Hochschule RheinMain
Fachbereich Design Informatik Medien
Studiengang Informatik

Master Thesis

Submitted in Partial Fulfillment of the requirements for the Degree of

Master of Science

# WORKFLOW AUTOMATION FOR EFFICIENT ON-DEMAND DISTRIBUTION OF SENSITIVE DATA IN THE FINANCE SECTOR

| | |
|---|---|
| Vorgelegt von | Rahul Thiruthinmel Premnavas |
| am | 14. April 2025 |
| Referent | Prof. Dr. Nikolay Tcholtchev |
| Korreferent | Dr. Sergio Staab |

## Eigenständigkeitserklärung (gem. ABPO, Ziff. 4.1.5.4 (3))

Ich bin verantwortlich für die Qualität des Inhalts dieser Arbeit, den ich mit geeigneten wissenschaftlichen Quellen belegt bzw. gestützt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Texte, Gedankengänge, Konzepte, Grafiken, technischen Inhalte und ähnliches in meinen Ausführungen habe ich eindeutig gekennzeichnet und mit vollständigen Verweisen auf die jeweilige Quelle versehen. Alle weiteren Inhalte der Arbeit (Textteile, Abbildungen, Tabellen etc.) ohne entsprechende Verweise stammen im urheberrechtlichen Sinn von mir. Ich versichere außerdem, dass ich die Bachelor-Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die vorliegende Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Mir ist bekannt, dass ein Verstoß gegen die genannten Punkte als Täuschungsversuch gelten und prüfungsrechtliche Konsequenzen haben kann. Insbesondere kann es dazu führen, dass die Leistung nicht bestanden ist und dass bei mehrfachem oder schwerwiegendem Täuschungsversuch eine Exmatrikulation droht.

Rahul

Wiesbaden, 14. April 2025

Rahul Thiruthinmel Premnavas

## Explanation on the use of AI tools

1. What tools were used?
   (a) CHATGPT
2. What are the prompts used?
   (a) How to write an abstract that is more general, not company-specific?
   (b) What could all be included in the architecture chapter if writing a thesis?
   (c) Could you help me to create a meaningful algorithm from Python code?
   (d) What all points should be included while writing a problem statement?
   (e) What points can all be added to your thesis while writing the introduction and motivation?

Rahul

Wiesbaden, 14. April 2025

Rahul Thiruthinmel Premnavas

# Erklärung zur Verwendung der Masterthesis

Hiermit erkläre ich mein Einverständnis mit den im folgenden aufgeführten Verbreitungsformen dieser Abschlussarbeit:

| **Verbreitungsform** | **Ja** | **Nein** |
|---|---|---|
| Veröffentlichung des Titels der Arbeit im Internet | × | |
| Veröffentlichung der Arbeit im Internet | × | |

Rahul

Wiesbaden, 14. April 2025

Rahul Thiruthinmel Premnavas

## Abstract

Timely Data Distribution in the financial sector is very crucial because of the changing business environments and the amount of competition in the ever-changing area. In this scenario, automation of tasks can significantly reduce the amount of time required to provide on-demand data. The Automation of the File-Transfer-System is carried out in this thesis, emphasizing optimizing everyday transactions and enhancing service delivery. FTS (File Transfer System), a vital component in an industry where various file transfers occur on a day-to-day basis across several time zones, is the primary focus of this research.

Another important component of this research is the internal platform for provisioning requests. A significant number of tasks that get accumulated in Request for Product Development (RPD), the above-mentioned internal platform for provisioning requests or tasks, are done manually, which obviously would take more time given the benefit of the doubt to the proposed idea of automating the frequent repetitive tasks. The tasks to be done can be of various types, which span throughout the various departments in a firm, from account creation in a particular application (business units) to changing the protocols, etc.

This research in FTS (File Transfer System) aims to improve customer experience management, thereby making the daily client to the organization and vice versa file transfers fluent, especially when the data that are handled are confidential in nature. Principally, there are five tasks to solve.

1. SFTP(Secure File Transfer Protocol) Key-generation
2. Password reset
3. FTS/SFTP protocol change
4. Management of Client IP address restrictions (add/modify/delete)
5. Creation of EH (Enterprise Hosts) accounts

Valid study and research regarding each service's REST APIs have been carried out; Python programming is used to implement the logic and the code being managed in the prominent version control system git/GitHub. Logging methods are implemented with the help of a Python module named logging, and steps are also carried out to parse the arguments from the command-line interface using the Python module argparse. As each task mentioned above uses different services, the API architecture had to be carefully examined and understood before proceeding from one task to another.

The first task, SFTP Key Generation, is completed at the beginning and yielded results; the code repositories are forwarded to the production servers. Further explanation and details regarding the tasks will be reported in the Implementation part of this report.

# Contents

# Chapter 1

# Introduction and Motivation

For an organization, it is paramount that processes and undertakings are expected to reach customers faster and more efficiently. It is also a vital factor in the firm's development, determined by how fast the services and products are delivered. In any financial firm, failure of service delivery can lead to monetarily financial loss and subsequently in market value; many financial software firms have immediate access to financial data and analytics, thereby making sure that the choices they make are relevant and quick. In Information Technology, FTS stands for File Transfer System, and it uses FTP (File Transfer Protocol) to transfer files between networks and hosts. Although outdated, it still has usage and implementation in places. The SFTP (Secure File Transfer Protocol) is the secure way of transferring files, which many companies are still using to achieve a secure line of connection between FTS Servers and the client hosts. The detailed description of the architecture will be discussed in later parts of the work. Ports 21 and 22 are used for both protocols.

Anyhow, it is also worth mentioning and diving into the networking aspects of this area of study. As we have already seen, the File Transfer System is a vital component in the firm, and FTP is the respective protocol; what is its architecture? And all the information that comes with it is also worth delving into.

## 1.1 The OSI Model

Firstly, we will have a look at the OSI model (Open System Interconnection Model). According to Sumit et al. [13], the OSI model, whose conception began in the 1970s, is the result of efforts by ISO (International Organization for Standardization) and made the layer-to-layer architecture of the communication channels, which in turn made it easier to categorize and comprehend a complex circuit of layers. The main objective of the OSI model is to make an interoperable network circuit that would help developers and vendors, says Sumit et al. [13]. Layering is the underlying idea of the OSI model, and seven layers are included to describe the OSI model.

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

### 1.1.1 Physical Layer

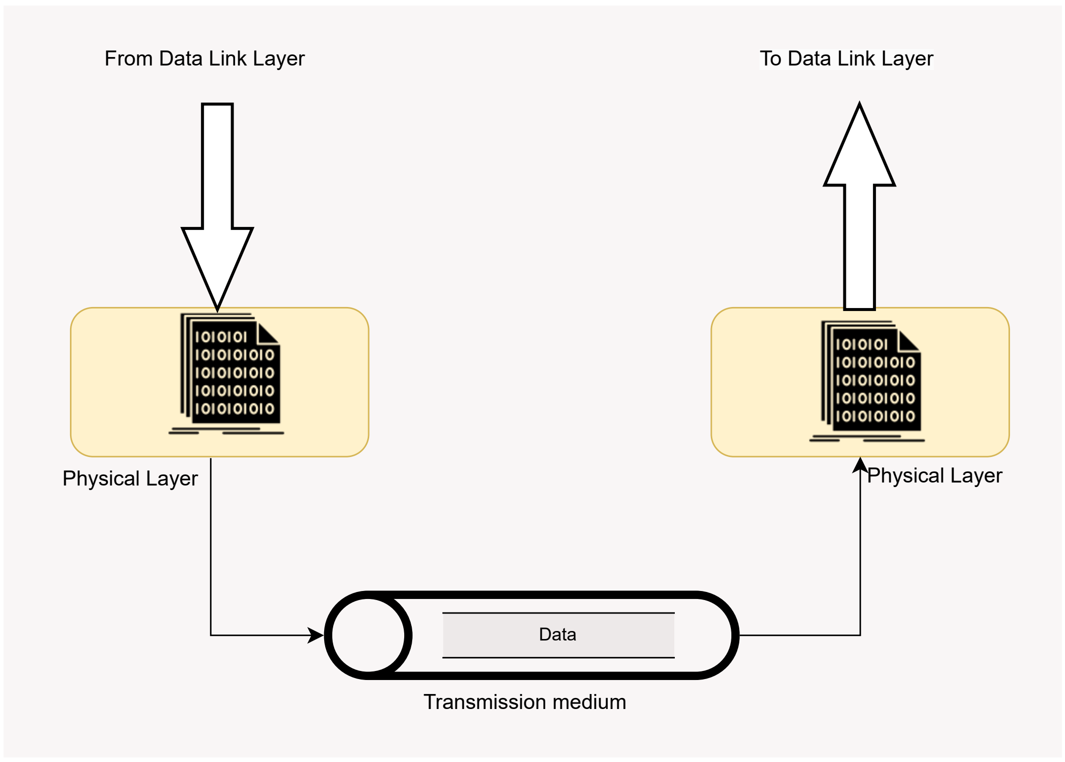


Figure 1.1: The physical layer [13]

It is the lowest layer of the OSI model, Figure 1.1 shows the physical layer and determines the medium of physical transmission and all the other physical components while connecting two nodes, for example, cables, pins, voltages, etc. Upon establishing connections between nodes, it also controls the flow and its modulation as described by Sumit et al. [13].

### 1.1.2 Data Link Layer

The main function of the Data Link Layer is to check the error using the Cyclic Redundancy Check (CRC) as per Sumit et al. [13], shown in figure 1.2. When the nodes are not directly connected, the checks will be performed through different links, which operate differently. The job of sending packets to the MAC addresses of hosts is done by this layer. The common protocols at the Data Link layer are Ethernet and HDLC (High-Level Data Link).

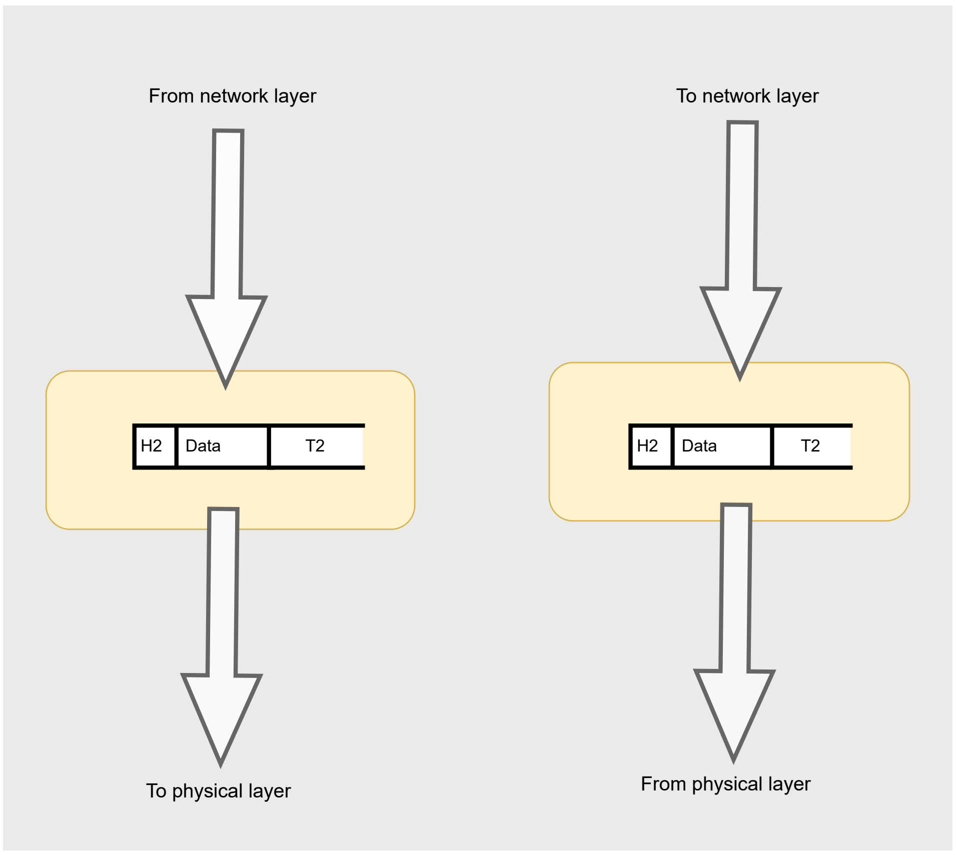


Figure 1.2: The data link layer [13]

### 1.1.3 Network Layer

The main function of this layer is to make a path and define the addresses of the nodes. In a network, every node is connected via addresses, and this is done by routing, by placing addresses of senders and receivers on headers, and this is done in this layer describes Sumit et al. [13]. The figure 1.3 below shows how the data is carried from the transport layer to the data link layer. The network may (or may not) implement message delivery in addition to message routing. This could involve breaking the message up into multiple pieces, sending each fragment over a different route, then piecing the pieces back together, reporting delivery failures, etc.

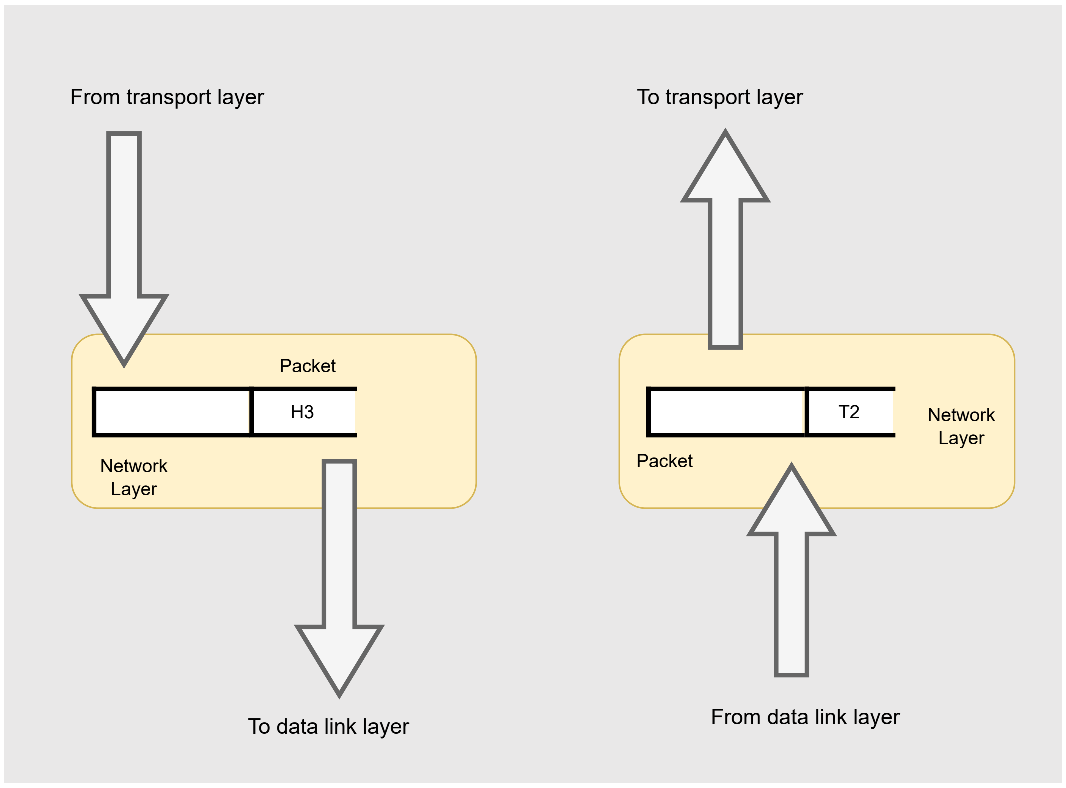


Figure 1.3: The network layer [13]

### 1.1.4 Transport Layer

Provides an end-to-end connection to users on different nodes. Provides reliability and security such as encryption, thus ensuring complete data transfer according to the work of Bora et al. [9]. Figure 1.4 shows how the data are carried in a transport layer.

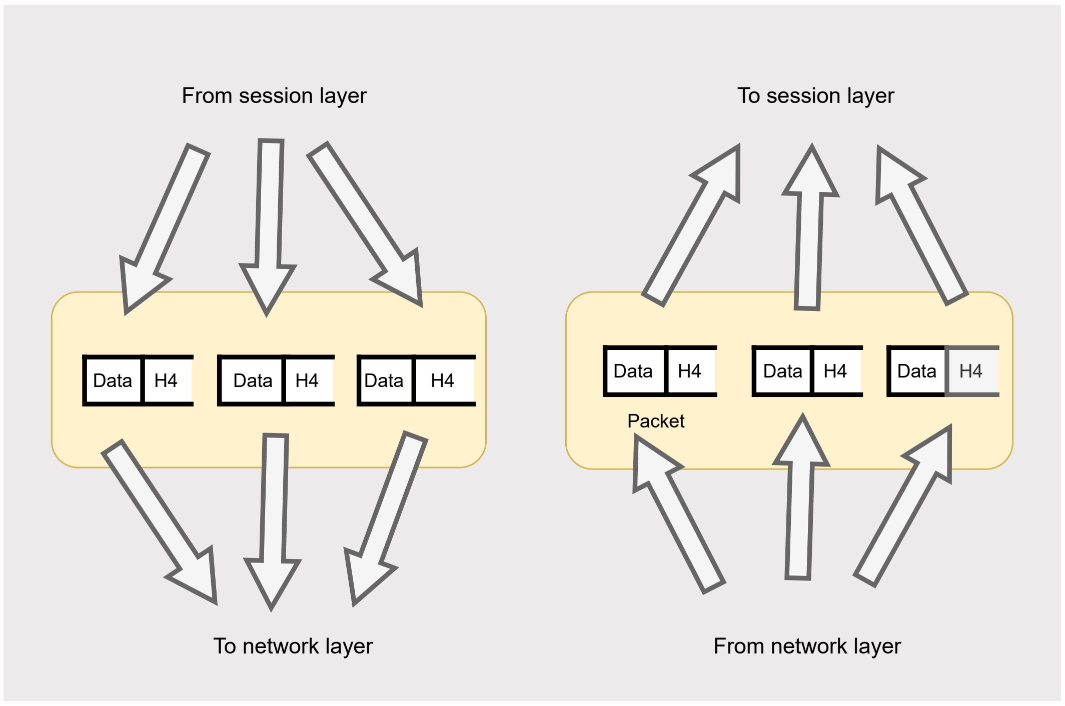


Figure 1.4: The transport layer [9]

### 1.1.5 Session Layer

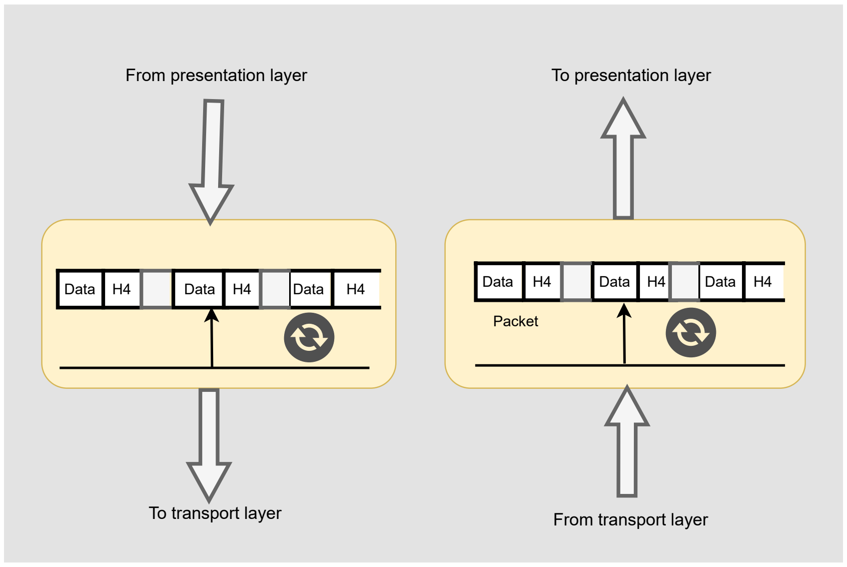


Figure 1.5: The session layer [13]

As its name suggests, it has a session between two users/nodes, the figure 1.5, the whole setup of the session and data exchanges, thereby monitoring the session and the security aspect as well [13].

### 1.1.6 Presentation Layer

It, figure 1.6, can be seen as a translator in a network[13]. It formats the data received and transfers them to the application layer, which is readable for humans and vice versa. In this layer, the following tasks are also performed.

- Character code translation
- Data Conversion
- Data compression
- encryption

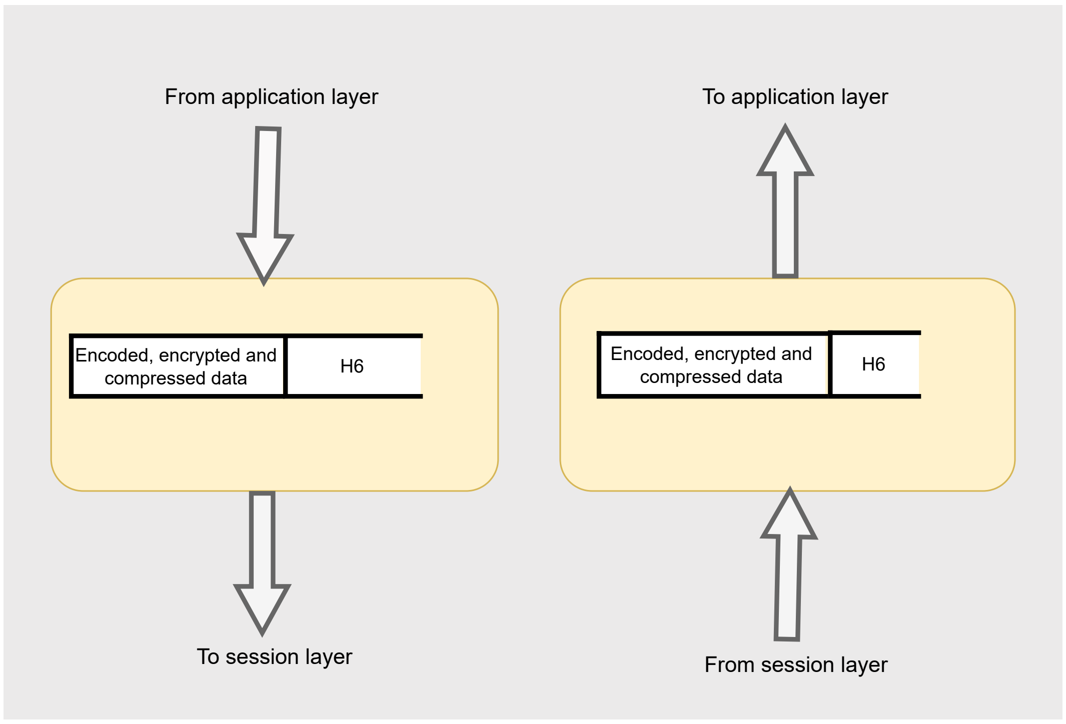


Figure 1.6: The presentation layer [9]

### 1.1.7 Application Layer

The user interaction occurs at this layer, figure 1.7. The familiar protocols running at this layer are HTTPS, HTTP, FTP, NFS, SMTP, and SSH.

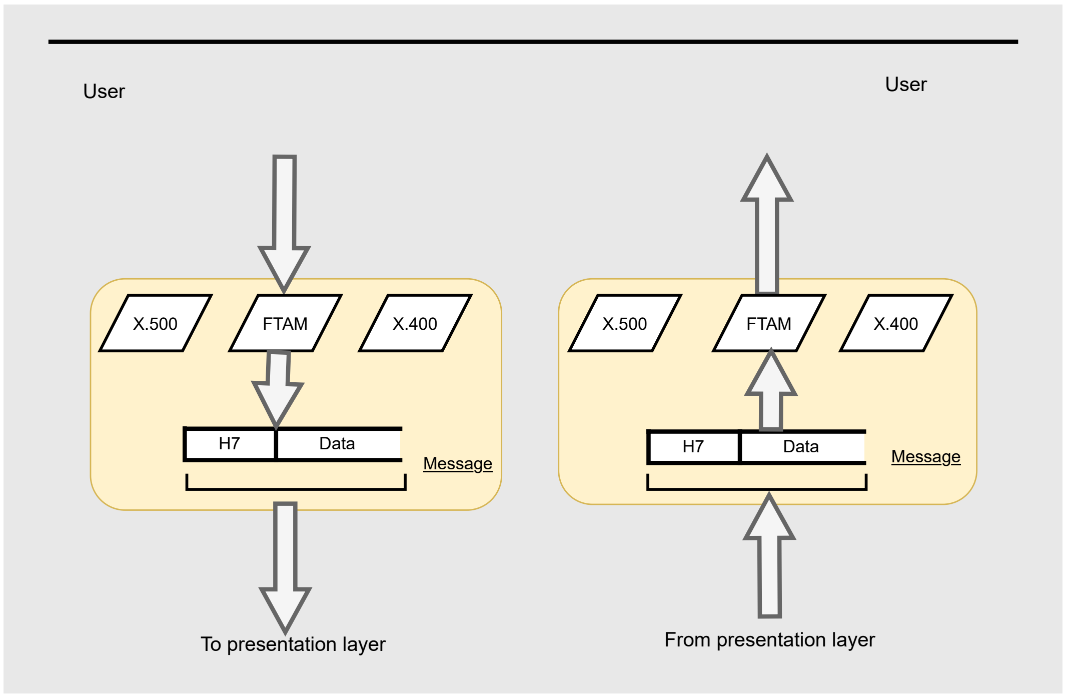


Figure 1.7: The application layer [9]

## 1.2 Other methods in FTS

In this area of work, the focus on FTP is in the foreground. File Transfer Protocol finds its usage in the application layer of the OSI model, and it is still widely used by many enterprises to transfer data between clients and servers. The major advantage of using FTP is that it uses no intermediary to carry out the data transfer, as we otherwise use browsers or emails to deliver data; in most cases, an application (FTP Client Applications) dedicated to the File Transfer System is used to perform the file transfer, says Robi et al. [17].

This paragraph discusses the various methods that are implemented in FTS. According to Lim et al. [14], they have proposed a C2CFTP which is an efficient and indirect file transfer between a client and a server structure. Because of the drawbacks they have pointed out in this paper, the main drawback is said to be indirect file transfer, which generates a lot of input/output of files that are processed by the server. This increases delay time and, thus, compromises efficiency. The C2CFTP proposes the idea of not storing the file in indirect transfers; in this case, the file needs not be opened or relayed to the subscriber.

In this article from Trabelsi et al. [20] they have introduced a covert file transfer protocol in which they try to covert the existence of communication, different from encryption, where the details of communication are encrypted. This is deployed in sending short messages and text files.CFTP is used unlike FTP, ICMP protocol, the IP record route option to transfer data, and the IP record route option is made some changes to get the desired result.

In the Master thesis provided by Alican [5], as the solution to the vast amount of data being shared and exchanged, they have introduced a Multi-File Transfer Protocol (MFTP), which solves the issue of time delay in downloading resources, whereby they distribute the files to be shared and then regather them, and found success in sharing and downloading the same data, even large files, without any delay.

## 1.3 Motivation

As the amount of data and transfers in a Firm increases, it is quite common that the employees, especially the developers and IT engineers, revert to the idea of automation. As clients increase, the File Transfer system is receiving a lot of incident enhancement and other requests daily; this is thereby the undertaking of the FTS engineers to minimize the effort to respond to the requests from clients, aiming at compatibility and effectiveness. Five tasks need to be addressed: log data need to be created for effective monitoring, and the front-end part needs to be taken care of as well, which might not be included in this part of the work.

The figure 1.8 explains how daily transactions are done in FTS manually; it has three main components: the Request for Product Development, first-level engineers, and the TSE team, who take requests and process them with their permissible rights. Then, FTS engineers have additional rights on the AdminFTS host and can perform more tasks with the admin rights. The RPD requests can be anything from creating an account in different business units to changing the protocols and changing login restriction policies.

RPD figures which would be allocated to a particular group, in this case to the first level of support or the FTS curators, take up the tasks, make necessary adjustments in the applications about the tasks and sometimes on the hosts, finish it, and resolve the RPD by commenting. The RPD can be of different severity and can be raised according to how the client wants to solve the problem.

The FTSAdmin server is the host with the host names, which can be invoked using the CLI to perform.
The tasks of the TSE team and FTS engineers are given briefly below.

Technology Solutions Engineering(First layer) is focused on the stability, performance, and functionality of the firm's products, and supports its product platforms

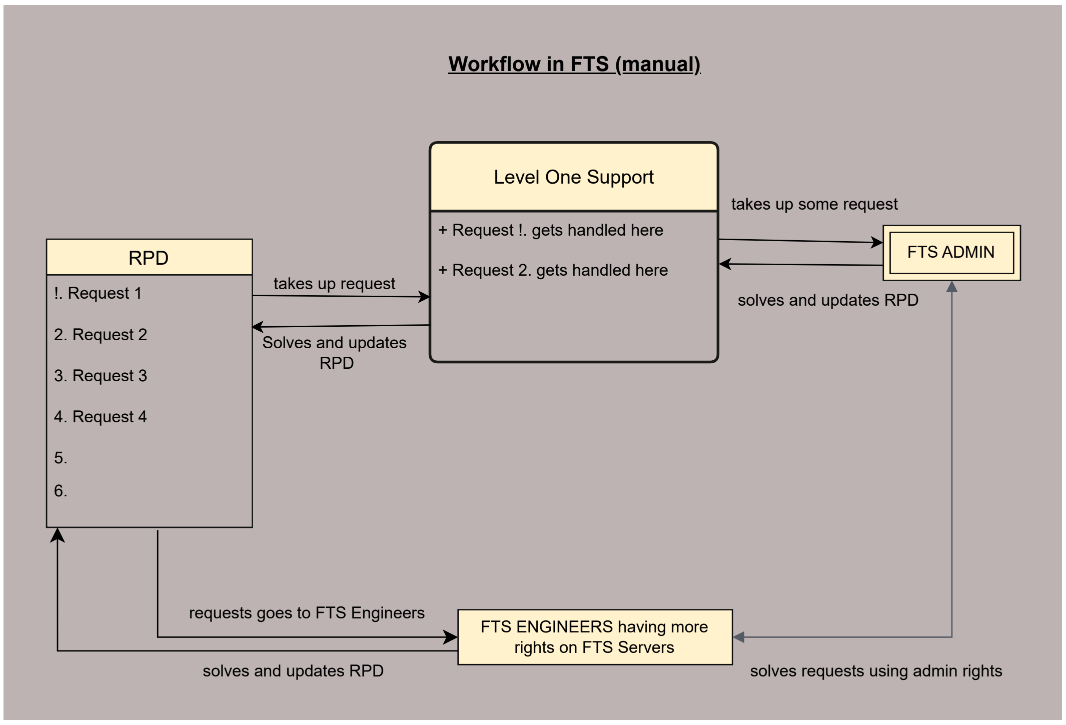


Figure 1.8: The manual process at Factset

at companies' client sites. Its 120,000+ workstations, 10,000+ web instances, mobile devices, and batch processes are supported 24/7, in close cooperation with the consulting support desk and operations staff. Client Technology Oversight, a subgroup of Technology Solutions Engineering, focuses on difficult (tier 3/4) technology issues and adapting products to be turnkey and seamless so that the firm minimizes the technology burden on its clients. Single-sign-on, virtualization, packaging, performance, and stability are driven by this team aligned with Engineering and Product Development. This team is also the escalation point for new technologies so that firms can be agile and adapt elegantly to changing technology conditions.

Strategic Client Technology, a subgroup of TSE, ensures that top clients by ASV are receiving personalized, directed, proactive assistance for rollouts and upgrades. This group also works to ensure that our clients have a single point of contact in our engineering group for roadmap discussions, as well as connectivity, performance, and stability conversations.

Desktop Solutions, a subgroup of TSE, focuses on automation, macros, and customized configurations that allow FactSet clients to build efficient workflows.

Again, the TSE group would hand over most of the scripts so that they could cater to the needs of the client. The automation tasks that we are attempting through this research would be production-ready, and the TSE team will run the scripts (python, shell scripts) on their CLI; that's how the processes go. In addition to that, they also perform troubleshooting techniques before escalating the request to FTS engineers. The troubleshooting steps can be given below to get an overview of what is happening under the curtain. Firstly, as soon as an RPD is filed, that means that when they get a request from the client, they would have to verify if the client has a valid subscription. In the RPD, they would have to check for the issues mentioned in the RPD about what is being questioned, and check whether it is an enhancement request, a question, etc. After that, it is necessary to check that the client can make a connection to the system servers. If everything is done and checked and works, it is recommended to add additional notes such as proxies, file lock permissions, etc. and later gather verbose logs using the command *sftp –vv*. The logs tell explicitly what the reason could be and, according to it, the TSE engineers either resolve the issue or reach out to the engineers in the File Transfer System.

The FTS engineers have additional rights on admin ftp, and their duties can be listed below.

- Manage scalable high availability File Transfer System (FTS) for use by Standard data feeds, Cornerstone, Enterprise Hosting, and Portfolio Batch services
- Installation, upgrades, and system configuration
- Internal tools for automation of client and vendor account management workflows
- FTS Monitoring and health check system
- Support of the FTS environment
- Migration to cloud-hosted infrastructure

Another important aspect of the research is the project management ticketing system, which is also essential for better communication between managers and customers; the main among them is Jira. In our work, we use RPD (Request for Product Management) as a ticket management system, and the API of it has been used to fetch details such as user names, host names, IP addresses, etc.

Surely, somewhere, somehow, in the history of computing, at least one manual has been written that you could at least remotely attempt to consider possibly glancing at.

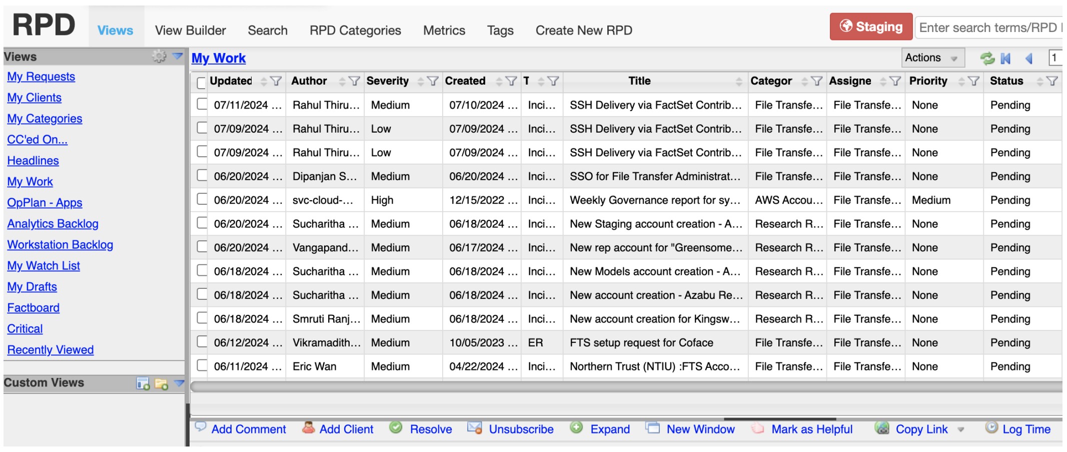
*Adam Rixey*

# Chapter 2

# Problem Statement

In Chapter 1 we have vividly gone through the importance of FTS and its various usages in enterprises, as we have understood if the clients become larger in number, to ease the process of transactions it is important to automate processes. Talking with the API of the application is the best way to automate the solution for robust processes, although bash scripts can also be composed to get the desired results. Python provides rather extended modules to ease processes and can be run in most operating systems.

The Request for Product Development is the ticketing system that is used in the firm to address issues such as providing access to hosts, creating users in a variety of services, for example AWS within the company, and also client requests for specific products, etc. A small overview of the RPD homepage is shown below 2.1



| Updated | Author | Severity | Created | T | Title | Categor | Assigne | Priority | Status |
|---|---|---|---|---|---|---|---|---|---|
| 07/11/2024 … | Rahul Thiru… | Medium | 07/10/2024 … | Inci… | SSH Delivery via FactSet Contrib… | File Transfe… | File Transfe… | None | Pending |
| 07/09/2024 … | Rahul Thiru… | Low | 07/09/2024 … | Inci… | SSH Delivery via FactSet Contrib… | File Transfe… | File Transfe… | None | Pending |
| 07/09/2024 … | Rahul Thiru… | Low | 07/09/2024 … | Inci… | SSH Delivery via FactSet Contrib… | File Transfe… | File Transfe… | None | Pending |
| 06/20/2024 … | Dipanjan S… | Medium | 06/20/2024 … | Inci… | SSO for File Transfer Administrat… | File Transfe… | File Transfe… | None | Pending |
| 06/20/2024 … | svc-cloud-… | High | 12/15/2022 … | Inci… | Weekly Governance report for sy… | AWS Accou… | File Transfe… | Medium | Pending |
| 06/20/2024 … | Sucharitha … | Medium | 06/18/2024 … | Inci… | New Staging account creation - A… | Research R… | File Transfe… | None | Pending |
| 06/20/2024 … | Vangapand… | Medium | 06/17/2024 … | Inci… | New rep account for "Greensome… | Research R… | File Transfe… | None | Pending |
| 06/18/2024 … | Sucharitha … | Medium | 06/18/2024 … | Inci… | New Models account creation - A… | Research R… | File Transfe… | None | Pending |
| 06/18/2024 … | Sucharitha … | Medium | 06/18/2024 … | Inci… | New account creation - Azabu Re… | Research R… | File Transfe… | None | Pending |
| 06/18/2024 … | Smruti Ranj… | Medium | 06/18/2024 … | Inci… | New account creation for Kingsw… | Research R… | File Transfe… | None | Pending |
| 06/12/2024 … | Vikramadith… | Medium | 10/05/2023 … | ER | FTS setup request for Coface | File Transfe… | File Transfe… | None | Pending |
| 06/11/2024 … | Eric Wan | Medium | 04/22/2024 … | Inci… | Northern Trust (NTIU) :FTS Acco… | File Transfe… | File Transfe… | None | Pending |

Figure 2.1: Request for Product Development

In the above figure, the queueing tickets can be seen; As the business day progresses, the tickets accumulate, and sometimes they can be unaddressed by the on-call engineer. This research in FTS is thereby planning to solve this particular problem whereby the engineer has more time to address important issues, taking care of servers other than clicking on the User Interface many times, hence saving time and speeding up the processes. As the RPD shown below comes with a well-arranged REST API structure, this attempt to automate cannot go wrong.

The other important requirement of this research is the Axway service; it is a French-American company that provides various solutions, including data monitoring, API development, and enterprise solutions. The other four tasks that exclude the generation of SFTP keys are performed using the Axway service. The engineers generally use the Axway interface to make adjustments to the requests, and now is again an attempt to talk to the API endpoints of Axway and thereby automate the processes.

In tackling the first problem, that is, the generation of SFTP keys for different hostnames and usernames, we are planning to perform our logic where we wrote a Python program to talk to the API of RPD, the ticketing system, fetch the mentioned details from the API JSON output, and use this information to generate keys with the respective RPD number, and save the private key to the database in Microsoft, simultaneously commenting out the RPD with the public key where the client can use this for inbound connections to the FTS servers.

In the other problems, we plan to carry out a similar procedure, where the REST API of Axway's service can be used to change the password for the existing users, change the protocol of given FTS/SFTP, etc.
The first task has been successfully carried out and seems to address the issues, the testing environment has given the desired results, and soon will be implemented in production environments. The project can be taken out, that is, making a front end, and can be implemented as a real problem solver. This is particular to the FTS department, similar steps can be made used in other departments in an enterprise such as back-end operations, database, orchestrations, and security.

Among all others, the major concern for the on-call engineers could be not so much a formulated Request for Product Development; in this case, it could be the missing information about server names, host names, etc., which in turn might cause the program to behave tangentially and might cause issues. Programmatically, we can make necessary adjustments to raise log errors, but in real scenarios, most of the time, the RPD would be again commented out to give the real-time information, which indeed is time-consuming. Basic error checks should also be performed on the client side to minimize these shortcomings.

In this study, the back-end portion of the automated application will be presented, and the Python code will be run from the command-line interface, where the production hosts run on Linux servers.

## 2.1 Technical Context

Most of the servers in the company where we carried out our research are derivatives of Red Hat (Linux Distribution), the Python programming language is used for scripting (Python 3.11 v), Visual Studio Code is used for the development environment, and iTerm is used for the UNIX command line interface.

The above information is put into the table below. 2.1.

| **Platform** | **Code Editor** | **Programming Language** |
|---|---|---|
| Linux/Unix | Vscode | Python/bash |
| MacOSX | Vscode | Python/zsh |
| Windows | Vscode | Python |

Table 2.1: Distributions and Editor as platforms

I can't go to a restaurant and order food because I keep looking at the fonts on the menu.

*Donald Knuth*

# Chapter 3

# State of the the Art

This chapter explores the current trends in this area, including its findings, the historical background of the attempts that have been made, the analysis of those structures and research, and different approaches.

## 3.1 Related Articles

According to Kornienko et al. [12], global digitalization is making tremendous changes in business solutions, revealing how this transformation has paved the way to server-client type architecture and addressed the back-end front-end structure of web applications, where the programming logic is performed on the server on the back end and the latter interacts with the user interface. They also listed some examples of back-end programming languages such as( Python, Ruby, Java, etc.) where they also need frameworks to make production-ready web solutions.

They also touch on API, where they define API as a set of rules used to communicate between the client and the software. API, according to them, secures the interaction. According to them, API has different types of architecture, and they are RPC, REST, SOAP, and GraphQL.

They also describe how the API schemes mentioned above differ from each other. The RPC specification provides API integration just like the HTTP API, thus getting almost the functionalities by forming a functional message to the contacting server. The server then processes this call and returns the result. The other schema, namely gRPC, paves the way for the old version, obliterating the bad functionalities and adding new capabilities of load balancing, message tracking, etc. They are used primarily in the field of IoT (Internet of Things).

The advantages of RPC/GRP are their simplicity in mechanism, excellent flexibility, and additional security available with it. Where the disadvantages count to the least abstraction is the inability to check the errors in queries.

Graph QL is introduced by Facebook, where data representation is done using graphs, unlike tables. It can get data of various complexities using one query and works very well as linking data; the major disadvantage would be low performance because of the nested or linked structure.

Another important schema or style is the Simple Object Access Protocol (SOAP), which uses the XML format and is specific to Web communication. In SOAP, the data are more secure and hence used in many protocols such as FTP, HTTP, and SMTP. The advantages are the way to use it collectively in various protocols, the existence of error handling, and the availability of metadata. Because it only supports XML and its need for a large bandwidth, it is heavy, making it difficult to parse the data.

REST is an alternative to existing styles and is a lightweight stateless architecture. By stateless, it means that no session information is saved on the server or anywhere; i.e., what is needed is in the query itself. XML and JSON formats are mostly used in communication stages. It also uses HTTP methods to communicate: GET, POST, PATCH, PUT, and DELETE. The advantages include its very simple architecture, caching availability, and acting as a direct agent to the server, with no third party involved. The disadvantages include the high load on the server side.

In the same article, they also introduce the requirements for using REST: they are authentication, the availability of logs, and the logic to connect to the application (program).

And for the API development, the access rights must be clearly established by the developers. That is, the checking of vulnerabilities of API endpoints. Since the REST API is stateless, the authentication also takes place from endpoints. The common authentication schemes are API Key, JSON Web Tokens, Cookie-based authentication, etc.

As we have briefly gone through the API architecture, its different schemas, and its methods, as much as that is the awareness of automation using Python as a programming language and related works, in the article written by Akash et al. [21] they have stressed the importance of CRM (customer relationship management), which somewhat resembles the FTS client-server transactions, using Python as the tool to perform the CRUD operations, whereas CRUD stands for Create, Read, Update, Delete. The characteristics of the Python library `requests` are reviewed and later implemented to make API calls and other tasks.

In the above-mentioned article, they provide code snippets, where they go step by step defining the functions to get the JSON data and then further manipulating the data for desired results. Firstly, they have done the authentication with the Microsoft Dynamics API, using the OAuth2 protocol , the important aspect of it is that the token is stored in memory and does not persist, while in our environment the firm, in which we carry out our research, provides its authentication mechanism, which we can discuss in later parts of this article. Requests library is implemented to perform a CRUD operation, whereby the corresponding Python variants would be: GET - READ, POST - Create, and PATCH - Update.

The workflow of their project is listed below to make it easier to comprehend. The authentication steps are as follows. Firstly, they initialize the parameters; it is important to get the authentication token, so afterward a POST request is created. The request contains various other parameters such as password, username, etc. Then the API performs the checks, the response is sent back in JSON format, and the required token is extracted using the Python dictionary.

Secondly, in the attempt to retrieve data from a random table, the workflow would be as follows. A GET request is made after a variable is set to the API URL, and then the corresponding function is created. The response is then encoded in UTF, and then the JSON data is parsed.

To make changes to the program, they have made a patch request to update the table present in it. The corresponding workflow would be the following. Using the credentials from the previous method, a function is created. In order to update the data, the keyword `patch` is included in the execution call. Check the responses after the JSON data received is passed using the appropriate Python library.

Similarly, another function that uses the `post` method to insert data into a table is also given in the project. Furthermore, performance and efficiency checks have also been carried out, which justify the quantitative measure that they promised for this particular project, which is quite improved when Python is used for the implementation. All relevant data can also be found on this Web page [3].

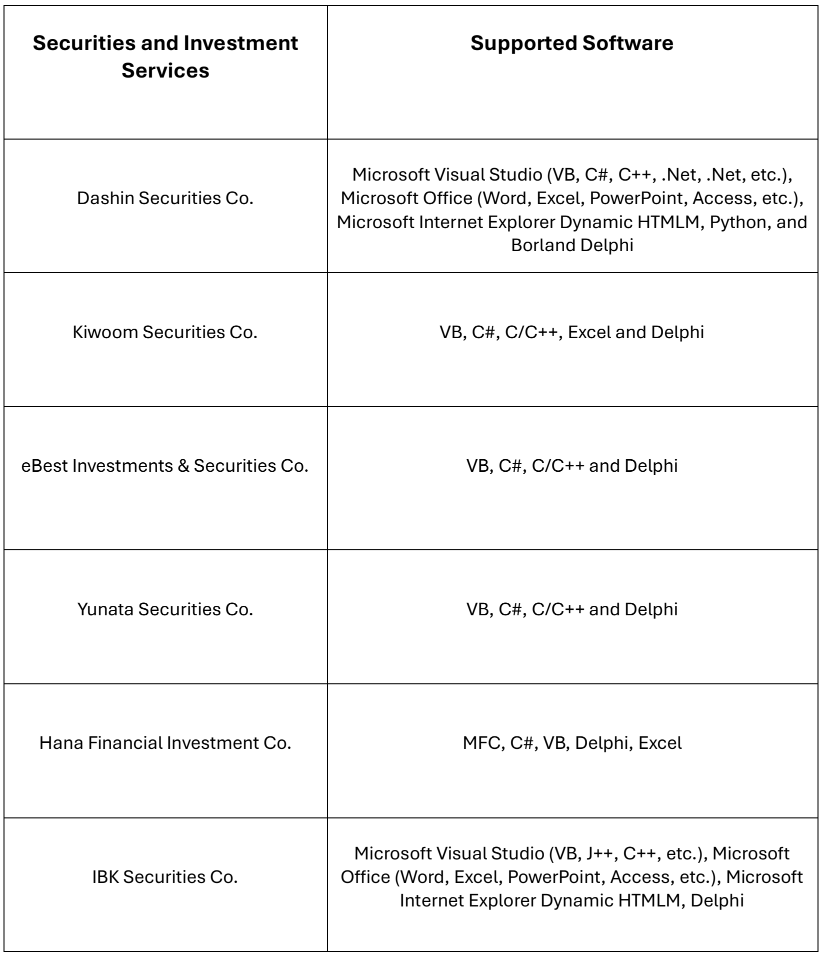
As we investigate further, the article presented by Batni et al. [8] provides an extensive overview of the automation of the testing process in software development using the Robot Framework. As the authors mention, it uses a keyword type of development, which means that the syntax is in table form, and the testing is done uniformly. One can also build custom libraries for it in the major programming languages, since it provides XML output. The results shown in the paper have produced significant findings that show the success of automating tasks rather than manually doing them, which we can be confident of for future work on the topic.

In a study conducted by Ryu et al. [18] they used Python scaling using large volumes of data in the investment and security sectors. The data is offered by such companies in large volumes on their websites, which subsequently can be accessed on particular open APIs, using particular programming languages, which in this case used Python. They conducted a study on two different companies and found differences in authentication and transfer methods. The table below shows the companies in South Korea where open APIs are supported; HTS is a system that is being used to trade stocks at any time.

| Securities and Invest Services | HTS | Open API |
|---|---|---|
| SK Securities Co. | Yes | No |
| Hanwha Investment & Securities Co. | Yes | No |
| Shinyoung Securities Co. | Yes | No |
| Bookook Securities Co. | Yes | No |
| Hanyang Securities Co. | Yes | No |
| Samsung Securities | Yes | No |
| Kyobo Securities Co. | Yes | No |
| KB Securities Co. | Yes | No |
| Hyundai | Yes | No |
| Daishin Securities Co. | Yes | Yes |
| Hi Investments and Securities Co. | Yes | No |
| NH Investments and Securities Co. | Yes | No |
| Kiwoom Securities Co. | Yes | Yes |
| eBest Investments and Securities Co. | Yes | Yes |
| Mirae Assests Daewoo Co. | Yes | No |
| Yuanta Securities Co. | Yes | Yes |
| Eugene Investment Co. | Yes | No |
| Korea Investment and Securities Co. | Yes | No |
| KTB Investments and Securities Co. | Yes | No |
| Hana Financial Investment Co. | Yes | Yes |
| IBK Securities Co. | Yes | Yes |
| Cape Investments and Securities Co. | Yes | No |
| Total | 22 | 6 |

Table 3.1: Companies in the study [18]

Of all the companies, they have taken out two companies with which they make API calls using Python; further studies show that many of the companies have yet to support Python in open API. The table is shown below.



| Securities and Investment Services | Supported Software |
|---|---|
| Dashin Securities Co. | Microsoft Visual Studio (VB, C#, C++, .Net, .Net, etc.), Microsoft Office (Word, Excel, PowerPoint, Access, etc.), Microsoft Internet Explorer Dynamic HTMLM, Python, and Borland Delphi |
| Kiwoom Securities Co. | VB, C#, C/C++, Excel and Delphi |
| eBest Investments & Securities Co. | VB, C#, C/C++ and Delphi |
| Yunata Securities Co. | VB, C#, C/C++ and Delphi |
| Hana Financial Investment Co. | MFC, C#, VB, Delphi, Excel |
| IBK Securities Co. | Microsoft Visual Studio (VB, J++, C++, etc.), Microsoft Office (Word, Excel, PowerPoint, Access, etc.), Microsoft Internet Explorer Dynamic HTMLM, Delphi |

Table 3.2: API supporting companies in the study [18]

Connecting to the open API of Daishin Securities CYBOS plus, which is the name of the open API, must be incorporated; thus, after coding CYBOS plus and improving the connection to the company server, authentication and all other checks are done using this CYBOS portal.

To connect to the open API of eBest Investments, they provide a package called

Xing-API, which, as above, enhances the connection to the server. The relevant data are given in the article by Ryu et al. [18].

The attempt of Chauhan et al. [10] to extract efficient data using the Python programming language and the application programming interface of the beam therapy equipment, which turned out to be much better for obvious reasons than manual extraction. It is a simple experiment, as shown in the picture below.

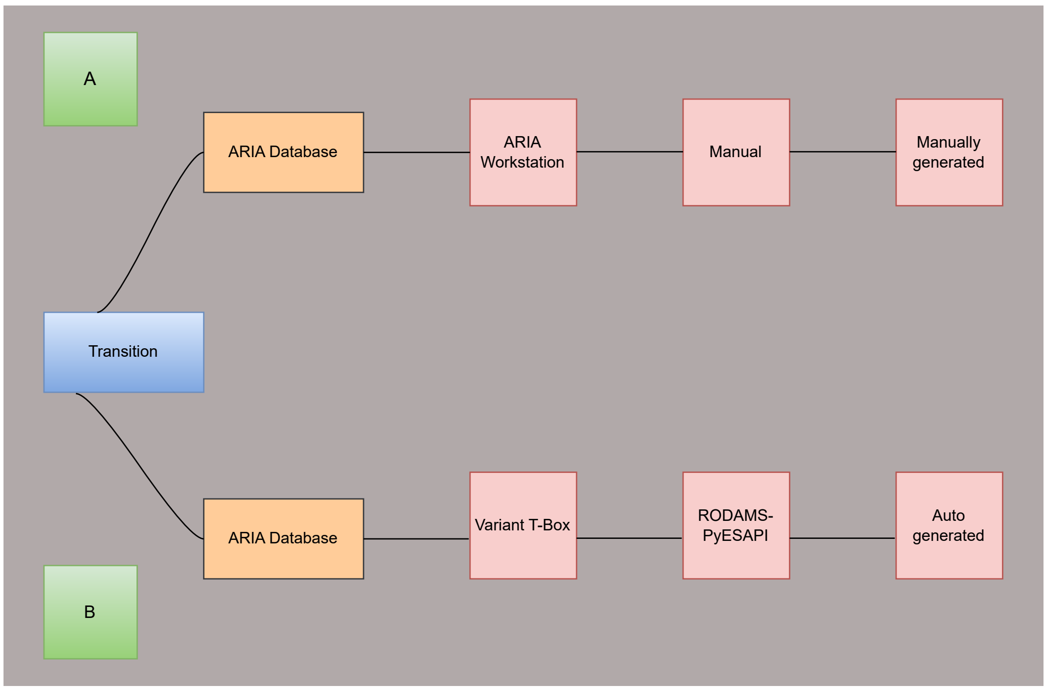


Figure 3.1: A. manual extraction, B. python implemented [10]

The facilities used in the study, particularly for radiation therapy and radiation oncology, developed an automation strategy, where their native instrument is named TPS (Treatment Planning System); in this approach, they have created an environment using the vendor's API, in this case Varian Medical Systems , and thus the newly created environment is PyESAPI, which retrieves data from the Varian TPS database. The script developed is named RODAMS, Radiation Oncology Data Mining Script, which has resulted in immense improvement. The details of the pilot project can be found in Chauhan et al. [10].

## 3.2 Standards

In this attempt to inform oneself about the existing standards, although the REST API (Representational State Transfer) structure that we discussed earlier in the other

capital, a study conducted by Murphy et al. [15] where they have analyzed 32 open guidelines for the design of the REST API, the attempt is to distinguish between the API and to obtain a best practice of designing a custom API. Many large companies are making their API available so that the public can access the data and services, says Brad et al. [16]. They have scrutinized 32 REST API documents found mainly through Git Hub.
REST API is a stateless protocol, that is, any change in the API implementation does not affect the user's code or data and has a separate server-client setup according to Roy et al. [11]. The study collected the most important topics 27, where they coded the rules and looked for similarities and differences in the given guidelines, and again categorized the set into the 10 most used ones. The 10 most stood-up categories of the 27 chosen are given below.

- Status Codes
- Response Structure
- Standard Methods
- Naming
- Versioning
- URL
- Error Responses
- Backward Compatibility
- Documentation
- Custom methods

They also highlight some differences in the API structure used by different agents, for example, when it comes to versioning, Google and Zalando recommend using three-number versioning, while others suggest a two-number versioning. Another issue is the placement of the version at the beginning or in the URL and also highlights some differences in the API structure used by different agents. For example, in versioning, Google and Zalando recommend using three-number versioning, while others suggest a two-number versioning. Another issue is the placement of the version at the beginning or in the URL.

Another important divergence could be seen in backward compatibility, which means the older version code could also be used in newer versions, then the newer version is said to be backward compatible, says the work of Murphy et al. [15], some guidelines discourage breaking compatibility, although it is costly to change codes for every new version. Some guidelines encourage backward compatibility.

The work of Tarkowska et al. [19] presents 11 tips to build usable REST APIs in a particular industry of health sciences. REST APIs are easier to develop than classical SOAP (Simple Object Access Protocol). Although the requirement documentation can be found on the two websites, namely [4] [7], about obscure explanations, the study has decided to present its findings. The suggested tips for building a useful REST API are as follows. It is good to always have good documentation attached while creating an API; explicitly declaring or presenting URLs for the API is also advised. In order to maintain data, it is recommended to use standardized formats such as JSON. The usage of standardized HTTP headers and responses is encouraged. It is also very important and highly recommended to ensure the safety of the products across networks. It goes without saying that one should share the links with clients for the execution of created APIs. For authentication, it is also advised to use standard ways such as SSO's. Ensure that the API is highly available, which is important according to the study mentioned above. Despite these suggestions, it is also important to version the API every time it is rolled out to be released.

I have always wished for my
computer to be as easy to use as my
telephone; my wish has come true
because I can no longer figure out
how to use my telephone.

*Bjarne Stroustrup*

# Chapter 4

# Architecture and Implementation

This chapter is divided mainly into two major subcategories: the architecture of the file transfer system setup and the implementation of the problem tasks , where each task is then separately done through Axway's User Interface. In the architecture part, we aim to give a general view about the set-up, what Axway has provided as a generic approach which can be completely independent of the companies' needs. The Implementation part ensures the necessary steps for the successful implementation of the aforementioned tasks, with appropriate workflow diagrams and some images pertaining to the tasks.

## 4.1 Architecture

In this chapter, we examine the system architecture we use for the setup: its key components and modules, data flow in the system, technologies used, and security considerations.

Firstly, we will examine the Axway secure file transfer setup where we conducted our study. Given in 4.1 is the setup for one DMZ and one Secure Transport Server, whereas in actual use cases there would be multiple servers. The secure transfer provided by Axway helps organizations send files securely inside and outside the organization. In addition, it supports file handling and routing, such as submitting files, keeping the files in FTP-like folders, and accessing ports with a customizable workflow. It provides a high level of security with easy-to-use governance and configuration features, such as configurable workflows, which we will later see in this chapter and administration.

### 4.1.1 Key Components

In the figure given below 4.1, we can see three different layers of networks (Secure Network, Perimeter Network, and Public Network) separated by dotted lines. The public network, which means the public Internet we use, contains various servers and clients; in terms of an establishment/firm, it is called partner servers and clients. The public network is self-evidently kept in check using the first layer of protection using Firewall 1. Requests from the public network pass through the DMZ(Demilitarized Zone) Server provided by Axway, check security credentials, and act as a primary layer of protection.

In the third section of the network, namely the secure network, where the secure transport server is again protected by another firewall, the figure can be identified by the name Firewall 2. In this secure layer, as per the figure, along with the secure transfer server, the internal server (in this case, FTP Server) and clients are located. The arrows show the requests made by each entity and how the workflow is being carried out. In simpler terms, requests from the outside are passed through the DMZ server to the secure transfer server, which regulates the traffic to the internal server.

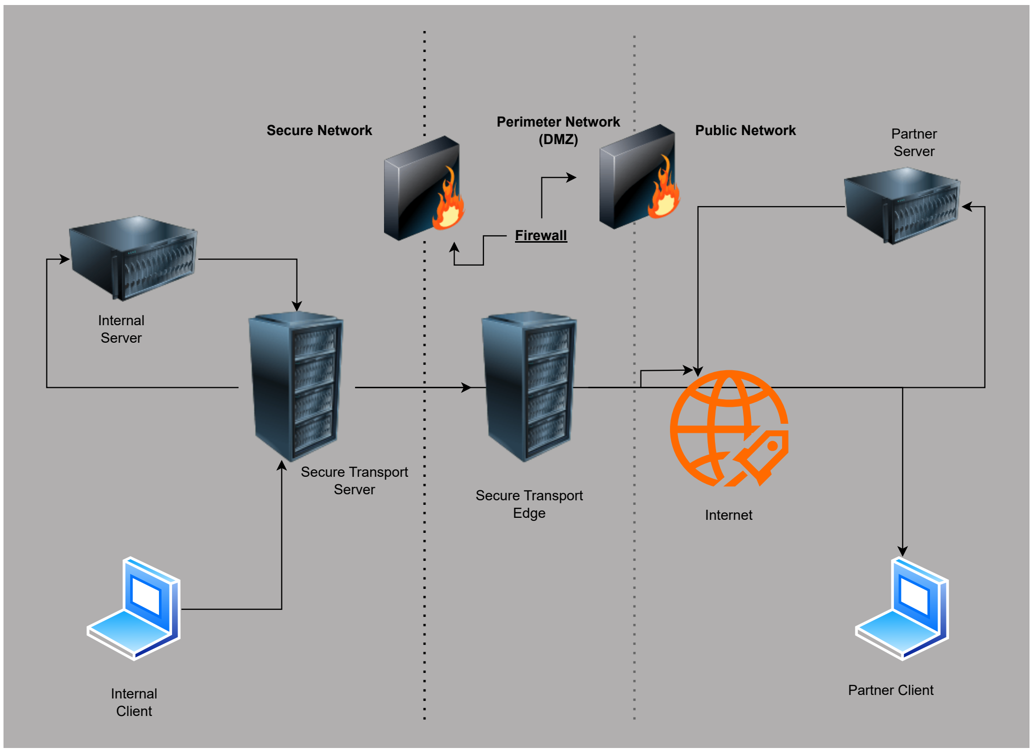


Figure 4.1: Axway Secure Transport Setup

Secure Transport is expected to work properly with any client or server software that complies with the following: FTP, SFTP, HTTP, etc. Axway supports the given platforms. The Operating Systems such as SUSE Linux (12.x,15.x), RHEL (8.x and 9.x), Rocky Linux 9, Oracle Linux (8.x, 7.x). The Microsoft SQL Servers (2017, 2019, 2022) and almost all the main Internet browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (except for Admin UI), etc. Authentication providers and directory services like RFC 4510 - LDAP v3 and SSO (Single Sign-On) SAML 2.0 (Administrators and end-users) and Kerberos for end-users and proxy software such as HTTP reverse proxy, SOCKS5, SecureTransport Edge.

### 4.1.2 On-Premises Installation

As we have used in this setup, an on-premise installation, the installation guide provided by Axway [1] gives a sufficient amount of information to proceed with the setup. There are three ways of deployment: Standard cluster deployments, Enterprise cluster, and standalone cluster deployments. The **standard cluster** uses an embedded PostgreSQL database in each node and does not depend on an external database. An **enterprise cluster** allows a high-performance cache management layer that significantly improves efficiency, provides near-linear scaling of up to 20 nodes, and enables very large-scale configurations. An enterprise cluster requires the organization to provide and maintain an external database and a high-performance shared file system for the user files, and the supported external databases on this are: PostgreSQL, Microsoft SQL Server, Oracle Database. **Standalone clusters** are suitable for small-scale operations. It can use either the embedded database or an external one and a local file system for user files.

Firstly, the system/hardware requirements if one is using secure transport on-premises setup.

| | **Secure Transport Server** | **Secure Transport Edge** |
|---|---|---|
| CPU's | 4 | 2 |
| RAM | 16GB | 8GB |
| Free disk space | 200GB | 100GB |

Table 4.1: Minimum hardware requirements

Secure transport can be deployed on Linux and Windows, while Linux is recommended. The required operating system package dependencies are glibc, glibc-langpack-en, zlib, libaio, libxslt, and libxml2. Furthermore, the Axway installer and operational script requirements are bash, awk, grep, ps, sed, which, tar, gzip, unzip.

### 4.1.3 Requirements for Microsoft SQL Server Databases

As we use Microsoft Database for this work, it is worth mentioning the necessary steps to set up such a database. Information for setting up other databases (e.g. Oracle, Postgres) can be found here [1].

- The Microsoft SQL Server collation must be case-insensitive
- SecureTransport can connect to an external Microsoft SQL Server database over a plain or secure connection. A certificate must be present for both secure and non-secure connections, and the external database server's certificates that contain keyUsage extensions MUST also have the digitalSignature indicator enabled
- The database must have READ-COMMITTED-OPTION enable, to check whether its is enable we could execute the following querry

  ```
  SELECT is_read_committed_snapshot_on FROM sys.databases
  WHERE name = yourdatabase
  ```

- These filegroups should be defined with at least one file in each filegroup
  1. ST-DATA- configuration, account, sites, and certificates
  2. ST-Filetracking- file tracking tables
  3. ST-Serverlog- severlog tables
- For the log-transfer or log-entry maintenance application, one would need to add two additional file groups: ST-Filetracking-Archive and ST-Serverlog-Archive. Obviously, the user must have writing permissions
- The default filegroup for the DB user must be ST-Data
- The login of the Secure Transfer must be mapped to the user of the database for database access
- MS SQL SERVER Authentication is the authentication mode used
- Either the db-owner role or one of the three fixed roles—db-datareader, db-datawriter, and db-admin—must be held by the user. Verify that the user is authorized to run the stored procedures if you have a custom setup. The admin role allows users to make modifications to the database schema. This position is necessary for new installations, service packs, and upgrades

- The DB password must not have curly brackets and special characters [1]
- Important information during installation
  1. Database port number
  2. Database username and password
  3. Database name

For the initial configuration, SecureTransport provides a configuration account with a default password. After the initial setup is completed, change the default password before starting the SecureTransport setup.

Following completion of the basic setup requirements, it is crucial to check the checklist provided on their website [2]. Type *https://<servername>: -<portnumber>* to log on to the server, where <servername> is the IP address or name of the server you wish to configure and <portnumber> is the SSL port number that we set for the Administration Tool when it was installed. When SecureTransport is operating as a non-root user, the default port number is *444* or *8444*. In the upcoming chapters, we will observe how the suggested tasks are carried out using the secure transport's graphical user interface at the outset, and in the implementation section, we will demonstrate our attempt to automate it using a command-line interface.

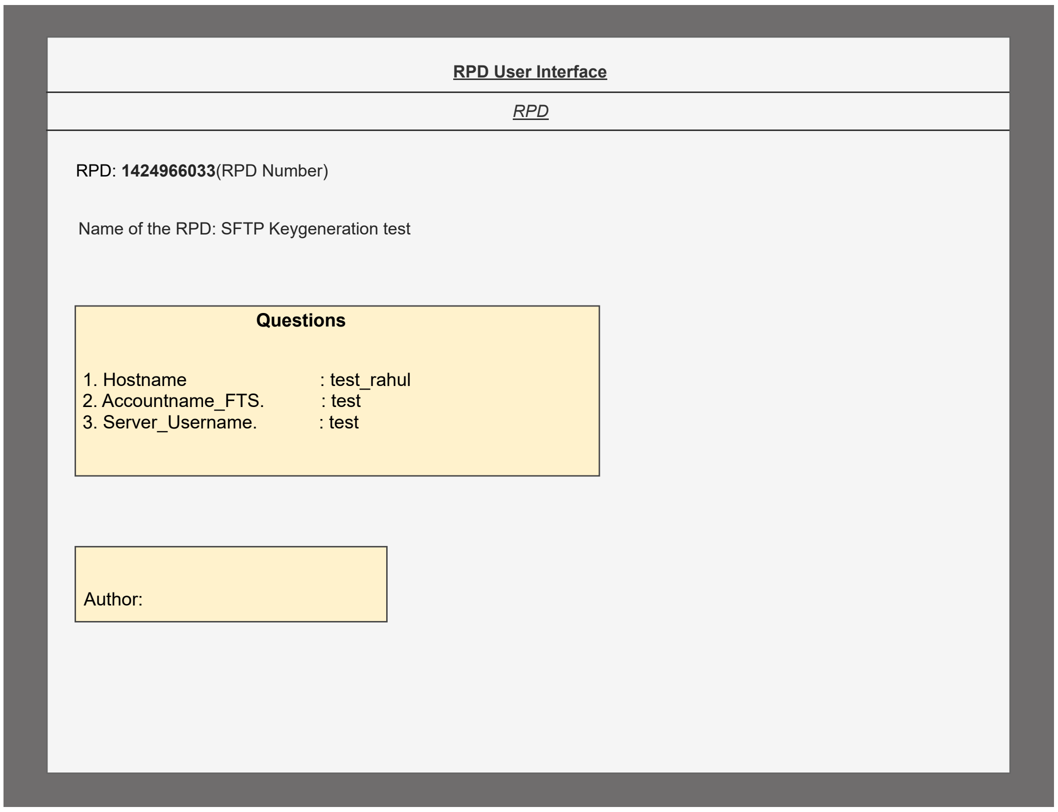
## 4.2 Implementation

In this section of the chapter, we will delve deeply into the steps of executing the five tasks mentioned at the beginning of this research. Images of the GUI and explanations of the steps to be carried out will be discussed in the first part, and in the latter part, the corresponding Python code will be presented.

The figure 4.2 is an abstract view of the RPD user interface. The figure captured for reference is also given below 4.3. From this diagram, we can deduce key information for starting the task, namely the RPD number, hostname, and server name. Additional data will also be present, like generic data that one requires to make sure that the RPD corresponds to the request for the File Transfer System. An overall idea of the task, primarily in this case of generating keys for secure transfer, could also be recognized. Another important aspect is the author who created this RPD and the person to whom this is aimed. In addition, the Application Programming Interface (API) is also available. It is well documented and is being used in later stages, especially in the automation/implementation part, to fetch the credentials. Further API calls can also be made; respectively, to the ticket system being used, it varies.

### 4.2.1 Usual Workflow (using GUI)

We will first discuss the outline of the RPD given in the figure 4.2. In it, the important entries for the task are also given. In this case, for the generation of SFTP keys, the main entries are the **hostname** and the **servername**. The department responsible for this particular request for product development is also given; it is the FTS (File Transfer System). After making this request by filling out the RPD Questionnaire, the RPD is requested to make the enhancement.



RPD: **1424966033**(RPD Number)

Name of the RPD: SFTP Keygeneration test

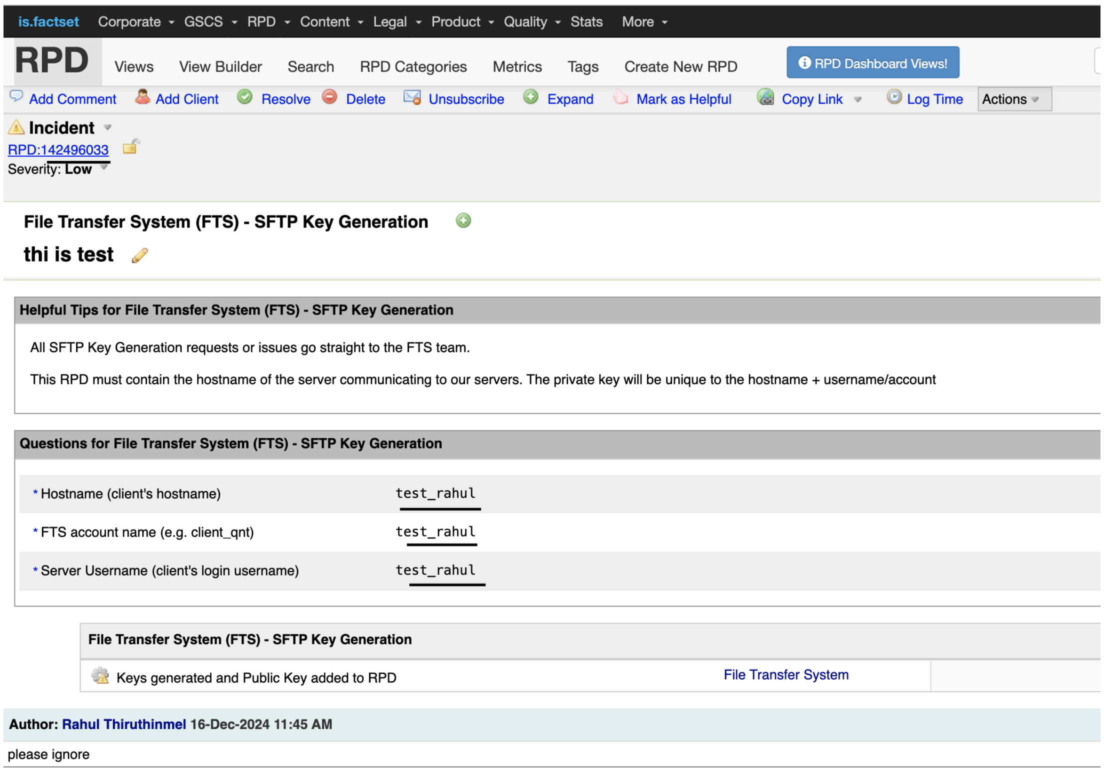
**Questions**

| | |
|---|---|
| 1. Hostname | : test_rahul |
| 2. Accountname_FTS. | : test |
| 3. Server_Username. | : test |

Author:

Figure 4.2: RPD GUI Representation

In the figure 4.3 provides a screenshot of the requested RPD, with the conditions needed for the creation of the SFTP key generation. The author of this RPD can also be seen, along with the type of request mentioned in the top left corner as **incident**. The other important options in this case would be to Add Comment, by which additional comments are added. If something needs to be addressed additionally or is missed, those can be added as a comment later. As the name suggests, add client adds different clients to the request, and lastly, resolution of RPD, which is important right after the problem is solved or the request is completed.



Incident
RPD:142496033
Severity: Low

File Transfer System (FTS) - SFTP Key Generation

thi is test

Helpful Tips for File Transfer System (FTS) - SFTP Key Generation

All SFTP Key Generation requests or issues go straight to the FTS team.

This RPD must contain the hostname of the server communicating to our servers. The private key will be unique to the hostname + username/account

Questions for File Transfer System (FTS) - SFTP Key Generation

| | |
|---|---|
| * Hostname (client's hostname) | test_rahul |
| * FTS account name (e.g. client_qnt) | test_rahul |
| * Server Username (client's login username) | test_rahul |

File Transfer System (FTS) - SFTP Key Generation

Keys generated and Public Key added to RPD    File Transfer System

Author: Rahul Thiruthinmel 16-Dec-2024 11:45 AM

please ignore

Figure 4.3: RPD for SFTP key generation

## SSH Key Generation

The first task is to create the SFTP key, and the steps we need to take on Linux hosts are given below. The steps would be the following.

- input 1 - client username get it from RPD
- input 2 - client hostname get it from RPD
- to generate SSH key-pair use

```
ssh-keygen -t rsa -b 2048 -N "" -f fts_rsa
```

- open the public key file, read the first line, and remove the last token (username who generated the file and on what host). Split the string by space delimiter and remove the third element of the string and write the string after the removal of the third element
- copy the SSH public key and post to the rpd

- simultaneously copy the SSH private key and store it in the database under the host and server names SQL Query for that is

```

DECLARE         @return_value int

EXEC    @return_value = [dbo].[sit_CreateSftpCredential]
                        @host = N'<client-hostname>',
                        @username = N'<client-username>',
                        @privateKey = N'private_key'

SELECT 'Return Value' = @return_value

GO
```

Listing 4.1: insert private key to database

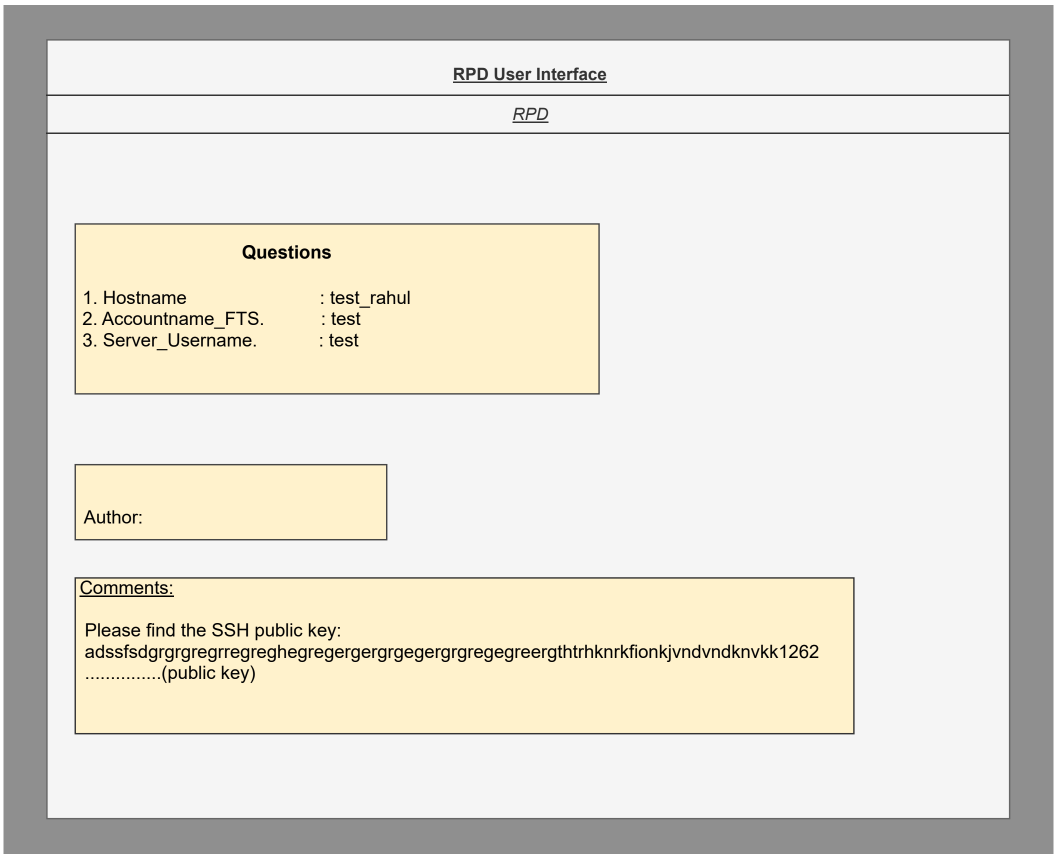
- update RPD with a comment stating that the SSH key pair was successfully generated for the client: <client username> and host: <client hostname>.

The RPD will then be commented on with the SSH public key after copying ; the representation of it is given below in figure 4.5. In normal business, if some questions exist, the RPD would be kept open. Let us say if the client cannot access the FTS account even after getting the SSH public key, the engineers would have to check and establish a connection; instances like that might occur. Normally, therefore, the RPD would be kept open for some more business days, and later, when there are no queries to the particular RPD number, it is then selected to resolve and save.

In figure 4.4, one can see that below the *questions* field there is the comment section where the support engineers, in some cases clients and engineers, comment out, make interactions, and later these recorded interactions can be used for references. This is exactly the case we discussed in the previous paragraph. The normal life cycle of RPD varies from two weeks to one to two months.

Adding to that, the author section on the RPD shows who has executed this RPD and the assignee section is the one or the team that is asked to carry out this task. In our test case, the author and assignees are the same. In use cases, one could see many comments or discussions in this section.

Figure 4.5 is the image captured for reference. Additional points to be remembered are the severity of the RPD, depending on how fast the client wants it to get resolved; the FTS team adjusts the severity; there is also a scope of parsing the severity and automating the process accordingly.



**Questions**

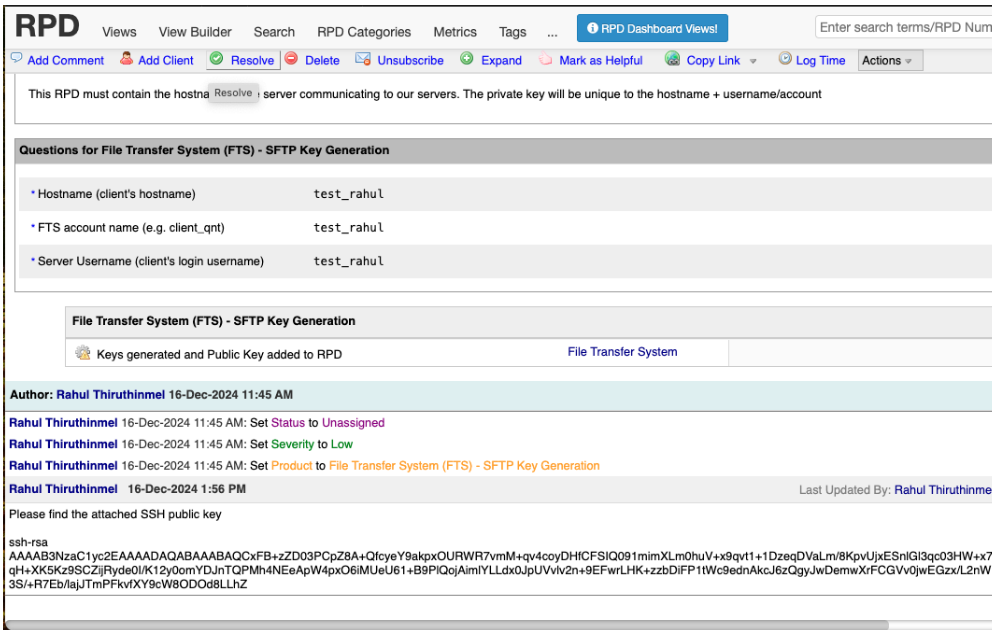
| | |
|---|---|
| 1. Hostname | : test_rahul |
| 2. Accountname_FTS. | : test |
| 3. Server_Username. | : test |

Author:

Comments:

Please find the SSH public key:
adssfsdgrgrgregrregreghegregergergrgegergrgregegreergthtrhknrkfionkjvndvndknvkk1262
...............(public key)

Figure 4.4: RPD GUI Commented with Public key



This RPD must contain the hostna Resolve server communicating to our servers. The private key will be unique to the hostname + username/account

**Questions for File Transfer System (FTS) - SFTP Key Generation**

| | |
|---|---|
| * Hostname (client's hostname) | test_rahul |
| * FTS account name (e.g. client_qnt) | test_rahul |
| * Server Username (client's login username) | test_rahul |

**File Transfer System (FTS) - SFTP Key Generation**

| | |
|---|---|
| Keys generated and Public Key added to RPD | File Transfer System |

**Author:** Rahul Thiruthinmel **16-Dec-2024 11:45 AM**

Rahul Thiruthinmel 16-Dec-2024 11:45 AM: Set Status to Unassigned
Rahul Thiruthinmel 16-Dec-2024 11:45 AM: Set Severity to Low
Rahul Thiruthinmel 16-Dec-2024 11:45 AM: Set Product to File Transfer System (FTS) - SFTP Key Generation

**Rahul Thiruthinmel 16-Dec-2024 1:56 PM** Last Updated By: Rahul Thiruthinme

Please find the attached SSH public key

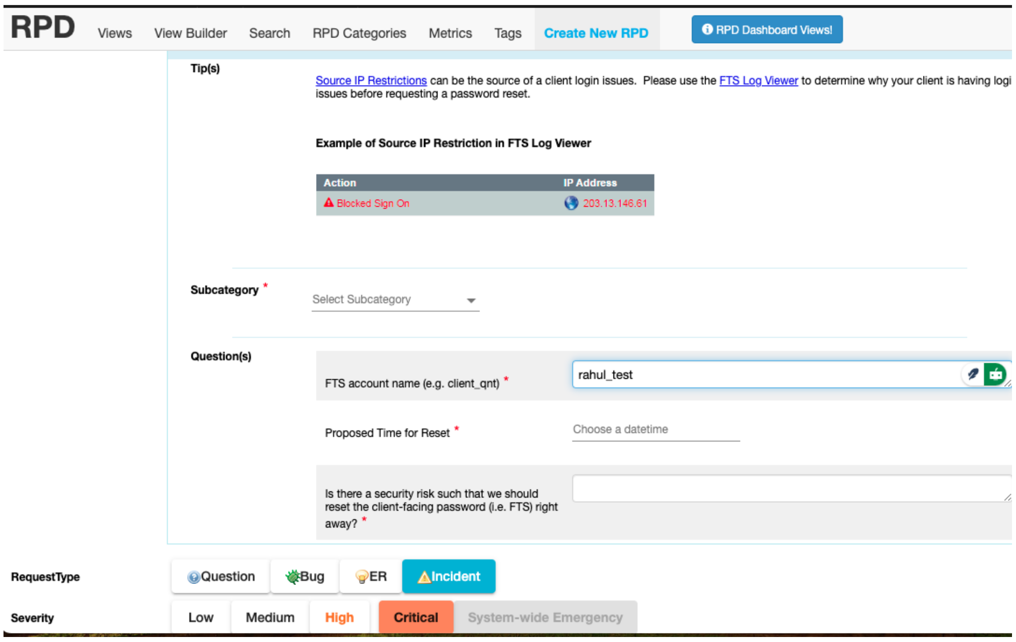
```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCxFB+zZD03PCpZ8A+QfcyeY9akpxOURWR7vmM+qv4coyDHfCFSIQ091mimXLm0huV+x9qvt1+1DzeqDVaLm/8KpvUjxESnlGl3qc03HW+x7
qH+XK5Kz9SCZijRyde0I/K12y0omYDJnTQPMh4NEeApW4pxO6iMUeU61+B9PlQojAimlYLLdx0JpUVvlv2n+9EFwrLHK+zzbDiFP1tWc9ednAkcJ6zQgyJwDemwXrFCGVv0jwEGzx/L2nW
3S/+R7Eb/lajJTmPFkvfXY9cW8ODOd8LLhZ
```

Figure 4.5: RPD commented out with public key

This will conclude the generation of the SFTP key. Next, we will take a look at how password reset functions in the Axway User Interface.

### Password Reset

On normal business days, we receive a request to change the password for a given username, given in the figure below. The figure 4.6 shows the account whose password should be changed; in this case, it is the *rahultest* test account. The type of request is an incident, and the severity is denoted as critical, and the modes can be changed according to how the client wants the request processed. In this case, it is critical, which means that it must be handled urgently in 10 to 15 minutes. Another important aspect is the resolution, as these RPDs take up memory and space on the central server; it is advised to resolve these RPDs, as this resolution takes up much lesser space compared to the actual storage when the RPDs are live; later, these RPDs can be revoked using their number if there occurs an instance where we have to refer to particular actions or processes which are interconnected, etc. Senior engineers normally assume this duty of resolution.



Tip(s)

Source IP Restrictions can be the source of a client login issues. Please use the FTS Log Viewer to determine why your client is having logi
issues before requesting a password reset.

**Example of Source IP Restriction in FTS Log Viewer**

| Action | IP Address |
|---|---|
| Blocked Sign On | 203.13.146.61 |

Subcategory * Select Subcategory

Question(s)

| | |
|---|---|
| FTS account name (e.g. client_qnt) * | rahul_test |
| Proposed Time for Reset * | Choose a datetime |
| Is there a security risk such that we should reset the client-facing password (i.e. FTS) right away? * | |

RequestType: Question Bug ER [✓ Incident]

Severity: Low Medium High [✓ Critical] System-wide Emergency

Figure 4.6: RPD User Interface for password reset

In the figure 4.7, the boxes show the important points (questions) that need to be answered when changing the password of a particular username, in this case, the account name. The figure 4.6 is the original view on the RPD GUI, where we also have to mention the reset time.

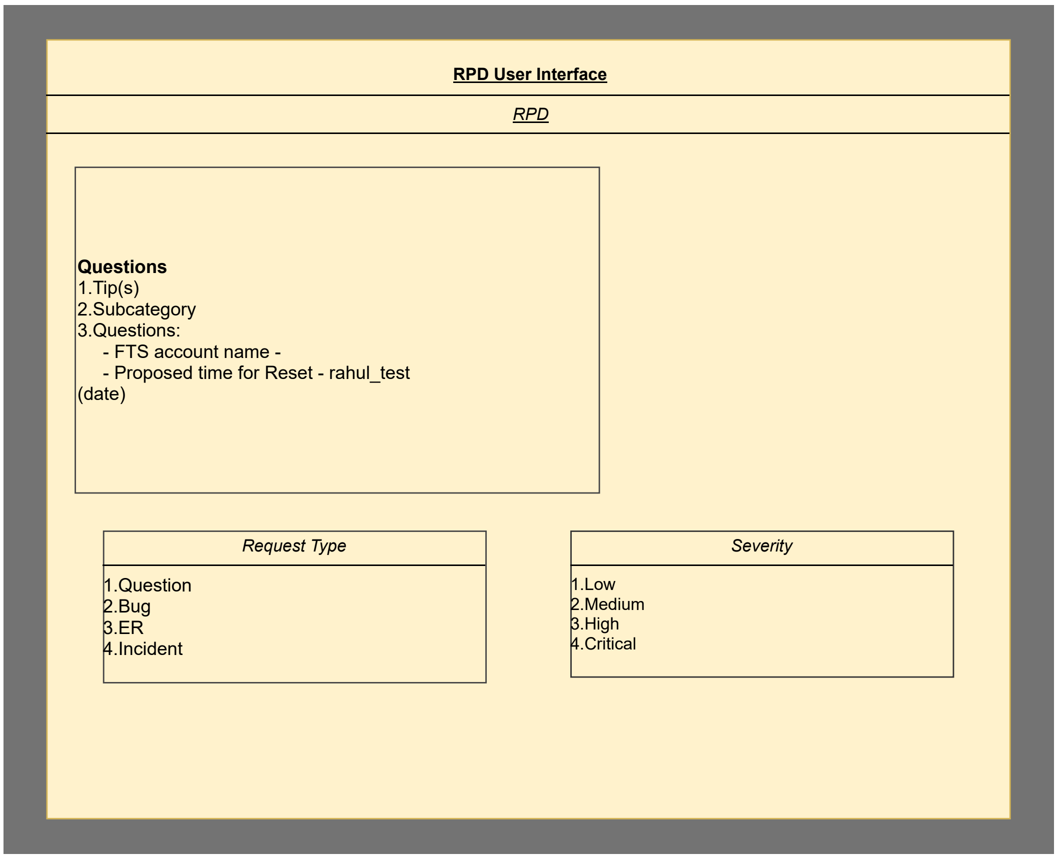


Figure 4.7: RPD Password reset

The workflow diagram is given below. Simple steps include signing in to the account, then entering the credentials, including the name and date of reset, and finishing the task.

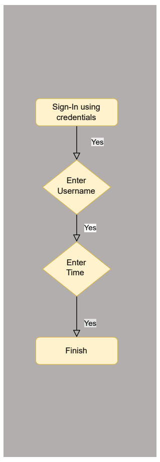
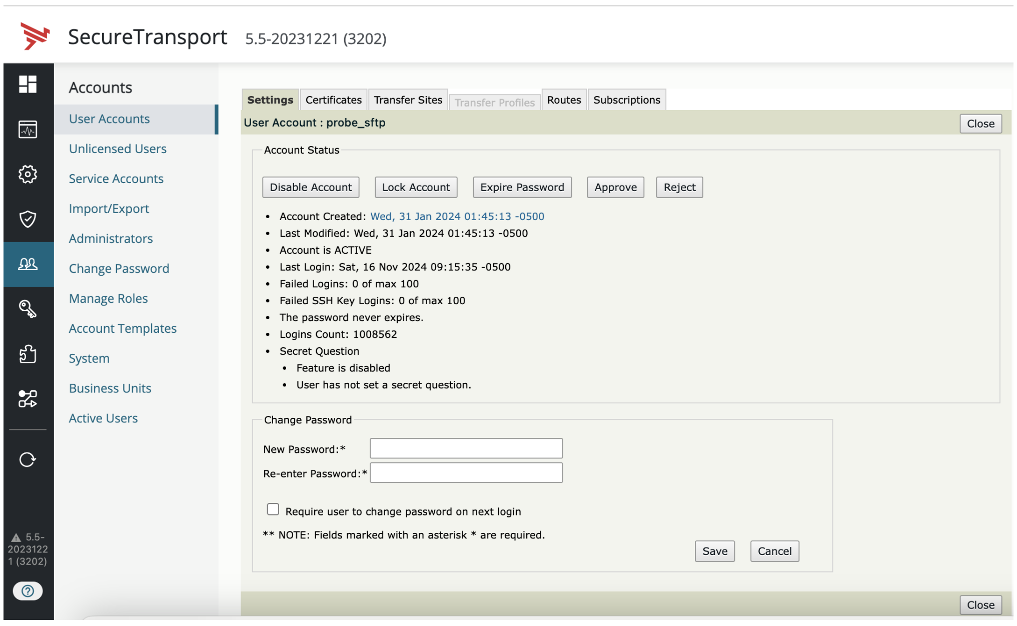


Figure 4.8: RPD Password reset workflow



User Account : probe_sftp

Account Status

- Account Created: Wed, 31 Jan 2024 01:45:13 -0500
- Last Modified: Wed, 31 Jan 2024 01:45:13 -0500
- Account is ACTIVE
- Last Login: Sat, 16 Nov 2024 09:15:35 -0500
- Failed Logins: 0 of max 100
- Failed SSH Key Logins: 0 of max 100
- The password never expires.
- Logins Count: 1008562
- Secret Question
  - Feature is disabled
  - User has not set a secret question.

Change Password

New Password:*

Re-enter Password:*

☐ Require user to change password on next login

** NOTE: Fields marked with an asterisk * are required.

Figure 4.9: RPD GUI to reset password

### Login Restriction Policies

As a third exercise, we will figure out how the log-in restriction policies are created and what the log-in restriction policies are. According to the definition, by configuring and utilizing log-in restriction policies, users can define and limit their ability to access SecureTransport Servers or SecureTransport Edges. In hierarchical inheritance and precedence order, the specified log-in restriction policies apply to business units, user accounts, and account templates [1]. Let us see what hierarchical inheritance and precedence order are.

The inheritance order is as follows:

- If a policy is not defined on the account or account template level, then the policy of the associated business unit is used
- If a policy is not defined either on the account or account template level or on the business unit level, but a default policy is set, then the default policy applies
- If no policy is set on the account or template, business unit, or default levels, then access is not restricted

The precedence order is as follows:

- If a policy is defined on the account or account template level, then it takes precedence over the policy assigned on the business unit level
- If no policy is defined on the account or account template level, then the policy assigned on the business unit level takes precedence

It is possible to choose and specify a single-user restriction policy as the default policy. When creating a new business unit, account, or account template, the default policy is the recommended user restriction policy. However, it also applies to users who do not match an account template or have a matching user account. This makes it possible to implement log-in limitations even for external users without an account definition within SecureTransport.

Login restriction policy rules can also be created using the Axway GUI, which predominantly declares the allow and deny type for client addresses, which is then attached to login restriction policies. The figure below 4.10 shows again the abstract layer of the Login Restriction Policy in the Axway GUI. When creating a login restriction policy, it is important to mention the type, type of business unit, and name of the rule.

The type of policy is predominantly in two versions: allow or deny, which means allowing the IP addresses of clients. We used IPv4 addresses, but there are other provisions available, too. The description option is optional, whereas we can add additional details, in particular, to the account. One more option of expression is available on the GUI, which is not shown in the abstract, which deals with the secure transport expression language, which in this case is also optional.

Login restrictions limit access to SecureTransport Servers and SecureTransport Edges by evaluating Allow then Deny and Deny then Allow Login Restriction Policies for end users.To manage the login restriction policies, after accessing Login Restrictions on the GUI, edit an existing policy by adding the name of the policy to be changed or altered, thus selecting the type to allow, then deny, or vice versa, and later to the policy add the corresponding IP addresses of the clients that might access the particular account. Optionally, we can also assign business units and descriptions, which give much more information about the policy, and also adding additional attributes can also be done.

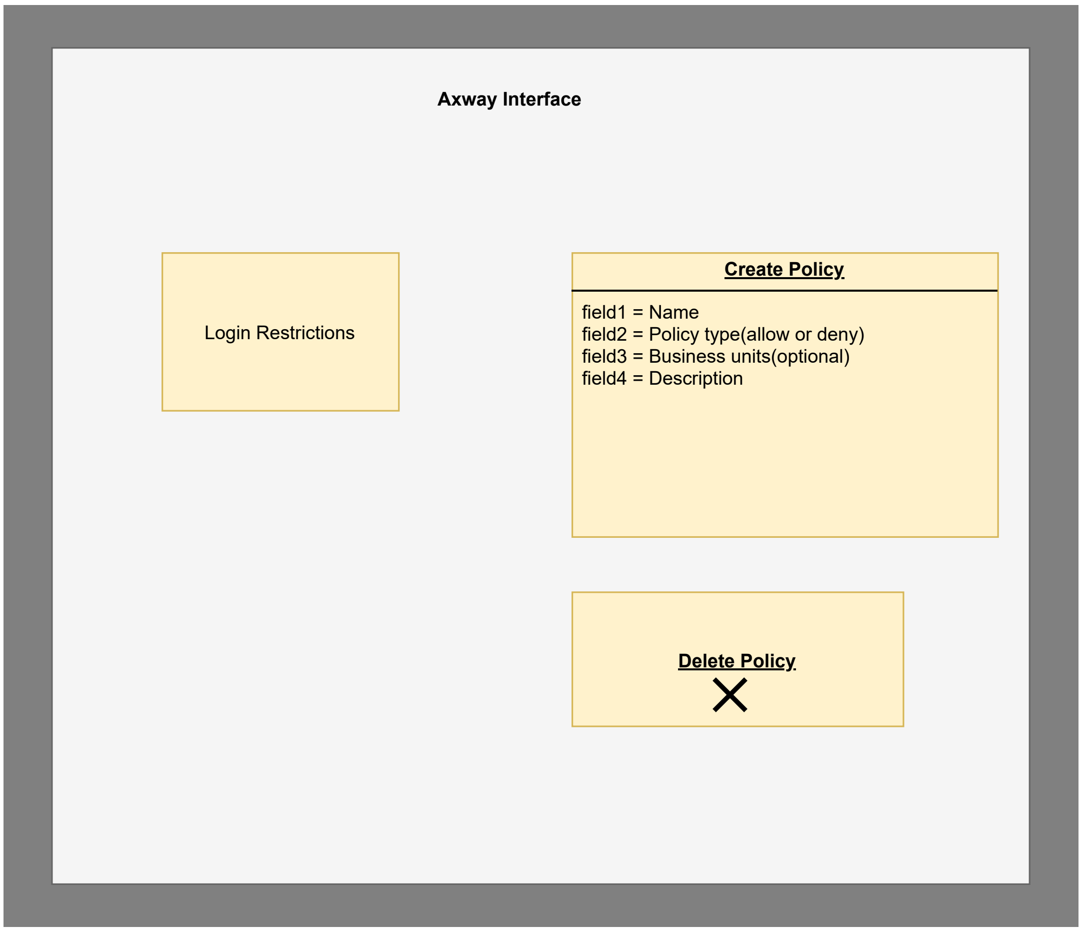
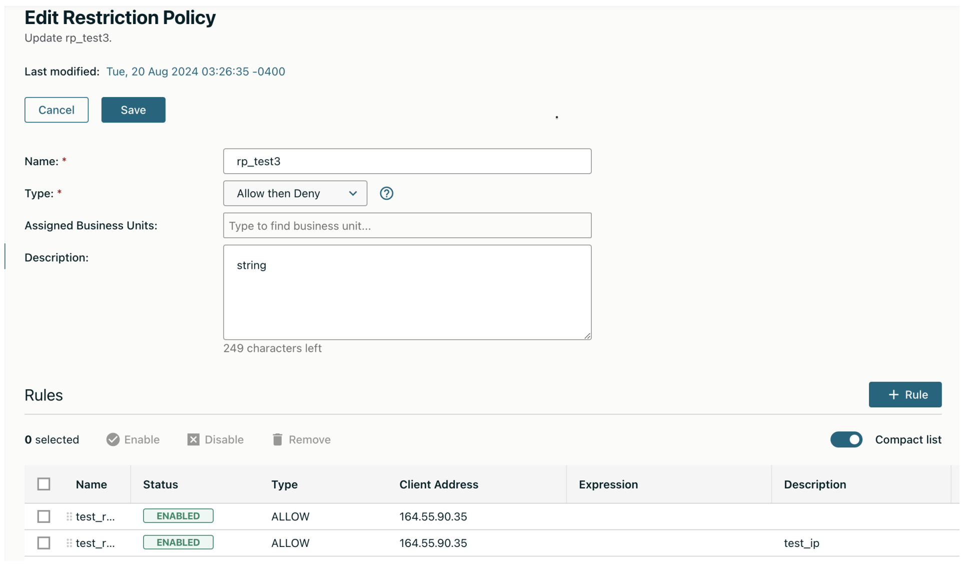


Figure 4.10: Login Restriction Policy Workflow Diagram

To delete the login restriction policy, the steps are as follows:

- Select the checkbox for the policy you want to delete
- Click the Delete button
- Confirm the deletion

The figure below 4.11 shows how it is seen in the Axway GUI. The important point to note is how the rules that are seen below can be added to the policy. This falls under the criteria for modifying the policy, which can also be automated. In the implementation in Python, we could add the corresponding entries and get, let us say, rules or descriptions attached additionally.



**Edit Restriction Policy**
Update rp_test3.

Last modified: Tue, 20 Aug 2024 03:26:35 -0400

Cancel | Save

| | |
|---|---|
| Name: * | rp_test3 |
| Type: * | Allow then Deny |
| Assigned Business Units: | Type to find business unit... |
| Description: | string |
| | 249 characters left |

Rules

0 selected | Enable | Disable | Remove | Compact list

| Name | Status | Type | Client Address | Expression | Description |
|---|---|---|---|---|---|
| test_r... | ENABLED | ALLOW | 164.55.90.35 | | |
| test_r... | ENABLED | ALLOW | 164.55.90.35 | | test_ip |

Figure 4.11: Login Restriction Policy

The figure below 4.12 shows how a rule can be created in the Axway GUI.

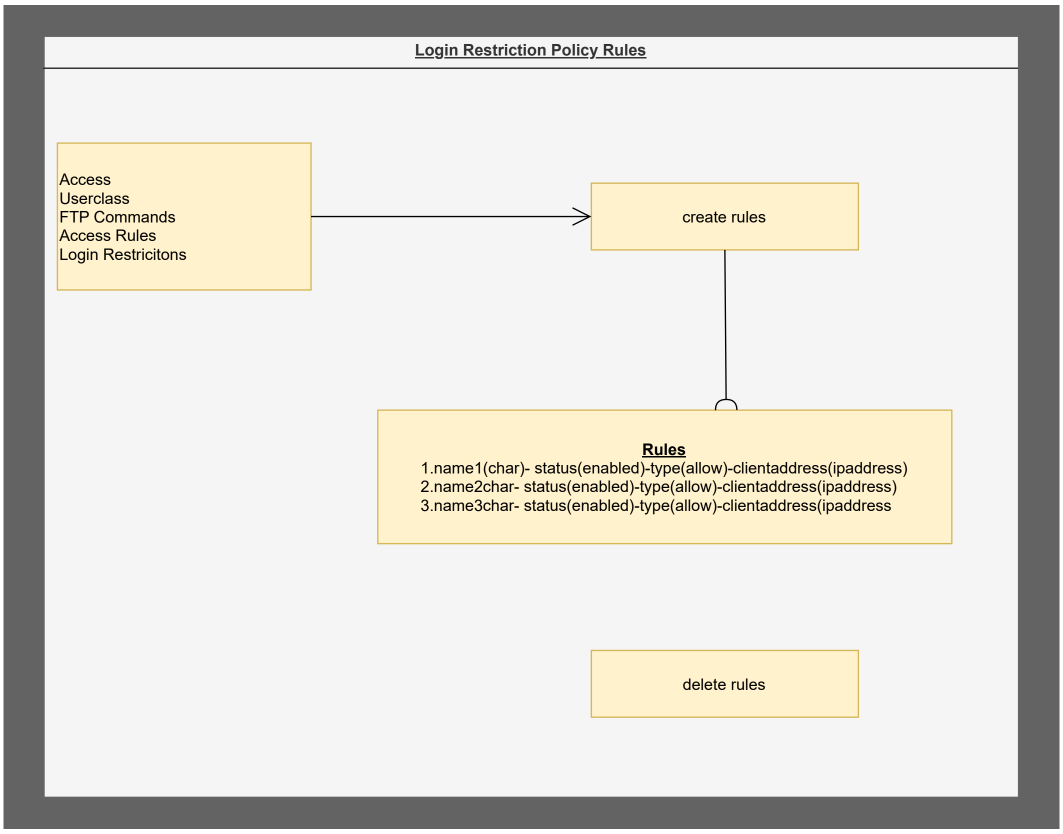
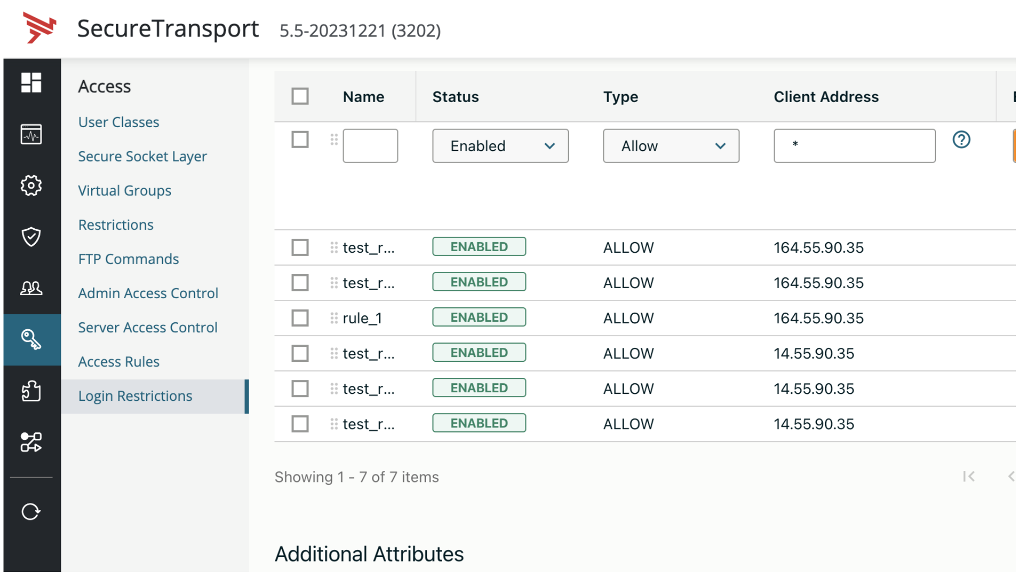

Login Restriction Policy Rules
Access
Userclass
FTP Commands
Access Rules
Login Restricitons
create rules
Rules
1.name1(char)- status(enabled)-type(allow)-clientaddress(ipaddress)
2.name2char- status(enabled)-type(allow)-clientaddress(ipaddress)
3.name3char- status(enabled)-type(allow)-clientaddress(ipaddress
delete rules

Figure 4.12: Login Restriction Policy Rules Workflow Diagram

Just like creating policies, the rules section attaches four important entries, namely the name, status (which can be enabled and disabled), the type to be followed (allow or deny), and the client's IP address. The GUI comes up with various options like FTP commands, user class, etc. Similarly to the delete policy, the delete option below can be used to delete any policy rule.

In the figure below 4.13, the user interface is shown to create the rules of the access restriction policy. To create the rule, we enter the details as mentioned above and check the boxes for the rules, and later those rules are added to the policies. Like creating the rules, deleting the rule can be done using the Axway GUI. However, we do not automate the deletion of rules, which is beyond the scope of this investigation.


SecureTransport 5.5-20231221 (3202)

| Name | Status | Type | Client Address |
|---|---|---|---|
|  | Enabled | Allow | * |
| test_r... | ENABLED | ALLOW | 164.55.90.35 |
| test_r... | ENABLED | ALLOW | 164.55.90.35 |
| rule_1 | ENABLED | ALLOW | 164.55.90.35 |
| test_r... | ENABLED | ALLOW | 14.55.90.35 |
| test_r... | ENABLED | ALLOW | 14.55.90.35 |
| test_r... | ENABLED | ALLOW | 14.55.90.35 |

Showing 1 - 7 of 7 items

Additional Attributes

Figure 4.13: Login Restriction Policy Rules

### Creation of EH Accounts

Lastly, we will describe the creation of enterprise host accounts (EH accounts) using the graphical user interface of the Axway program. Before we go further, we will take some time to understand what business units are, how they are created, and how to create an account using them. Enterprise hosts are a type of business unit.

Business units are used to encapsulate certain information necessary for transfers into a single entity. When creating accounts and templates, you can specify a business unit to represent a particular set of information about the transfer. The information contained in a business unit includes the business unit name, the base folder, a parent business unit, whether administrators are allowed to modify the base folder or the home folder, and which HTML template to use when users belonging to this business unit log in using the web client. In Axway, only master administrators and delegated administrators with permission to manage business units can create and delete business units and modify business units [1].

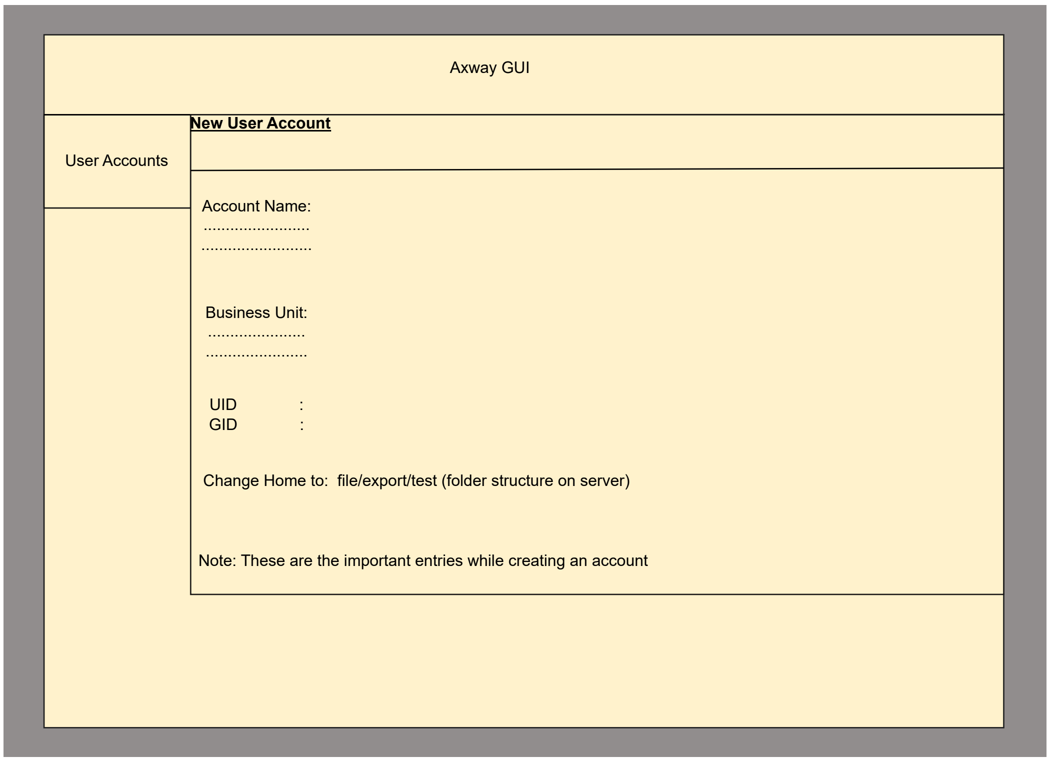
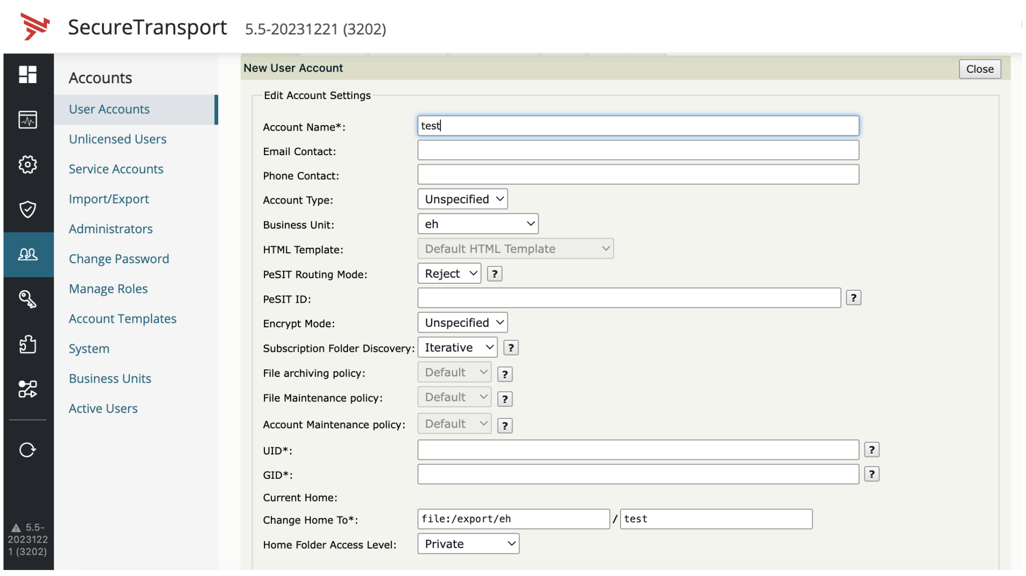


Figure 4.14: EH account graphical diagram

The figure 4.14 shows the abstract view of how an Enterprise Host (EH) account can be created on the Axway interface. Whilst creating the user account, in the accounts section after selecting EH Account as the type of business unit, it is important to specify the base folder field as it is the standard used in most of the firms. UID and GID in most cases for one particular system would be constant and should be kept intact. Also, log-in restriction policies can be attached; under that come the allow or deny rules that can be added as per requirements. This would conclude how it could be done in the Axway User Interface.

The following figure 4.15 shows the creation of EH accounts in the Axway GUI. As discussed above, the important fields to keep in mind when creating the EH account are the name of the account, the home folder, UID, and GID. Other additional arguments can be added as per clients' requirements, and additional descriptions can also be attached.



| | |
|---|---|
| Account Name*: | test |
| Email Contact: | |
| Phone Contact: | |
| Account Type: | Unspecified |
| Business Unit: | eh |
| HTML Template: | Default HTML Template |
| PeSIT Routing Mode: | Reject |
| PeSIT ID: | |
| Encrypt Mode: | Unspecified |
| Subscription Folder Discovery: | Iterative |
| File archiving policy: | Default |
| File Maintenance policy: | Default |
| Account Maintenance policy: | Default |
| UID*: | |
| GID*: | |
| Current Home: | |
| Change Home To*: | file:/export/eh / test |
| Home Folder Access Level: | Private |

Figure 4.15: EH Account Creation

# Chapter 5

# Automated Approach

In this chapter, we will see the Python code used for the automation process. In this program, we have used Python modules such as PyODBC(for connecting to the MySQL database), OS, Requests, JSON, argparse, requests, and dotenv (for working with env variables).

**SSH Key Generation(automated)**

The whole program can then be divided into four subparts. In the first part of the program, later shown in the figure below 5.1, the variables are read from *.env* files stored in the main directory. Later, these credentials can be used to call the function **getcredentials** so that the server name and hostname are filtered from the RPD. After parsing the hostname and server name from the RPD, along with the RPD number, we created a folder in the /home directory, which has the name of the RPD number; then using *ssh-keygen* command, we generated SSH key pairs. Later in the third part of this program, the SSH private keys are placed in the database under one particular username and hostname. Creating multiple entries of the same hostname or server name in this database is not allowed as per the MySQL schema; the duplication problem is thus excluded, which in turn was helpful during testing. And this part includes the simultaneous posting of the public key in the commenting section of the RPD, from which the server and hostname are parsed. The steps with the workflow diagrams and pseudocode pertaining to each step are given below.

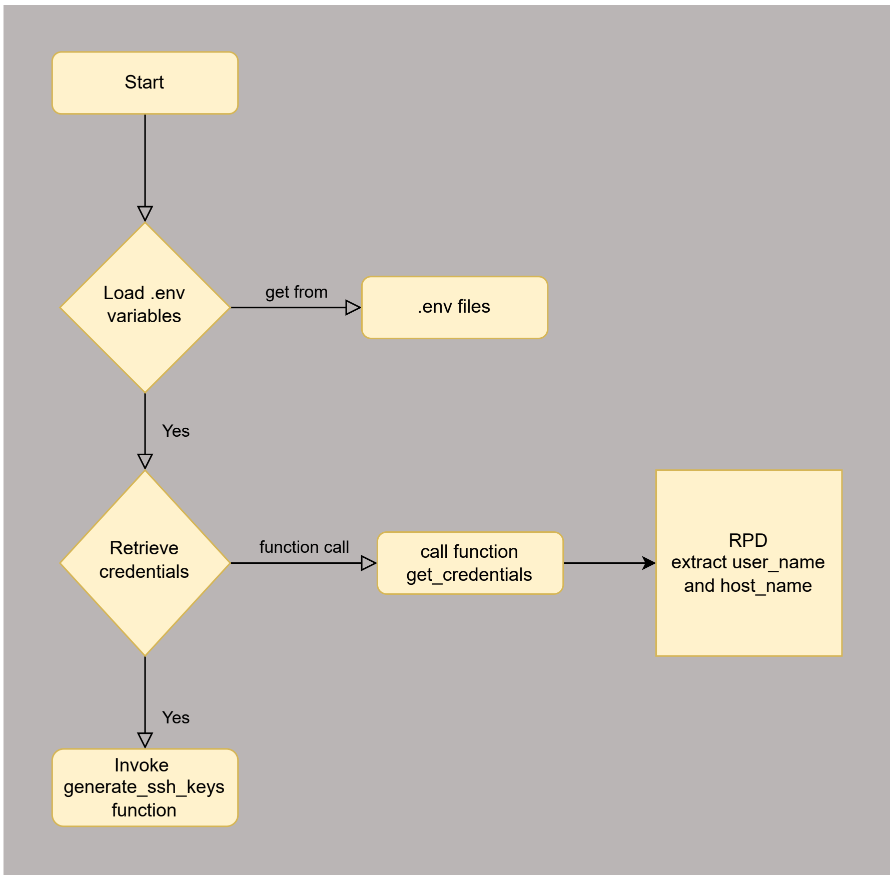

Start
Load .env variables
get from
.env files
Yes
Retrieve credentials
function call
call function get_credentials
RPD extract user_name and host_name
Yes
Invoke generate_ssh_keys function

Figure 5.1: Getting credentials from RPD

The corresponding pseudo-code is given below.

```

Function get_credentials(rpd_id):
   Create a session for making HTTP requests
   Authenticate using NTLM with the provided user and password

   Perform a GET request to retrieve comments
   Parse the response JSON into comments_dic

   Define identifiers for hostname and server username
   Initialize an empty dictionary for relevant data

   If 'Questions' key exists in comments_dic:
      For each question in comments_dic['Questions']:
         Get 'Id' from the question

```

```
        If hostname_identifier is found in id_field (case-insensitive):
            Add hostname to relevant_data with the key 'Hostname'

        If servername_identifier is found in id_field (case-insensitive):
            Add server username to relevant_data
            with the key 'Server Username'
Else:
    Print message indicating no 'Questions' key found

Return relevant_data
```

Listing 5.1: Get credentials

In the second part of this function call, we generate private and public SSH keys using the function ‘generatesshkeys() and then parse it to make a clear text so that the public key can be posted in the comment section in RPD. The figure below shows the workflow diagram.

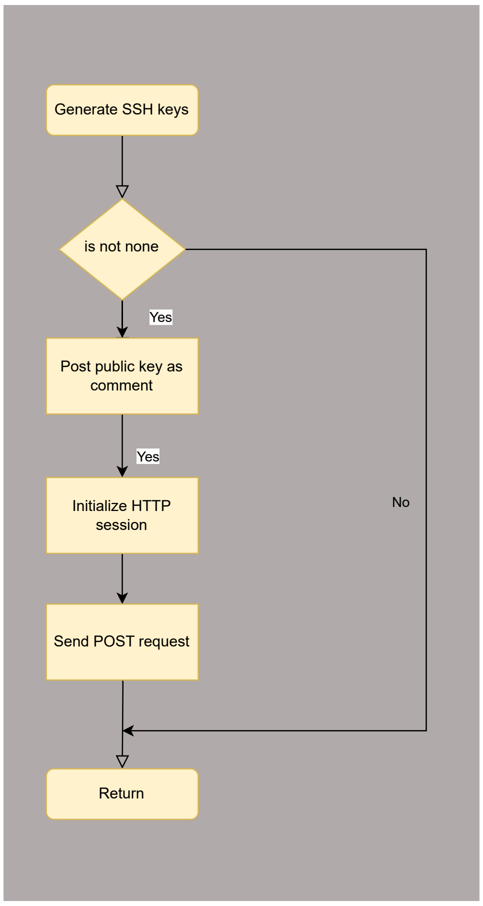


Figure 5.2: Generate SSH keys

The pseudocode below corresponds to the actual code being used in the function.

```

Function generate_ssh_keys(key_name):
   Define the command to generate SSH keys using `ssh-keygen` with:
      - Type as 'rsa'
      - Bit size as '2048'
      - No passphrase
      - Output file with name key_name

   Try:
      Execute the command to generate the SSH keys
      Check for successful execution

      Open the private key file with name key_name for reading
      Read the private key content

      Open the public key file with name key_name.pub for reading
      Read the public key content
      Remove trailing space and handle specific format issues

      Return private key and public key
```

Listing 5.2: Generate SSH keys

The Python pseudocode below shows how the connection to the MySQL database is carried out using the PyODBC driver and, consequently, inserting the private key into the database. The workflow diagram corresponding to it is shown here 5.3.

```


connection_string = f'DSN=MYMSSQL;UID={user_db};PWD={password_db}'
   cnxn = pyodbc.connect(connection_string)
   crsr = cnxn.cursor()
   private_key = private_key.replace("\n", " ").replace("'", "''")
   query = """
        INSERT INTO [ftsdb_test].[dbo].[ServerInitiated
        TransferSftpCredentials]
        (host, username, privateKey)
        VALUES (?, ?, ?)
        """
   crsr.execute(query, (host_name, user_name, private_key))
```

```
#inserting private key to db
#rows = crsr.execute("select * from [ftsdb_test].[dbo].
[ServerInitiated
TransferSftpCredentials] where username='rpremnavas_test2' ").
fetchall()
cnxn.commit()
crsr.close()
cnxn.close()
print(" the private key has been succesfully inserted to
the database and the rpd has been commented out
with the public key")
```

Listing 5.3: Insert private key to database

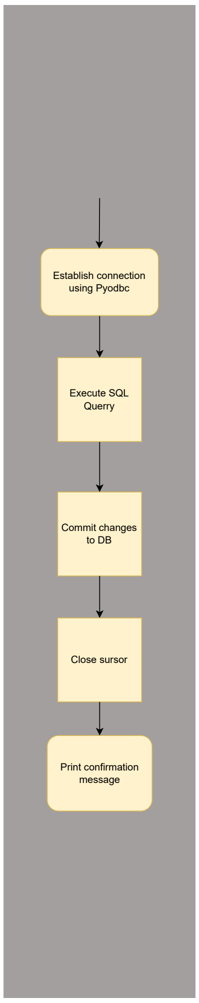


Figure 5.3: Mysql connection string

Lastly, the main() function is called along with the command-line parser so that the RPD-ID given to the parser executes the function calls above (insert-key-into-database() ). The workflow diagram is given below figure 5.4.

```
def main():
   parser = argparse.ArgumentParser()
   parser.add_argument("rpd_id", help= "RPD ID to get credentials and
   subsequently insert keys to db")
   args = parser.parse_args()
   print(insert_keys_into_database(args.rpd_id))
```

Listing 5.4: Main Function

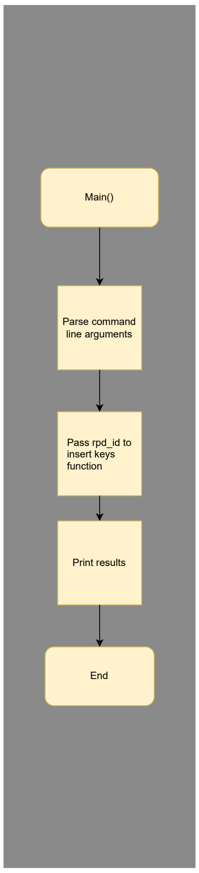

Main()
Parse command line arguments
Pass rpd_id to insert keys function
Print results
End

Figure 5.4: Main function

### Password Reset (automated)

The following diagram 5.5 illustrates the steps to resetting the passwords for the accounts. From the onset, we imported the required libraries and variables, as we did with the majority of the programs till now. We set up an API request, namely a GET request, to check whether the user exists, and according to the response, if needed, we made a JSON object to create a new password. In this case, we used a POST request to put the credentials in the database. The corresponding pseudocode for the workflow is given below the diagram.

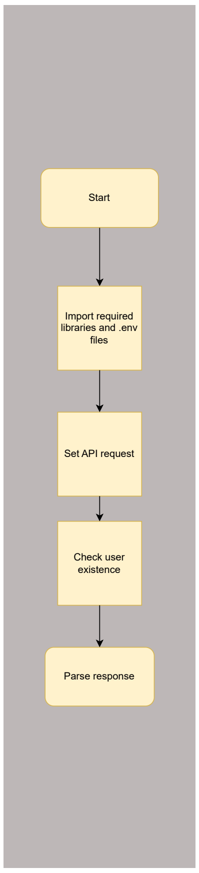


Figure 5.5: Password Reset

```

Function change_password(username, password):
   Construct the URL for checking if the user exists
   Send a GET request to the URL with basic authentication
   Parse the response JSON into response_dic
   Print the response status code

   If the status code of the user existence check is 200:
      Construct the URL for changing the user’s password

      Create a data dictionary with:
         - Username
         - Password credentials containing:
            - Password
            - Username

      Convert the data dictionary to JSON format

      Send a POST request to update the password using:
         - URL for the user
         - Headers and authentication
         - JSON data for the new password
         - Disable SSL certificate verification

      If the password change request status code is 204:
         Print "password successfully changed"
      Else:
         Print "check for mistakes"
      Print the status code of the password change request
```

Listing 5.5: Password reset

**Login Restriction Policies (automated)**

In the third section, for creating/deleting login restriction policies and creating rules, we have three programs. The first one creates a login restriction policy, and as per requirements, we also add login restriction rules to each policy.

The workflow diagram is shown below. After importing necessary libraries and variables, we started setting up an API session, including a function that contained a dictionary with the necessary keys to make a POST request, which in turn talks to the API interface and produces an output. The 201 response from the API server ensured that the POST request was successful.

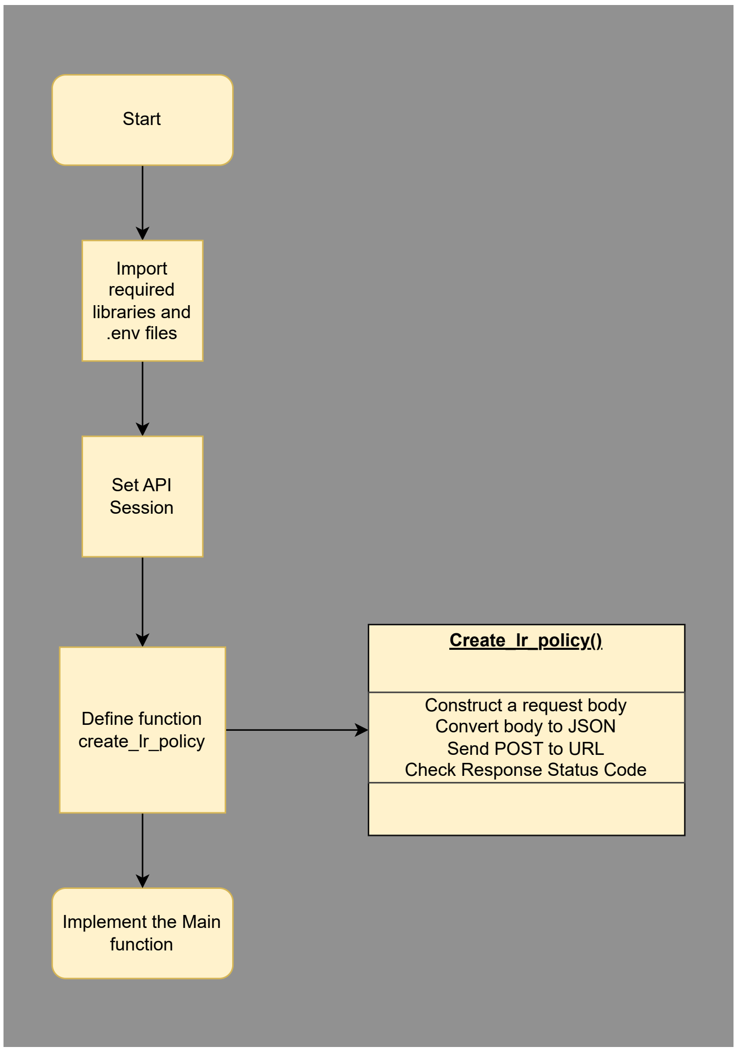

Start
Import required libraries and .env files
Set API Session
Define function create_lr_policy
Create_lr_policy()
Construct a request body
Convert body to JSON
Send POST to URL
Check Response Status Code
Implement the Main function

Figure 5.6: Create login restriction policy

The following listing shows the function we used to create a policy using a JSON object using the POST method. In this pseudocode, the function *create-lr-policy* takes up an argument which is the name of the policy to be created and inside the function call. After setting up the execution call using the Python dictionary, convert the type of the instance to JSON and then send a POST request. This POST request results in the creation of the Login-restriction policy on the Axway server.

```

Function create_lr_policy(name):
   Construct the request body as a dictionary with:
      - "name" set to the provided name
      - "type" set to "ALLOW_THEN_DENY"
      - "description" set to "test"
```

```

  Convert request body dictionary to JSON format

  Send a POST request to create the policy using:
     - URL for creating the policy
     - JSON data as request body
     - Headers and basic authentication
     - Disable SSL certificate verification

  If the response status code is 201:
     Print "login restriction policy created"
  Else if the response status code is 401:
     Print "check your credentials"
  Else:
     Print "something went wrong"
```

Listing 5.6: Create login restriction policy

To delete the log-in restriction policy, as we discussed above, we create a function in this case, ‘delete-lr-policy(), making a deletion operation in the API request. The following Python code is the code that we used for this undertaking.

```python
def delete_lr_policy(name:str):
 response = requests.delete(url=url_2+'/'+name, headers=headers,
 auth=HTTPBasicAuth(api_user, api_psswd), verify=False)
 if response.status_code == 204:
  print("successfull operation")
 else:
  print("something went wrong, check the data")
```

Listing 5.7: Delete login restriction policy

We have the following workflow diagram to create a login restriction policy rule. Figure 5.7 shows the workflow of the function, where we create a Rule Object with the necessary credentials (client address and name of the rule) and then formulate a function to make an API request. In the listing given below, the function takes the arguments policyrule, policy name, and ipaddress, then constructs an object named rule to take in variables. Then, this data object is converted to json format and then creates an API call using the POST method. The if statements are included to read the server response. A 201 response means that the rule is created.

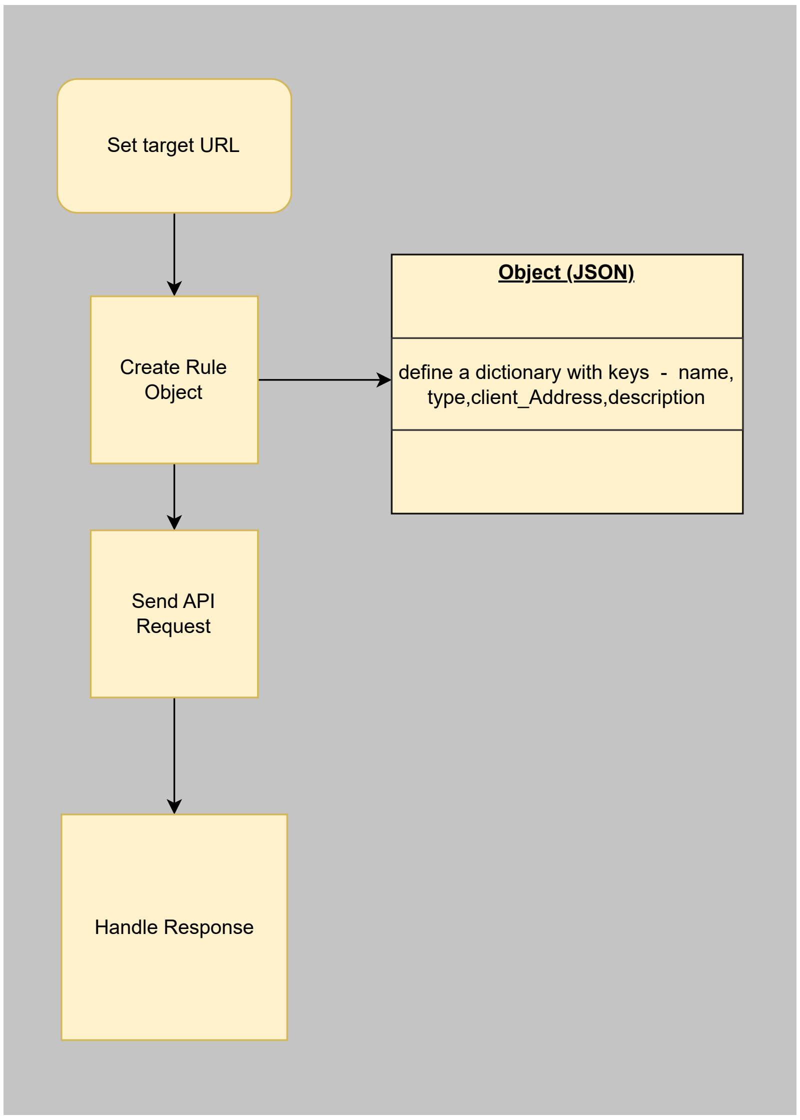


Figure 5.7: Create Rule

```


Function create_rule(policy_name, name_of_the_rule, client_address):
    Construct the URL
     Construct the rule data as a dictionary with:
        - "name" set to name_of_the_rule
        - "enabled" set to "true"
        - "type" set to "ALLOW"
        - "clientAddress" set to client_address
        - "description" set to "test"

    Convert rule data dictionary to JSON format

    Send a POST request to add the rule using:
       - Constructed URL for the rule
       - JSON data as the request body
       - Headers and basic authentication
       - Disable SSL certificate verification

    If the response status code is 201:
       Print "Rule successfully created"
    Else if the response status code is 400:
       Print "bad request"
    Else if the response status code is 401:
       Print "unauthorised login credentials"
    Else:
       Print "check your entries, take fts support"
```

Listing 5.8: Create login restriction policy rule

**Creation of EH Accounts (automated)**

Our last task involves creating EH accounts. The workflow diagram in Fig. 5.8 illustrates this. Firstly, we imported the required libraries and variables, and as we did in previous exercises, we then created an API session, authenticating it, and then later building a function ('create-eh-account') to create a PUT request and talk to the API. The function call includes a dictionary, which is then converted to JSON and made available in the PUT request.

The pseudocode related to the function is given below. It is evident that the request body contains variables which are the name of the account to be created, that is, the only argument in the function call, the home folder being explicitly mentioned, UID, GID(values that are constant, created during the Axway server setup), and the name of the business unit. In addition, the contact details for each business unit can also be added, which is an optional parameter. The optional parameters include notes(set to test), homefolderAccesslevel(set to private), disabled (set to false), type(set to user). The API server responded with a 201 response, which means that the request to make the EH account was successful.

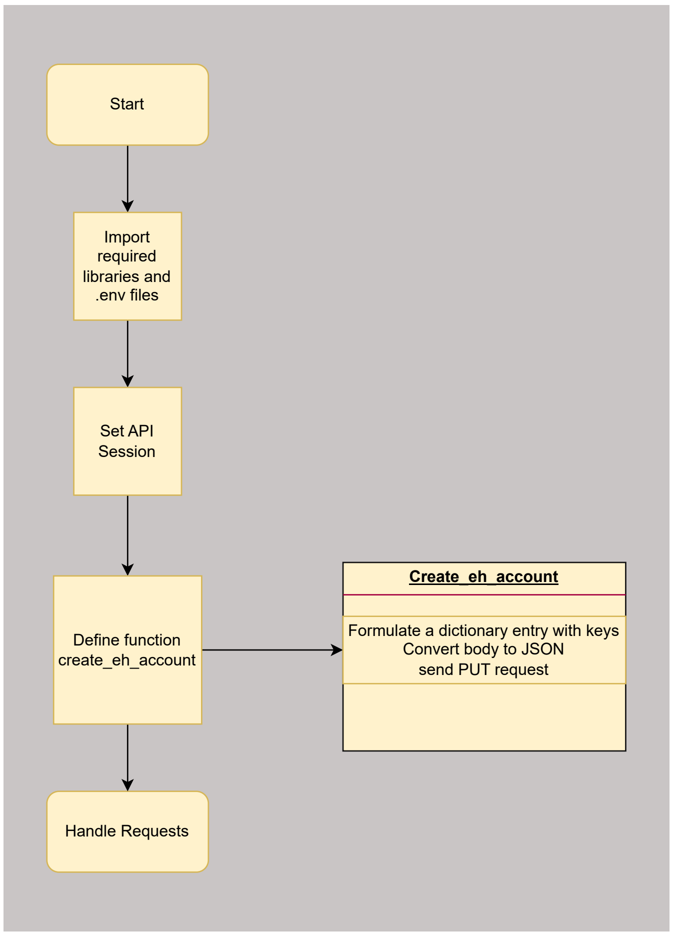

Start
Import required libraries and .env files
Set API Session
Define function create_eh_account
Create_eh_account
Formulate a dictionary entry with keys
Convert body to JSON
send PUT request
Handle Requests

Figure 5.8: Create EH-Accounts

```
Function create_eh_account(account_name):
    Construct the URL for creating the EH account

    Define headers with 'Content-Type' set to 'application/json'

    Create the request body(As a dicitonary datatype)
        - "name" set to account_name
        - "homeFolder" set to "/export/eh"
        - "homeFolderAccessLevel" set to "PRIVATE"
        - "uid" set to "2123" - doesnot change
        - "gid" set to "2020" - doesnot change
        - "disabled" set to "false"
        - "type" set to "user"
        - "notes" set to "test account"
        - "businessUnit" set to "eh"
        - "contact" with nested dictionary containing:
            - "email": "abc1@test.com"
            - "phone": "1234553421"

    Convert the request body dictionary to JSON format

    Send a PUT request to create the account using:
        - Constructed URL
        - JSON data as the request body
        - Headers and basic authentication
        - Disable SSL certificate verification
```

Listing 5.9: Creation of EH Accounts

This concludes the automation and GUI steps that we carried out in this investigation. We will discuss the results in the next chapter.

# Chapter 6

# Results and Discussion

In this final chapter of this research, we will examine the results and consider how implementing the functions would affect the overall execution process, and how much time each task would take in both executions, namely the GUI approach and the Command-Line approach. This undertaking could affect the monthly transactions in the File Transfer Department in a positive way, by which the rate at which the RPD's are responded to and solved could be increased to a good extent. The time taken for each task is recorded and summarized in later parts of this chapter as graphs.

We look at the instructions for running the program. The first task is to read the RPD (Request for Process Development) number, generate SSH keys with the corresponding RPD number, and copy the private key to the database. Moreover, the public key will be posted in the comment section of the RPD.

The function on the CLI can be called as follows.

**python3 privatekeydb.py '134532722'**

The program will run the function defined in the previous chapter, saving the private key in the database, and giving the output to the console.

```

Entry:
        Statement = 0x7f79d1863400
        SQL = [Insert into [ftsdb_test].[dbo].[ServerInitiatedTransferSftp
               Credentials] (host ,username ,privateKey)
              values ('mft.int-np.wlifea...][length = 1827]
[ODBC][44136][1718632798.693048][SQLExecDirectW.c][456]
                Exit:[SQL_SUCCESS]
[ODBC][44136][1718632798.693156][SQLRowCount.c][173]
```

```
                Entry:
                        Statement = 0x7f79d1863400
                        Row Count = 0x7ff7b026f800
[ODBC][44136][1718632798.693216][SQLRowCount.c][247]
                Exit:[SQL_SUCCESS]
```

Listing 6.1: Output inserting private key

Simultaneously, in the RPD commenting Section, in fig 4.4 will be added with the private key, thus ending the automation.

The automation has resulted in a significant time reduction, which normally might have taken 4-6 minutes, is reduced by 10 seconds. The following graph shows 6.1 how the CLI(automation approach) significantly reduces the time taken rather than performing the same operation using the GUI(normal approach through User Interface).

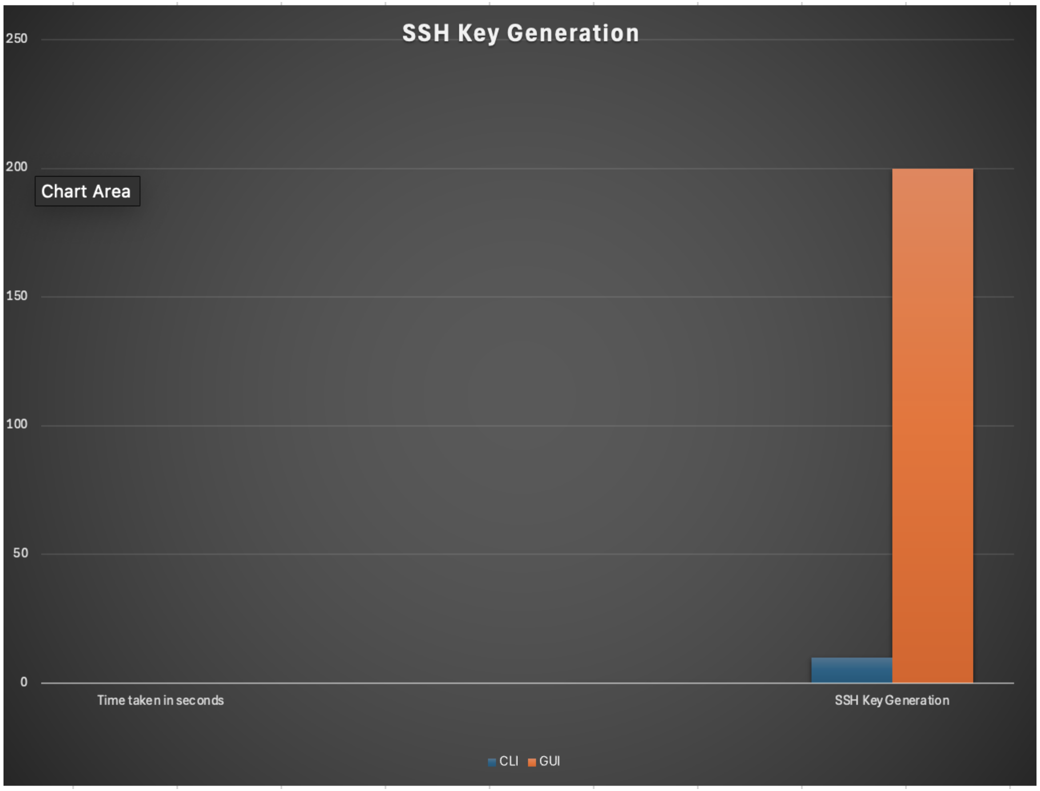
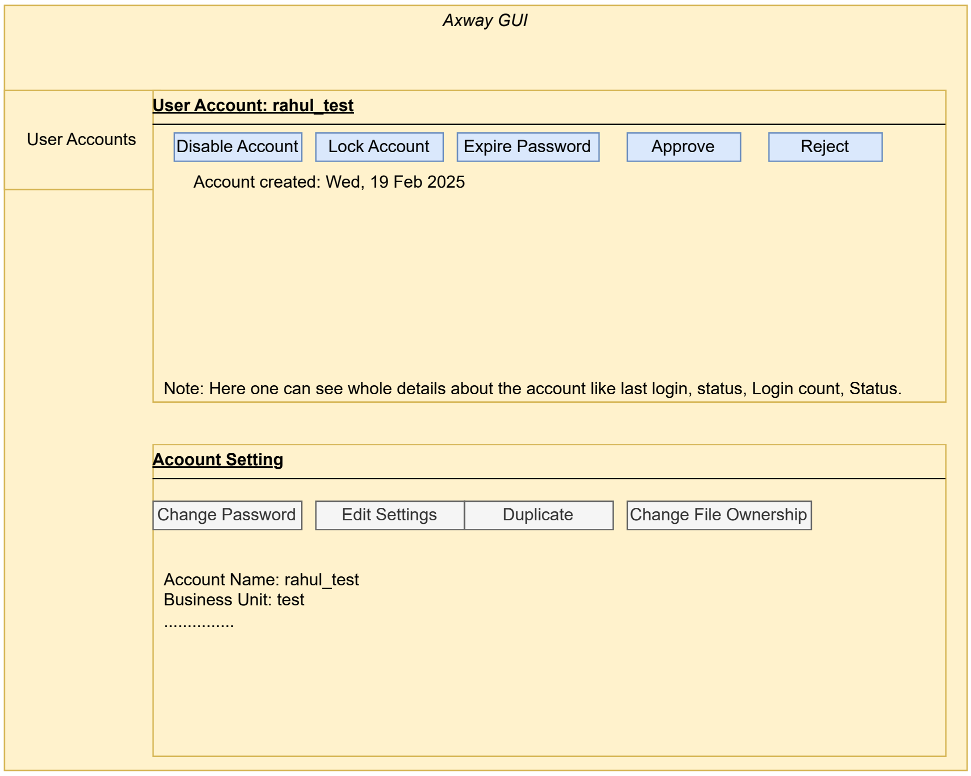


Figure 6.1: Time taken in average SSH Key Generation

As for the second test, which is the password reset for the accounts you have created before, the password change of an existing user can be done using the away-test.py program (given in the previous chapter); it can be done on CLI; a sample prompt is given below.

**python3 axway-test.py your-user-name your-new-password**

The following figure 6.2 shows the Account Status, and upon running the program, we should get a successful response from the API server, in this case a 201 response.



Account created: Wed, 19 Feb 2025

Note: Here one can see whole details about the account like last login, status, Login count, Status.

Account Name: rahul_test
Business Unit: test
...............

Figure 6.2: GUI Axway for test account

The time taken to carry out this task is around 5 seconds, which normally might have taken 3 minutes. The bar graph given below 6.3 shows the time distribution in both approaches, namely the GUI and the CLI approach. The amount of time taken to perform the task using CLI has changed significantly, which can dramatically increase the speed of transactions on a business day. Given the scenario if the account of one particular user in a particular business unit is locked due to false password usage , or there are chances that one forgets their passwords(rarely), they would not have to wait for more time getting approved from the FTS engineers until the RPD is answered; instead, this automated approach can ease the process by which the first level of support can interfere and run the function on FTS servers to get the password changed, thereby easing the way of doing business.

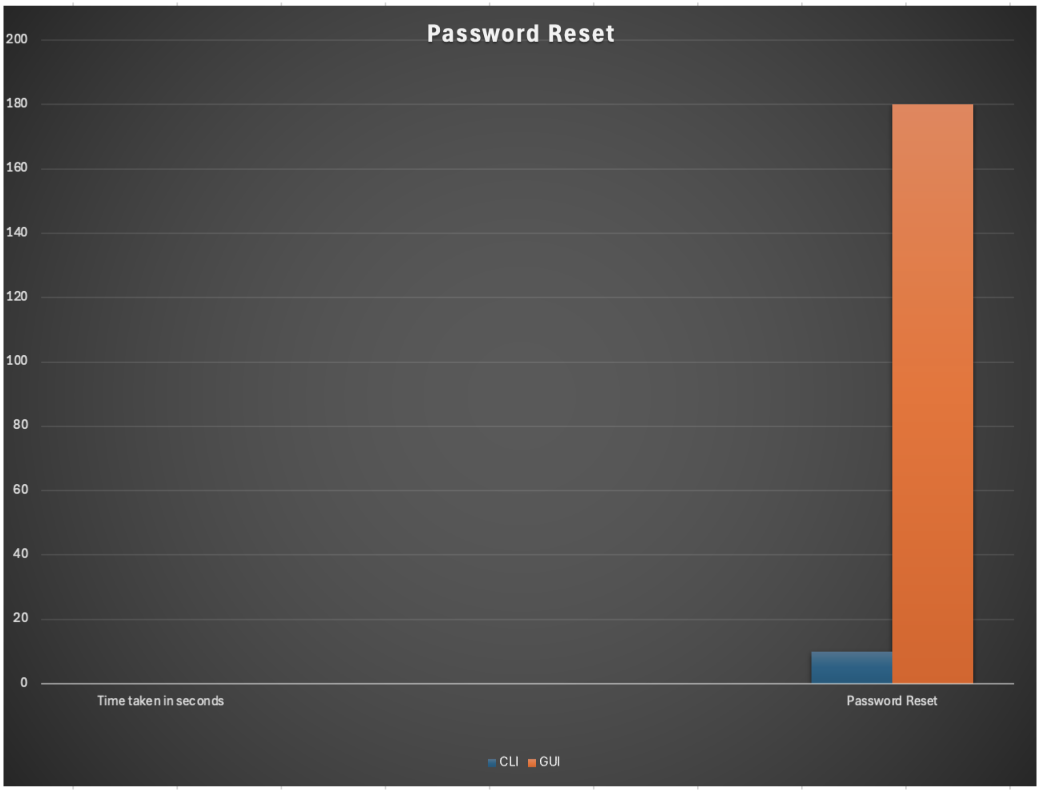


Figure 6.3: Time taken in average password reset

For the third task to change/delete/modify login restriction policies, you can create/delete a login restriction policy by using the command line and calling the Python code by the example below:

**python3 create-lr-policy.py name-of-the-policy** - to create policy

**python3 delete-lp.py name-of-the-policy** - to delete the policy

In order to create a rule for a specific login restriction policy, call the program given below.

**python3 create-rule.py name-of-the-policy name-of-the-rule ip-address-of-the-client**

e.g.: *python3 create-rule.py rp-test3 test-rule5 14.55.90.35*

The time taken to create and delete the log-in policy would normally take 3-5 minutes depending on how many rules to add, etc.; by automating the process, it can be done in 5-10 seconds.

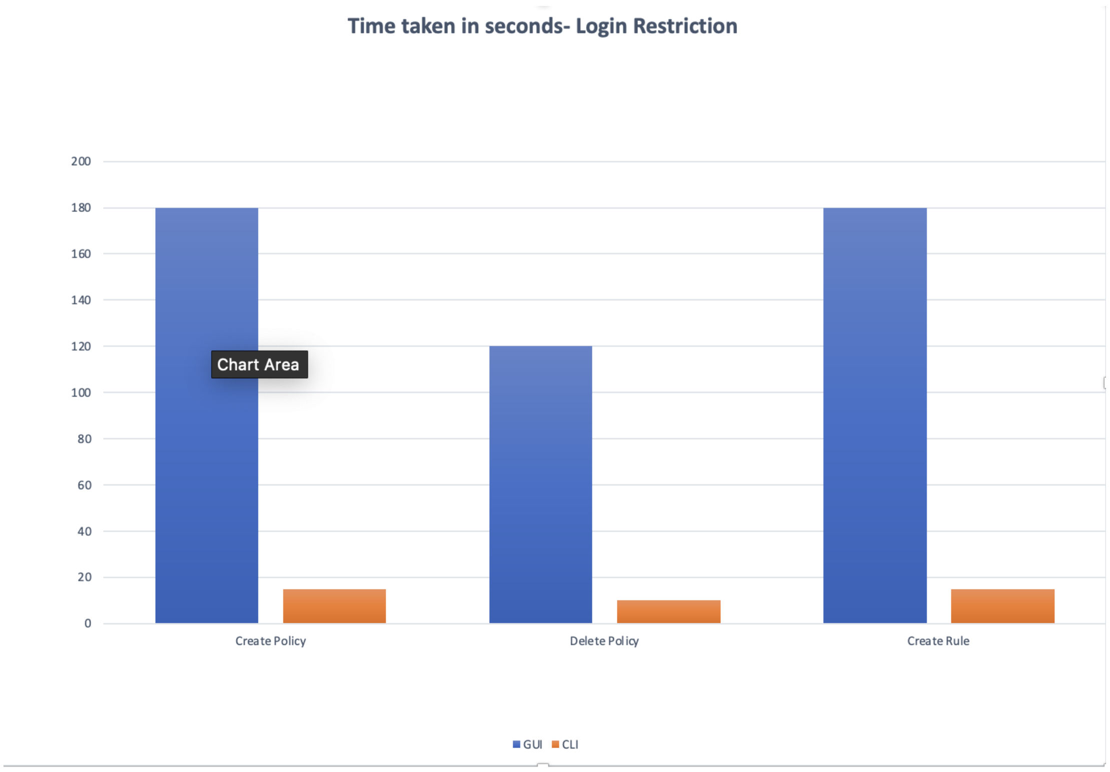


Figure 6.4: Time taken in average

Finally, create the Enterprise Hosts Account on FTS servers and run the program *python3 create-eh-account.py name-of-YOUR-account*

**python3 create-eh-account.py eh-test**

The time taken to run the program is 10 seconds, which normally would have taken 3-5 minutes of GUI usage. The graph shown below 6.5 shows the time taken by both approaches, where it is quite evident that the automated approach has considerably reduced the time to 60 times less than what it could have used up until this test. As discussed in the previous case, approval to get RPD addressed in this case can also bring many disadvantages, particularly at market risk by loss of data/time/money, as these processes are carried out during stock exchange opening hours. The enterprise hosts are one type of business unit in a particular firm; there could be tens and hundreds depending on the size of the business sector, and this automation process can pave the way to bring a more automated approach to creating business units or whatever tasks related to creating/changing/adding using the automation tools.

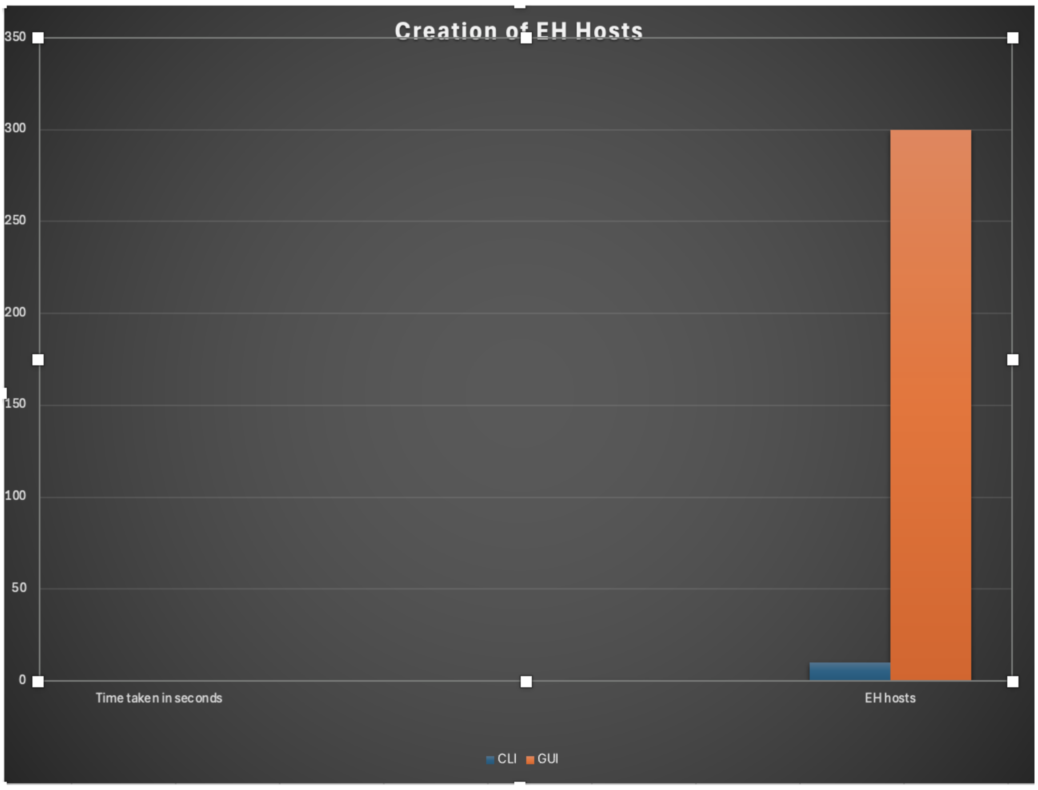


Figure 6.5: Time taken in average EH Hosts Creations

With this undertaking to automate these four tasks, the results have been proven to be successful and after being tested in a test environment, they will be deployed on production servers. As another scope, FrontEnd to this program can also be added, depending on the requirements, which in this case is beyond the scope of our research.

The fifth task is to change the protocol we used to transfer the files/sensitive data from one server to the other. That is, from the normal FTP (File Transfer Protocol) to the secure FTP. As the new Axway component comes along with both options at the mid-way of this approach, there was no need to automate, and we refrained from the task. Information about this new update can be found on the Axway home page [1].

# Chapter 7

# Conclusions and Future Work

In this last chapter, we will summarize the results from a broader perspective and look at possible future work.

Firstly, this research helped us to significantly improve the time taken for the whole task. Despite improving the time, we managed to keep the accuracy and reliability of the data transfer very good. One of the major challenges while undertaking this research was the connection of Python to MySQL Database via PyODBC Driver, as there were very limited data available for this particular problem. As different users have different operating systems, we recommend reading this GitHub page [6] for Mac Users. This challenge was overcome by installing a FreeTDS driver and adding the hostname of the database server to *freetds.conf* in */usr/local/etc*.
This research/approach can be implemented in other departments like UNIX teams, where the tickets (RPD) can be addressed by using the API of host building applications such as Foreman, or getting specific logs from Kibana, etc.

In terms of futuristic perspectives, we can also add user interfaces to this program, which we mentioned in the previous chapter. We can also optimize algorithms using various features of the programming language used, and multiple other data sources can also be incorporated into the program for the enhancement of the program. For larger workflows, much more ordered algorithms can also be used. We could integrate AI where the system identifies a potential risk and reacts accordingly. Client feedback can be collected over a period of time and enhance the functionalities of written programs.

## References

[1] Axway-installation. https://docs.axway.com/bundle/SecureTransport_55_InstallationGuide_allOS_en_PDF/resource/SecureTransport_InstallationGuide_allOS_en.pdf. Accessed: 2024-11-13.

[2] Checklist-axway. https://docs.axway.com/bundle/SecureTransport_55_GettingStartedGuide_allOS_en_HTML5/page/Content/GettingStartedGuide/setup/Starting_Setup.htm. Accessed: 2024-16-13.

[3] data-processing. https://www.semanticscholar.org/paper/Concurrent-Data-Processing-in-Microsoft-Dynamics-Yadav-Sehgal/. Accessed: 2024-09-30.

[4] ietf. https://www.ietf.org/. Accessed: 2024-09-30.

[5] masterthesisbern. http://rvs.unibe.ch/research/pub_files/Ge10.pdf. Accessed: 2010-09-30.

[6] pyodbc. https://github.com/mkleehammer/pyodbc/wiki/Connecting-to-SQL-Server-from-Mac-OSX. Accessed: 2024-09-30.

[7] w3. https://www.w3.org/.

[8] Neha S Batni and Jyoti Shetty. A comprehensive study on automation using robot framework. 2020.

[9] Gaurav Bora, Saurabh Bora, Shivendra Singh, and Sheikh Mohamad Arsalan. Osi reference model: An overview. *International Journal of Computer Trends and Technology (IJCTT)*, 7(4):214–218, 2014.

[10] Rohit Singh Chauhan, Anirudh Pradhan, Anusheel Munshi, and Bidhu Kalyan Mohanti. Efficient and reliable data extraction in radiation oncology using python programming language: A pilot study. *Journal of Medical Physics*, 48:13 – 18, 2023.

[11] Roy T. Fielding. Architectural styles and the design of network-based software architectures"; doctoral dissertation. 2000.

[12] DV Kornienko, SV Mishina, SV Shcherbatykh, and MO Melnikov. Principles of securing restful api web services developed with python frameworks. In *Journal of Physics: Conference Series*, volume 2094, page 032016. IOP Publishing, 2021.

[13] Sumit Kumar, Sumit Dalal, and Vivek Dixit. The osi model: overview on the seven layers of computer networks. *International Journal of Computer Science and Information Technology Research*, 2(3):461–466, 2014.

[14] Mingyu Lim. C2cftp: Direct and indirect file transfer protocols between clients in client-server architecture. *IEEE Access*, 8:102833–102845, 2020.

[15] Lauren Murphy, Tosin Alliyu, Andrew Macvean, Mary Beth Kery, and Brad A Myers. Preliminary analysis of rest api style guidelines. *Ann Arbor*, 1001(48109):4, 2017.

[16] Brad A. Myers and Jeffrey Stylos. Improving api usability. *Commun. ACM*, 59(6):62–69, may 2016.

[17] Robbi Rahim, Solly Aryza, Pristisal Wibowo, Adek Khadijatul Z Harahap, Abdul Rahman Suleman, Edi Epron Sihombing, Yuswin Harputra, Muhammad Rahman Rambe, Andysah Putera Utama Siahaan, H Hermansyah, et al. Prototype file transfer protocol application for lan and wi-fi communication. *Int. J. Eng. Technol*, 7(2.13):345–347, 2018.

[18] Gui-Yeol Ryu. A study on comparison of open application programming interface of securities companies supporting python. *The International Journal of Advanced Smart Convergence*, 10:97–104, 2021.

[19] Aleksandra Tarkowska, Denise Carvalho-Silva, Charles E Cook, Edd Turner, Robert D Finn, and Andrew D Yates. Eleven quick tips to build a usable rest api for life sciences. *PLoS computational biology*, 14(12):e1006542, 2018.

[20] Zouheir Trabelsi and Imad Jawhar. Covert file transfer protocol based on the ip record route option. *Journal of Information Assurance and Security*, 5(1):64–73, 2010.

[21] Akash Yadav and Jai Sehgal. Concurrent data processing in microsoft dynamics crm using python. *International Journal of Innovative Research in Computer Science and Technology*, 2023.