengineering, which was also performed but is outside the scope of this paper.

3 Networking and Security in EURO-CITI

Access to EURO-CITI services should be possible from home PCs and networked public PCs/kiosks installed at commonly visited places. Local Authorities aim also to exploit recent advances in mobile data access, and offer their services through the industrially established Wireless Application Protocol (WAP) [5]. Since, the EURO-CITI platform should cater for complete network - level security, public access will be based on the SSL/TLS framework.

Each EURO-CITI server node is responsible for supporting and implementing services that are provided by Local Authorities of a specific geographic area to local citizens. However, an equally important aspect of EURO-CITI is that individual servers should be able to participate in networks[a] of EURO-CITI nodes. Such networks will be configured:

- on demand (for example for the purposes of a specific voting procedure and then they will cease to exist either as a whole or partially)
- on a permanent basis. In such networks, participants will be cities that share common interests and are willing to co-operate for long time periods.

The requirement for establishment and configuration and maintenance of EURO-CITI networks has a serious impact on the design and implementation of the architecture of each server. Each server needs a well-defined, XML-based interface for communication with the other servers.

In each Local Authority, a server managing security issues is installed and operated. Such node is referred to as S_Host and is responsible for implementing the required security policy. S_Hosts need to securely communicate with each other over typical Internet connections. To accomplish this objective, S_Hosts form a EURO-CITI VPN by establishing dynamic security associations.

IPsec is considered to be the most appropriate technology for communication between S_Hosts. The IPsec interfaces that are integrated in the EURO-CITI platform must support the capability to dynamically establish/release IPsec Security Associations (SAs). In IPsec, secure channels between communicating parties are defined through these security associations, and each association has its own security characteristics. In our network design, we adopted a solution involving the establishment of Security Associations at the application end-point level (i.e. defined by the IP addresses and port numbers of the endpoints) instead of at the node level (i.e. defined only by the IP addresses), which is more flexible and provides direct support for EURO-CITI services.

Moreover, we adopted the permanent establishment of a secure communication channel (to be referred to as P-Channel) between co-operating Authorities. This channel is used as a bearer for control information between the nodes of Local Authorities. It conveys the specific signalling that is required for the ad-hoc set-up of security associations (supplementing the internal structure of the VPN) that support new service applications, such as new voting sessions. This scheme is illustrated in Figure 1.

[a] Here, the word network denotes a logical association between the involved parties

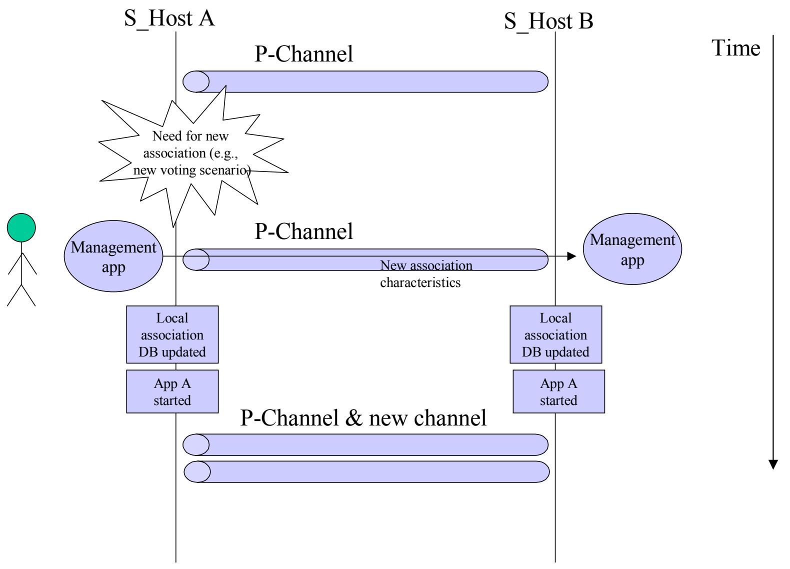

S_Host A
S_Host B
Time
P-Channel
Need for new association (e.g., new voting scenario)
Management app
P-Channel
New association characteristics
Management app
Local association DB updated
Local association DB updated
App A started
App A started
P-Channel & new channel

Figure 1: Security associations between S_Hosts

The internal architecture of the EURO-CITI Security Manager (ESM), the key component for the secure, networked operation of EURO-CITI services will be discussed in more detail in section 6.

As already discussed, the two end-points are linked through a pre-installed secure channel (P-channel). The specifications of such a channel are statically configured into the appropriate hosts, as specified by the controlling server software.

The ESM capitalizes on the APIs provided in contemporary VPN software (e.g., Windows 2000, Checkpoint VPN-I, Linux FreeS/WAN) for gaining access to native IPsec functionality and bringing the necessary security associations (SA) into effect. In particular, the ESM is capable of signalling, for each dynamic connection, the following information for IPsec use:

- A source IP address, subnet mask and port
- A destination IP address, subnet mask and port
- Security parameters for negotiation (e.g., the encryption algorithm like 3DES [1], [4]).
- The IPsec connection mode (tunnel vs transport [1]).
- preshared key authentication
- Algorithms for IKE [3], [6] negotiations (e.g., using 3DES, SHA [1], [4], [6]).

ESM keeps track of the security associations installed so far, the socket numbers reserved on both endpoints (a specific range of socket numbers are reserved – a pool of ports) and proceeds with new channel establishments as needed. The ESM classes on the inviting end-point create new keys / pass-phrases for the newly requested scenarios and provide them (along with a selection of other security parameters) to the invited endpoint through the P-channel. The invited end-point receives the security settings along with the specification of the newly requested voting scenario parameters (in XML [9] format) and passes those to the management application. The administrator on the receiving end-point may then decide whether to accept the voting scenario, possibly with modifications in the suggested parameters, or reject it. Responses are transmitted back to the inviting authority again through the P-Channel. Provision is also taken for acknowledgements in this negotiation (3-way handshake as shown in the UML diagram in Figure 2). Following the agreement on a new voting scenario and the establishment of the appropriate channels, all communication pertaining to this scenario is realized through this new secure channel.

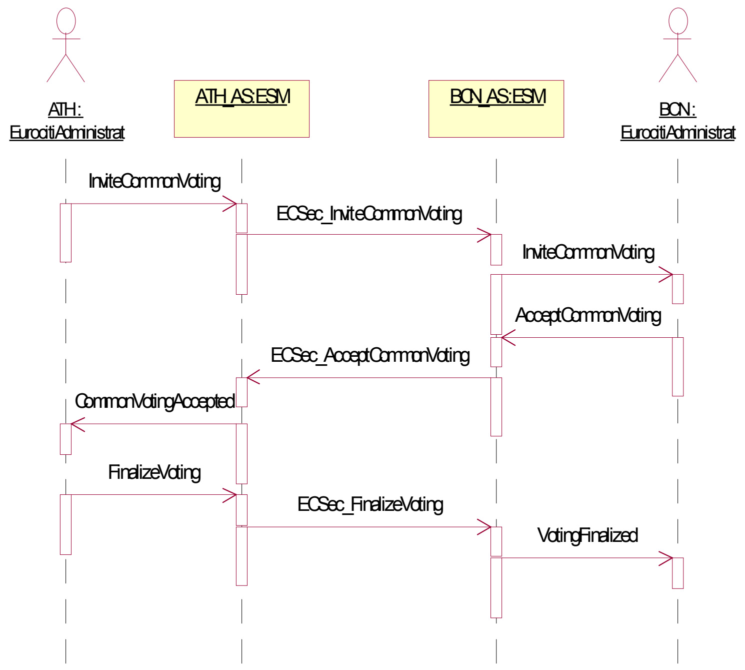


Figure 2: 3-way handshake for new cross-network Voting scenarios

ESM provides a specific interface to the Java-based application (based on Java Server Pages (JSPs) and servlets) that is developed for the set-up/administration and operation/running of EURO-CITI applications. This interface is using the Remote Method Invocation (RMI) technology. Through this interface it is possible to:

- request the establishment of a new voting scenario that involves another Local Authority (EURO-CITI) node,
- request the transmission of statistics (periodic or on-demand),
- request the transmission of final results and,
- authenticate remote users.

The establishment of dynamic communication networks is performed in such a way so as to allow a good degree of scalability for the platform. Although each node needs to know about every other node to achieve cooperation, direct connections between the nodes are not required. Instead, the node that initiates a scenario acts as an intermediate communication point, coordinating the entire session centrally, and therefore the total processing and bandwidth requirements of the platform increase linearly with the number of EURO-CITI servers. The details of session management are presented in more detail in section 5.

3 XML Messages

Secure hosts exchange XML messages in order to provide the specified e-Government functionality. These messages utilise the XML protocols specified for supporting the services provided by the EURO-CITI platform. The Election Markup Language (EML [12]) proposed recently by the OASIS consortium specifies a schema for voting services. Our protocol, apart from the tele-voting service, also supports services such as tele-consulting (enabling citizens to ask for advice) and electronic submission of forms.

In order to fulfill the requirements for the services provided within the platform we introduce the XML messages drafted in Table 1. More details on the functionality and the use of these messages are provided in sections 4 and 5.

Table 1. XML message elements

Message	Parameters
ECSec_AuthenticateUser	• Username (e.g.,citi0134@bcn), • Password (sta12thes), • Unique Scenario ID
ECSec_UserAuthorised	• Username, • Unique Scenario ID • Status Code
ECSec_InviteCommonVoting	• Unique Scenario ID (e.g., 1023@Athens) • Voting subject, • User groups eligible to vote, • Extra info on the voting issue, • Voting options (in multiple-choice format), • The start/end date of the voting period, • Partial/final results option • Security Algorithm option list • Keys/Passphrases
ECSec_RejectCommonVoting	• Unique Scenario ID • Reason
ECSec_AcceptCommonVoting	• Unique Scenario ID • Changes
ECSec_FinalizeVoting	• Unique Scenario ID
ECSec_Statistics	• Unique Scenario ID • VotingOptions/Counts
ECSec_RequestStatistics	• Unique Scenario ID • Statistics Kind
ECSec_TerminateVoting	• Unique Scenario ID • Options for releasing secure connections, backing-up voting results, removing data from local databases, etc.
ECSec_Alert4Authentications	• Unique Scenario ID (e.g., 1023@Athens) • Voting subject, • User groups eligible to vote, • Voting options (in multiple-choice format), • Security Algorithm options' list • Keys/Passphrases

The XML DTDs (Document Type Definition) of the two most important messages, namely the ECSec_InviteCommonVoting and ECSec_Alert4Authentication are provided in Figure 3 and Figure 4 respectively, while the full set of DTDs for the rest of the messages is available in [10]. The ECSec_InviteCommonVoting message is used for the initialisation of a network tele-voting scenario and contains information such as voting information, security options and identification of the EURO-CITI node that initialises the scenario. Correspondingly, for the initialization of a European scope scenario an ECSec_Alert4Authentication message is used. More details on network and European scope scenarios are presented in the following section.

```
<?xml encoding="UTF-8"?>
<!DOCTYPE ECSec_InviteCommonVoting [
<!ELEMENT ECSec_InviteCommonVoting (sendingNodeID, scenario)>
<!ELEMENT sendingNodeID (#PCDATA)>
<!ELEMENT scenario (scenarioID, scenarioTitle, language, subject, userGroup,
info, administratorId, votingOption+, partialResults, passPhrase,
securityOption, categoryId, notificationTimeout, keyword+)>
<!ATTLIST scenario
    mode        (RL|RR) #REQUIRED
    startDate   CDATA   #REQUIRED
    endDate     CDATA   #REQUIRED>
<!ELEMENT scenarioID (#PCDATA)>
<!ELEMENT scenarioTitle (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT userGroup (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT administratorId (#PCDATA)>
<!ELEMENT votingOption (mchoice+)>
<!ATTLIST votingOption
      seqNumber      CDATA         #REQUIRED
      requiredOption (Yes|No)      #REQUIRED
      type           (Radio|Check) #REQUIRED
      text           CDATA         #IMPLIED>
<!ELEMENT mchoice (#PCDATA)>
<!ATTLIST mchoice
      seqNumber CDATA #REQUIRED>
<!ELEMENT partialResults (#PCDATA)>
<!ELEMENT passPhrase (#PCDATA)>
<!ELEMENT securityOption (#PCDATA)>
<!ELEMENT categoryId (#PCDATA)>
<!ELEMENT notificationTimeout (#PCDATA)>
<!ELEMENT keyword (#PCDATA)>
]>
```

Figure 3: DTD of the Invite Common Voting Message

```
<?xml  encoding="UTF-8"?>
<!DOCTYPE ECSec_Alert4Authentication [
<!ELEMENT ECSec_Alert4Authentication (sendingNodeID, scenarioID,
parameters)>
<!ELEMENT sendingNodeID (#PCDATA)>
<!ELEMENT scenarioID (#PCDATA)>
<!ELEMENT parameters (administratorId, subject, userGroup, info?,
startDate?, endDate?, notificationTimeout, notifyDate?, categoryId?, type)>
<!ELEMENT administratorId (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT userGroup (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT startDate (#PCDATA)>
<!ELEMENT endDate (#PCDATA)>
<!ELEMENT notifyDate (#PCDATA)>
<!ELEMENT notificationTimeout (#PCDATA)>
<!ELEMENT categoryId (#PCDATA)>
<!ELEMENT type (#PCDATA)>
]>
```

Figure 4: DTD of the Alert For Authentication Message

5 Service Negotiation Protocol

A EURO-CITI server is capable of initiating applications setup by the administrator and accessed by the citizens, to provide services such as tele-voting and tele-consulting. Services are internally managed by Scenario objects, which are responsible for maintaining service state. They are also responsible for managing connections between collaborating EURO-CITI servers, when services are provided at a network level, as is the case with network voting and voting with European scope.

Communication between servers is achieved through the use of Scenario objects. For each service provided by one or more servers, there is a Scenario object on each server. Every such Scenario instance is aware of one or more Scenario objects with which it communicates.

Scenario creation can be performed in two ways. The operator can initiate a new Scenario with the desired parameters, which will provide services for the local server. If the Scenario is of local scope, no network activity takes place. For Scenarios that require cooperation with other servers, an XML based protocol is used to create Scenario objects on the remote servers with the corresponding service parameters. This second method for automatic Scenario creation still needs to be approved by the operators on the remote servers, in order to become active and start providing the actual service.

The EURO-CITI platform supports two types of cooperation between servers. Loose cooperation is provided for services that only use the authentication or authorization functionality from remote users. Strong cooperation is provided for services that require the active participation of their remote peers.

5.1 European Scope Scenarios

A loose type of cooperation is available for Scenarios with European scope where a master Scenario can request remote servers to authenticate users which have the right to participate on a locally provided service but are not registered with the Local Authority. The negotiation for such a communication (presented in Figure 5), involves sending a message from the master Scenario to selected (or all) remote servers, (Alert for Authentications). The remote server operators are notified upon receipt of the message (see for the ESM Notifier in subsequent paragraphs) and can decide to accept or reject the request by sending an Ack/NAck message (see Figure 6). The operator of the master Scenario can then decide to proceed with the service excluding the servers, which did not wish to participate or cancel the proposed Scenario.

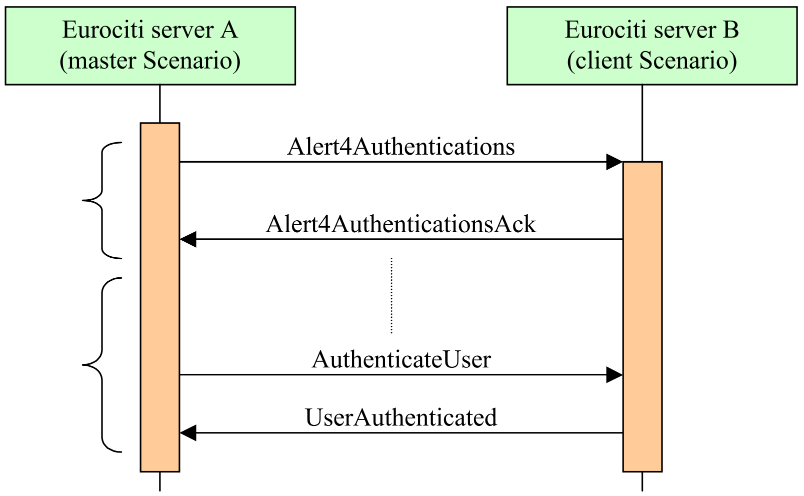


Figure 5: A simple Scenario with European scope

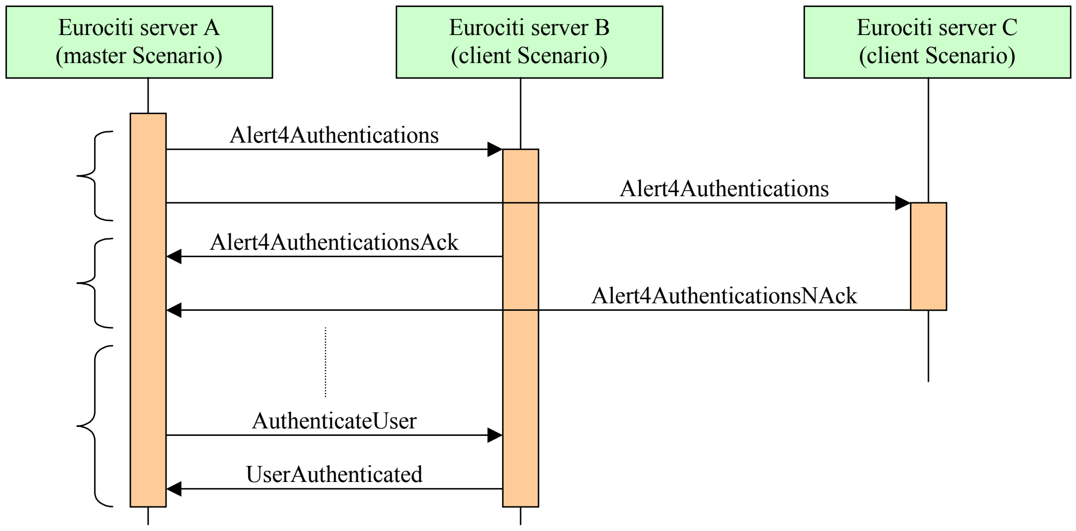

Eurociti server A (master Scenario)
Eurociti server B (client Scenario)
Eurociti server C (client Scenario)
Alert4Authentications
Alert4Authentications
Alert4AuthenticationsAck
Alert4AuthenticationsNAck
AuthenticateUser
UserAuthenticated

Figure 6: One of the servers rejects the participation request

When multiple servers participate in a Scenario, client Scenarios are not directly communicating with each other, but they always send their requests and responses to the master Scenario. The master Scenario is responsible for coordinating communications between all clients (star-like communication).

5.2 Network Scenarios

A more complex communication protocol exists for Network Scenarios. Network Scenarios are executing on multiple servers, the service is provided on all servers locally (authentication/authorizations may be done remotely) and the local results/entries are made available to all participating servers.

A network Scenario is created on a server acting as a master. An InviteCommonVoting XML message is then sent to selected EURO-CITI servers for participation. Server operators can respond to such a request either by unconditionally rejecting the Scenario (using the RejectCommonVoting message) or by accepting the Scenario and sending the AcceptCommonVoting message. Acceptance of a Scenario can be conditional upon one or more Scenario parameters. The master Scenario operator collects all answers from the invited servers and can decide how to handle proposed changes and rejections.

For network Scenarios, the most trivial case is when all remote servers accept the Scenario parameters unconditionally (see Figure 7). In this case, the master Scenario sends a notification message (FinalizeCommonVoting) to all remote servers indicating the successful completion of all negotiations and that all remote servers can launch the service.

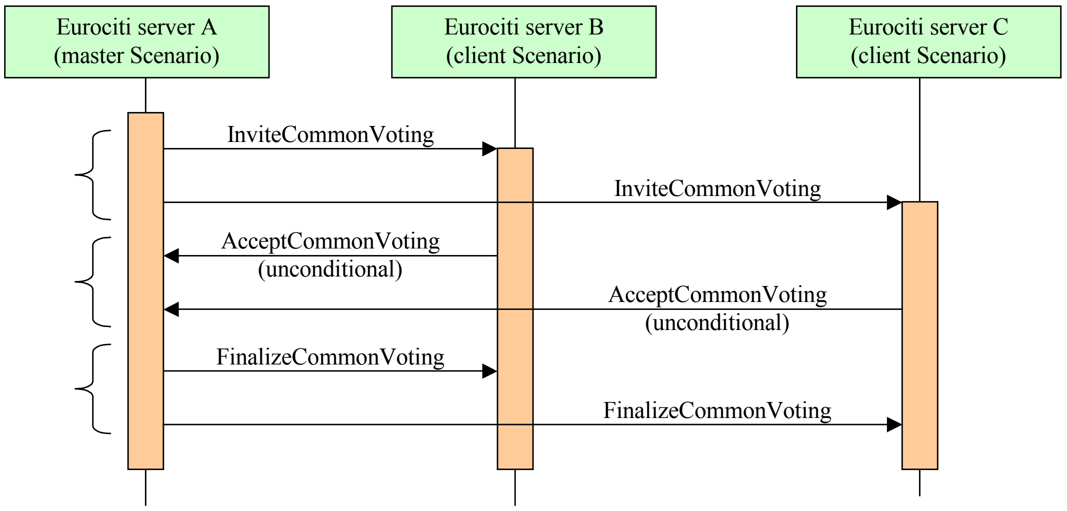

Eurociti server A (master Scenario)
Eurociti server B (client Scenario)
Eurociti server C (client Scenario)
InviteCommonVoting
InviteCommonVoting
AcceptCommonVoting (unconditional)
AcceptCommonVoting (unconditional)
FinalizeCommonVoting
FinalizeCommonVoting

Figure 7: Trivial negotiation of network Scenario

When some of the servers respond with rejects (see Figure 8), the operator may choose to finalize the Scenario with all servers that accepted the service unconditionally. The master Scenario operator may also choose to reject the proposed Scenario, by sending to all accepting servers a RejectCommonVoting message, indicating an unsuccessful negotiation phase.

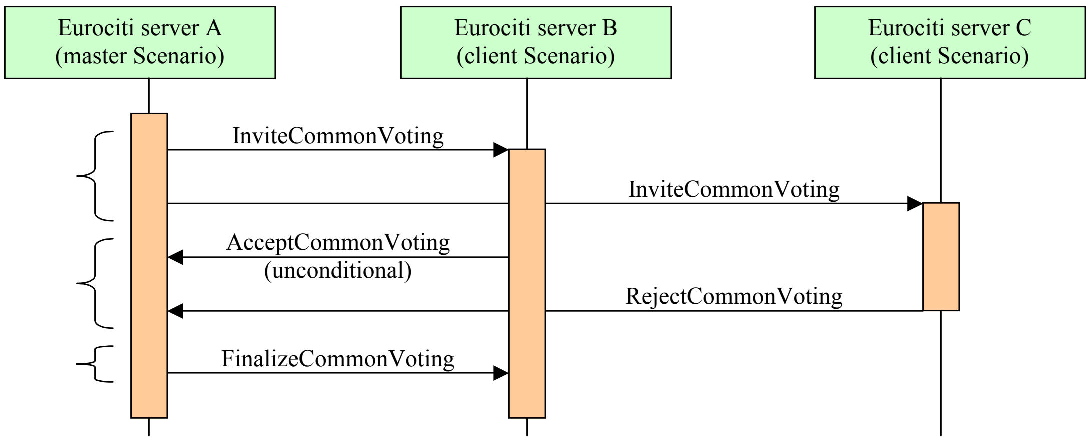

Eurociti server A (master Scenario)
Eurociti server B (client Scenario)
Eurociti server C (client Scenario)
InviteCommonVoting
InviteCommonVoting
AcceptCommonVoting (unconditional)
RejectCommonVoting
FinalizeCommonVoting

Figure 8: Rejecting servers do not participate in the Scenario

Conditional acceptance requires a renegotiation of the Scenario parameters (presented in Figure 9). Each conditional acceptance carries an indication of the Scenario parameters, which are not accepted as well as suggestions on how these parameters can be modified for the Scenario to be accepted. All answers are collected by the master Scenario operator who can decide on how to proceed with the negotiations. The most common action would be to send rejection to all accepting servers and send a new invitation to all servers incorporating the requested changes. Alternatively, the master Scenario operator may choose to finalize the Scenario with the servers, which accepted unconditionally.

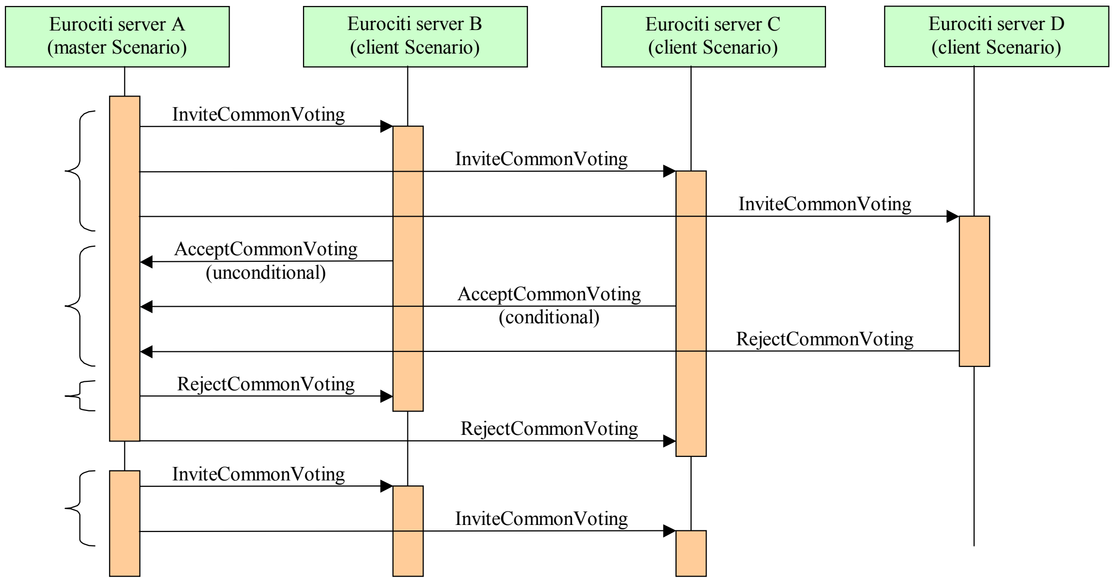

Eurociti server A (master Scenario)
Eurociti server B (client Scenario)
Eurociti server C (client Scenario)
Eurociti server D (client Scenario)
InviteCommonVoting
InviteCommonVoting
InviteCommonVoting
AcceptCommonVoting (unconditional)
AcceptCommonVoting (conditional)
RejectCommonVoting
RejectCommonVoting
RejectCommonVoting
InviteCommonVoting
InviteCommonVoting

Figure 9: Scenario renegotiation with accepting servers

6 ESM Architecture

The EURO-CITI Security Manager (ESM) is a software kernel that incorporates J2EE technologies for supporting the operations of the EURO-CITI server. ESM provides interfaces towards the LDAP and database (DB) servers, the set of EURO-CITI applications and the underlying network infrastructure (Figure 10).

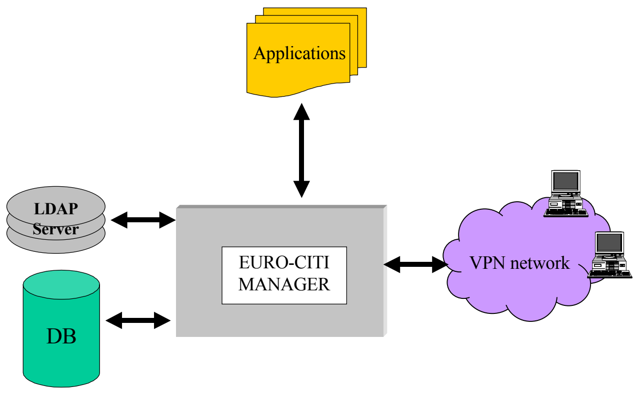


Figure 10: ESM Interfaces

ESM consists of a number of components with very specific tasks. These components are:

- The **ESM Coordinator**,
- The **DB/Directory Manager**,
- The **XML component (Composer and Parser)**,
- The **NET component (Receiver, RX and Transmitter, TX),**
- The **Queue**,
- The **Notifier**, and
- The **Dispatcher**.

The above components are integrated in ESM as shown in Figure 11. The internal structure of ESM resembles that of basic communication transceivers. A set of components (i.e., the XML Composer and Net/TX) is exclusively devoted to outbound communication (transmitter). Inbound traffic is handled by the functional combination of Net/RX, XML Parser, the Queue, the Dispatcher and the Notifier.

The Coordinator is the basic module of the architecture, as it undertakes the task of coordinating the other modules, as well as the communication with external entities through the RMI interface. Those entities correspond to the EURO-CITI applications, intended for access either by simple users or administrators. RMI is used by the system to trigger the dynamic creation of new ESM objects like Scenarios or Connections. Following their instantiation, such objects may have direct access to internal ESM components like the Net Component (TX part), the XML Composer and the DB/Directory Manager. Hence, after they are instantiated, ESM internal objects undertake all the operations needed (see Figure 12) in order to communicate with their peers.

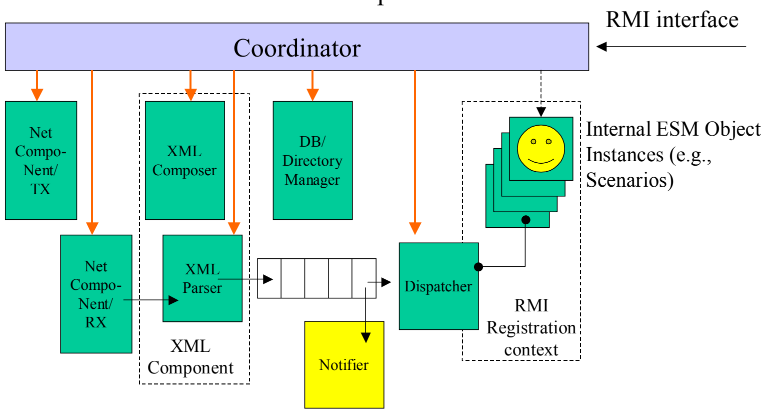


Figure 11: Structure of the Security Manager

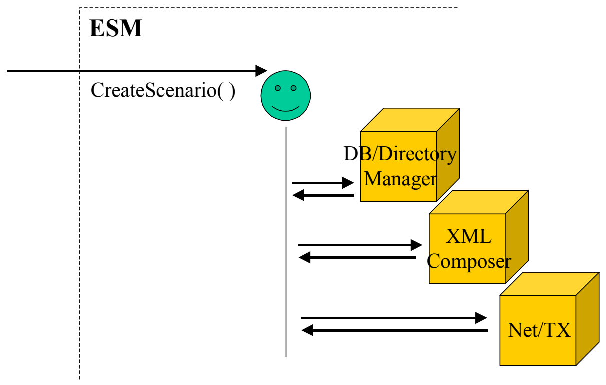


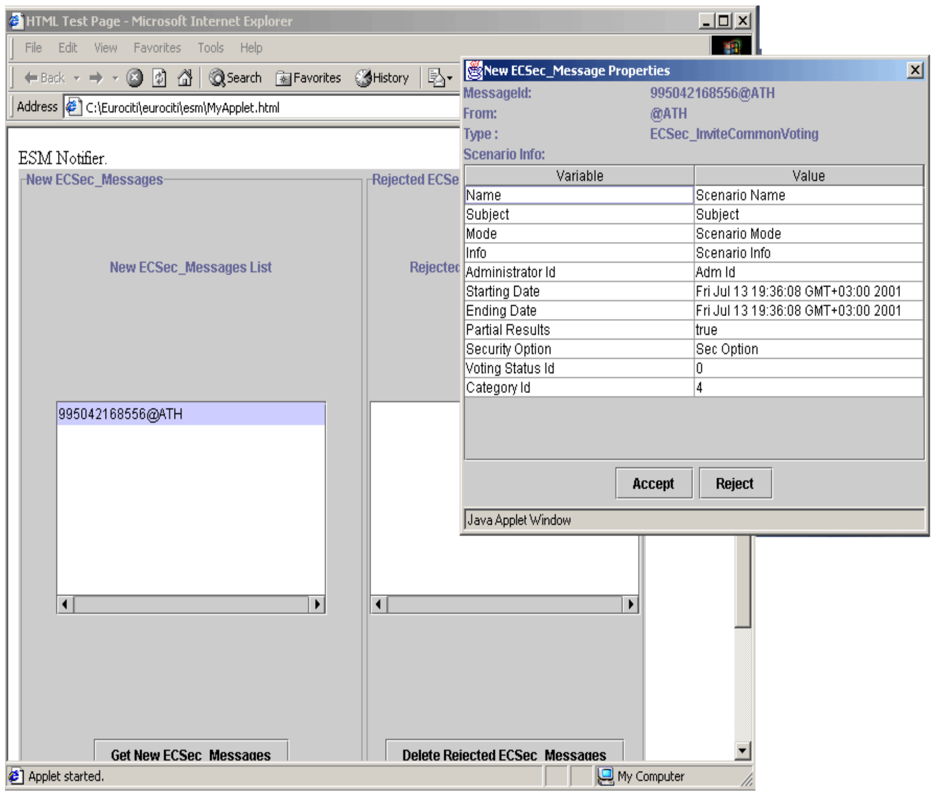
Figure 12: Control flow for outbound messaging

The discussed internal organization of ESM allows the thread-safe operation of the various components. The design of ESM components for outbound messages is strongly coupled with the Scenario object instantiated (i.e., control is on the Scenario object). On the reverse direction (i.e., inbound messages destined to instantiated scenario objects or remote requests - invitations pertaining to new scenarios) things are handled in a different way, following a loosely coupled approach. The RX part of the Net component acts as a Socket server awaiting idle for inbound messages. Following the reception of a message, the contents of the Socket buffer are passed to the XML parser for further processing. The XML parser (based on the Simple API for XML, SAX [11]) is capable of identifying the nature of the inbound message and verifying its validity. The construction of objects that correspond to each message is performed in a generic way, using the Java Reflection mechanism. This allows for flexibility in the specification of XML messages, since once the DTD is specified, creation of a corresponding class is the only requirement for the message to be recognized, processed by the XML parser and inserted into the JMS queue.

The contents of the ESM queue can be consumed either by the Notifier or by the Dispatcher components. Messages that need administrator’s approval in order to be further processed are passed to the Notifier while the Dispatcher manages the rest “silently”.

The Notifier is a Java applet (Figure 13) that runs on each EURO-CITI server. The Notifier entity has been introduced in order to present to the administrator, inbound messages such as requests (Invitations) for new voting scenarios or alerts for future authentications by the remote end. Such signals are not bound to existing ESM Object instances (e.g. Scenarios). Hence, it is required to obtain administration approval (and possibly have their parameters changed) prior to their processing. If they are approved, they are also passed to the Dispatcher, which handles them appropriately. The Dispatcher, after interpreting the signals retrieved by the queue, uses the RMI registration service in order to:

- locate the proper object instance and invoke the appropriate method on it
- create a new one if this is the case


HTML Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address C:\Eurociti\eurociti\esm\MyApplet.html
ESM Notifier.
New ECSec_Messages
Rejected ECSe
New ECSec_Messages List
Rejectec
Get New ECSec_Messages
Delete Rejected ECSec_Messages
Applet started.
My Computer
New ECSec_Message Properties
Accept
Reject
Java Applet Window

995042168556@ATH

MessageId: 995042168556@ATH
From: @ATH
Type : ECSec_InviteCommonVoting
Scenario Info:

Variable	Value
Name	Scenario Name
Subject	Subject
Mode	Scenario Mode
Info	Scenario Info
Administrator Id	Adm Id
Starting Date	Fri Jul 13 19:36:08 GMT+03:00 2001
Ending Date	Fri Jul 13 19:36:08 GMT+03:00 2001
Partial Results	true
Security Option	Sec Option
Voting Status Id	0
Category Id	4

Figure 13: Screen dump from ESM Notifier

The DB/Directory Manager provides a double interface towards the LDAP server and the DB component. The former is required in order to handle remote requests for authentications and authorizations, while the latter is used for accessing application specific data (e.g. the statistics to be transferred between the servers throughout the progress of a ballot). The DB interface is also used, upon initialisation of a new voting scenario, for creating a persistent replica of the scenario object within the underlying DB.

Due to the adoption of J2EE components, ESM can be operated on any platform (e.g, UNIX, Windows 2000). In a UNIX or Windows NT node, the Windows 2000 security kernel that we currently use for VPN setup is not available. In that case, the Local Authority has to employ a different IPsec package with programmable interface (API). A typical example is the CheckPoint VPN-I software. ESM is capable of handling both VPN kernels. A layer has been designed and placed between the ESM (specifically the Net component) and the security kernel. This layer (ESM VPN Convergence Layer) behaves much alike a driver that supports many IPsec packages. It bridges the generic, package-independent functionality of ESM with the specific functionality that the VPN kernel provides (Figure 14).

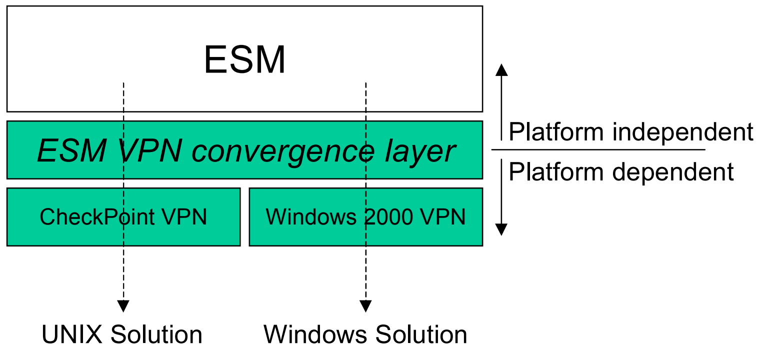


Figure 14: ESM support for multiple IPsec kernels

An important component of the overall architecture was the need to maintain consistency between components within the ESM, as well as between cooperating components in separate ESMs. The need for consistency was identified early in the development process and was subsequently verified during the platform tests. The use of the J2EE transaction and persistence mechanisms, which are pervasive throughout the platform, along with the specified message confirmation mechanisms, allow a high degree of integrity to be achieved and ensure that even temporary failures (both in the network and even in the various components) do not invalidate the status of executing scenarios.

7 Conclusions

The use of widely available technology such as SSL/TLS, XML and IPsec allows for the creation of a flexible EURO-CITI base platform, on top of which E-Government services can be deployed. The service front-end is WWW/WAP based and allows access to services from a wide variety of terminals such as PCs, kiosks and WAP-enabled mobile phones. Security support for accessing EURO-CITI services is provided through the SSL/TLS/WTLS mechanisms, which are available on existing WWW/WAP browsers. Services for which this security level is sufficient should be able to reach a wide audience, improve upon the administrative procedures of Local Authorities and promote direct democracy.

An XML based protocol is implemented and used, to support the EURO-CITI services. For services where more than one Authority participates, such as tele-voting, the EURO-CITI platform ensures the security of transactions using IPsec.

The performed trials have verified that the design goals have been accomplished, with the processes of scenario establishment as well as end-user access being sufficiently simple so as to allow wide-scale application. This applied to all service categories, ranging from local scenarios to network scenarios, with the participation of three servers handled by the Local Authorities and actual usage by a select number of citizens.

In this paper, we presented the design/implementation of the EURO-CITI architecture, the J2EE platform, and the XML protocol. We discussed the different modes of collaboration between EURO-CITI nodes and their impact to the XML-based protocol and we also discussed the internal organization of ESM and present its constituent modules. Finally, we showed how outbound and inbound communication is handled and the interfaces of ESM to other modules such as the LDAP and the IPsec kernel, and how these components are coordinated to provide a highly functional and robust platform for the provision of e-Government services for Local Authorities.

References

[1] S. Kent and R. Atkinson, “Security Architecture for the Internet Protocol”, IETF RFC 2401, November 1998.

[2] A. Valencia, K. Hamzeh, A. Rubens, T. Kolar, M. Littlewood, W. M. Townsley, J. Taarud, G. S. Pall,B. Palter and W. Verthein, “Layer Two Tunneling Protocol-L2TP”, Internet Draft, March 1998.

[3] L. Barriga, R. Blom, C. Gehrmann, and M. Naslund, “Communications security in an all-IP world”, Ericsson Review 2000, No 2, pp 96-107.

[4] O. Kallstrym, “Business solutions for mobile e-commerce”, Ericsson Review 2000, No 2, pp 80-92.

[5] “WAP Architecture Specification, WAP Forum, April 30,1998.

[6] D. Harkins, and D. Carrel, “The Internet Key Exchange (IKE)”, IETF RFC 2409, November 1998.

[7] ANSI X3.106, “American National Standard for Information Systems-Data Link Encryption”, American National Standards Institute, 1983.

[8] D. Maughan, M. Schertler, M. Schneider, and J. Turner, “Internet Security Association and Key Management Protocol (ISAKMP)”, RFC 2408, November 1998.

[9] M. Leventhal, D. Lewis, and M. Fuchs, “Designing XML Internet Applications”, Prentice Hall, 1998.

[10] “EURO-CITI D121: Functional Network Architecture Specification”, March 2001, URL: http://www.euro-citi.org/publications/deliverables/d121/WP12_UOA_D121.pdf

[11] “SAX: A Simple API for XML”, URL: http://www.megginson.com/SAX/

[12] OASIS Election and Voter Services Technical Committee, EML, http://oasis-open.org/committees/election/