
Vesa Kaikkonen

# Improving project delivery of eInvoicing projects

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

2 June 2025

PREFACE

I started thinking about a topic for my thesis in summer 2023, in fall 2023 after I started my studies I had written an essay of 3 pages about that topic, so I had all planned out and it remained like this since I started working at Apix Messaging Oy in January 2024. The company provided me with a few real topics which might help our company and our team. So, it made more sense to do a thesis about one of those topics.

During the year 2024 writing has been progressing slowly. My idea was to write a thesis during working hours, since the topic was heavily related to work, during spring and early summer there were unexpected changes in resourcing which meant that I had to leave writing to later time. During fall the resource situation took turn to better and I had more time to focus on writing. Now, almost 12 months after work was started the thesis is done!

What did I learned while writing my master's thesis? Writing takes quite a long time; you should start as soon as you can. I also learned a lot about the areas investigated in Apix systems and processes.

Finally, I would like to thank my co-workers for their support and comments regarding topics. Special thanks go to the family, which enabled my studies during the time of quite hectic moments of childcare.

Espoo, Finland, 02.06.2025
Vesa Kaikkonen

## Abstract


| | |
|---|---|
| Author: | Vesa Kaikkonen |
| Title: | Improving project delivery of eInvoicing project |
| Number of Pages: | 49 pages |
| Date: | 2 June 2025 |
| Degree: | Master of Engineering |
| Degree Programme: | Information Technology |
| Professional Major: | Networking and Services |
| Supervisors: | Aapeli Takala, Consulting Manager |
| | Aarne Klemetti, Researching Lecturer |


---


The purpose of this study was to investigate ways to improve working in eInvoicing project in project team. The study describes the current state of project delivery processes, ways of working, message handling system and tools used when building project in case company and clarifies reasons behind this study.

The theoretical background gives a brief look at topics related to eInvoicing in general, defining eInvoicing, standards used in eInvoicing, legislation in Finland, also briefly looking at the history and the future of eInvoicing.

The study elaborates target processes of mentioned areas. Target processes is a list of ideas of functions which could further help organization to achieve more flexible and reliable environment. In the future some of these ideas are taken into drawing board.

As a result of the thesis, programs that will help setting up project and remove manual works were identified. More in depth planning was done to one of the programs. That program will remove manual work from initializing phase in project implementation.



Keywords: eInvoicing, ways of working, improvement


---

# Contents

# List of Abbreviations

| | |
|---|---|
| ACL | Addressing and Capability Lookup |
| AIBSL | Association Internationale Sans But Lucratif |
| API | Application Programming Interface |
| CEN | Comité Européen de Normalisation / European Committee for Standardization |
| channel_in | Channel with specific components used when receiving data to Incus |
| channel_out | Channel with specific components used when sending data from Incus |
| CI/CD | Continuous Integration, Continuous Delivery |
| EDI | Electronic Data Interchange |
| EDIFACT | Electronic Data Interchange for Administration, Commerce and Transport |
| eInvoicing | E-Invoicing, e-invoicing, electronic invoicing |
| ERP | Enterprise Resource Planning |
| ETL | Extract Transform Load |
| Incus | Electronic messaging platform |
| ML | Machine Learning |
| PEPPOL | Pan-European Public Procurement On-Line |
| Peppol BIS | Peppol Business Interoperability Specifications |
| PKI | Public Key Infrastructure |
| PowerBi | Microsoft Power BI is an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence |
| SFTP | Secure File Transfer Protocol |
| SML | Service Metadata Locator |
| SMP | Service Metadata Publisher |
| UAT | User Acceptance Test. Data is send to users for verification |
| YTJ | The business information system |
| XML | Extensible Markup Language |

# 1 Introduction

The purpose of this study is to investigate how project delivery could be standardized and automated, so that it would be more predictable, easier to schedule, to have more quality and less manual work and human errors. First, it begins by explaining materials and methods used in this study and investigate eInvoicing landscape generally. Second is a very important step as current processes are analysed for their possible problems relating project delivery by gathering data from employees and the system itself. Finally, it proposes target processes, selects area for improvement and lastly implements program selected and planned.

**Case Company**

Thesis is done for Apix Messaging Oy. Apix is a messaging operator which acts as operator for eInvoicing, eSalary, and Electronic Data Interchange (EDI). Apix offers electronic business transactions to handle orders, order confirmations and invoices. Services will cover data transformation, routing and distribution to operator networks or to physical channels. Cybersecurity is also considered, Apix acts according to ISO-27001. [5] [28]

Apix was founded in 2010, and from 2021 Apix has been part of Accountor Group. Apix is part of Accountor Groups FMS-cluster whose software's are used in over 3000 accounting offices serving over 200 000 businesses in Finland, Nordic and Baltic countries. [5]

Apix act as Access Point (AP) for Pan-European Public Procurement On-Line (PEPPOL) network and is also connected with Danish eInvoicing infrastructure NemHandel. [5]

Apix Messaging Oy key figures [8]:

- 34 employees (2023)
- 8 EUR million revenue (2023)

- 8,2 EBIT %

**Business challenge and Objectives**

The challenges behind this thesis are mostly the complexity of the message handling system used, lack of documentation for processes of project delivery, ways of working and tools used to create a project. There are no clear ways of working how actual project work is supposed to be carried out. Lack of overall working instructions on how different parts of project related work should be done correctly is causing delays and human errors. Lack of tooling is also causing issues, delays and uncertainties, because people do not know how to use existing tools properly or there are no tools which could help with certain tasks. The team is doing tasks manually which again increases the risk of unnecessary errors.

The objective of this thesis is to analyze and document, define target processes and pick certain areas which are chosen to be improved to defined state. Areas to investigate are project delivery process, ways of working, message handling system and tools that are needed to help with creating projects.

This study aims to reach the following goals:

- analyze and document current message handling system, project delivery processes, ways of working and tools used.
- define target process of previously mentioned areas.
- select areas which to improve to defined target process.
- create tools which help to achieve automation and better quality in project deliveries.

**Scope of study**

This study will include investigation of the current setup for sales invoices, and how implementation is done in the most standard way in the current system used with existing infrastructure. This study will not include deep infrastructural investigation of the current system. It will also not have a user-friendly User

Interface (UI), as it is mostly used from terminals. UI presented is mostly for visualization.

**Structure**

This thesis has been divided into 9 sections.

- Section 1 introduces the subject of study, case company, business challenges and objectives of the study.
- Section 2 contains methods and materials used in the study. Also, a research design graph is presented.
- Section 3 represents theoretical background. Section consists of defining eInvoicing, eInvoicing standard, explaining the standard, legislation of eInvoicing in Finland. Section also explains the short history of eInvoicing and glance for the future and lastly Peppol.
- Section 4 introduces current state analysis, explains the process flow of the message handling system.
- Section 5 looks at target processes of selected areas.
- Section 6 introduces selected areas for improvement.
- Section 7 planning of the selected component.
- Section 8 contains implementing the tool.
- Section 9 represents conclusions. The section summarises the research and implementation and covers the conclusions and results of the project. It also analyses any successes or failures including recommendations on improvements and what can be worked on in the next phase.

# 2 Method and Material

This section introduces materials and methods used in this study.

### Research Design

Research design consists of background investigations, solution development, solution testing and closing with conclusions. Background investigations include current state analysis and technical background as shown in Figure 1.

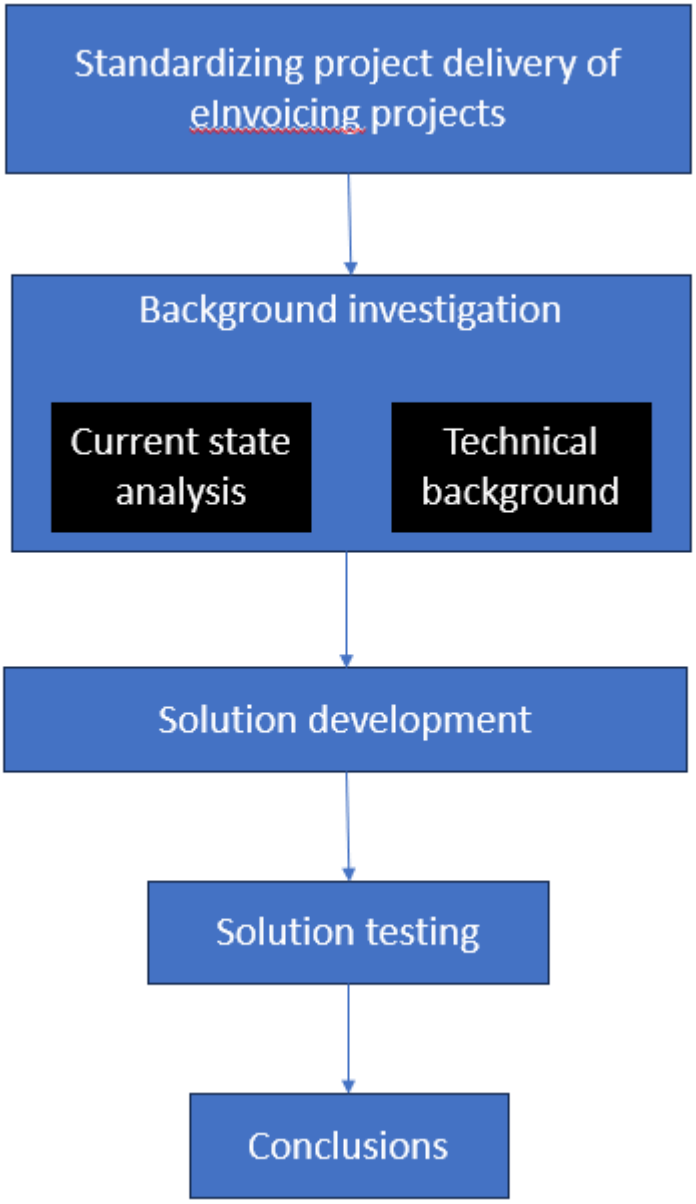


Figure 1. Research design.

**Research materials**

Materials for research are gathered from discussions with colleagues, solving problems in customer cases, current system, various databases and from internet. Company production databases are used to gather information of most used components, which are later used as background information when deciding which components are standard for the project.

# 3 Theoretical Background

This section introduces theoretical background of the study. It gives definition of eInvoicing, explaining it as the electronic exchange of invoices between buyers and sellers using structured data formats like Peppol Business Interoperability Specification (BIS) or Finvoice.

Next, outline the eInvoicing standards, focusing on the European standard EN 16931-1:2017, developed by the European Committee for Standardization (CEN). This standard defines a semantic data model for core invoice elements, enabling interoperability across systems and countries.

The section then discusses the legislation of eInvoicing in Finland, particularly law 241/2019. It mandates that companies must send eInvoices to public sector entities, and from April 1, 2020, recipients can reject non-electronic invoices if they’ve declared a preference for eInvoices.

Following this, the history of eInvoicing is presented, tracing its roots back to the 1960s with the emergence of EDI. It covers key developments through the decades, including the introduction of EU directives and the rise of eInvoicing platforms, leading up to the digital transformation of the 2020s.

The section continues with a look into the future of eInvoicing, predicting that Peppol will become the dominant format. It also anticipates the integration of emerging technologies like AI, IoT, and blockchain to enhance automation, security, and transparency in invoicing processes.

It then introduces Peppol, an international network and standard launched in 2008 to streamline electronic document exchange. Managed by OpenPeppol Association Internationale Sans But Lucratif (AISBL), Peppol supports the secure and standardized exchange of business documents across borders. The section also describes the implementation of Peppol in Finland, where the State Treasury acts as the Peppol Authority.

Finally, the Peppol network architecture is briefly explained. It is based on a four-corner model involving senders, receivers, and their respective service providers.

**Defining eInvoicing**

eInvoicing, e-Invoicing or electronic invoicing, sending/receiving electronic invoices via electronic channels between business parties (buyers and sellers) in a structured data format, such as Peppol BIS or Finvoice. According to qvalia “eInvoicing enables organizations to create, send, and manage invoices electronically, ensuring the exchange of billing documents is more secure, efficient, and faster”. [1]

**eInvoicing standard**

Electronic invoicing European standard was formally published along with supporting technical specifications and reports in 6 parts by the European Committee for Standardization (CEN) on 2017. Part 1 of standard is named “Semantic data model of the core elements of an electronic invoice” EN 16931-1:2017. [2]

**Explaining the standard**

A main objective of the eInvoicing standard is to enable eInvoice sending between organizations by using single structured format message. This removes the need for system adjustment with individual trading parties. To accomplish this, the standard defines semantic data model that lists business terms how they should be used in compliant invoices. By using semantic data model messages can be processed automatically. [3]

**Legislation of eInvoicing in Finland**

Law that applies on the electronic invoicing of procurement units and traders in Finland is 241/2019. Law consists of 5 paragraphs.

5. paragraph has extra clause which concerns paragraph 4, clause states that paragraph 4 will be applied starting from 01.04.2020. Paragraph 4 states in short, ”You must be ready to send invoices as online invoices if your customer is a company, municipality, or the Finnish state. The recipient of the invoice does not have to pay invoices received in other ways if they have declared their recipient's invoices as an online invoice.” [4]

**History of eInvoicing**

History of electronic messaging started already back in 1960s when first EDI messages were sent.

- 1990s: Large companies started to implement EDI-systems between business parties.
- 2000s: e-invoicing has gradually been implemented in the legislative framework In Europe, Directive 2001/115/EC was one of the first steps to modernize the legislation based on improving VAT obligations and electronic formats.
- 2010s: eInvoicing platforms started to emerge and be more usable. Platforms began integrating financial software, providing automation and be more popular between all sized companies.
- 2014: EU passed directive 2014/55/EU, which authorized commission (CEN) to create European standard for eInvoicing.
- 2020s ->: eInvoicing entered another phase where digitalization and cloudification has been picking up speed. Finland legislation 241/2019 paragraph 4 was activated 01.04.2020, which requires companies to send eInvoices whose customer is company, municipality, or the Finnish state. [1] [4]

**Future of eInvoicing**

Challenges that have been around from early 2000s, having lot of different file formats which different companies would use, are still here but nowadays Peppol is starting to rise above others. Peppol format is predicted to be default format for private and public sectors.

Technology wise we might see very interesting development of eInvoicing in form of Internet Of Things (IoT), Artificial Intelligence (AI), Blockchain. Predicting that we will see more secure, smarter, faster solutions. Blockchain Smart contracts could create major leap how business transactions are conducted and making transactions more secure and transparent. Digitalization and automate processes will spread across to world. [1]

**Peppol**

Peppol is an international network and standard, which purpose is to enhance electronic invoicing and business processes between public and private sector procurements. Peppol was founded 2008 as a joined collaboration European commission and European countries. It will provide common and standardized communication protocol enabling exchange business documents between different countries and organizations. With the help of Peppol network, sellers and buyers can exchange electronic business documents such as catalogs, orders, order confirmations, despatch advices and invoice directly from system to system. [21] [22] [23]

Peppol is maintained by OpenPeppol AISBL. Association is responsible of defining, service development, maintaining and executing rules and regulations of Peppol network. OpenPeppol is non-profit international organization which is located in Bryssels Belgium and acts under Belgian law. OpenPeppol consists of central organization OpenPeppol AISBL, national authorities and national service providers. [21] [22] [23]

**Peppol in Finland**

Central government introduced Peppol network 1.4.2024.
Peppol Authority in Finland is State Of Treasury, which promotes Peppol network usage in Finland and supports and oversees operating national Peppol service providers. State Of Treasury also monitors the development of national legislation, coordinates training and increases competences and offers Finnish companies the opportunity to participate and influence the development of the

international Peppol network. State Of Treasury states that “The development work is carried out openly and closely with stakeholders”. [21]

**Peppol network**

Peppol network relies on four-corner model. It consists of four participants, sender, sender service provider, receiver service provider and receiver.
Network contains four core elements, APs, Addressing and Capability Lookup (ACL) previously known as Service Metadata Publisher (SMP), Service Metadata Locator and Public Key Infrastructure (PKI). [21]

**Access Point**

APs are Peppol certified service providers that connect each other through ACL. Each participant is registered to ACL with document types what they can send and receive. Sending AP are required to validate outgoing messages before sending. Message compliance is verified with Peppol BIS. Validation tools are provided within the Peppol BIS. [7] [21]

**Addressing and Capability Lookup**

Peppol certified Service Providers offer ACL services, which let buyers and suppliers publish their electronic addresses and supported message types like a digital business registry. These services can be provided independently or alongside AP services. [21]
Peppol Authorities can choose between:

- A decentralized model with multiple ACL providers, or
- A centralized model with a single provider.

**Service Metadata Locator**

Service Metadata Locator (SML) defines which ACL an AP needs to connect with to discover the addressing details of any trading partner in the Peppol Network. This approach is like how the internet can find websites based on their domain

names. The Peppol SML is a single centralized service provided by OpenPeppol. [21]

**Public Key Infrastructure**

Peppol uses PKI to ensure secure and trusted communication across its network. When a service provider becomes Peppol certified by signing the Service Provider Agreement, they receive a digital PKI certificate. This certificate authenticates their identity and secures message exchanges. It remains valid as long as the agreement is active but can be revoked if the provider violates the terms, ensuring that only trusted entities can operate within the Peppol network. [21]

Shown in Figure 2 is simple order example to demonstrate four-corner model. Buyer wants to order product from seller. Buyer sends order message to their service provider, which then forwards it to seller service provider which again forwards order message to sellers Enterprise Resource Planning (ERP)-system. [24]

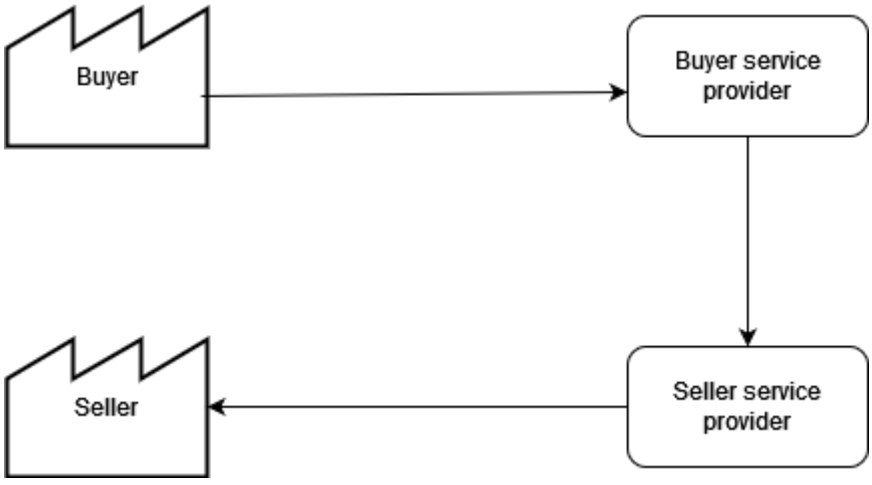


Figure 2. Example of ordering process in peppol network.

# 4 Current processes

This section introduces current processes of message handling system, project delivery processes, ways of working and tools used when creating customer project. First the Message Handling System, section describes the flow of incoming data and the various processing stages it undergoes before being delivered to the recipient. Next, section describes the Ways of Working from a project engineer's perspective. This section also introduces the Project Delivery Process going through different phases of the process. Lastly, the section outlines the various tools used in project creation.

**Message handling system**

Currently used Extract, Transform, Load (ETL)-system is very complex, this section gives brief overview of it. [9]

When message is received, it goes through message handling system shown in Figure 3.

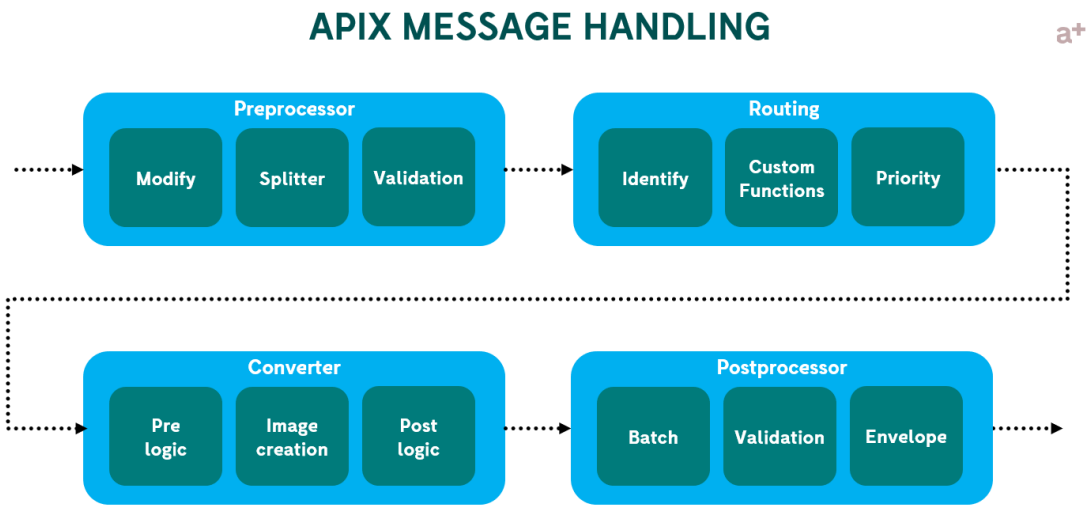


Figure 3. Overview of Apix message handling system.

In default case data is a zip file including metadata Extensible Markup Language (XML) + layout of invoice in PDF-format and maybe attachments. Data can also

be just metadata XML, in this case invoice layout is created dynamically with predefined layout.

Data to Apix messaging platform is received via SFTP or Application Programming Interface (API). API is an interface to application for external users to access data that is not accessible otherwise. Through API user can collect, receive and send data. [26] [27]

SFTP is SecureFTP that is secure method of transporting data between parties. Connection uses SSh-keys to create secure connection between parties before sending data. [25]

**Preprocessor**

**Modify**

After data is received, preprocessing phase will conduct several checks to verify data is what was agreed with customer and that data is processable. General rule is that data is not modified, but sometimes there is a need to modify data to get it to pass validation checks. All modifications need to be agreed with customer.

**Splitter**

Current system expects metadata XML to contain only 1 invoice, if multiple invoices are found data is split to 1 invoice batches in this phase. Other tasks might be done at this phase; tasks can be creating image or format mapping. Depending on implementation there is possibility that some modifications to data will be performed at this phase.

**Validation**

At validation phase it is made sure that invoice layout PDF can be opened and that metadata XML is in correct format. Main validation that is performed is schema validation [10]. Finvoice format compliance is validated against schema

received from Finance Finland webpage [6]. Other important validation which is done at this stage is schematron validation. [11]

Schematron is a rule-based validation language for transforming business and using case-specific requirements into technical validation rules. Schematron helps to tighten and narrow down message structure by stating structures that should or should not be used. It also supports integrity requirements like sum checks, date comparisons, dependent elements (either-or, if-then, one-of, all-or-none, etc.), and conditional value requirements. Schematron should be used as an additional validation layer on top of schema validation. [11]

Postal addresses validation, if data is meant to be printed as paper invoice, it needs to have valid postal addresses. Other possible validation can be bank account, sender / receiver references, VAT-codes.

In case data only includes metadata XML and not invoice layout PDF, default invoice layout can be created at this point. If data is incorrect and there is no agreement of data modification with customer data is rerouted back to customer with comments.

**Routing**

**Identify**

At this phase message recipient is identified.

**Custom functions**

Receiver address found from data matched to routing database, using route found from eInvoice address webpage supported by Tieke. [12]
There are many possibilities how routing can be controlled:

- Overriding data address with preselected address stored in customer data. In this case sender is responsible from all errors that might occur caused by data.
- Suggest routing, if other means of routing are not successful, system tries to suggest route. This is turned on in stored customer data.
- Channel in, default channel is defined. If no route find, message is sent to printing service where invoice is printed and mailed.

**Priority**

There are 4 priorities used
Priority 1, if eInvoice address is correct and route is found.
Priority 2, if eInvoice address I not found or there are multiple routing possibilities.
Priority 3, if priorities 1 or 2 fail, channel out priority is used. Bigger number, higher priority.
Priority 4, if other have failed, trying to use company name to find route.
If all previous options fail message is redirected to print or default channel preselected for customer.

**Converter**

**Pre logic**

Possible data manipulation or validation for converter, can be done at this phase.

**Mapping/Image creation**

Creating mapping from input format to output format.
Images can also be created here, if not found already.

**Post logic**

Data validation and manipulation is possible also at this phase.

**Postprocessor**

**Batch**

Possibly renaming outgoing file, packing/unpacking, this phase may also contain fixing some XML elements to suite customer needs.

**Validation**

Outgoing message is validated according to format and matching schema and schematron document.

**Envelope**

Finally, message is ready to be forwarded. Message is dropped to desired file path. Determining path is based customer set parameters or based on outgoing channel.

After postprocessor data is taken into process which will route it to correct recipient, based routing decision done in Routing phase.

**Ways of Working**

First step is to look for similar kind of setup that could be used as foundation. Based on format of customer message, project engineer will select most suitable already built solution they can think of, if it can be found. Already build solution which does not need lot of modifying and preferably would be using generic components.

After solution is found and selected, components used will be renamed and modified to suite customer needs.

Changes may need to be done. To receiving paths, paths are changed to customer specific receiving component. Changes may be needed to customer

specific modules, also data mapping may need to be changed. Invoice image will be created, if not provided. When building of image layout, internal tests and data validation are done. After data, image and process tests are passed it is time to start pilot test with receivers. More on tests conducted at section 8.

Customer will contact their customers to agree which ones would like to be used as pilot receivers. In general level customer should be sending all possible combinations of invoices which they might send in production and maybe some other that might come relevant at some point. Testing data should be as production like as it can be, not some made up data, this will usually hit you hit back at some point during production run. Also test invoices should contain lot of rows in invoice to see how page breaks fit to layout.

After invoice image is approved it is time for User Acceptance Test (UAT). In UAT invoices are send to real pilot customers with production like data to confirm that invoices are received correctly to receiver Enterprise Recourse Planning (ERP)-system. Test invoices should be sent to different customers with different ERP-systems, as ERP-systems may differ from each other. ERP-system integrates organizations resources, data and operation, eliminating isolated work environments, streamlining processes. [13]

After customer confirms UAT is accepted it is time to turn invoicing on and start sending production invoices.

**Project delivery process**

The process begins when the customer requests a solution offer. Following this, the salesperson contacts the customer to finalize the contract details. Once the contract is agreed upon, the project team is assigned to the task. The project kick off meeting are held where all project details are discussed and agreed upon, including the schedule.

With the schedule in place, the project work commences. Customer review meetings are held as needed or according to the agreed schedule. Throughout

the project, the customer frequently sends test data and receives responses from the team. Based on the feedback provided by the customer, the project team makes necessary changes for example to invoice layout image, if customer does not provide it.

Testing continues until the test results are approved by the customer. Once approved, the next set of test data is sent via production to chosen pilot customers. After the pilot tests are successfully completed and approved, the project is ready to move into full production. Following production, a hypercare phase ensures everything runs smoothly. In hypercare, production is monitored more closely for two weeks. After hypercare phase is over, project is handed over to customer service.

**Tools**

Different tools are used to create a project, here are tasks that are executed with those tools, first sftp-folder creation tool is used to create customer specific folders to sftp-server for customer data. Then, Laskumappi account is created with specific tool designed for this task. Next, data mapping, as customer data usually is some other format than invoice receiver format. Then, layout of invoice image is created with specific tool. Finally, for database entries separate tool is used.

Addition to these tools, there can be also other tools used. For example, searching tool which you would use to search suitable components for upcoming project.

# 5 Target processes

This section introduces target processes of areas of message handling system, ways of working, project delivery processes and last part looks at tooling. The section gives an overview of suggestions on how each area could be improved

**Message handling system**

This section introduces some ideas which could improve in current message handling system.

A message handling system could include several key aspects. Firstly, it could include version control for components, as well as customer-specific modules for tracking, reverting changes if necessary.

Standardized converters could be used, with customizations should only be applied through customer-specific modules. As converters can be complicated and require a lot of work, it would make sense to create converters that relieve strictly on standards and then modifications would be done via special modules. Centralized logging might be implemented to enhance visibility and facilitate issue resolution. Improved monitoring would provide a real-time view of all activities within the system.

A common validation tool would be employed to check data, selecting the appropriate validation tool based on the data received. If validation fails, data would be returned to sender automatically with notice of return reason. This way resources could be saved, and sender could receive feedback much faster than currently.

Machine learning could be utilized to train a model that automatically selects the validation / converting tool. Another way to save resources and remove the possibility for manual error.

Finally, modular components would be used to ensure flexibility and ease of switching components without impacting the entire system.

**Ways Of Working**

This section introduces some ideas which could improve ways of working to be more efficient and cause fewer manual errors.

Tool designed to search setups with similar components. This would speed up setting up the foundation of the project setup.

Database of past projects and solutions, storing keywords that can be sorted by customer type and project complexity. Additionally, there could be a library of templates for common modifications and receiving paths.

An automated testing environment could be integrated with the mapping tool, allowing packets to be moved to the test environment upon completion of mapping conversion, where automated tests are run and results reported.

Finally, a CI/CD pipeline to automate the deployment process.

**Project delivery process**

This section introduces some ideas which could improve project delivery process.

To improve the project delivery process, customers could fill out a project request form in a digital portal. Once the request is submitted, it is sent to sales for confirmation of requirements. If everything is agreed upon, a contract is created using an intelligent contract management tool to ensure official and accurate agreements.

Project team members could be listed in the system with their skills, field of expertise, and availability, allowing the best-suited individuals to be assigned to the project. The project team follows and works on tasks within the project

management tool, while the customer has visibility into certain tasks and areas, enabling them to regularly monitor project progress.

For the kick-off meeting, whether virtual or live, an automated task list and schedule could be generated from the project management tool.

A realistic schedule could be set using historical data, with the project management tool tracking progress and providing alerts.

An automated checklist to ensure production readiness.

**Tools**

This section introduces some ideas which could improve tooling.
First, a tool could be developed to combine the creation of folders with SFTP-server and Laskumappi accounts. This tool could be written in Bash, utilizing existing scripts. Additionally, a cloning tool could be created to facilitate setup cloning or the creation of setups from templates. A validation tool would be useful for checking incoming data. Lastly, a component search tool could be designed to search for specific strings within filenames and inside files located in certain folders.

# 6 Area selected for improvement

This section explains area that was selected for improvement. First it explains selected area for improvement. Next recognized tools to develop. The third and last part introduces selected tool.

### Selected area

Area that was selected for improvement was Tools. Improving tools used to create project was seen as the most valuable for project engineers. Improving tooling could provide benefits faster than improvements in other areas.

### Recognized tools to develop

The programs that were identified as bringing the most value include a component search tool, which searches setups made with similar components. Additionally, a validation tool for incoming data serves as a common validation mechanism. A combining tool integrates existing tools used to create folders to SFTP-server and Laskumappi user accounts. Lastly, a cloning tool searches for the best setup match based on given criteria and clones the setup. Alternatively, setups can be created from pre-made templates.

### Selected tool

Selection was done based on need and rough estimate of time needed to develop the tool.

Program that was selected, tool for cloning. That was seen as the most useful tool that would create most value for our team. Other tools maybe developed outside of this thesis.

# 7 Planning selected component

This section outlines planning of selected component, tool for cloning customer setups. First introducing purpose of the tool. This is followed by objective, requirements, data gathering for most used components, data processing. Final two sections introducing template for creation of standard setup and finally representing cloning program.

### Purpose

Purpose of the tool is to automatise creation of customer setup as much it is possible and reasonable without lot of manual work.

### Objective

Objective is to provide customer setup helper tool for technical engineers and project managers. Tool is meant for internal use only to help project personnel to setup customer implementation. Objective is to build solution that will combine manual phases in single tool. Phases to combine creation of Laskumappi account, setup of file_checker to handle data transfer from sftp-folder to system, setup of einvoice_sender to connect channel in and customer to data folder.
Idea is to make it possible to setup each phase individually or all phases at once.

### Requirements

### Functional

Functional requirements are shown in graphical UI in Figure 4.

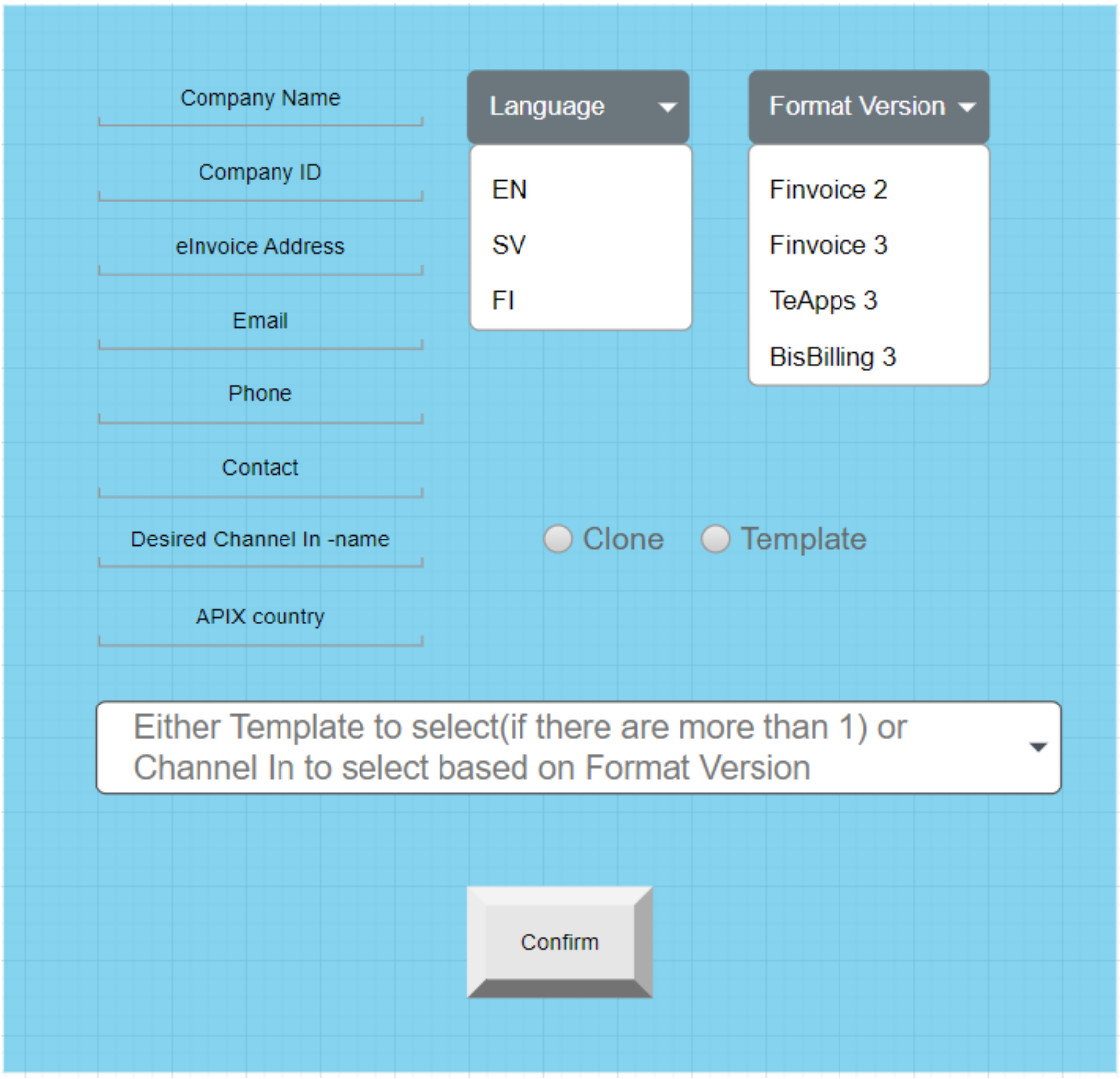


Figure 4. Example of graphical User Interface.

- User interface:
  - Graphical form.
    - It has text fields for Company name, Company ID, eInvoice Address, Email, Phone, Contact person and dropdown menus for file format and file format version and for Language.

    - Radio button to select Template or Clone.
      - Clone. Dropdown lists consist of contents of DB table channel in fields software_name, software_version,

pre_processor, splitter, validator, ack_instructions, format, version.

- Template. Preselected data will be shown in dropdown menu.

- Confirm button at bottom of page.

- Command line
  - tool can be used directly from command line.

**Non-functional**

To ensure quality product, ISO/IEC 25010 standard must be followed. The ISO/IEC 25010 standard is a norm for product quality and is part of the ISO model that was created by International Organization for Standardization. [19] [29].
ISO/IEC 25010 defines nine characteristics:

- Functional Suitability
- Performance Efficiency
- Compatibility
- Interaction Capability
- Reliability
- Security
- Maintainability
- Flexibility
- Safety

Each of these items has been divided into sub-characteristics as shown in Figure 5.

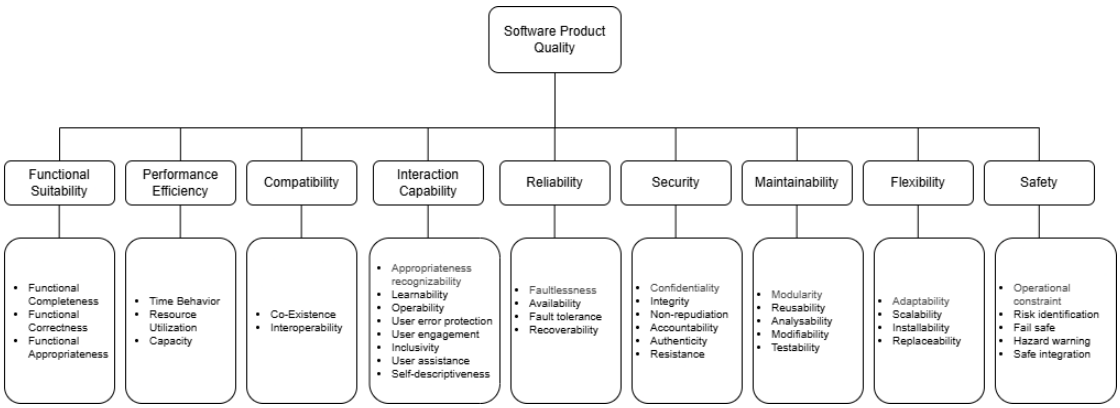


Figure 5. Software Product Quality characteristics. [19]

Non-functional requirements for the cloning tool were grouped according to ISO/IEC 25010 model as shown in Table 1. Values were based on current setup and estimate of future developments. All sub-characteristics were not applied.

Table 1. Non-functional requirements for the cloning tool.

| **ISO/IEC 25010 Quality characteristic** | **Non-functional requirements for the Cloning tool** |
|---|---|
| Functional Suitability | • The program must support all required functionalities, SFTP, file checker, contracts and einvoice sender. |
| Performance Efficiency | • The program should execute tasks (database queries, subprocess calls, etc) within an acceptable time frame.<br>• The program should use system resources (CPU, memory, disk) efficiently.<br>• The program should handle concurrent executions without performance degradation. |
| Compatibility | • The program should not interfere with other applications running on the same system.<br>• The program must integrate seamlessly with database server and subprocess scripts. |

| | |
|---|---|
| Interaction Capability | • The program should provide clear outputs and logs to help users understand its purpose and results.<br>• The program should be easy to learn and operate, with clear documentation and meaningful error messages.<br>• Users should be able to operate the program easily, with minimal manual intervention.<br>• The program should validate inputs (for example mandatory fields, valid formats) to prevent user errors. |
| Reliability | • The program should be stable and free of critical bugs.<br>• The program should handle unexpected errors (database connection failures, invalid configurations) gracefully without crashing.<br>• The program should provide mechanisms to recover from failures, such as retry logic or error logs for debugging. |
| Security | • Access to solution must be restricted from outsiders.<br>• Sensitive data (passwords, configuration and server details) must be protected from unauthorized access.<br>• The program should ensure data integrity during processing, avoiding corruption of configurations or logs.<br>• User actions (configuration changes, executed tasks) should be logged to ensure accountability.<br>• The system keeps log of logins. |

| | |
|---|---|
| Maintainability | • The program should have a modular structure, with separate functions for each task.<br>• Solution should be easy to modify and further develop.<br>• Components (for example validation, configuration loading) should be reusable in other projects.<br>• The program should provide clear logs and error messages to help developers analyse and debug issues. |
| Flexibility | • The program should support running in different environments (test, production) by dynamically detecting the server environment.<br>• The program should allow setup customization via configuration files without modifying the source code.<br>• It should create / remove necessary directories automatically when needed.<br>• The program should be modular, allowing individual components to be replaced or updated without impacting the rest of the system.<br>• It should follow standard coding practices to ensure compatibility with alternative tools or systems performing similar tasks. |
| Safety | • It should restrict operations to authorized users by leveraging system-level permissions (for example: for logging user actions).<br>• The program should log all critical actions (for example: setting configurations, |

| | |
|---|---|
| | sending emails) to identify potential risks or failures.<br>• It should detect and report invalid configurations (for example: missing sftp username, invalid file paths, invalid configurations) before executing operations.<br>• In case of failure (database/subprocess errors), the program should terminate gracefully and provide meaningful error messages.<br>• Operations performed to database should include mechanism to revert actions in case of failure.<br>• The program should provide warnings for potentially unsafe operations (for example: missing mandatory configurations, invalid file patterns) to allow users to take corrective action.<br>• It should log all warnings and errors for later analysis.<br>• The program should ensure safety during integration with external components by validating inputs and outputs. |

When creating Laskumappi account, display created password, customer id and transfer id and transfer key on screen for user to store them for later use.

When creating einvoice sender and file checker, display data that was used to create each item.

Good coding practises must be followed. These include, for maintainability, well-written code is easier to understand, modify and extend. For readability, clear and consistent coding helps developer quickly understand the logic and structure. For efficiency, good practices often lead to more efficient code in performance and resource usage wise. For error reduction, following best practices can minimize likelihood of bugs and errors. Maintain readability and include, line indentation, maximum of characters written in line, correctly setting imports. [14] [15] [16] [17] [18]

**Restrictions and limitations**

Exceptions, in following conditions proper and clear error message needs to be shown in pop-up window:

- Customer enters invalid email address
    - Max 64 characters long.
    - Only 1 ‘@’-sign.
    - No special characters (comma separated list), “,\[,].
    - No underscore (_) in domain part.
- Special characters <> are not allowed in any field.
- Valid company ID, Id is checked from ytj.

**Required information**

Items required when creating Laskumappi account, Company name, Company ID, eInvoice address, Email, Language, Desired name for new channel_in or (File format, File format version).

Meaningful items required to be used to setup file_checker are type, directory, filepattern, fileage, filetype and cmd.

Items used to setup einvoice_sender are type, path, id_customer, id_channel_in.

**Gathering data to determine most used components**

This part is essential when predicting components user might need to build most efficient customer setup. Data will be gathered from two databases, transactionlog and apix. Tables used, from apix database, channel_in to determine components that are used in particular channel_in.

From transactionlog event_history-table. In this table there all traffic information that has gone through message handling platform.

Following listings 1. 2. and 3. display SQL-queries used to list most used components. Listed components are used to determine order where components are introduced to user to select. Listing outputs are saved as csv.

Listings display type, pre_processor/splitter/validator count, format, version, invoicing_type, name of the pre_processor/splitter/validator, software=name of the channel_in and lastly version of the software.

SQL script will leave out rows that are acknowledgement messages and messages that are sent before 01.01.2023. Results are grouped by type, format, version, invoicing_type, name of the pre_processor/splitter/validator, name of channel_in = software and by software_version. Also, the notable fact is that the left join is used, it will return all records from left side table, in this case apix.channel_in table and it will return matching records from table on right side, transactionlog.event_history-table.

```
Select type, count(pre_processor), format, version, invoicing_type,
pre_processor,software_name, software_version#,
from apix.channel_in C
left join transactionlog.event_history E
on E.channel_in_name = C.software_name
where 1=1
and type != 'ack'
and E.sent_time > "2023-01-01"
group by type, format, version, invoicing_type,
pre_processor,software_name,software_version
```

Listing 1. SQL-query for Pre_processor.

```
Select type, count(splitter), format, version, invoicing_type,
splitter,software_name, software_version#,
from apix.channel_in C
left join transactionlog.event_history E
on E.channel_in_name = C.software_name
where 1=1
and type != 'ack'
and E.sent_time > "2023-01-01"
group by type, format, version, invoicing_type,
splitter,software_name,software_version
```

Listing 2. SQL-query for Splitter.

```
Select type, count(validator), format, version, invoicing_type,
validator,software_name, software_version#,
from apix.channel_in C
left join transactionlog.event_history E
on E.channel_in_name = C.software_name
where 1=1
and type != 'ack'
and E.sent_time > "2023-01-01"
group by type, format, version, invoicing_type, validator,software_name,
software_version
```

Listing 3. SQL-query for Validator.

**Data processing**

Data doesn’t need any special processing, data is only sorted by pre_processor/splitter/validator count, and it is ready to use. Due to complexity of the system, data gathered from databases need to be manually handled to determine which component can be used. Many items on top of lists are considered as special cases such as components used in operator traffic or with other partners. At this stage tool is aimed for cases that can be considered as regular/standard.

**Templates for standard setup**

Using data received from SQL-queries, most used components are included in some channel_in component. Channel_in which includes wanted components that are placed to txt-file as key=value pair as seen in Listing 4.

```
CHANNEL_IN=best_channel_in_for_teapps30.pl
```

Listing 4. TeApps 3.0 components selected as template.

# 8 Implementation

This section introduces implementation of Cloning tool with its features.
First looking at Cloning program with process flow chart and example of customer configuration file. The architecture of the Cloning Program is divided into frontend and backend, components are displayed. Next, section introduces various test types that were conducted to code and program. Finally, section presents structure of tool, it contains six main components.

Cloning program is a solution for creating similar customer setup from which have been proven to work. User can select template or copy existing setup. User can also select to use graphical UI or use it from command line with configuration file.

The flow of Cloning program is very simple as illustrated in Figure 6.
At first step user gives required information via web-UI or configuration file, example shown in Listing 5. If user uses web-UI, data from UI is stored to same configuration file. At second step based on information given data is gathered from database. Next step will modify data collected from database. Changes needed to data is reset record id and dates. After data is ready, it is uploaded to database using SQL. At final stage, confirmation of performed actions are shown with newly created customer id.

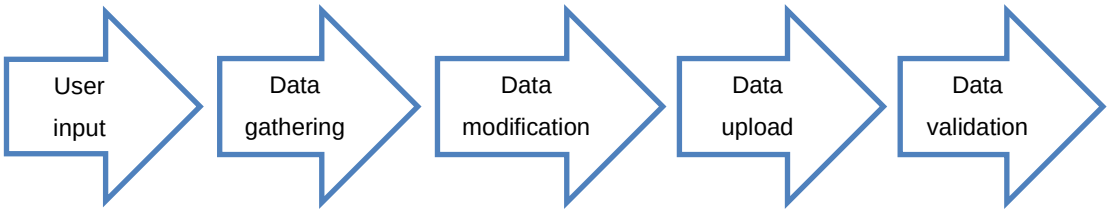


Figure 6. Process flow of the Cloning program.

```
[LASKUMAPPI]
SET                             = 1
EMAIL                           = vesa.kaikkonen@apix8.com
Y_TUNNUS                        = TESTI81234567-8
COMPANY_NAME                    = Apix Oy
# 0/1, DEFAULT = 0
PERMISSION_TO_EXPORT            =
#fi/en/sv. DEFAULT = FI
LANGUAGE                        = fi
#FI/SV, DEFAULT = FI
APIX_COUNTRY                    = FI
CONTACT_PERSON                  = Vesa Kaikkonen
PHONE_NUMBER                    = +358 40 123 4567
FORMAT                          = finvoice
FORMAT_VERSION                  = 3.0
CHANNELIN                       =
# finvoice / teapps / peppol
TEMPLATE                        =
CUSTOMERID                      =

[CONTRACTS]
SET                             = 1
CONTRACTS                       = NoRouteEmail;myemail@apix.fi

[SFTP]
SET                             = 0
#INCOMING/OUTGOING
#DIRECTION   = INCOMING
SFTP_USERNAME                   = my_sftp

[FILECHECKER]
SET                             = 0
# MANDATORY
FILECHECKER_DIRECTORY           = /file/path/OUT/
# MANDATORY
FILECHECKER_NAME                = Apix Oy Outgoing
FILECHECKER_CMD                 =
FILECHECKER_TYPE                =
# DEFAULT = .*(Zip|zip|ZIP)
FILECHECKER_FILEPATTERN         =
# DEFAULT = 1
FILECHECKER_FILEAGE             =
# DEFAULT = f
FILECHECKER_FILETYPE            =
# DEFAULT = 0
FILECHECKER_ACTIVE              =

[EINVOICE_SENDER]
SET                             = 0
# 0/1, IF EMPTY, DEFAULT = 0
EINVOICE_SENDER_ACTIVE          =
# normal / gateway / edi, IF EMPTY, DEFAULT = normal
EINVOICE_SENDER_TYPE            =
EINVOICE_SENDER_PATH            =
# IF EMPTY, DEFAULT  = @SFTP_USERNAME@/sent
EINVOICE_SENDER_ARCHIVE         =
# normal / big, DEFAULT         = normal
EINVOICE_SENDER_MODE            =
# 0/1, IF EMPTY, DEFAULT        = 0
EINVOICE_SENDER_ACK_SENT        =
# 0/1, IF EMPTY, DEFAULT        = 0
EINVOICE_SENDER_BIGINT          =
# DEFAULT  = empty
EINVOICE_SENDER_COMMENT         =
```

Listing 5. Example of customer configuration file. Own section for every module.

### User access roles

Cloning program access roles are based on accesses to production system. There are no different access roles, every user has the same privileges.

### Architecture

Cloning program architecture can be divided into frontend and backend as shown in Figure 7. Frontend consist of graphical interface and command line interface. Backend consists of database, engine and configuration file.

Also seen in Figure 7 data flows from graphical interface and command line interface to configuration file. Cloning program engine then uses configuration file to execute user request.

Cloning program is hosted by current production Linux environment, using default system components.

Cloning program is programmed using Python coding language. Program may use existing components which are programmed using Perl language.

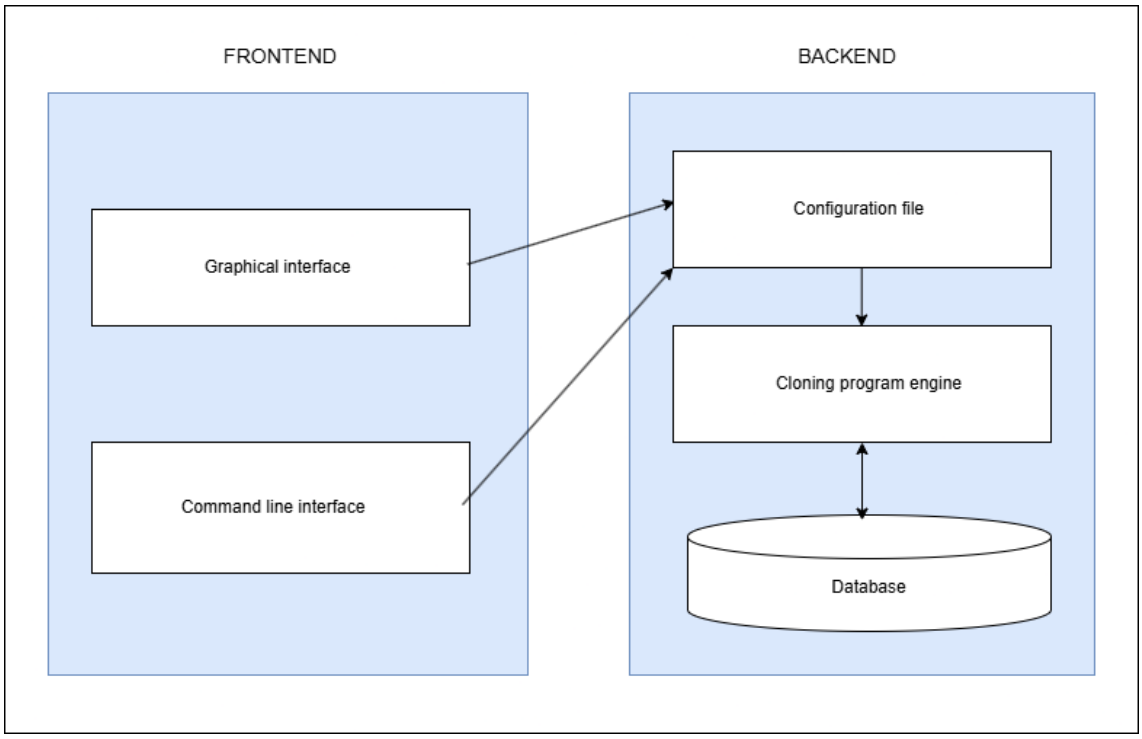


Figure 7. Architecture of the Cloning program.

**Testing**

To guarantee solution quality and reliability it was essential to incorporate various testing methodologies and frameworks. As much as possible tests were automated, but mostly performed manually, due to lack of knowledge about testing tools that could be used in this kind special program. Once feature or functionality was completed, it was then manually tested and if passed, it could consider to be complete. This method is comparable to agile development process used in agile Scrum teams.

Test types that were performed to the program shown in Table 2 below.

Table 2. Tests performed to the program. [20]

| **Test type** | **Description** |
|---|---|
| Unit test | Very low level and close to the source of an application. They consist of individual methods and functions of the classes, components or modules. |
| Integration test | Verify that different modules or services work well together. Usual integration test is when testing connection to database and performing queries. |
| Functional test | For example, verify of an outcome database query. |
| End-to-end test | Verifies that user can use all features of the program. Test that will verify if user can create Laskumappi account, add contracts to new account, setup SFTP connection + folders, setup einvoice_sender and file_checker. Tries to simulate real world situation. |

| | |
|---|---|
| Smoke test | Are basic and quick tests to program main functionalities and features. |

**Structure of Cloning tool**

Tool is divided into 6 different modules. Main, Database, Configuration, Laskumappi, SFTP and Validate. This section briefly introduces these modules.

**Main**

Acts as orchestrator of the program. It collects data from configuration files. Loads customer configuration file and templates. Determines server used, test or production. This will affect parameterization of scripts and tasks executed.
Based on user needs, given in configuration file, executes tasks for selected modules.

**Database**

Database module opens connection to database. Executing queries with timeout. Handles data insertions for einvoice_sender and file_checker. Fetches various customer and connection data.

**Configuration**

Module provides methods to manage configuration settings for the program.
Loads data from configuration, constants and template initialization files.
It uses specific class method structure of loading data for every selected section.

**Laskumappi**

This module will provide classes and functions to manage and validate Laskumappi configuration data for the program. Validate module is used to validate country, language and email address. Uses script that will create laskumappi user with given data.

**Validate**

Validate module provides validation functions for various configuration parameters used in the program. It offers functions to validate email, corporate id, language, country. It also provides validation function for mandatory items. It will check presence of data given and gives error if data is missing.

# 9 Conclusions

The objective of this master thesis was to investigate improvements in areas of messaging handling system, ways of working, project delivery process and tooling.
The objective was divided into four research goals, which were:

- analyze and document current message handling system, ways of working, project delivery processes and tools used,
- define target processes of mentioned areas,
- select areas which to improve to defined target process,
- create tools which could help to achieve automation and better quality in project deliveries.

In conclusion current processes were successfully analyzed and documented in this thesis. Target processes were defined, and improvement suggestions were given to all areas. One area was selected which got more focus for improvements. The tooling area was selected for improvement and tools that were built improved automation and quality in project tasks.

As a result of this thesis learned a lot about case company processes, tools and ways of working. It was very noticeable how lack of documentation and silent knowledge is affecting ways of working. Also, a new tool was introduced in March 2025 to project team. As being new tool and it requires some learning of new ways of working there might be some starting difficulties.

**Summary**

Section 1 introduced case company, business challenges and objectives, scope of study and structure of thesis. Lack of documentation, instructions, tooling, knowledge and manual tasks was seen as challenges that caused delays, and also human errors, issues and uncertainties were identified.

Also, in section 1 study goals were introduced. Goals to reach are to analyze current processes of message handling system, project delivery, ways of working and tools used to create project. Goals are also to define target processes, identify the area that would deliver the quickest benefits for project engineers and as the last goal based on improvement ideas, create tools that would provide most value to project implementation team.

Section 2 concentrated on methods and materials used in this study. Section 3 focused on theoretical background, defining eInvoicing, outlining eInvoicing standards and legislation, history and short future of eInvoicing. Section also presented Peppol.

Section 4 consisted of current processes of message handling system, ways of working, project delivery process and tools that are used when building project. From current message handling system overview of were presented along with short explanation of each phase. Phases in message handling system, Preprocessor, Routing, Converter and Postprocessor. In Ways of working section, the process of setting up a customer sales invoice project from a project engineer's perspective, including finding a suitable existing solution, modifying it to meet customer needs, conducting internal tests, and performing user acceptance testing (UAT) with real pilot customers before starting production invoicing. In the Project delivery process section, we looked at current sales invoice project delivery process, it involves customer requests, contract finalization, project team assignment, meetings, schedule setting, project work, customer review meetings, test data exchanges, feedback-based changes, testing, pilot tests, production and hypercare phase, finally handover to customer service. In the Tools section were introduced tools that are used to set up customer implementation projects. These tools include sftp-tool to create folders to sftp-server, customer account creation tool, data mapping tool, image creation tool. Addition to these there can be some other tools used as well.

Section 5 took a glance at how areas presented at section 4 could be improved. A message handling system could be enhanced with version control for components, customer-specific modules, centralized logging for better issue

resolution, real-time monitoring, and standardized converters with modular components for flexibility. Machine learning could assist in selecting validation tools, while a common validation mechanism ensures accurate data handling. Ways of working could be improved by introducing tools such as a component search tool, a database of past projects for reference, a library of modification templates, an automated testing environment integrated with the mapping tool, and a CI/CD pipeline for streamlined and automated deployments.

To improve the project delivery process, a digital portal could streamline customer requests, and contracts could be managed with intelligent tools for accuracy. The system would match project team members based on their expertise and availability. A project management tool would facilitate task tracking, customer visibility, and scheduling using historical data, with automated task lists and checklists ensuring readiness and progress monitoring.

The tooling section could benefit more sophisticated tools, including a combined tool for creating SFTP folders and Laskumappi accounts, a cloning tool for replicating setups, a validation tool for checking incoming data, and a component search tool to locate specific strings in filenames and files.

Section 6 presents the area that was selected for improvement. Selected area tools were seen as fields that would give benefits for engineers faster than other areas. Tools that would create the most value in a short time was recognized, including a component search tool, a validation tool for incoming data, a combining tool for integrating tasks like creating SFTP folders and Laskumappi accounts, and a cloning tool for setup replication or creation from templates. The cloning tool was chosen as the most valuable, with the highest potential to streamline processes and provide significant benefits.

Section 7 introduced planning of select component. Section presented purpose, objectives and requirements (functional and non-functional) for the tool. For non-functional requirements using ISO/IEC 25010 model to ensure quality of the program developed. Model consists of nine characteristics, such as Functional

Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, Flexibility and Safety.

Section also presented information required to create Laskumappi account.
Data was gathered to determine most used components to create templates. SQL-listings of data gathering was provided.

Section 8 focused implementation of selected tool with section 7 introduced requirements. The implementation of the Cloning tool simplifies the creation of customer setups by utilizing templates or existing configurations, and it supports both a graphical user interface and a command-line interface, focused on command-line interface. The tool is designed to handle data through configuration files, modify essential fields such as IDs and dates, and upload the prepared data to the database.

The architecture of the tool is divided into two main components: the frontend, which includes the user interface and command-line features, and the backend, encompassing the database, engine, and configuration files. Python and Perl serve as the primary programming languages for its development.

To ensure reliability and quality, extensive testing was performed on the tool. These tests included unit testing, integration testing, functional testing, end-to-end testing, and smoke testing.

The tool comprises six key modules: the Main module, which orchestrates operations; the Database module, which handles query execution; the Configuration module, which manages program settings; the Laskumappi module, which handles account management; the SFTP module, which manages connections; and the Validate module, which validates configuration parameters including email addresses, corporate IDs, languages, and countries.

**Challenges**

There were no major challenges when creating this project. Maybe the biggest one was the complexity of the message handling system. Since that needed to be figured out before any decisions about any tooling could be made. Some challenges were caused by lack of documentation of the system and current tooling, but those were cleared quickly. Some minor challenges were faced during coding of the tool.

**Future**

The aim in the future is to increase usage of the program and develop new features. Features already planned, proper logging for user actions and system failures. Some improvements have been planned to coding of the modules.

Other target process discoveries that were made, tool for searching components have been developed and it is in production use.

Version control systems have been activated for source codes of mapping and image creation files. Also, CI/CD pipeline is in development.

Production changes have been focused on customer specific modules instead of general components making implementations more modifiable and modular.

## References


1 Per Holmlund CMO, The history of e-invoicing: from paper to pixel. https://qvalia.com/the-history-of-e-invoicing/ Blog post.04.01.2019. Accessed 18.02.2024

2 Obtaining a copy of the European standard on eInvoicing. https://ec.europa.eu/digital-building-blocks/sites/display/DIGITAL/Obtaining+a+copy+of+the+European+standard+on+eInvoicing Accessed 30.05.2025

3 Compliance with the European standard on eInvoicing: Explaining the standard, https://ec.europa.eu/digital-building-blocks/sites/display/DIGITAL/Compliance+with+eInvoicing+standard Accessed 18.02.2024

4 Verkkolaskulaki 2020 - mitä se tarkoittaa pienyrittäjälle. https://www.isolta.fi/verkkolaskulaki-2020/ Accessed 18.02.2024.

5 Apix Messaging Oy. https://apix.fi Accessed 20.02.2024

6 Finvoice3.0.xsd. https://file.finanssiala.fi/finvoice/css/270923/Finvoice3.0.xsd Accessed 25.02.2024

7 Peppol BIS UBL schematron validation rules. https://docs.peppol.eu/poacc/billing/3.0/files/PEPPOL-EN16931-UBL.sch Accessed 25.02.2024

8 Apix Messaging Oy, financials. https://www.asiakastieto.fi/yritykset/fi/apix-messaging-oy/23327487/taloustiedot Accessed 24.03.2025

9 Understanding ETL, O'Reilly Media, Inc: Matt Palmer, 2024.

10 Schema validation. https://www.truugo.com/en/xml_validation/ Accessed 03.04.2025

11 Schematron validation. https://www.truugo.com/en/schematron_validation/ Accessed 03.04.2025

12 Verkkolaskuosoitteisto. https://verkkolaskuosoite.fi/client/ Accessed 03.04.2025

13 Toiminnanohjausjärjestelmä. https://fi.wikipedia.org/wiki/Toiminnanohjausj%C3%A4rjestelm%C3%A4 Accessed 23.04.2025

14 Style guide for Python Code. https://peps.python.org/pep-0008/ Accessed 28.04.2025

15 Coding standards and best practices to follow. https://www.browserstack.com/guide/coding-standards-best-practices Accessed 28.04.2025

16 Coding best practices. https://en.wikipedia.org/wiki/Coding_best_practices Accessed 28.04.2025

17 Top coding practices for modern developers. https://www.linkedin.com/pulse/top-coding-best-practices-modern-developers-taazaa-inc-rcarf Accessed 28.04.2025

18 What are some code optimization best practices for memory management and garbage collection? https://www.linkedin.com/advice/0/what-some-code-optimization-best-practices-memory Accessed 28.04.2025

19 ISO/IEC 25010. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 Accessed 29.04.2025

20 The different types of software testing. https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing Accessed 16.05.2025

21 Peppol in brief. https://www.valtiokonttori.fi/en/service/the-state-treasury-is-the-finnish-peppol-authority/#peppol-in-brief Accessed 23.05.2025

22 PEPPOL. https://en.wikipedia.org/wiki/PEPPOL Accessed 23.05.2025

23 Peppol: Yhtenäisen sähköisen laskutuksen verkosto – mitä sinun tulisi tietää siitä? https://apix.fi/ajankohtaista/peppol/ Accessed 23.05.2025

24 Peppol Network. https://peppol.org/learn-more/peppol-interoperability-framework/ Accessed 24.05.2025

25 SSH, The Secure Shell: The Definitive Guide, 2nd Edition. O'Reilly Media, Inc: Daniel J. Barrett, Richard E. Silverman, Robert G. Byrnes, 2025.

26 Mastering API Architecture. O'Reilly Media, Inc: James Gough, Daniel Bryant, Matthew Auburn, 2022.

27 Rest API for external usage. https://wiki.apix.fi/ Accessed 26.05.2025

28 ISO/IEC 27001. https://www.iso.org/standard/27001 Accessed 26.05.2025

29 ISO: Global standards for trusted goods and services. https://www.iso.org/home.html Accessed 02.06.2025