E-vote 2011

SSA – U Appendix 3

Customer Technical Platform

Project: E-vote 2011

Change log

Version	Date	Author	Description/changes
1.0	26.10.2009		First version

CONTENT

1 Technical Introduction

This document covers the proposed technical solution for the Election System.

The Election System is divided into different modules, each covering different parts and use case requirements given by KRD. This document outlines the Election System architecture, the high level module division and the abbreviations used for each module, and a brief explanation of what use cases each module supports.

This document also describes the infrastructure in terms of hardware, machines and operating system required to run the different parts of Election System and the different environments used to develop, test and deploy the final product.

In addition the software stacks and development tools used to adapt and build the different parts of the Election System are documented.

As requested by KRD in the SSA-U Appendix 2B Requirements table document, there is also a chapter with "Added headings for elaboration" for the elaboration of specific requirements.

2 Architecture

2.1 Modules

The figure below shows the high level modules the complete Election System is divided into. The arrows between the modules indicate in which direction the main flow of information between the various modules goes.

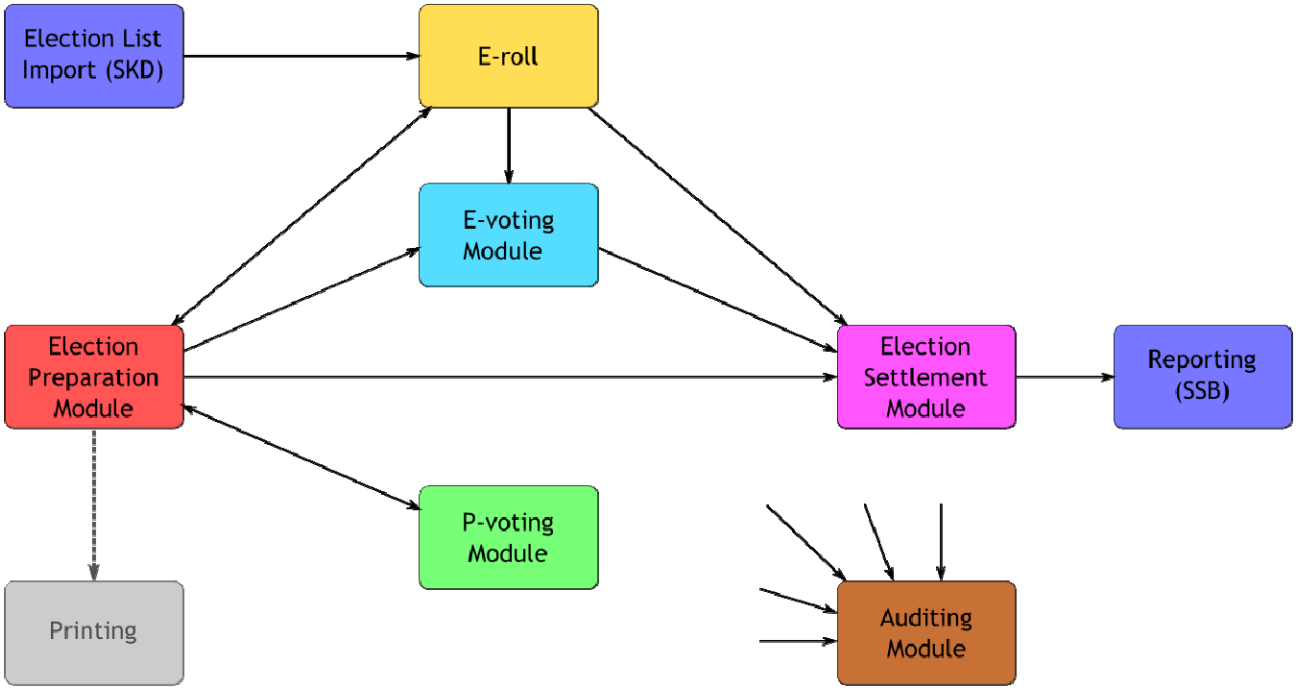


Figure 1 - Election System Modules

The E-roll is the module that will hold all information around the Electoral Roll. It will provide services to import data from SKD, have a self-service website where voters can verify that they are listed correctly, and the Polling Committee and Election Committees can use it to update the Electoral Roll when necessary. This module also serves as the electoral roll data source to the other modules in the Election System, including the Election Preparation Module, E-voting Module and Election Settlement Module.

The Election Preparation Module handles a.o. list proposals and the definition of roles and users throughout the Election System. It will export data to a printer supplier to print out the ballots and the polling cards, and receive data from the P-voting Module that takes care of the e-counting of the paper votes.

The E-voting Module is the module that handles the e-voting process.

At the end of the election process, the Election Settlement Module enters the game. It will receive data from the E-roll, the E-voting Module and the Election Preparation Module to perform the final counting, distribution of seats, and the generation of protocols and reports. The module is also responsible for the export of all relevant information to SSB.

The last module to present is the Auditing Module. This module will receive auditing information from all other components in the Election System. It will create a central audit trail in addition to all the local audit trails on the various components.

The table below gives an overview over which modules that cover which use cases on a high level. (Some of the functional requirements in a use case may be covered by another module.)

E-roll	UC 0.3 Electoral Roll UC 0.4 Exception Process for Electoral Roll UC 3.1 Registration of p-votes in Electoral Roll UC 3.5 Approval of P-votes and Ballots
Election Preparation Module	UC 0.1 Definition of Roles UC 0.2 Configuration of Election System UC 1.1 Submission of List Proposals UC 1.2 Checking and Processing List Proposals UC 3.2 Manual Registration of P-vote Results
E-voting Module	UC 2.1 E-voting
P-voting Module	UC 3.3 Electronic Counting of P-votes (OCR)
Election Settlement Module	UC 3.4 Counting E-votes UC 4.1 Reporting of Results to SSB UC 4.2 Settlement UC 5.1 Reporting
Auditing Module	UC 5.2 Auditing

Table 1 - Module/Use Case Coverage

UC 9.1 Authentication is to a large extent cross-cutting concern, and therefore not represented as such in the figure and table above.

2.2 Election Preparation and P-voting Module

This section gives the architectural overview of the Election Preparation and P-voting module architecture. The following figure shows the components of these modules and how they interact with other modules of the election system.

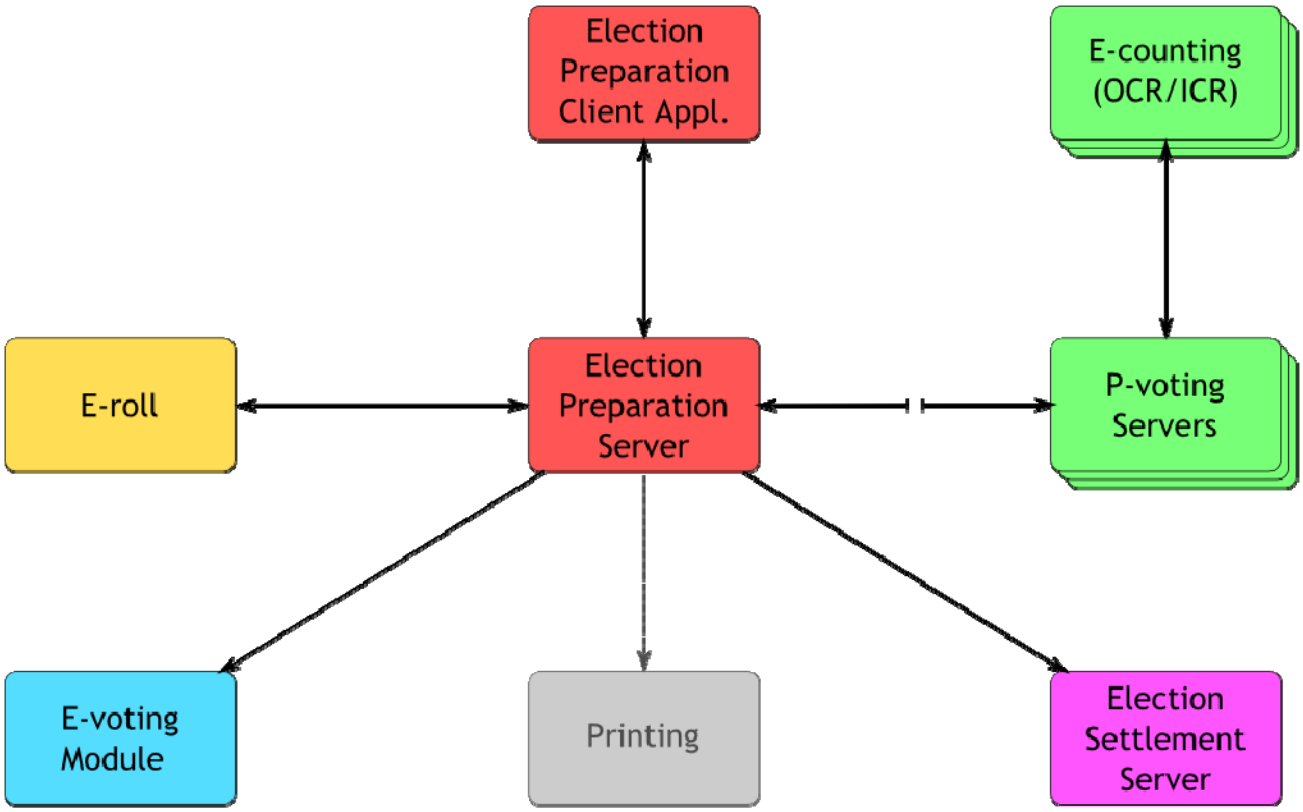


Figure 2 - Election Preparation and P-voting Module Architecture

The solution will be deployed as a thin-client solution building on an existing system currently in use within the U.K. election arena. The current product is used by 50+ U.K. local authorities to manage the workflow of data from submission, through verification and approval and finally on to poll card, ballot paper and postal pack production by selected print suppliers. In 2009, this product managed the preparation of over 4.8 million ballot papers for the European elections in the U.K.

As a thin client solution it will provide the necessary accessibility and usability, include multi-lingual and multi-browser capability and be available to all locations removing any deployment issues.

The system is coded in Java and constructed using the Spring framework. The deployment uses a number of open source technologies and implements a MySQL database at its persistence layer, abstracted by a Hibernate layer. The use of such technologies permits deployment on a number of platforms including Unix and Windows.

In essence the solution is a workflow management tool. Data is captured or imported and progresses through a number definable states, each requiring 1 or more levels of approval if desired and ensuring that any and all changes to the information remain controlled by the one system, being fully audited and thus ensuring ongoing

integrity and authenticity. In addition the existing solution already provides notifications to relevant users at each stage of the workflow process by email and/or SMS.

To facilitate the ICR and signature comparison requirements, the system will be further enhanced to using the same 3rd party image processing engines deployed in another existing solution. This scanning solution reads and compares the date-of-birth and signature entries on postal application forms for 60+ local authorities in the U.K. and has been successful in the detection of fraudulent behavior during a number of local government elections.

The figure below shows the internal architecture of the Election Preparation Server in more detail. Please refer to the chapter Software Stack for more information about the software components and products mentioned in the figure.

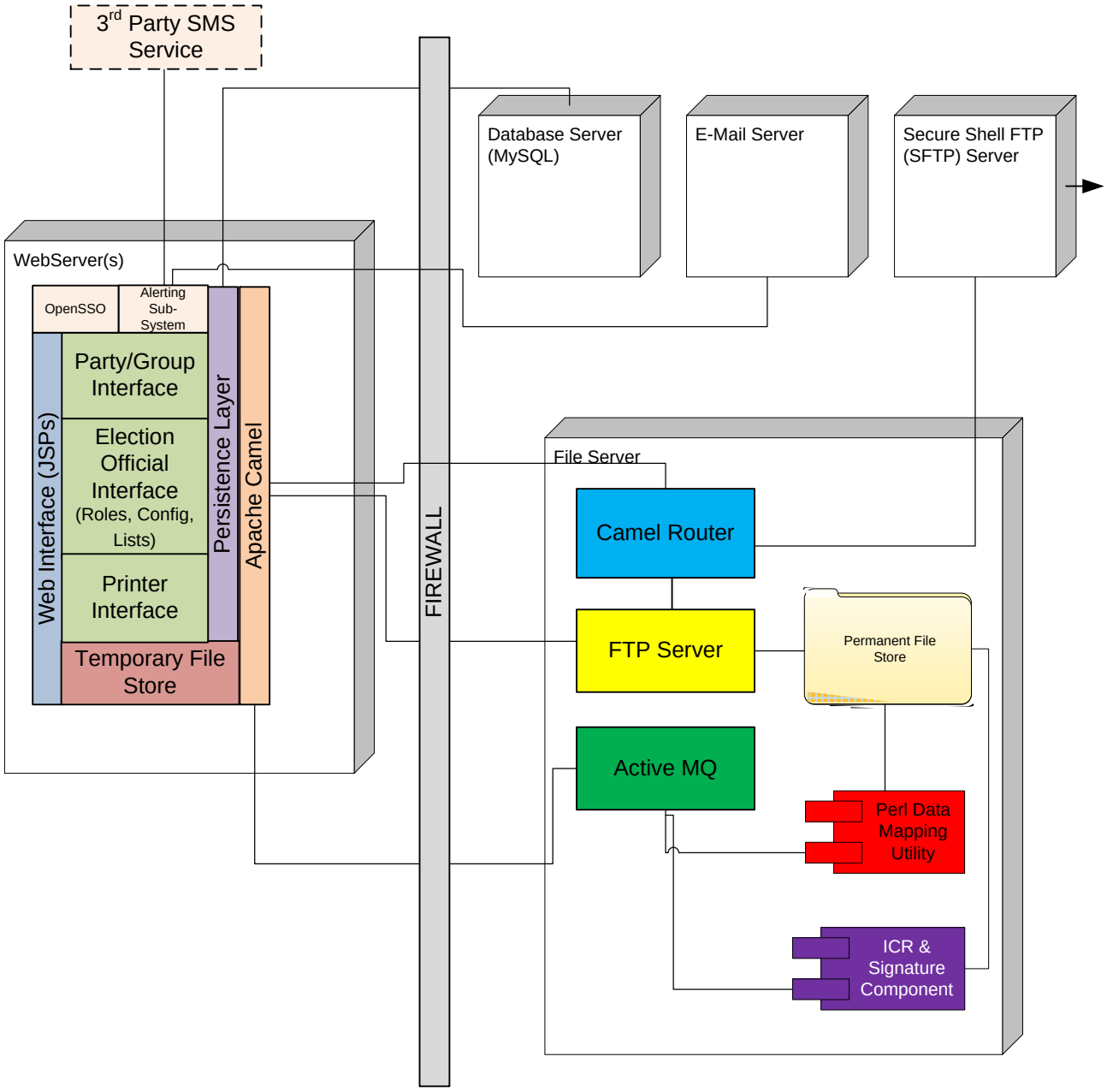
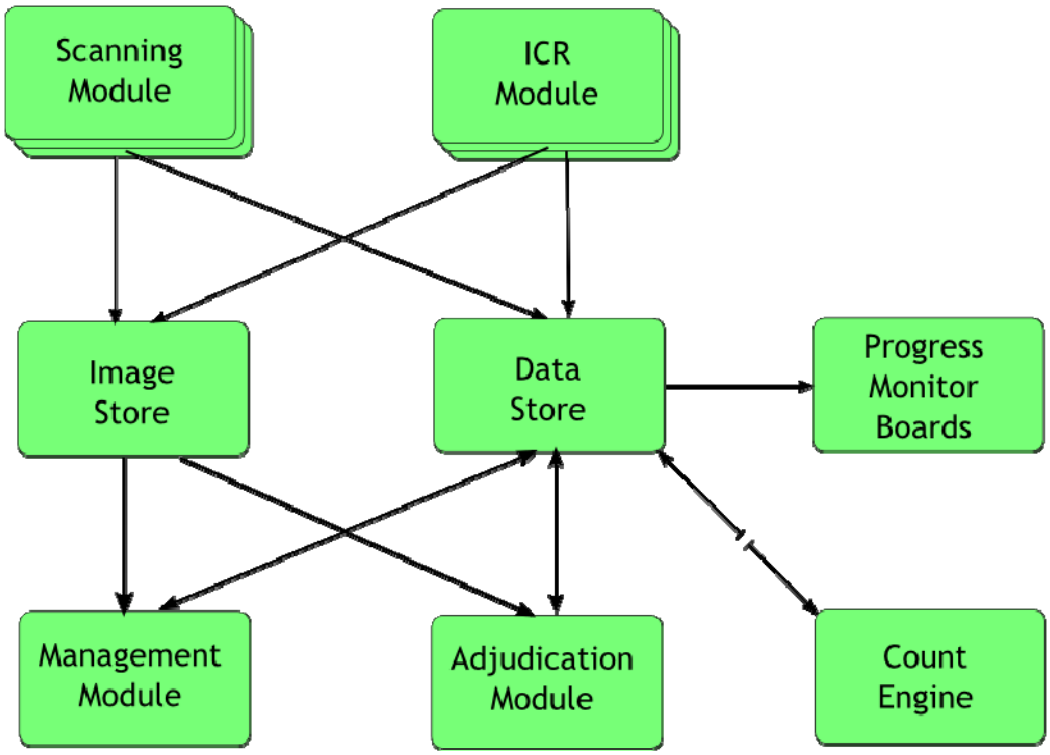


Figure 3 - Election Preparation Server Internal Architecture

2.3 P-voting Module

This section gives an architectural overview of the P-voting module. The following figure shows the components within the P-voting solution and how they interact with each other during the scanning, verification, adjudication and counting phases.



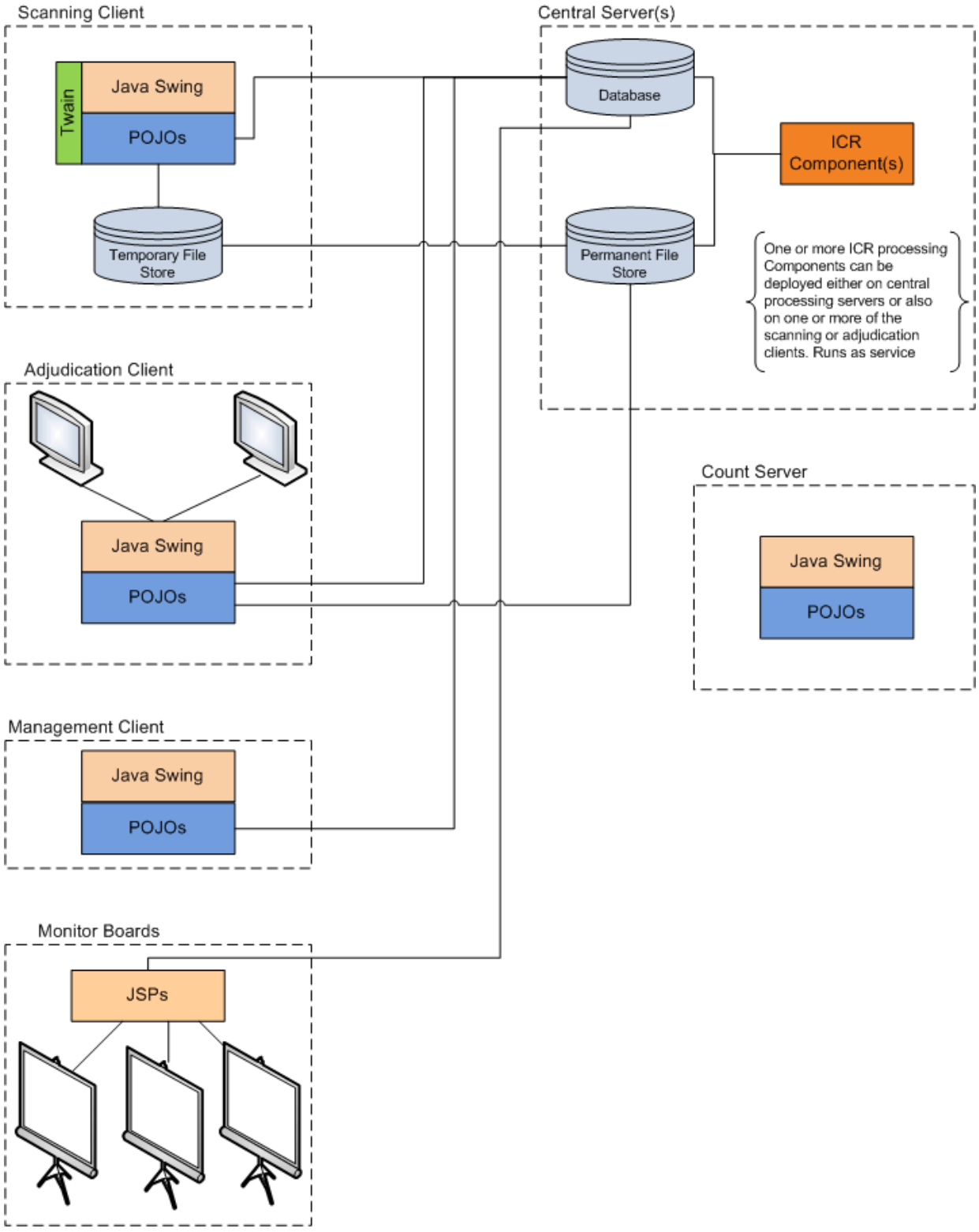
Figur 4 - P-voting Module Architecture

The P-voting solution will be based on an existing product successfully deployed during the 2007 electoral pilots in the U.K. and recently used in a number of local elections within Scotland with further counts in the calendar.

The solution architecture is principally a thick-client deployment maximizing the volume of documents through the scanning hardware, processing of images immediately on capture and allow maximum flexibility by scaling horizontally as required. The system is also designed to run entirely using commercial off the shelf scanning hardware providing the TWAIN protocol is supported.

As the product is developed in Java, it can be deployed on a number of platforms as required and functions entirely on its own air-gapped network for additional security.

The figure below shows a more detailed internal architecture of the P-voting Module. Please refer to the chapter Software Stack for more information about the software components and products mentioned in the figure.



Figur 5 - P-voting Module Detailed Architecture

2.4 E-voting and Election Settlement Module

2.4.1 Introduction

This section gives an architectural overview over the E-voting and Election Settlement Modules. The following figures show the internal technical architecture of both modules, including the air-gapping. These two modules together constitute the core of the Election System, and are therefore subject to strict security requirements, as expressed by the certification requirement.

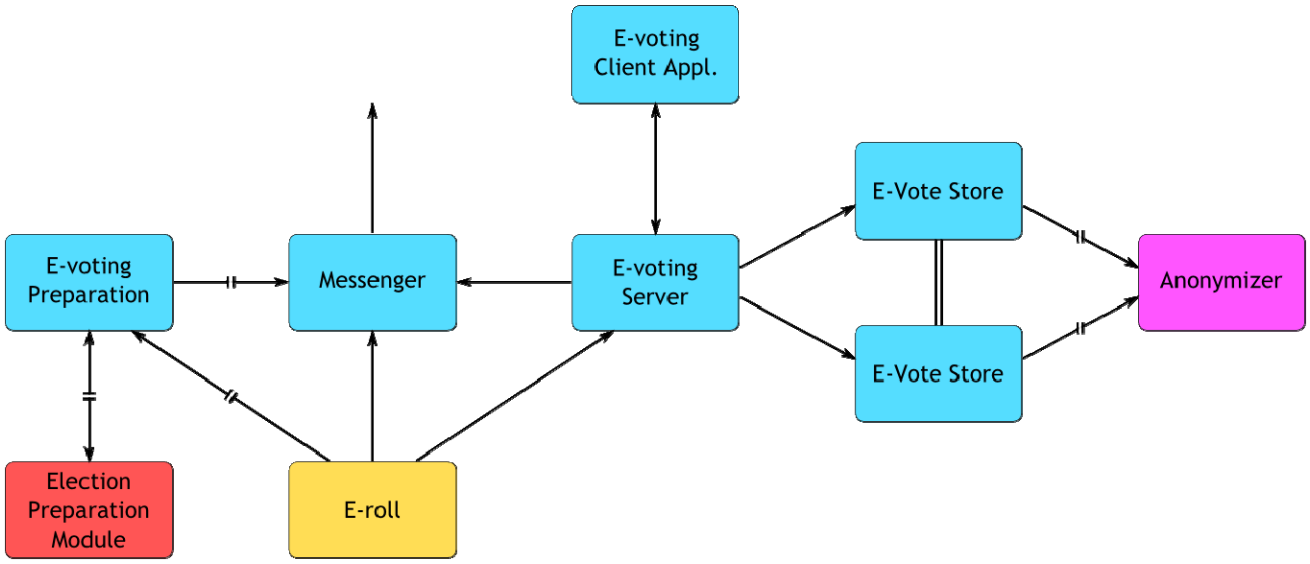


Figure 6 - E-voting Module Architecture

The E-voting Module takes care of the e-voting process. In the center of this module is the E-voting Server, with which e-voters will communicate through the E-voting Client Application. When a voter logs into the E-voting Server, he will authenticate himself with an eID, and then proceed in the E-voting Client Application to fill out his ballot. When he is finished, he submits his ballot to the E-voting Server in a double envelope. The E-voting Server will verify that the double envelope is valid, and send parts of it to the Messenger. The Messenger will construct a check code, and send it back to the voter over SMS. The other part of the double envelope will be sent to the E-vote Store, where the e-votes will be stored in a secure manner. The last component in the E-voting Module is the E-voting Preparation application, an air-gapped computer that will produce all software images for all other components in the Election System.

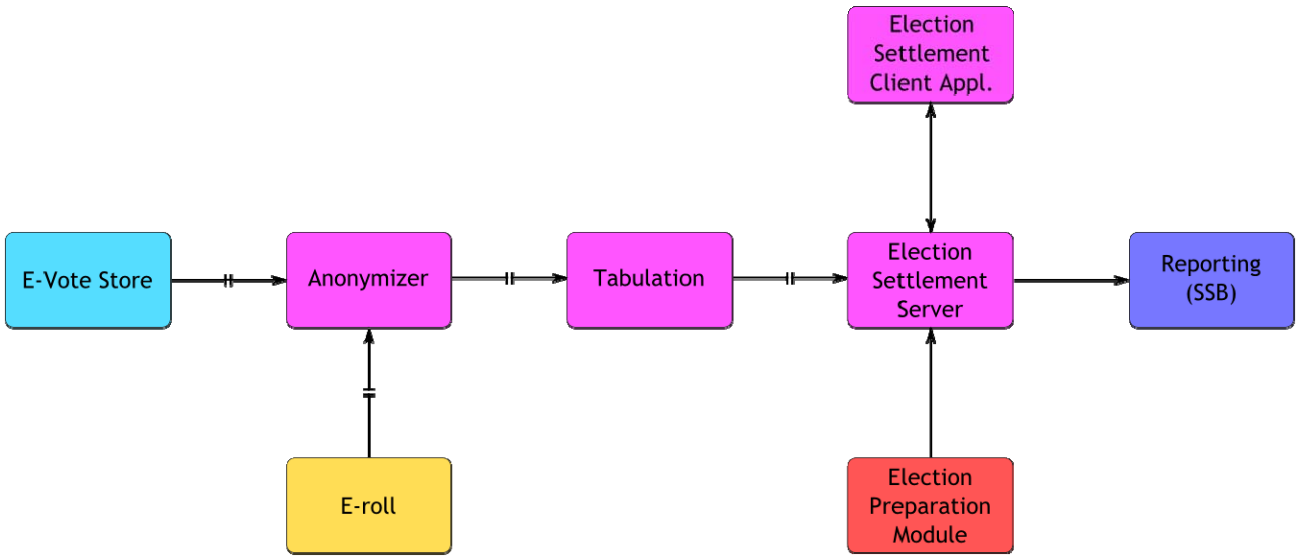


Figure 7 - Election Settlement Module Architecture

The Election Settlement Module consists of an Anonymizer, Tabulation application, an Election Settlement Server and an Election Settlement Client Application. Notice that both the Anonymizer and the Tabulation applications reside on air-gapped computers. When the time has come to count the e-votes, the Anonymizer will receive all e-votes from the E-vote Store and a list of all voters who have voted in the various environments from the E-roll. When the votes have been filtered and anonymized, they are sent to the Tabulation application, where the tabulation is done. When all the results are calculated, they will be sent to the next component in the Election Settlement Module, the Election Settlement Server. Election Officials will have the possibility to browse the results on this server using the Election Settlement Client Application, create and execute reports, protocols, etc. They will also be able to approve the results before they are sent to SSB for further reporting to the media. The Election Settlement Server also receives the results from the P-voting Module through the Election Preparation Module.

2.4.2 General Principles

The E-voting and Election Settlement Modules will be open sourced: Their source code and documentation of the modules will be made publicly available. The protocol specifications and interfaces to other modules use standard tools such as EML, and are therefore also open and publicly available. The cryptographic tools that will be used are published for public review by the cryptographic community. Whenever possible, open source technologies will be used.

These two modules are internally a modular system: the architecture divides the system into components with well defined responsibilities and interfaces. Those components could be implemented by any other party, not just the authors of the architecture. This also leaves room for further protocol development to add features such as mixnets in the future. Security is a process not a product.

The modules are designed with the security requirements in mind. Vote integrity is provided by a novel approach to e-voting. Double-envelope scheme is applied.

The E-voting Module is an autonomous system. It provides its services with input from the Election Preparation Module and the E-roll, but does so without relying on the availability of them. Possible failures in the E-voting Module won't affect the p-voting election procedures.

2.4.3 State Machine

The E-voting and Election Settlement Modules together constitute a state-machine that resembles the generic election process. In each state only certain functionality of the machine is available to the authorized personnel. The state of the machine is protected from unauthorized modification.

Following states are recognized:

- Pre-election stage

  In this stage the E-voting and Election Settlement Module components and election data are prepared by the authorized personnel. The input to this stage comes from Election Preparation Module, E-roll and the E-voting Module source-code repository. Here the configuration for components and the installation packages are prepared.

- Election stage

  In this stage voting occurs. Only minor modifications to configuration under strict control of election officials might be allowed.

- Pre-tallying stage

  Active voting has stopped. In this stage the merging of e-votes and p-votes and anonymization of e-votes to be tallied occurs.

- Tallying stage

  The process of tallying the ballots is executed here.

2.4.4 Cryptographic Protocol

The E-voting Module implements a cryptographic protocol that protects the vote integrity even in the presence of Trojans on the voter's computer, while maintaining easy and intuitive user interfaces.

The security and trust aspects of this scheme have been further elaborated in the paper "Security and Trust for the Norwegian E-voting Pilot Project", presented at the NordSec Conference in Oslo 14 – 16 October 2009. Exact cryptographic details and accompanying proofs are provided in the paper "On Achieving E-Vote Integrity in the Presence of Trojans" that is submitted to the Eurocrypt 2010 conference. Both papers are included to the delivery as attachments to appendix 2A. It is recommended to read them in addition to this document.

The cryptographic protocol is based on the double-envelope principle: the inner envelope contains only the ballot and no information about the voter. The outer envelope is digitally signed by the voter. The inner envelope can only be opened by the tallying sub-process. The double-enveloping scheme ensures integrity and secrecy of the ballot.

The cryptographic protocol uses the following components:

- E-voting Client Application
- E-voting Collection Server consisting of sub modules
  - Collection Module
  - Messenger Module
  - Storage Module

- Tabulation Application

The inner envelope is encrypted using ElGamal public encryption. The outer envelope consists of a digital signature that is produced through an interface that should be provided by eID.

The E-voting Client Application must have access to two public keys – one for the Messenger and one for the Tabulation application. The basic idea of the scheme is that after the double-enveloped ballot is sent to the Collection Module, the Collection Module forwards part of the request to the Messenger which calculates a certain check-code that the voter can use to verify that the ballot he cast also is the ballot that will be stored. The Messenger sends this check-code to the voter via a post-channel like SMS. Note: It is important to understand that those check-codes can be implemented in such a way that neither the Collection Component nor the Messenger gain information about the voter's choice. The scheme uses homomorphic encryption, oblivious transfer and zero-knowledge proofs. Please refer to "On Achieving E-Vote Integrity in the Presence of Trojans" for a detailed specification.

The protocol goes as follows:

Pre-election stage:

- Each voter in the electoral roll receives a polling card with the party names and corresponding 4 digit codes. The polling card is voter-specific.

Election stage:

- The voter votes using the E-voting Client Application. The ballot is cast and an inner envelope is produced by encrypting the ballot with two previously published public keys of the e-voting system. The inner envelope is digitally signed by the voter. The double-enveloped ballot is sent to the E-voting Collection Server.
- The E-voting Collection Server receives the double-enveloped ballot and verifies that the ballot is consistent. If so, a part of it is sent to the Messenger for integrity checking, and another part of the ballot is sent to the Storage Module for storage. The E-voting Collection Server knows the voter's identity, but has no means to see how the voter has voted.
- The Messenger computes a check-code and sends it to the voter via a third channel (e.g. SMS).
- The voter checks the code. If it corresponds to the code of the party his choice, everything is OK. If not, there might be a Trojan on his computer modifying his choices.
- The Storage components receive only those double-envelopes that are approved by the E-voting Collection Server, i.e. those double-envelopes produced by voters who have authenticated successfully to the E-voting Collection Server. The Storage components are the central storage unit where the e-votes are kept until the end of the polling phase. The Storage Module knows all the e-voters, but cannot see how the voters have voted.

Pre-tallying stage:

- The outer envelopes are removed and the inner envelopes are sent to the air-gapped Tabulation application.

Tallying stage:

- The Tabulation application holds the private key that can be used to open the inner envelopes for tabulation. The Tabulation application can decrypt all the ballots, but has no information about the voters who have cast those ballots.

It is important to understand the following:

- This scheme uses the ElGamal cryptosystem. This means that the key used to produce the inner envelope is based on ElGamal, not RSA.
- The use of a third channel is important - it is less probable that the Trojan is in control of both the computer and the mobile phone.
- The fact that all necessary prerequisites hold, i.e. only the Tabulation application has access to the private key etc., must be guaranteed via suitable organizational measures.

The following algorithms and key lengths are used in the cryptographic protocol:

- SHA2 or better for hashing
- ElGamal with key length of 2048 bit or ECC with key length of 224 will be used. These key lengths are considered equivalent to RSA 2048 key length security.

2.4.5 Pre- and Post-Channels

The cryptographic protocol for E-voting with integrity protection introduces the need for two additional channels: a pre-channel and a post-channel.

Via the pre-channel, polling cards with integrity check-codes are sent to the voters. These check-codes are later used to verify that the code returned by E-voting Collection Server via SMS corresponds to the party and candidates the voter intended to vote for.

The implementation of these two channels can be considered to be the extra cost introduced by the integrity security requirement. The pre-channel can be implemented using the polling cards, i.e. the postal service. If the government uses the regular postal service to send election information to voters, then this information could also include the personalized e-voting data. For the post-channel, the E-voting Collection Server depends on the ability to determine post-channel data from the electoral roll.

The post-channel can be implemented reusing information and services from eID/MinID and MinSide. For SMS we need to get hold of the mobile phone number of the voter. MinSide/MinID has this number registered in the profile of most of the users (totally more than 1,500,000 users/citizens are registered). eID is expected to build on MinID. MinSide/MinID has a lookup service for users that returns information including the mobile phone number. eID is expected to have the same service. The E-roll will then be populated with mobile phone numbers using the lookup service or through the initial import of the Electoral roll. When a voter checks that he/she is in the E-roll and the mobile phone number is not included, the voter should be informed to register his mobile phone number in eID/MinID/MinSide. MinSide has also a secure closed e-mail service for secure communication with the citizen. It should also be considered to use SMS and/or Minside e-mail services as a post-channel. Other commercial actors also offer mobile phone numbers in connection with the social security number (fødselsnummer/personnummer), but we recommend using the governmental controlled sources both for cost and security reasons.

Note that it is also possible to vote without receiving integrity check-code, but this would render the integrity protection useless.

2.4.6 Dependence on eID

The E-voting Collection Server and other components of the electronic voting system have certain infrastructural requirements:

- Authentication: There has to exist an authentication method, a way to determine from which person the query comes and whether the person has a right to vote or not.
- Digital signature: There has to be means to digitally sign encrypted ballots so that it can be later said: this encrypted ballot represents, after anonymized decryption, the intent of this person, who has a right to vote.
- Validity: There has to be means to determine whether the credentials used for authentication or giving the digital signature are currently valid.

In Estonia, all these services are provided by the governmental PKI via X509 certificates, RSA keys on ID-card and OCSP protocol. They are the building blocks for the E-voting protocol, and it is assumed that eID will offer an API through which it provides these services.

2.4.7 Communication Policies

All sensitive communication between the components in the E-voting and Election Settlement Module - online or offline - is (mutually) authenticated and also encrypted if necessary. In case integrity has to be preserved - digital signatures are used.

For all on-line protocols, SSL is used as basis. Each component must have a SSL private key and certificate issued by a trusted certification authority. These certificates are distributed between components and only known parties are accepted for communication. Optionally the SSL private keys can be stored inside the hardware security module.

Components that need offline input from other components, shall verify the digital signatures. They will only accept the input when the signatures are correct.

Components that need to output offline information shall sign the information with the private keys of the election officials who initiated the process producing the output.

For digital signatures, the Norwegian eID infrastructure shall be used.

2.4.8 Components

To implement the election process as a state machine and the given cryptographic protocol, the E-voting and Election Settlement Modules contain the following components:

- E-voting Preparation: Offline computer to prepare the e-voting and settlement modules, creating server configurations in a trusted environment and generating check-codes for voters.
- E-voting Client Application: The end-user application to be used for e-voting over the internet.
- E-voting Collection Server: A web-accessible server, that directly communicates with E-voting Client Application.
- Messenger: A firewall protected server responsible for sending integrity check-codes via SMS.
- E-vote Stores: Firewall protected servers that securely store the votes before the tallying stage.
- Anonymizer application: An air-gapped application used to anonymize the e-votes to be tabulated right before the tabulation.

- Tabulation application: An air-gapped application that tabulates the ballots and thus gives the results.

The following sections describe each component in terms of:

- Responsibilities: the exact role of the component in the E-voting System
- Input: the input that the component needs to perform its tasks
- Output: the output that is given by the component
- Other: additional important information about the component
- Optional: optional features of the component

Wherever HSM is mentioned in the rest of this document, we refer to a FIPS 140-2 level 3 validated device, supporting an N of M authentication scheme for private keys.

It should also be noted that all operations are subject to the same role-based access policy as the rest of the Election System.

2.4.8.1 E-voting Preparation

Responsibilities: The E-voting Preparation application resides on an offline computer. This application is among other things responsible for the code-book generation.

Input: The E-voting Preparation application gets its input from the Election Preparation Module and the E-roll. It needs list of candidates, list of voters and their distribution between municipalities.

Output: Signed configuration for server-side components.

Other: There will be only one E-voting Preparation application in the system.

2.4.8.2 E-voting Client Application

Responsibilities: The E-voting Client Application will provide a voter with user interface to participate in the election. The application authenticates the voter, displays a list of available parties and candidates, encrypts and signs the ballot, and sends it to E-voting Collection Server (to Collection Module to be more precise).

Input: The E-voting Client Application is programmed to be able to participate in the cryptographic protocol with the Messenger and the Tabulation application. It receives the list of parties and candidates from E-voting Collection Server.

Output: The E-voting Client Application sends double-enveloped ballot to E-voting Collection Server according to the cryptographic protocol.

2.4.8.3 E-voting Collection Server

Responsibilities: The E-voting Collection Server is responsible for the voter authentication and checks against the electoral roll. The server distributes list of parties and candidates to the client application, collects the double-enveloped ballots from the client application and verifies signatures on them. The server performs the validity checks on the certificates, verifies the zero-knowledge proofs, and participates in the oblivious transfer protocol with Messenger and the storage protocol with the E-vote Stores.

Input: The E-voting Collection Server is configured with a list of voters, parties and candidates. It also needs a code seed table containing a random number for each voter–party and voter–candidate number pair. Except for

the double-enveloped ballots coming from the E-voting Client Application, all the information comes from the E-voting Preparation computer via DVD-ROM.

Output: The E-voting Collection Server sends the ballots to E-voting Stores and the check-data to the Messenger. The E-voting Collection Server sends digitally signed responses to the E-voting Client Application.

Other: The E-voting Collection Server is connected directly to web on the one side, and routed through a firewall at the other. The E-voting Collection Server is characterized by its read-only state - nothing will be stored there, making it easy to swap in case of trouble. There can be more than one E-voting Collection Server in the system for load balancing.

Optional: The E-voting Collection Server could use cryptographic accelerator card in order to speed-up the election process. In order to increase efficiency the E-voting Collection Server could be separated into two subcomponents: one responsible for serving clients, the other responsible for the cryptographic protocol. For SSL key storage a HSM could be used whereas one HSM could be shared by several E-voting Collection Servers.

2.4.8.4 Messenger

Responsibilities: The Messenger participates in the cryptographic protocol with the E-voting Collection Server and sends integrity check-codes to the voters via SMS.

Input: The Messenger receives a mapping between voters, their mobile phone numbers and randomized check-codes from the E-voting Preparation application via DVD-ROM. The Messenger accepts on-line requests only from the E-voting Collection Server only.

Output: The Messenger sends SMS messages to the voters.

Other: The Messenger is characterized by read-only configuration - nothing will be stored here, making it easy to swap in case of trouble. The Messenger has an HSM protected private key. It is enough to have one Messenger in the system.

Optional: The Messenger could use an HSM for SSL key storage.

2.4.8.5 E-vote Stores

Responsibilities: The E-vote Stores accept double-enveloped ballots for storage from the E-voting Collection Server. The E-vote Stores maintain the integrity of the votes, and they provide a list of voters who have cast an e-vote.

Input: The E-vote Stores need a list of voters from the E-voting Preparation application. The E-vote Stores accept double-enveloped ballots from the E-voting Collection Server.

Output: List of p-voters and the double-enveloped electronic ballots.

Other: There will be 2 E-vote Stores in the system, and they will be kept synchronized.

Optional: The E-vote Stores could use an HSM for SSL key storage.

2.4.8.6 Anonymizer Application

Responsibilities: The Anonymizer application is an air-gapped application that is responsible for removing the outer envelopes from the e-vote ballots.

Input: The list of double-enveloped ballots stored inside an E-vote Store and the list of voters who have cast a p-vote.

Output: List of encrypted ballots to be tallied.

Other: The Anonymizer will be completely offline. It could be implemented as a LiveCD prepared by the E-voting Preparation application.

2.4.8.7 Tabulation Application

Responsibilities: The Tabulation application is an air-gapped application that is responsible for the correct calculation of the e-election result from to the list of encrypted ballots.

Input: List of parties and candidates per constituency through the E-voting Preparation application and List of encrypted ballots from the Anonymizer.

Output: Election results to be imported into the Election Settlement Server.

Other: The Tabulation application has a private key stored in the HSM. The Tabulation application will be completely offline. It could be implemented as a LiveCD prepared by the E-voting Preparation application.

2.4.9 Trusted Execution Environments

Some security requirements, such as the capability to ensure that only audited code gets executed, can only be achieved if the server-side components of the E-voting and Election Settlement Module have access to a trusted execution environment such as tamper-resistant cryptographic coprocessor with capability to store code and data. Security critical code will be loaded digitally signed into this environment and executed there. A remote attestation protocol will be executed to ensure that the state of the environment is unaltered. Only approved functionality will be executed.

Each server-side component of the E-voting and Election Settlement Modules must have access to a trusted execution environment in order to guarantee the state of the system.

2.4.10 Security Aspects

2.4.10.1 Vote Confidentiality

Vote confidentiality is achieved through a public key encryption scheme. The inner envelope is created using a public key encryption method (ElGamal) on the ballot in the e-Voting Client Application. The inner envelope can only be opened by the owner of the corresponding private key, in our case the Tabulation application. No information about the voter's identity will be inside the inner envelope.

2.4.10.2 Vote Integrity

Vote integrity is achieved by the double envelope scheme. The inner envelope is digitally signed by the voter using eID. This digital signature is carried along with the encrypted ballot and is verified during major steps in the election process. In the E-vote Stores, the vote is stored together with the outer envelope, but it is removed by the Anonymizer before the tabulation process starts.

2.4.10.3 Vote Anonymity

The anonymity of the vote through-out the system is guaranteed by technical and cryptographical methods. There is no place on the server side where the voter's identity and the clear-text ballot can be linked with each other. The only part of the system where a clear-text ballot and voter's identity are available at the same time is in the E-voting Client Application. On the server side the vote is stored securely, i.e. in a double-enveloped manner. The components that can see the voter's identity will only see double-enveloped ballot and cannot open the inner envelope. The Tabulation application can open the inner envelopes, but will receive anonymized votes, i.e. with the outer envelopes carrying the digital identities removed. Decrypted ballots contain no information that will link them back to the actual voter. The decryption can be initiated only when N out of M election officers are present.

2.4.11 User Roles

The off-line components of the E-voting and Election Settlement Modules will use the role system of underlying operating system. The user and roles will, however, be managed in the Election Preparation Module, and exported from there when the components are installed on the respective servers. Wherever necessary, additional mechanisms can be implemented upon the operating system level, e.g. to provide additional authentication schemes.

2.4.12 Availability

The E-voting Module will be designed with high-availability requirements in mind. All of the components, except the E-vote Stores, are easily replaceable in case of failure. For that reason there will always be two synchronized E-vote Stores at the system at all times. In case of failure in one component a replication procedure can be started.

The high-availability features in the system architecture support the use of organizational measures and high-availability hardware:

- All server components must have RAID controllers.
- All server components must have doubled power supply units.
- Authentic back-up configurations must exist at all times.
- Private keys must be backed up or redundant HSM's must be installed.
- Different Internet Service Providers should be used for connecting the the E-voting Collection Server to outside world.

2.4.13 Performance

The proposed algorithm for integrity protection relies on the E-voting Collection Server’s ability to effectively verify zero-knowledge proofs created by the voters. These proofs introduce additional computational overhead compared to a straight-forward encrypt-and-sign e-voting solution. On the other hand, if one wants to protect the electronic voting system against large-scale client domain manipulations, this is the price that has to be paid.

A non-optimized proof-of-concept implementation of the integrity protection protocol using ElGamal key lengths of 2048 bits (which offers similar security to RSA 2048 bit length keys as stated by security

requirements) was tested. A simulation was run using 32 choices, which would be equivalent to party choices in the real system.

The E-vote Collection Server, which proves to be the bottleneck of the protocol, was executed on a platform capable of doing approximately 400 exponentiations per second. On this platform the E-voting Collection Server was able to serve 56 queries per minute. This adds up to 3360 queries per hour.

This shows that when the E-vote Collection Server would have access to a cryptographic accelerator capable of doing 6000 1024 bit RSA-operations per second, we could reach a performance of 800 queries per minute or 48,000 queries per hour. With simple optimizations a performance of 900 queries per minute (54,000 queries per hour) could be achieved. In addition, the E-vote Collection Server is a component using read-only devices only, which makes it very scalable. One can therefore have several E-vote Collection Servers in the system.

In the test, the Messenger could serve up to 2,500,000 queries per hour with the help of a similar HSM as the E-vote Collection Server. Tabulation is even faster.

During the Estonian 2009 local government elections, approximately 106,000 e-votes were collected, and 104,313 of them were tabulated. The statistics shows that the interest on e-voting is highest on the first and on the last hour of the election. The minute peak was 107 queries. The hour peaks was approximately 4500 queries. 100 e-voters also p-voted, and approximately 2000 e-votes were automatically revoked because the voter e-voted once more. The vast majority of the e-voters used the e-voting system only once.

Apart from the first and last hour, the load on the voting system divided evenly over the 7 voting days. Saturdays and Sundays were quieter, but the daily pattern repeated itself there as well: the most active e-voting hours were between 20.00 and 22.00.

2.5 Auditing Module

2.5.1 Introduction

The auditing module consists of two components, an Auditing Server and an Auditing Application. The Auditing Server is a server protected by a firewall, and provides a time-stamping service for the components that need secure logging capability. It will provide a secure log based auditing trail to the system. The Auditing Application is an on-line application within the secure perimeter of the Election System, used to monitor the election process. In addition, audit daemons will have to be run on all the Election System server-side components that require auditing. These daemons will be configurable and able to filter data and create warnings depending on the circumstances.

2.5.2 Auditing Server

The Auditing Server will implement a secure log, based upon a digital time stamping system to audit events that need an auditing trail. Digital time stamping based on binary linked schemes enables the cryptographic connection of events in a way that there exists a fixed order between events, no events can be removed from the timescale afterwards, and no new events can be inserted between two other events. The server will be an on-line component using an HSM to store its private key. There will be only one Auditing Server in the system.

A scientific paper called "Time-Stamping with Binary Linking Schemes" (Buldas, Laud, Lipmaa, Villemson) explaining the technology is attached to Appendix 2A in the delivery.

2.5.3 Auditing Application

Election officials will be able to use the Auditing Application to actively monitor what's happening on the server-side components, perform integrity checks, analyze the election process, configure daemons and work with data received from them. It will have functionality to execute a self-checking protocol to ensure that all system components are functioning as intended.

2.6 E-roll

The figure below details the E-roll architecture. The module consists of an E-roll Server, which serves three applications: the E-roll Administration Client Application, used by the Electoral Committee to update the electoral roll, the E-roll Polling Station Client Application used at the polling stations to registers who has voted, and the E-roll Self-Service application where voters can check whether they are registered correctly in the electoral roll, and from where they can send in applications.

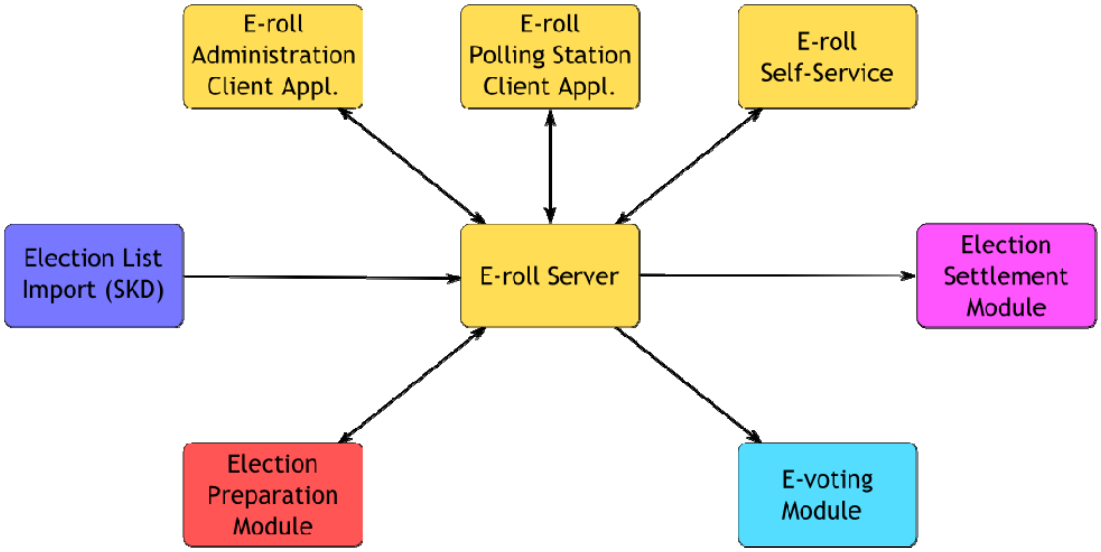


Figure 8 - E-roll Module Architecture

The E-roll Server receives its data from the Election List Import component, which reads and handles the files coming from SKD. The E-roll Server also offers its services to the Election Preparation Module, e.g. in order to check whether candidates on a party list are registered on the electoral roll. The E-roll Server also provides the electoral roll to the E-voting Module, not as a direct service, but through database replication. Finally, the E-roll Server provides data about which voter has voted on which media and in which environment to the Election Settlement Module.

3 Infrastructure

3.1 Environments

To successfully develop, test, train people, accept, deploy and maintain the full Election system, several separate environments must be established and operated. The Tenderer proposes that the following environments are set up:

- Development environment
- Test/Reference environment
- Training environment
- QA environment
- Production environment

More details on specific software and hardware for the different environments will be prepared during the specification phase. These specifications can then be used as input for the Customer towards its' operational partner for preparation of the operational environments that the Customer is responsible for. Which environments the Customer is responsible for, are described below.

3.1.1 Development Environment

The Tenderer will be the owner, the main user and responsible for the Development environment. Development will be done on Tenderer's PCs with the Tenderer's development licenses.

The system will be developed and unit tested within the Development environment. This environment will not reflect the real Production environment due to cost and resource reasons.

The Developer environment will include developer tools (Integrated Developer Environments like Eclipse), unit testing tools and either local (or centralized databases) for development and unit testing.

The Development environment will include common shared components that are not tied to users in a specific environment, but that also will be shared with the Test/Reference environment:

- Version control system: Subversion
- Bug tracking: JIRA
- Wiki: Confluence

3.1.2 Test/Reference Environment

The Tenderer will (also here) be the owner, the main user and responsible for the Test/Reference environment.

The Test/Reference environment is where the Tenderer will do all the integration and system tests. The tests done in this environment will be mostly functional. The Tenderer will also perform the final System Test in this environment.

The Test/Reference environment will be a down scaled variant of the QA and Production environments.

3.1.3 Training Environment

The Customer will be the owner, the main user and responsible for the Training environment. This is where the customer training activities are performed. This environment can also be used to provide demos and partial testing of the different sub deliveries for the Customer.

The Training environment can be equally equipped as the Test/Reference environment. The Tenderer will assist the Customer's operational partner in the set up of this environment.

3.1.4 QA Environment

The Customer will be the owner, the main user and responsible for the QA environment.

The QA environment is where the Customer performs its' acceptance test, and based on the results of this, approves whether the system is ready for deployment to the Production environment or not.

The Tenderer will also use this environment to run the necessary performance, scalability and security tests that must be carried out for verification and tuning before accepting and deploying the final system.

The QA environment should be equally equipped as the Production environment. The Tenderer will assist the Customer's operational partner in the set up of this environment.

3.1.5 Production Environment

The Customer will be the owner, the main user and responsible for the Production environment.

The Production environment is where the tested and approved system will be installed, base lined and prepared for Election configuration.

3.2 Run-Time Infrastructure

The illustration below outlines the main components in the run-time infrastructure of the Election System:

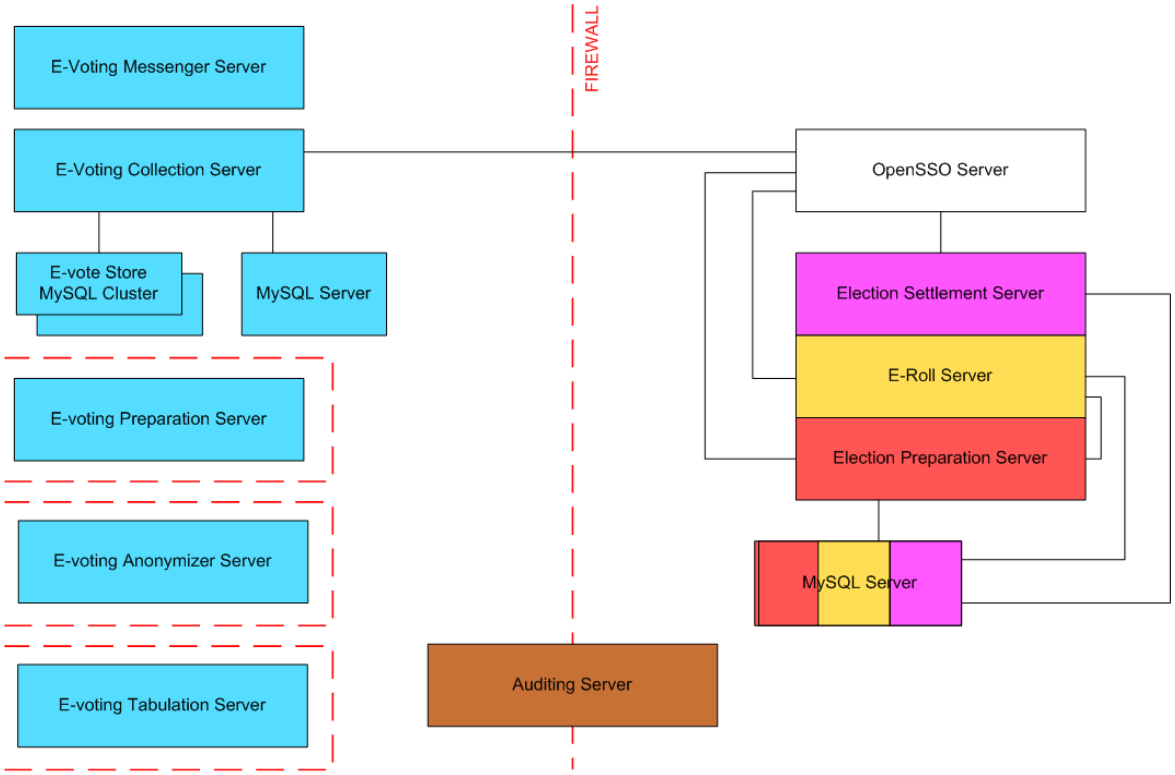


Figure 9 – The Run-time Infrastructure of the Election System

Note that the run-time infrastructure sketched above is due to changes during detailed design. More details on the infrastructure hardware and software will be specified then – and for each of the proposed environments (See Environments above). These specifications can be used as input for the Customer towards its' operational partner for preparation of the operational environments that the Customer is responsible for (See Environments above).

3.3 Networking

The following modules will be air-gapped, and therefore have no direct access to internet:

- E-voting Preparation application
- Anonymizer
- Tabulation application

The following modules will be behind a firewall:

- Messenger
- E-vote Stores
- Auditing Server
- Auditing Application

The Messenger additionally needs access to an SMS service to send out the check-codes to the voters.

The following component will have a connection to both internet and to the modules behind firewall:

- E-vote Collection Server

3.4 HSM

The following components need a dedicated HSM and a trusted execution environment:

- Tabulation application
- Messenger
- Auditing Server

This can e.g. be implemented with 3 Thales/nCipher nShield Connect 6000 HSM (with SEE and ISO smartcard feature activations). If multi-party computation is to be used for cryptographic protocol, then 2 additional HSM units are necessary.

The following components can share a trusted execution environment:

- Configuration application
- E-vote Collection Server
- E-vote Stores
- Anonymizer
- Auditing Application

This could be implemented with e.g. 2 Thales/nCipher nShield Connect 6000 HSM (with SEE activations), but the Tenderer proposes that a decision on this is postponed until we have a more detailed overview over the concrete requirements for these modules.

The E-vote Collection Server needs acceleration for the public key cryptography. The nCipher HSM has functionality to accelerate the cryptographic operations.

3.5 High-Availability

The following modules will need high-availability hardware:

- Messenger Module
- Storage Module
- Collection Module

3.6 Alerting Infrastructure (SMS and E-Mail)

The Election System will be able to provide status alerts or system notifications in two formats: e-mail and SMS. For the voting protocol, it is the opinion of the Tenderer that SMS seems to be the most appropriate channel to send the out-of-band check codes back to the voter.

SMS can be provided by a 3rd party interface with a variety of permissible protocols, including SOAP and HTTPS. This interface is easily replaceable to permit other suppliers or an internal implementation. The Government has several SMS services related to high volume/high availability common services such as eID/minId, Altinn and Minside. The Tenderer assumes that one of these services and agreement for services could be used by the Election System. To provide e-mail alerts a mail server with SMTP access will be required.

4 Technologies

This chapter gives an overview over some of the most important and basic technologies used in the modules and the components of the Election System.

4.1 Election Markup Language (EML)

Election Markup Language (EML) is an XML-based standard to support end to end management of election processes. The communication between the various modules and components in the Election System relies heavily on EML. The figure below illustrates this, showing the numbers of all the EML Messages that flow between the modules.

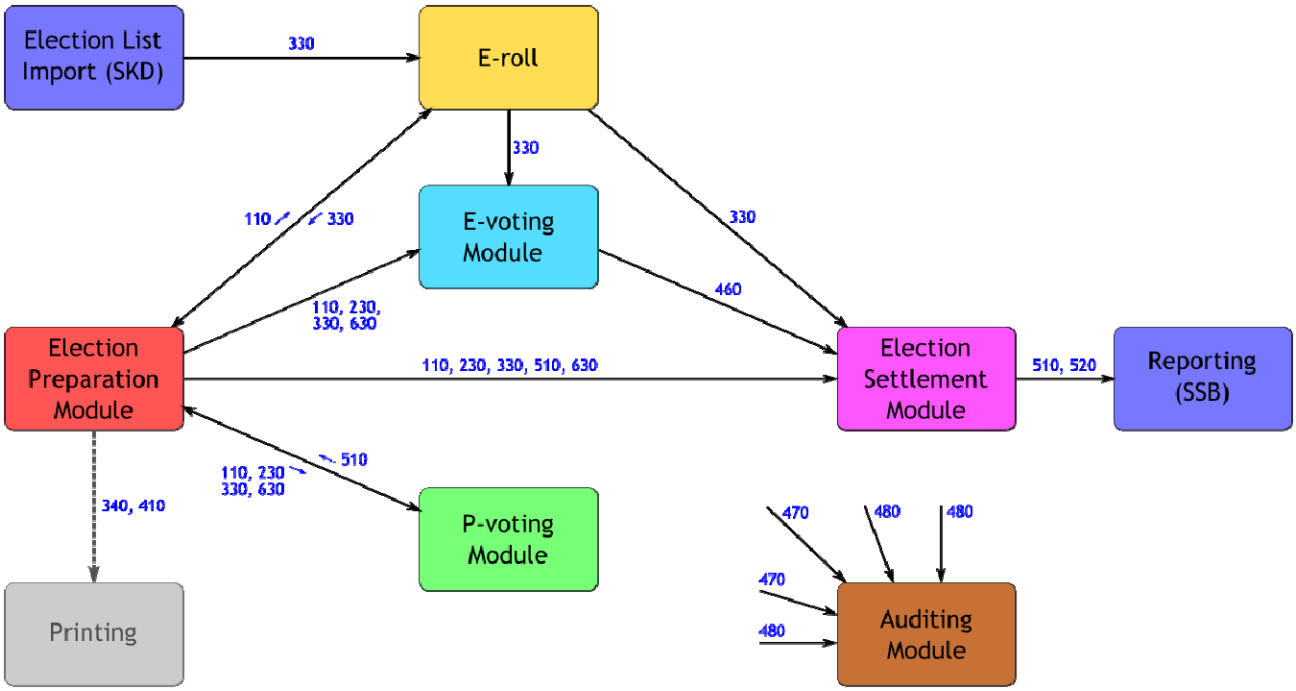


Figure 10 - EML Message Flow in the Election System

As an example of how these EML messages can be used, the figure below illustrates how the E-voting Client Application and the E-voting Server communicate with each other.

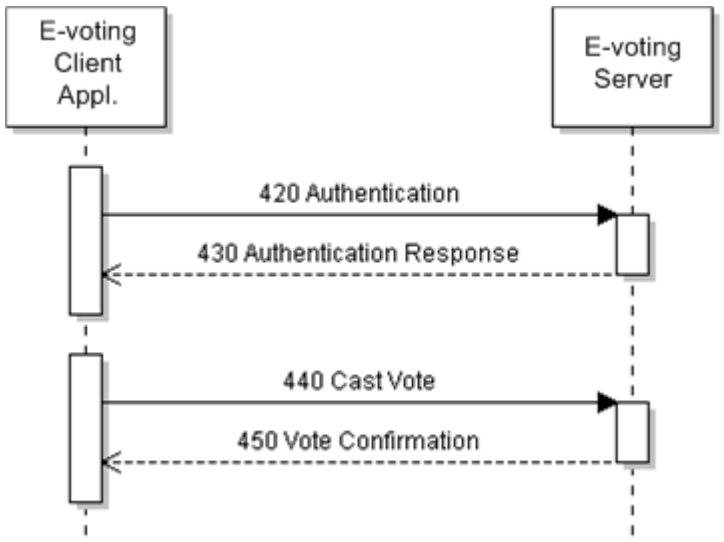


Figure 11 - EML messages between the E-voting Client Application and the E-voting Server

4.2 Representational State Transfer (REST)

The client applications that use GWT (ref 5.5) in the Election System, and in particular the E-voting Client Application, use a client/server communication model which, thanks to the EML domain model, almost naturally leads to an architecture based on representational state transfer (REST). This architecture makes it also easier to describe and understand the communication and API between the client and the server, and decouples the client and the server from each other to a larger degree than would be the case with an RPC (Remote Procedure Call) based architecture. This means also that this architecture makes it easier to allow multiple clients to be developed against the same E-voting Server using the same API, if the Principal would wish to do so.

4.3 Secure Sockets Layer (SSL)

Secure Sockets Layers (SSL) will be used extensively in the Election System. All communication with the Internet will be protected by SSL, i.e. all the client applications in the Election System and the integration points with other systems (SKD and SSB). SSL will also be used internally wherever appropriate, so that components will communicate securely and authenticate each other.

4.4 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) will be used extensively to style client application user interface elements, like e.g. fonts, colors, positions etc. This makes it easy to implement another skin for look and feel if required, but the software will be delivered with only one set of cascading style sheets.

5 Software Stack

This chapter gives an overview over the different software components that will be used to build the Election System. In general, the Tenderer has tried to reduce the number of different software components and products used, in order to minimize the complexity of the development. Many of the software components and products will therefore be used in many of the modules.

5.1 Solaris 10 with Trusted Extensions

The various components of the Election System can run on different operating systems. The Tenderer proposes to use Solaris 10 with Trusted Extensions[1], certified for Common Criteria EAL4+, throughout the solution, both in the production and the test environment. One of the few exceptions to this will probably be the ICR software, as explained in the next section. During the development phase, other operating systems may be used, both on the server side and on the developers' workstations.

5.2 Windows Server 2003

At the moment, some of the 3rd party components require another operating systems than Solaris 10 with Trusted Extensions. An example of this is the ICR software that runs on Windows only. In those cases, other operating systems, like Windows Server 2003[2] will have to be considered and used.

5.3 MySQL

MySQL[3] is a popular open source database, and will be used throughout the Election System. It works on many different operating systems, including the Solaris 10 with Trusted Extensions. Both a command-line interface and GUI-tools are available for administration, migration, querying etc. In addition, several other commercial and non-commercial tools integrate with MySQL.

5.4 GlassFish

GlassFish[4] is an open source application server. It uses a derivative of Apache Tomcat as the servlet container for serving web content, with an added component called Grizzly which uses Java NIO for scalability and speed. In addition, the Jersey framework will be used to build RESTful Web Services in Java. JAXB (Java Architecture for XML Binding) is used for easy mapping of EML message to and from Java objects both on the client and server side.

Special care should be taken for the set-up of the application server of the E-voting Server. The Tenderer proposes that GlassFish V3 be used, since it is possible to trim that version down to minimum of features and resource usage. This will be beneficial both from a security and a performance perspective.

[1] Refer to http://www.sun.com/software/solaris/trustedsolaris/ for more information about the Trusted Solaris Operating System.
[2] Refer to http://www.microsoft.com/windowsserver2003/default.mspx for more information on Windows Server 2003.
[3] Refer to http://www.mysql.com/ for more information about MySQL.
[4] Refer to http://glassfish.dev.java.net/ for more information about GlassFish.

5.5 Google Web Toolkit (GWT)

Google Web Toolkit[5] (GWT) is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. It will be used to create the user interface of the E-voting Client Application, the E-roll Polling Station Application and the E-roll Self-Service. The toolkit adheres to the WAI-ARIA specifications, the WAI specification relevant for rich Internet Applications.

5.6 Spring Framework 2.5

Many of the client interfaces will be developed as JSPs and Java objects built on the Spring framework[6]. As an example, the figure below shows how this framework is structured within OPT2VOTE's existing solution for the Election Preparation Module.

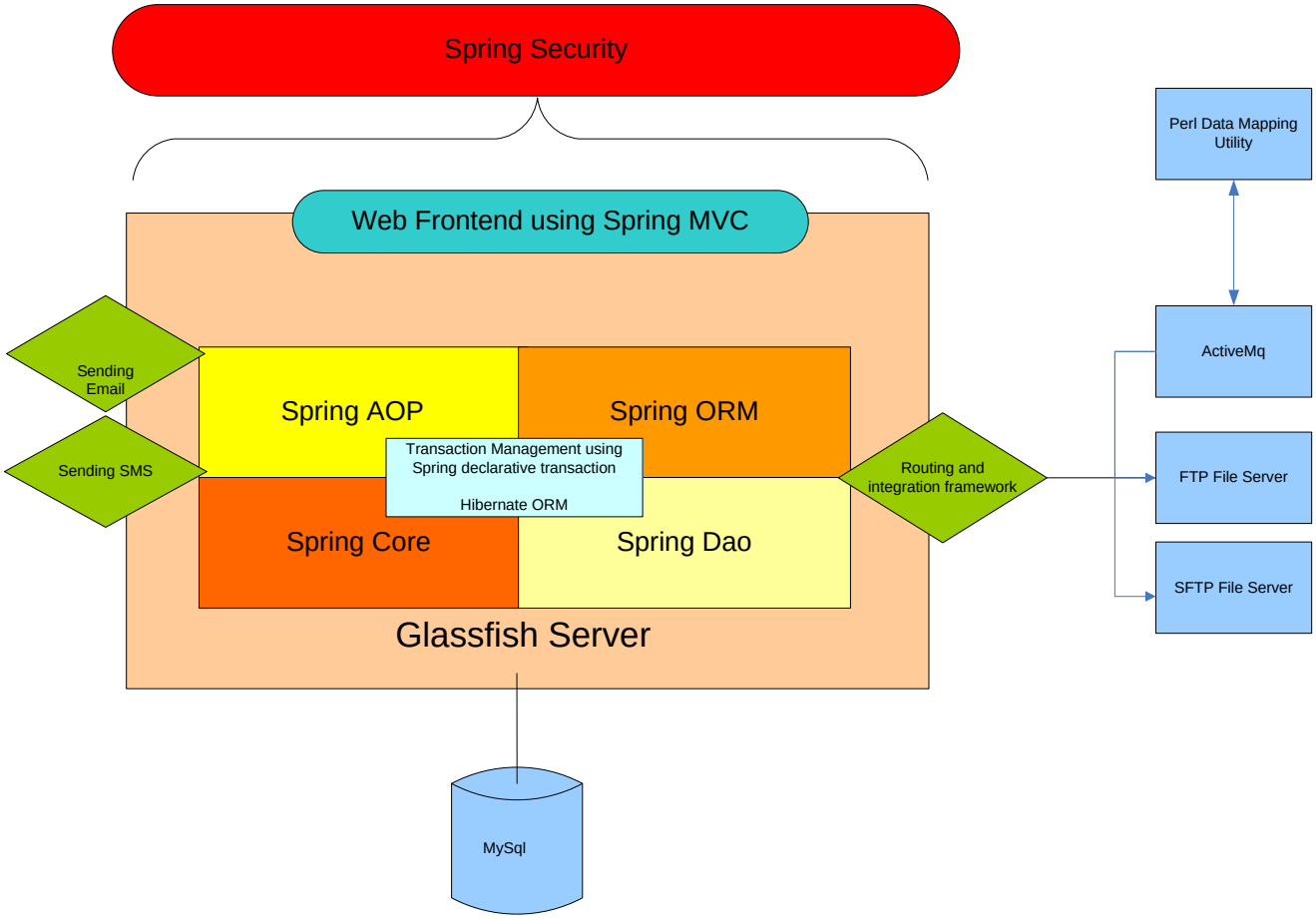


Figure 12 - Spring Framework in the Election Preparation Module

[5] Refer to http://code.google.com/webtoolkit/ for more information about Google Web Toolkit.
[6] Refer to http://www.springsource.org/ for more information about the Spring framework.

5.7 Hibernate

Hibernate[7] is an open source object/relational mapping (ORM) toolkit that relieves the need to make direct use of the JDBC API. The Hibernate layer provides abstraction between the application and persistence layers.

5.8 OpenSSO

OpenSSO[8] is an open source access management and federation server platform. It provides single sign-on capability with integrated support for SAML, and will be used to integrate with eID.

5.9 Apache Camel

Apache Camel[9] is an open source routing and integration framework. It facilitates the definition of rules controlling the routing of data files across the system from submission onto the temporary file store, through persistence within the permanent file store and copied onwards to the SFTP distribution server. This software product will be used primarily in the Election Preparation Module, but will be used in other modules too when appropriate.

5.10 ActiveMQ

Apache ActiveMQ[10] is an open source message broker which fully implements the Java Message Service (JMS). This product will be used in the Election Preparation Module to manage all communications between the primary administration application and the data mapping utility providing seamless, fully automated processing of multi-format data files into a standardized format for printers.

5.11 OpenSSH

OpenSSH[11] is an open source version of the SSH (Secure Shell) connectivity tool, facilitating the transfer of files internally throughout the solution in a secure way.

5.12 DataMapping Utility

The DataMapping Utility is a Perl application that manipulates multiple file formats controlled by various configuration files into standard output formats suitable to forward onto printers. The responsibility of data formatting therefore resides with the application and not printer.

5.13 Mitek Quickstrokes & Mitek Signprotect

Mitek Quickstrokes and Mitek Signprotect are 3rd party ICR and signature encoding libraries respectively, to be used in the Election Preparation Module. The implementation of these components will be loosely coupled allowing libraries to be replaced in the future if required. The ICR component will be deployed as a service, and can be deployed either on the scanning clients or centrally on the database server or dedicated image processing server(s).

[7] Refer to https://www.hibernate.org/ for more information about Hibernate.
[8] Refer to http://opensso.org/ for more information about OpenSSO.
[9] Refer to http://camel.apache.org/ for more information about Apache Camel.
[10] Refer to http://activemq.apache.org/ for more information about Apache ActiveMQ.
[11] Refer to http://www.openssh.com/ for more information about OpenSSH.

5.14 Gears

Gears[12], formerly called Google Gears, is a piece of software offered by Google that adds new features to a web browser, including a LocalServer module, which caches and serves application resources, and a Database module which can store data locally. The software is free and open source software, released under the BSD license.

The LocalServer module is a specialized URL cache that a web application can control. Requests for URLs in the cache are intercepted and served from the user's local disk. This mechanism makes it possible to run web applications locally when the computer is off-line.

The Database module is based on SQLite, an embedded relational database management system that is contained in a relatively small programming library. Its source code is in the public domain. There are no special limitations on how large the database can grow other than the theoretical limit of 32 TB[13]. In practice, the available bandwidth needed to synchronize data and the size of the disks is the real limitations.

5.15 JasperReports

JasperReports[14] is an open source Java reporting tool. It can write to screen, to a printer or into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values (CSV) and XML files.

JasperReports can be used in Java-enabled applications, including Java EE or Web applications, to generate dynamic content. It reads its instructions from an XML or .jasper file.

The JasperReports engine also allows for report definitions to include charts, with the rendering provided by the JFreeChart library which supports many chart layouts, such as Pie, Bar, Stacked Bar, Line, Area, Scatter Plot, Bubble, and Time series.

JasperReports can merge multiple data sources for easy usage in reports. Scriptlets may accompany the report definition to provide hooks for additional processing in reports. The scriptlets is built using Java. JasperReports also allows for creation of re-usable sub reports that can be used in several different main reports.

5.16 Swing

Swing[15] is a widget toolkit for Java. It is part of Java Foundation Classes, the API for providing a graphical user interface for Java programs. The P-voting solution is a thick client Java implementation using Swing.

5.17 Morena

Morena[16] is an image acquisition framework, used by the P-voting solution. It operates through TWAIN to drive the scanning hardware.

[12] Refer to http://gears.google.com/ for more information about Gears.
[13] The maximum size of each database page is 32 kB, and the maximum number of pages in a database is 1024^3.
[14] Refer to http://jasperreports.sourceforge.net/ for more information about JasperReports.
[15] Refer to http://java.sun.com/products/jfc/tsc/articles/architecture/ for more information about Swing.
[16] Refer to http://www.gnome.sk/Twain/jtp.html for more information about Morena.

6 Development Tools

This chapter gives an overview over the development tools that the Tenderer intends to use during the development phase. Note that when the need arises, or when better alternatives become available, these tools will be considered.

6.1 Eclipse

Eclipse[17] is a multi-language software development environment comprising an IDE and a plug-in system to extend it. One of the plugins that will be used in Eclipse is the Google Plugin for Eclipse[18], to develop the various GWT applications in the Election System. Eclipse will be used as the main tool to produce Java code during the development phase.

6.2 Maven

Apache Maven[19] is a software tool for Java project management and build automation. It will be used to compile all Java code in the project and maintain the dependencies on 3rd party libraries.

6.3 Hudson

Hudson[20] is an extensible continuous integration server that will be used to build all the software artifacts during the development phase. It ensures that software changes do not affect core functionality by running automated regressions tests after the build process. It can also be set up to generate parts of the system documentation on a regular basis based on the current state of the source code.

6.4 Nexus

Nexus[21] acts as a configurable proxy between the build processes and the public Maven repositories, and can also assist with the deployment of internally generated artifacts. It isolates the build processes from the availability of external Maven repositories, thus ensuring the stability in the developers' team.

6.5 Subversion

Subversion[22] (SVN) is an open source revision control system. It is used to maintain current and historical versions of files such as source code, web pages and documentation.

6.6 Testing Frameworks

Testing frameworks will be used to automate the task of unit testing, regression testing, performance testing, et Appendix 5 lists a set of tools that will be used during the project with their specific goals and tasks.

[17] Refer to http://www.eclipse.org/ for more information about Eclipse.
[18] Refer to http://code.google.com/eclipse/ for more information about the Google Plugin for Eclipse.
[19] Refer to http://maven.apache.org/ for more information about Apache Maven.
[20] Refer to https://hudson.dev.java.net/ for more information about Hudson.
[21] Refer to http://nexus.sonatype.org/ for more information about Nexus.
[22] Refer to http://subversion.tigris.org/ for more information about Subversion.

6.7 Code Quality Tools

In addition to the testing frameworks, a combination of code quality tools will be used throughout the project to ensure the quality of the source code. Sonar[23] is an open source code quality management platform, dedicated to continuously analyze and measure the source code quality. Another tool is PMD[24], which scans the source code and looks for potential problems, possible bugs, unused and suboptimal code, over-complicated expressions and duplicate code. CheckStyle[25] helps programmers write Java code that adheres to a particular coding standard, the default being Sun Code Conventions. Finally, Jalopy[26] will be used to format all source code, such that it streamlined across the whole project.

6.8 C/C++ Development Tools

If performance should become an issue, especially in the E-voting or Election Settlement Module, C or C++ will be considered to implement parts of the functionality. Standard GNU developments tools will be used, including `gcc`, `g++`, `automake`, `autoconf`, `libtool`, with `vi` as the standard editor.

6.9 nCore API and Codesafe Development Kit

The nCore API and Codesafe development kit will be used to develop the software components interfacing with the HSMs.

[23] Refer to http://sonar.codehaus.org/ for more information about Sonar.
[24] Refer to http://pmd.sourceforge.net/ for more information about PMD.
[25] Refer to http://checkstyle.sourceforge.net/ for more information about CheckStyle.
[26] Refer to http://jalopy.sourceforge.net/ for more information about Jalopy.

7 Added Headings for Elaboration

This section covers the request by KRD made in the SSA-U Appendix 2B Requirements table document for elaboration of specific requirements.

7.1 Fallback Solution (E-roll)

The E-roll Polling Station Client Application will use Gears so that the web application can still be available locally on the client PCs in the polling stations in the event of a loss of communication. The application will be set up such that data registered locally during the down-time will be synchronized automatically to the E-roll Server. When the communication with the E-roll Server goes down, the user at the polling station will be notified, and when the communication is restored, the user will be notified again.

As to limit the data volume that has to be synchronized between the local PC and the E-roll Server, the Tenderer proposes that only the Electoral Roll for the local municipality will be stored locally. This means that when communication with the E-roll Server is lost, the functionality of the E-roll Polling Station Client Application will be limited to the following:

- Voters from registered to the local polling station can still be handled, and all information will automatically be synchronized when communication is restored.
- Voters from the same municipality but belonging to other polling stations, and voters from other municipalities during the advance voting period, can be registered with their name and ID (*personnummer*) when they want to vote in the polling station. They will also get a unique number assigned, that should be written down on the cover envelope together with their name, and the unique number will be stored in the local database. When the communication line is restored, information about all voters from outside the municipality will be synchronized to the server.

7.2 Open Source

7.2.1 Elaboration of Requirement MC2

The ownership of copyright and other related rights for software components developed for the Customer is transferred to the Customer in line with clause 10.1.1 of the main agreement.

Any software developed by the Contractor, including standard software, shall be open source. The term «open source» is limited to mean that the source code shall as a minimum be made available to the Customer.

The core system, i.e. those parts directly involved in e-voting or counting of e-votes or that otherwise use, store or manipulate election data where no paper audit trail exists, shall be open source. However, third party closed source standard components are allowed in the core provided certification requirements, as set out below, are met.

With the exception of third party closed source standard components, the core system components must as a minimum be licensed to allow the Customer to make the source code available to the public and allow anyone to copy, modify, inspect, compile, debug and run the core for testing purposes.

With the exception of third party closed source standard components, the license to all system components must as a minimum allow the Customer or anyone the Customer authorizes to copy, modify, inspect, compile, debug, run and make available the source code and derivative works based on the source code for use in academic research or for further development of the system for use in Norwegian elections.

To avoid doubt, for third party standard components, the requirement for open source or a relevant certification applies only to core software components.

The open source requirement and the certification requirement does not apply to third party software used for backup, monitoring or similar operational tasks. Nor does the open source requirement and the certification requirement apply to third party ICR/OCR-software, as a paper audit trail will exist.

Database software that uses, stores or manipulates election data, cryptography software and card-reader software is considered a part of the core, and thus the open source requirement or the certification requirement for third party closed source standard components applies.

The operating system on which the system runs shall either be licensed under a generally recognized open source license that is accepted by The Open Source Initiative or meet the same certification requirements as set out below for closed source core components.

The following software modules and related documentation from Computas and Cybernetica will be made available for the customer as open source:

- E-Roll
- E-voting
- Election Settlement server

In addition, the following software modules and related documentation from OPT2VOTE will be made available to enable the customer to do further development for Norway:

- Election preparation
- P-voting
- E-Counting

All open source software is public.

For software modules from OPT2VOTE, the Tenderer agrees the that core directly involved in the e-counting, will be licensed to allow the Customer to make the source code available to the public and allow anyone to copy, modify, inspect, compile, debug and run the core for testing purposes.

7.2.2 Elaboration of Requirement MC3

It must be possible for anyone to compile the open source components of the core system, but not necessarily to set up a complete e-voting system.

The Source code for the E-voting Client Application and the E-votes Counting Module will include scripts and instructions – in good Open Source manner - so that anyone with competence (e.g. programmers) – in addition to review - can compile it. This does not mean that anyone can setup and test these core parts because they are

dependent on other components in the Election system, databases, authentication etc.

7.3 Scope for Deliverables and Implementation

7.3.1 Elaboration of Requirement GR3.6

The Principal must have an option to acquire software licenses for full scale implementation if this is not included in the software delivered for the pilots in 2011. The Principal prefer to have a license for unlimited use in Norway. But if such license is not offered, the Contractor must base pricing e-counting on 200 scan-centers with the average of 3 scanners (600 scanners totally).

The Tender agrees to offer the Principal the option to acquire software for a full scale implementation. The Tenderer has supplied pricing for the Principal to acquire software licences for both the 2011 pilots and also a full-scale implementation based upon the assumption of 200 scan centers.

7.4 Implementation Environment of the System

7.4.1 Elaboration of Requirement MC6

The operations partner is yet to be decided. The Contractor must describe the conditions for running the system as defined by the requirements.

The Principal's operational partner must be able to operate servers with Solaris operating system with trusted extensions. In addition they must be able to operate GlassFish application servers and MySQL databases. More details on specific software and hardware for the different environments will be prepared during detailed design. These specifications can then be used as input for the Principal towards its' operational partner for preparation of the operational environments that the Principal is responsible for (See Environments above).