UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Aleksander Parelo

Development of a virtual printer and print driver for Print in City

Master's Thesis (30 ECTS)

Supervisor(s): Urmas Tamm, MSc
Meelis Roos, PhD

Tartu 2021

Development of a virtual printer and print driver for Print in City

Abstract:

With the advancement of digital solutions, the need for printing is becoming less relevant every year. This also means that the need for personal printers is decreasing every year, and services like Print in City, which allow users to access public printers for a small fee, are replacing personal printers. Currently, these services still have a downside compared to regular printers – they require the user to use a webpage to upload their documents. This complicates the service and discourages the users from using it. This thesis details the development of a virtual printer and printer driver to solve this problem. The virtual printer is a web service that acts as a regular printer and makes the experience of using Print in City very similar to the experience of using a regular printer. A working prototype of the proposed solution is developed and presented.

Keywords:

Printing, print driver, virtual printer, Print in City

CERCS: P175 Informatics, systems theory

Virtuaalse printeri ja printeri driveri arendamine Print in City jaoks

Lühikokkuvõte:

Digitaalsete lahenduste arenguga väheneb iga aastaga vajadus dokumente välja printida. See tähendab, et väheneb ka vajadus omada kodus isiklikku printerit ja suureneb printimisteenuste, nagu *Print in City*, turuosa. Need teenused võimaldavad kasutajatel väikese tasu eest kasutada avalikke printereid. Hetkel on selliste teenuste üheks probleemiks see, et kasutaja peab tööde printimiseks need veebilehele üles laadima. See teeb printimise kasutajatele keerulisemaks ja aeganõudvamaks. Käesolevas lõputöös pakutakse välja lahendus, milleks on virtuaalne printer ja printeri driver. Virtuaalne printer on veebiteenus, mis käitub nagu tavaline printer. See võimaldab kasutajatel kasutada *Print in City*'t sarnaselt tavalisele printerile. Töö käigus luuakse virtuaalse printeri ja klientrakenduste töötavad prototüübid.

Võtmesõnad:

Printimine, printeri driver, virtuaalne printer, Print in City

CERCS: P175 Informaatika, süsteemiteooria

Table of Contents

List of Figures

List of Tables

1 Introduction

With most of the world going digital, the need to print out documents is decreasing every year. This also means that owning a personal printer is no longer necessary. However, sometimes people still need to print things out. The company Overall has introduced Print in City to solve that problem. Print in City is a web service that allows users to print out documents using public printers by paying a small fee. Such print points are available in Estonia, Latvia and Finland with plans to launch in more countries in the future.

Currently, Print in City requires users to upload their documents using a web service. This complicates the process of printing and turns off some users from using Print in City regularly. Furthermore, Print in City currently supports only PDF and JPEG files which also makes it more cumbersome to use.

This thesis proposes a solution for these problems – a virtual printer and printer driver. A virtual printer is a web service that acts like a regular printer that can be added to a user's computer similarly to a regular printer. This makes it easier for users to print documents using Print in City since they no longer have to convert their documents to the correct format or use a web page to upload them. They can simply click print on their computers and select Print in City as the printer.

This thesis is divided into two larger sections, the first section consists of chapters 1-8 and discusses the different technologies that were used for the development of the virtual printer service and client applications. Chapter 3 gives an overview of the technology and history of Print in City. Chapter 4 talks about Internet printing, starting with a short overview of its history. Then the technologies used for Internet printing are discussed. Chapter 5 describes printing on Windows, detailing the different protocols supported by Windows for talking with printers and giving an overview of the architecture of printer drivers on Windows. Chapter 6 talks about printing on macOS and Linux. It gives an overview of the printing protocols supported by these operating systems and talks briefly about the driver architecture of both platforms. Chapter 7 describes printing on mobile devices, specifically iOS and Android, and gives an overview of the supported printing protocols and methods for accessing printers on these platforms. Chapter 8 gives a brief overview of Azure Service Bus and the technology behind it.

The second part of the thesis consists of Chapters 9-11 and talks about the implementation of the virtual printer and client applications. Chapter 9 lays out the design requirements for the solution and gives a general overview of the result. Chapter 10 describes the virtual printer in more depth. Chapter 11 covers the client applications that were developed for this thesis.

2 List of Abbreviations

EMCA – European Computer Manufacturers Association

IETF - Internet Engineering Task Force

IPP – Internet Printing Protocol

PDF – Portable Document Format

JPEG – Joint Photographic Experts Group

RFC – Request for Comments

PWG – Printer Working Group

HTTP – Hypertext Transfer Protocol

DNS – Domain Name System

mDNS – multicast Domain Name System

NAT – network address translation

NAT-PMP – Network Address Translation Port Mapping Protocol

DNS-SD – Domain Name System service discovery

PCLm – Printer Command Language mobile

URF – Universal Raster Format

BLE – Bluetooth Low Energy

GDI – Graphics Device Interface

XPS – XML Paper Specification

UWP – Universal Windows Platform

XML – Extensible Markup Language

LPD – Line Printer Daemon protocol

SMB – Server Message Block

USB – Universal Serial Bus

MDM – mobile device management

AMQP – Advanced Message Queuing Protocol

SSO – single sign-on

REST – representational data transfer

API – application programming interface

GPD – generic printer description

3 Print in City

Print in City is a web service that allows users to print to public printers from a web browser. These public printers are located in different public places like libraries, universities, and schools. [1]

3.1 History of Print in City

Print in City (originally named Pilveprint) is developed by Overall Eesti AS. Print in City was first publicly launched in 2011. [2]

Over the following years, new print locations were added and the service was expanded to new countries. Print in City is currently publicly available in Estonia, Finland and Latvia and there are plans to launch Print in City in several new markets in the near future. [1]

3.2 Design of Print in City

Print in City is designed in a way that allows users to print to printers that are on a private network the user does not have access to. This allows institutions to provide their printers for public use without allowing unauthorized users to connect to their internal network. An overview of the general design of Print in City can be seen in Figure 1.

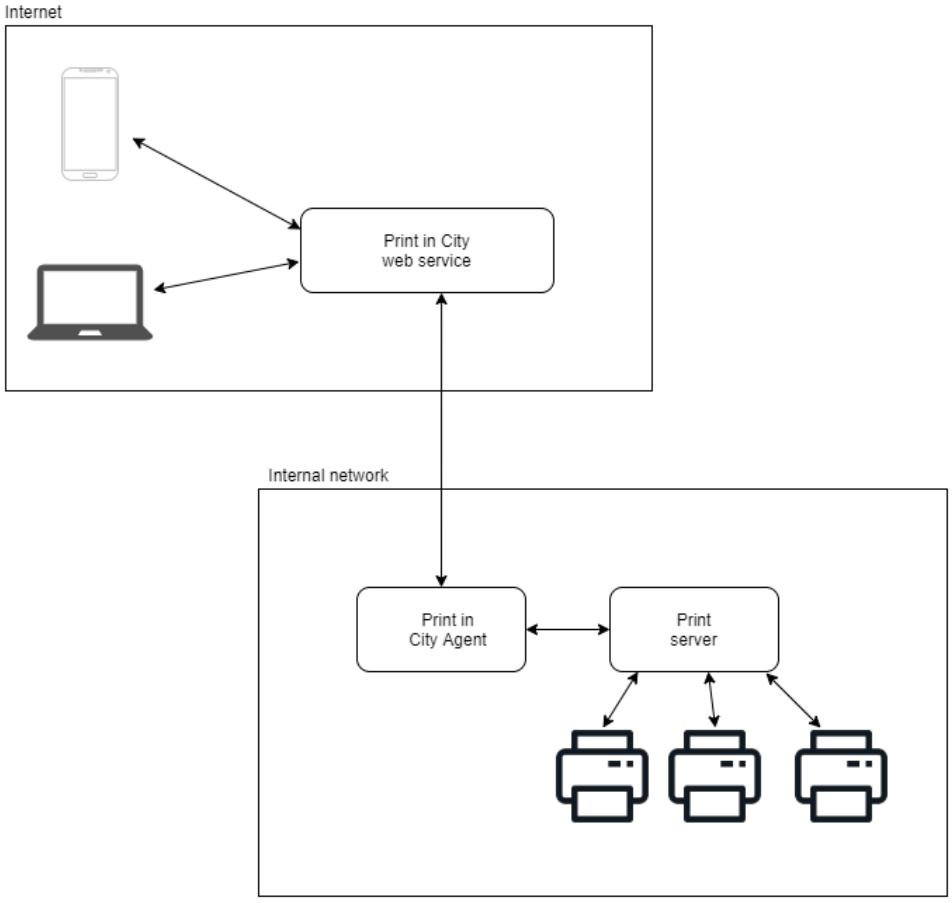

Figure 1: Overview of Print in City design.

Separating users from private networks is achieved by using a Print in City agent that is installed on a server on the local network. The agent is responsible for communicating with the local print server, including uploading user information and print jobs to the server and

downloading data about printed documents from the server. The agent is connected to the main Print in City server using a secure network connection.

Users interact with Print in City through a web interface that is also hosted on the main server. The web service handles the authorization of users and accepts payments from users. The web service also collects data about printed documents from all the agents and keeps data synchronized between the agents.

3.3 Print in City workflow

To print a document using Print in City, a user first has to create an account on the Print in City website and add credit to their account. They also have to add a supported authentication method to their account for authentication at the printer. The currently supported authentication methods are PIN code and different cards, for example, Estonian ID cards or different transportation cards.

After the user has successfully set up their account they can upload documents to Print in City using the web interface. The web interface currently supports documents in either PDF or JPEG formats. After the documents have been uploaded the user can choose the print settings they wish to use for the print job. Currently, the user can change the colour mode, duplex mode and page size of the print job. After choosing the desired print settings the user can select a print point where they wish to pick up their print job and submit the job to Print in City.

To print out the job the user has to go to a printer at the print point they chose and authenticate themselves. They can then choose the uploaded document and print it out. They can also make copies and scan documents at the printer. Scanned documents are sent to the email address associated with the user's account.

4 Internet Printing

4.1 History of Internet printing

Network printing is the process of printing a document to a printer that is connected to the device over a network. The history of network printing has two different start points: the EMCA-140 standard published by European Computer Manufacturers Association (EMCA) in March 1990 and RFC 1179 standard published by Internet Engineering Task Force (IETF) in August 1990. [3], [4]

Both of these standards had many shortcomings. To overcome these problems, several printer manufacturers decided to start working on a new standard called Internet Printing Protocol (IPP). The IPP project was formally proposed by Novell in the summer of 1996. [5]

4.2 Internet printing protocol

Internet printing protocol history

Version 1.0 of the IPP standard was published in April 1999. The first version of IPP was published as an experimental standard since all the problems had not been fixed yet. It was also decided that the published standard will soon be updated to version 1.1 and that version will no longer be experimental. [5] The draft of the IPP 1.1 standard was published in September 2000 [6], [7]. This draft was updated in 2002, 2005 and 2015 [8]–[12]. These drafts were combined into a single standard that was published in 2018 [13].

IPP 2.0 was published in 2009 as an update to IPP 1.1. It was separated into IPP 2.0 for printers and IPP 2.1 for print servers. IPP 2.2 for production printers was added in 2011. IPP 2.0 was adapted as Printer Working Group (PWG) standard in 2015. [14], [15]

IPP specification

IPP is an application-level protocol that can be used for distributed printing on the Internet. IPP allows clients to submit print jobs, inquire about the submitted jobs and the capabilities of the printer. The printer can be a physical device that produces output or a web server that communicates with a real printer.

IPP uses Hypertext Transfer Protocol (HTTP) 1.1 POST requests for communication. This communication method was chosen because it provides all the necessary features required by IPP and is already an Internet standard. By using an existing protocol there is no need to develop a custom protocol. Furthermore, it was expected that many printers will already implement HTTP to provide other services to the user. For example, a web interface for changing printer settings.

A simple binary encoding was chosen to encode the IPP messages, this was done to make it simpler for manufacturers to develop embedded printers. These embedded printers might have limited processing power and memory which would make it difficult for them to decode a more complex encoding scheme in real time. [16]

4.3 Zero-configuration networking

The following chapter is based on Bonjour documentation [17] provided by Apple.

Zero-configuration networking is a system that allows network devices to acquire configuration parameters automatically and without a central service. This is needed for dynamic networks where there are no central devices to keep track of the configuration. The most

widely used Zero-configuration networking protocol is Bonjour. Bonjour was originally developed by Apple and released as an open-source standard.

Bonjour is based on the Zeroconf protocol proposed by the Internet Engineering Task Force (IETF) and consists of 3 main components. These components are addressing (allocating IP addresses to hosts), naming (using names to refer to hosts instead of IP addresses) and service discovery (finding services on the network automatically).

Addressing

Addressing is solved by using link-local addressing. Link-local addressing uses a range of IP addresses that are reserved for a local network. It works by having devices pick a random IP address from the reserved range and testing if the address is already in use. If the address is already in use the device picks another random address. If the address is not in use then the device starts using it.

Naming

Bonjour uses multicast Domain Name System (mDNS) for network address translations. It is a variant of DNS where DNS queries are sent over the local network using multicast. This means that no single DNS server with global knowledge is required. Each service can provide its own DNS capability by listening for DNS queries for its name and responding to them.

Bonjour also provides built-in support for Network Address Translation Port Mapping Protocol (NAT-PMP). This can be used with supported routers to create and destroy port mappings to allow hosts to connect to services that are behind a firewall.

A unique name is required for network address translation to work properly. Since the local names only have significance on the local network devices can choose their own names. They do this by choosing a name and sending an mDNS query with the chosen name to see if the name is already in use on the network. If there is no response to the query the device can use the name.

Service discovery

Service discovery allows applications to find services on the local network and maintain a list of services and ports. The application can then resolve the name of the service into the IP address of the service using the methods described in the last section.

Bonjour uses the DNS service discovery (DNS-SD) protocol for service discovery [18]. DNS-SD uses standard DNS messages to allow clients to discover services on the network and service (SRV), text (TXT) and pointer (PTR) records for describing the services. SRV records let devices on the network know the hostname and the port where a service instance can be located. A TXT record with the same name as the SRV record gives additional information about the service. A client can discover SRV and TXT records for a certain service type by querying for a PTR record for that service type. [19]

4.4 Driverless printing

Driverless printing is a method of printing where a device requires no additional printer-specific software to print to a printer. The printer must inform the device about its capabilities upon request and support a common page description language for the print job.

There are a variety of methods for driverless printing, for example, sending the job as an email attachment to the printer, using a web interface to upload the job or using a driverless printing protocol, such as AirPrint or IPP Everywhere. [20]

Driverless printing protocols

There are four driverless printing standards available: Apple AirPrint, Printer Working Groups (PWGs) IPP Everywhere, Mopria and Wi-Fi Direct. All of these protocols use DNS-SD to advertise printers on the local network and IPP for communication between a client and the printer. All four standards support a selection of the four page description languages (PDLs), with the only difference between the protocols being which page description languages are supported.

These four page description languages are PDF, Apple Raster, PWG Raster and Printer Command Language mobile (PCLm). Table 1 shows the supported page description languages for each protocol.[21]

Table 1: Overview of supported PDLs for different Driverless printing protocols

Driverless printing protocol	**Supported PDLs**
Apple AirPrint	Apple Raster/Universal Raster Format (URF), PDF
PWGs IPP Everywhere	PWG Raster, PDF
Mopria	PCLm, PWG Raster, PDF
Wi-Fi Direct	PCLm, PWG Raster, PDF

AirPrint

AirPrint is a network printing and discovery service created by Apple for use on iOS and macOS devices [22]. AirPrint was launched for iOS in 2010 and macOS in 2012 [23], [24].

AirPrint uses IPP 2.0 standard for communication with print devices and the Bonjour protocol for device discovery on a local network. It includes support for device discovery using iBeacon Bluetooth Low Energy (BLE) beacons. AirPrint is the recommended way for printing on iOS and macOS devices. [25] AirPrint is a proprietary protocol and documentation about the protocol is only available from Apple on request. Certification by Apple is required to advertise a device as AirPrint compatible.

IPP Everywhere

IPP Everywhere is an open-source driverless printing standard developed by the PWG. The first version of the IPP Everywhere specification was published in 2013, an update to the standard was published in 2020. [26], [27]

IPP Everywhere uses DNS-SD for printer discovery and IPP for communication with the printer. PWG provides a self-certification toolkit on their website that can be used to certify that a printer meets the IPP Everywhere specification. Devices that have been successfully certified using the self-certification tool can be listed on the PWG website as officially supported. [28]

Mopria

Mopria is a driverless printing standard developed by the Mopria Alliance. Mopria Alliance was founded in 2013 by Canon, HP, Samsung and Xerox, it currently has 21 members [29]. Mopria is the default printing method for Android devices and it is also supported by Windows 10 [30].

Mopria is a closed standard and is available only to members of the Mopria Alliance. [31]

Wi-Fi Direct printing

Wi-Fi Direct is an extension of the existing Wi-Fi standard that allows devices to connect directly without the use of an access point. It is developed by the Wi-Fi Alliance and is publicly available. The first version of the Wi-Fi direct standard was published in 2009 and the latest version of the standard was published in 2020. Wi-Fi direct supports driverless printing using DNS-SD for device discovery and IPP for printing. [32]

5 Printing in Windows

Windows requires the use of printer drivers for printing. These drivers enable applications to call device-independent functions to print documents. A printer driver consists of a rendering component and a configuration component. The rendering component is used to convert the graphics coming from an application into a format that the printer uses to output the graphics to a page. The configuration component contains the user interface that users use to configure the printer's selectable options. The configuration component is also responsible for communicating with the printer to retrieve the printer's configuration and features. [33]

Windows printer drivers are divided into versions. The latest available version of drivers is version 4 (V4). V4 printer drivers were introduced with the launch of Windows Server 2012 and Windows 8. V4 printer drivers were meant to replace version 3 (V3) drivers over time, but V3 drivers are also still supported. [34]

5.1 V3 printer drivers

V3 printer drivers use a Graphics Device Interface (GDI) for rendering print jobs to a format that is supported by the printer. For V3 drivers, developers are responsible for creating these GDIs for each supported device. This makes drivers platform- and device-specific, with new drivers needed for each new device and platform. Figure 2 shows what happens when an application creates a print job using a V3 driver. [35]

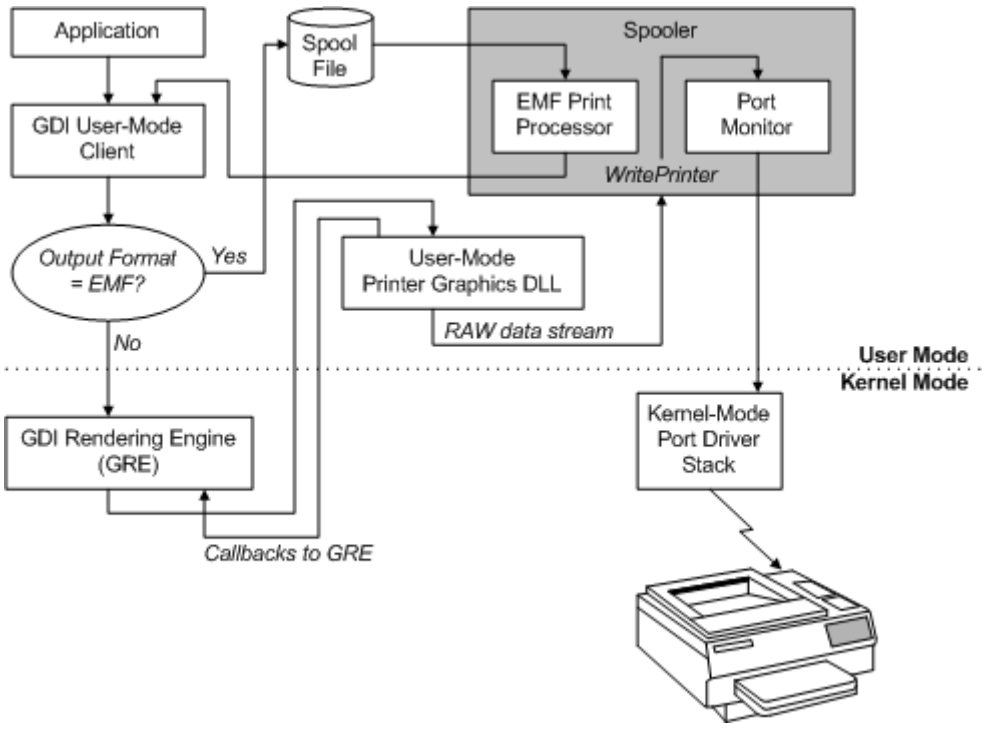

Figure 2: Printing workflow for V3 printer drivers [35]

5.2 V4 printer drivers

The main design goal of V4 printer drivers is architecture independence. To achieve this Microsoft uses the XPS format as the page description language. This XPS file can be sent directly to a supported printer or filters can be used to convert it to a format that is suitable

for the printer. Microsoft provides premade filters for Postscript and PDL6 printers. Microsoft also provides a module for rasterizing XPS documents. Custom filters can then be used to convert that raster output into a format suitable for the printer. An overview of the architecture of V4 printer drivers can be seen in Figure 3. [36]

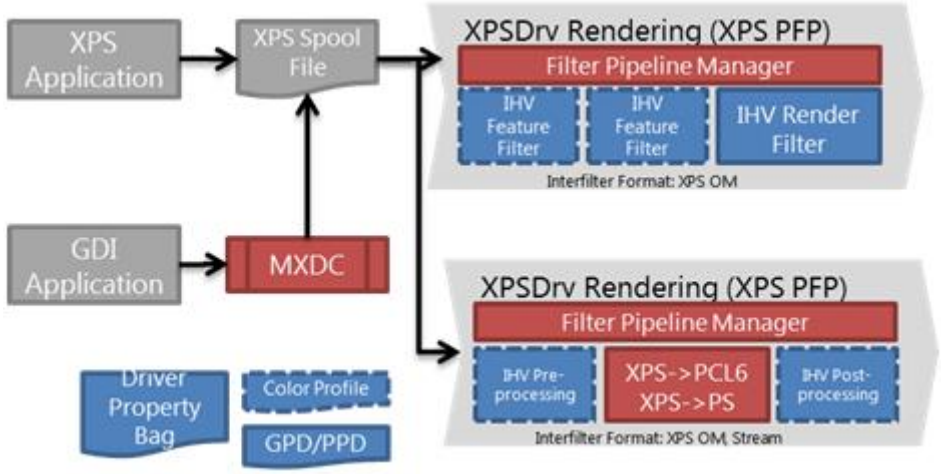

Figure 3: V4 printer driver architecture [36]

V4 printer driver user interfaces (UI)

With V4 drivers Microsoft introduced support for Universal Windows Platform (UWP) applications for printer drivers. This allows developers to create more complex applications for printers that make it easier for users to control complex print devices. V4 printer drivers also retain support for Windows desktop UI applications for printer drivers.

Both the UWP application and the Windows desktop UI based applications can be used to change print settings. The main difference between the two approaches is that UWP applications are designed to work best on touch-based interfaces and Windows Desktop UI is meant to integrate into existing Windows desktop applications. Even though the two approaches are quite different at the UI level they can still share the business logic and the connection to the print device. A high-level overview of the two different UI systems and their connection to the print device can be seen in Figure 4.

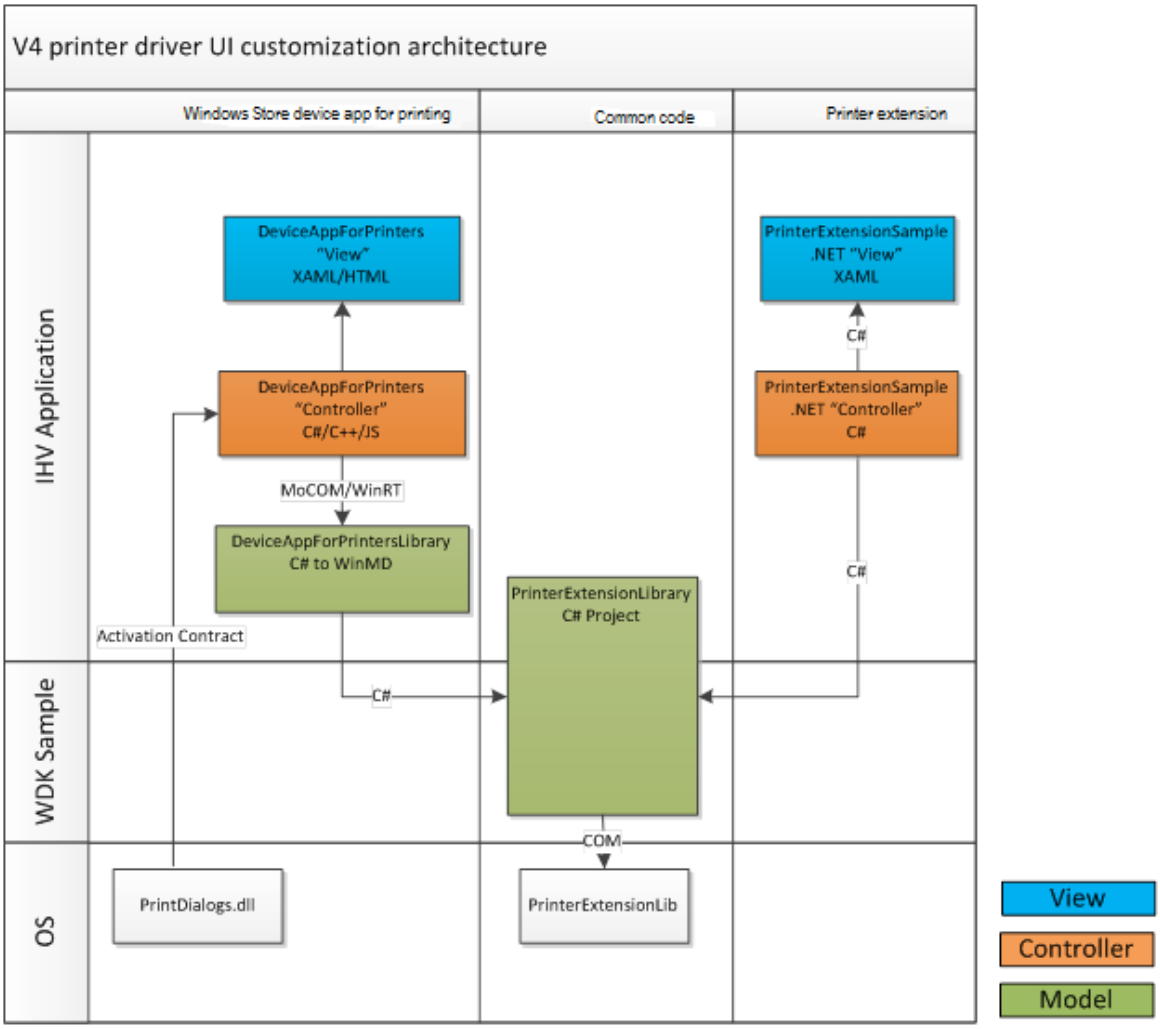

Figure 4: High-level overview of different UI approaches for V4 printer drivers [37]

UWP applications for printers also add support for working with print jobs after printing, this allows developers to capture the print job before it is sent to the printer and modify it or provide the user with extra customization options. [37]

XPS document format

XML Paper Specification (XPS) is a cross-platform document format created by Microsoft and used by Windows as the standard page description language in V4 printer drivers.

XPS uses a ZIP file as the packaging format and XML as the data format of the document. XPS is a fully contained format. This means that the XPS file contains all the data necessary to render the page correctly, including all the resources like fonts and images. In addition to the elements required for rendering the document, XPS documents can also include optional elements that are not required for rendering. These include print tickets that describe the print parameters that should be used for printing the document.

XPS documents consist of one or more documents, each document can contain one or more pages. Each page consists of different elements like text, images and vector graphics. Figure 5 shows the structure of an XPS document and all the possible elements it can contain. All the possible relationships between these elements are also shown. [38]

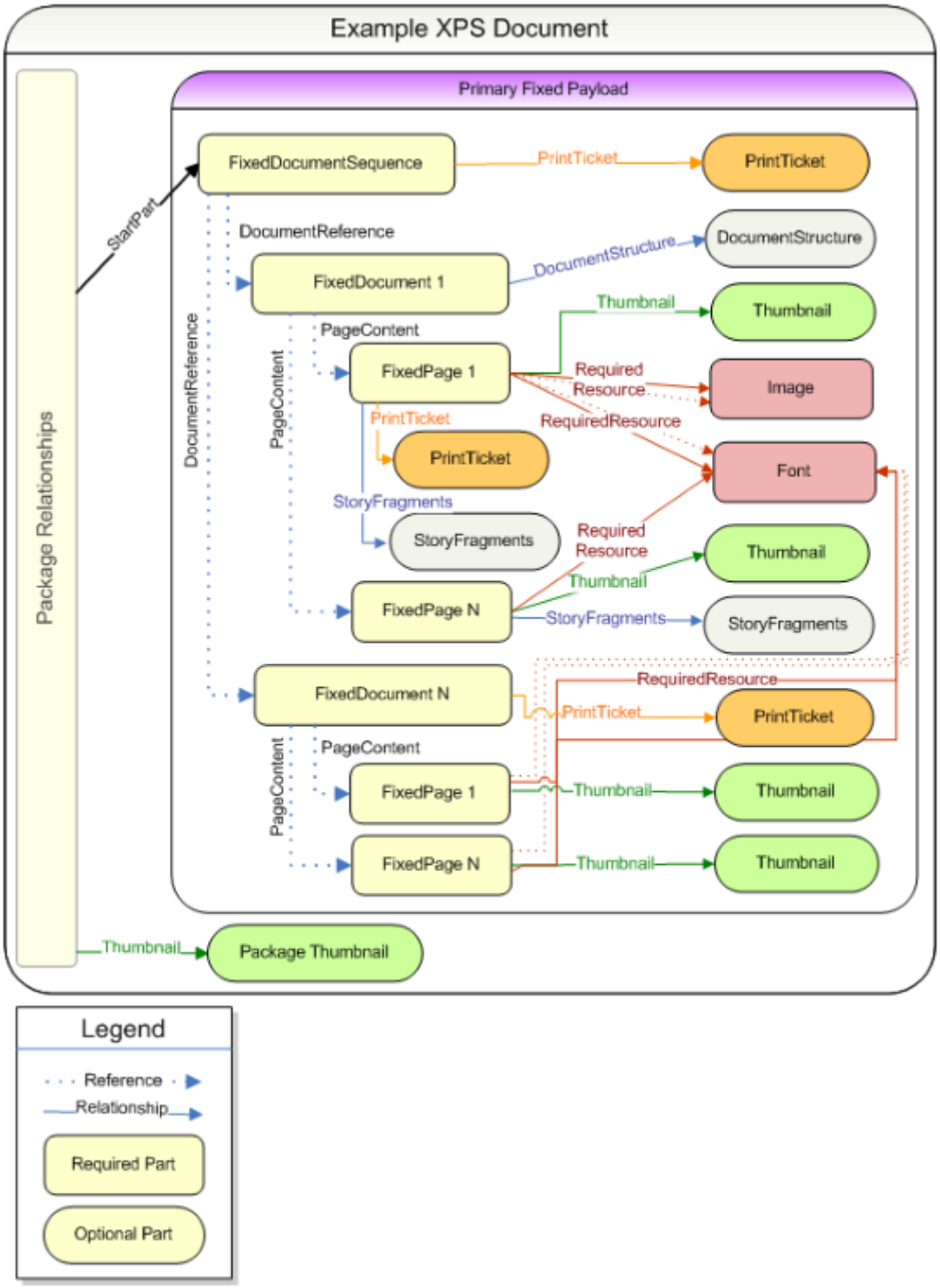

Figure 5: XPS Document structure [38]

5.3 Network printing in Windows

Microsoft Windows supports many different protocols for network printing. Among the supported protocols are IPP, Line Printing Daemon (LPD) protocol and Server Message Block (SMB) [39]. Figure 6 shows a full overview of all the supported printing protocols for Windows. Most of these protocols do not support driverless printing.

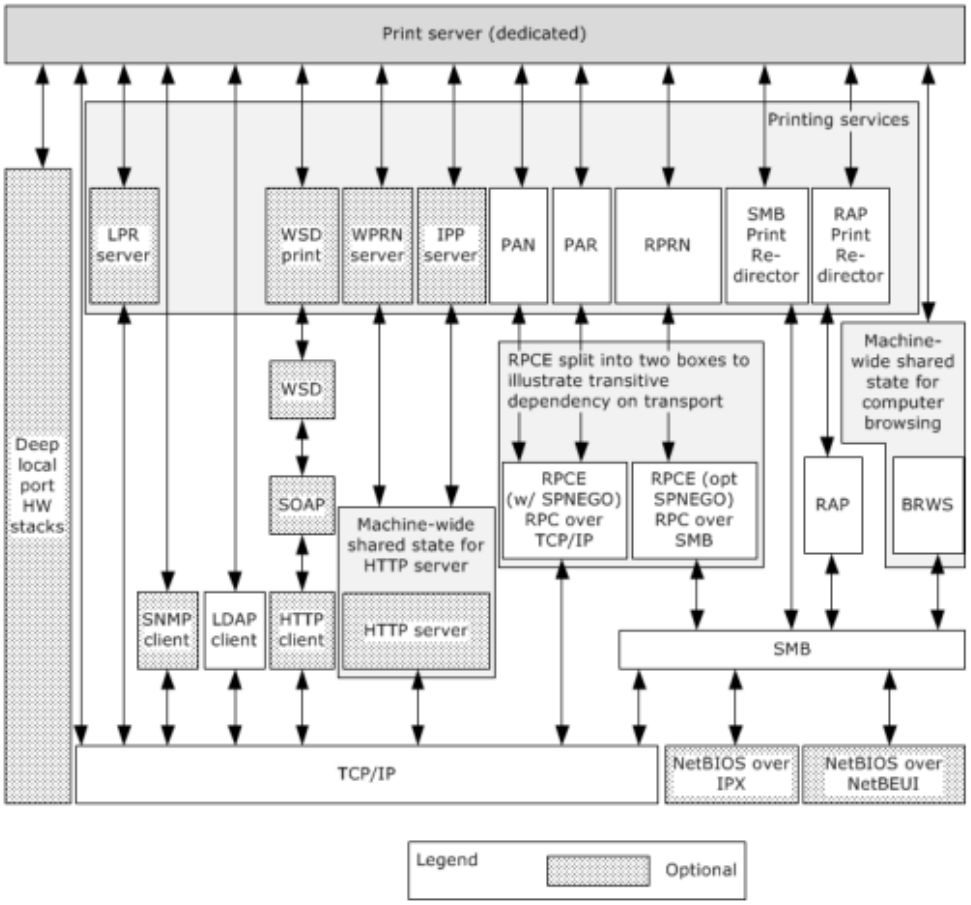

Figure 6: Overview of supported printing protocols for Microsoft Windows [39]

Windows only supports version 1.0 of the IPP protocol. This means that only a very limited set of IPP functions are supported. Printing using IPP also requires a driver on Windows, this driver provides the capabilities of the printer to Windows and handles the conversion of the printed document to the correct format.

Driverless printing in Windows

Starting with Windows 10 October 2018 update, Windows supports Mopria for driverless printing. Mopria devices can be used in Windows without a driver if access to Windows Update is not available. If access to Windows Update is available, then a suitable driver is downloaded from Windows Update. [40]

6 Printing on macOS and Linux

Both macOS and Linux use CUPS as a printing back-end. CUPS is an open-source printing back-end developed by Apple. CUPS has native support for IPP and different driverless printing protocols that are based on IPP, like IPP Everywhere and AirPrint. CUPS also has support for filters that can be used to convert a print job into a format that is suitable for the printer. [41]

6.1 macOS

The recommended method for printing in macOS is AirPrint. When using AirPrint, no additional software is required to use a printer. AirPrint printers are discovered automatically by macOS using Bonjour. AirPrint printers can also be added manually by using the printer's IP address.

macOS also supports HP Jetdirect and LPD protocols for network printers. Printers using these protocols usually require the installation of drivers for full functionality. Limited functionality may be available without drivers as well.

USB and Bluetooth printers can also be used with macOS. Depending on the printer model, drivers might be needed for full support. [42]

6.2 Linux

Almost all Linux distributions use CUPS as their printing back-end. CUPS has native support for IPP and IPP Everywhere. CUPS also supports LPD and HP Jetdirect protocols for network printing. [43]

The default page description language in CUPS is PDF. CUPS uses filters to convert jobs into a format that is supported by the printer. Filters are similar to drivers on Windows. [44]

7 Printing on mobile devices

Printing on mobile devices is mostly accomplished by using a driverless printing protocol. For iOS this protocol is AirPrint and for Android it is Mopria.

7.1 Printing on iOS

The default and only natively supported printing protocol for iOS is AirPrint. There are multiple ways for discovering AirPrint printers on iOS. The most common way is to find printers from the local network using Bonjour. These printers are discovered automatically by the device and shown to the user when they attempt to print. AirPrint printers can also be advertised using regular DNS servers. The DNS server has to have records for the device which usually means that the user is connected to an internal network with a local DNS server. [25]

Another way for discovering AirPrint printers is the use of specialized iBeacons. iBeacons are specialized BLE beacons that can be used for different purposes. One of these use cases is advertising AirPrint printers. [25]

AirPrint printers can also be added to iOS devices manually by using a Mobile device management (MDM) profile. [45]

7.2 Printing on Android

The default print service for Android devices starting from version 8.0 is Mopria. Mopria devices are discovered automatically and users can print to them without installing any extra software. [46]

Android also supports custom print services. These services use the Android API to receive print jobs from the default print dialog and then pass these jobs onto a print device. Custom print services are similar to regular Android applications and have access to similar features as regular apps. This allows developers to create complex print services that can use different protocols to communicate with printers. The custom print service also handles the discovery of new printers. [47]

8 Azure Service Bus

Azure Service Bus is a fully managed message broker with message queues and publish-subscribe topics. It uses messages to transfer data between different applications. A message consists of metadata and message data and uses the Advanced Messaging Queuing Protocol (AMQP) 1.0 as the transport protocol for messages. [48]

Azure Service Bus has three main components – sender, broker and receiver. The sender is responsible for creating the messages and sending them to the broker. The broker collects messages from senders and stores them until they can be sent to the receiver. The receiver listens for messages from the broker and accepts them. [48]

There are two different ways of sending messages over the service bus – queues and topics. [48]

Queues are usually used for point-to-point communication. Figure 7 shows an example of a queue. The sender sends messages to a queue that is stored on the broker. The broker receives these messages and stores them until a receiver is available to receive the messages. A receiver connects to the broker and requests messages from the queue. The messages are sent to the receiver only when the receiver requests them. [48]

Figure 7: Azure service bus queue [48]

Topics are usually used in a publish-subscribe configuration. A single topic can have multiple subscribers who all receive messages sent by a sender. Figure 8 shows an example of a topic. The subscribers act similarly to subscribers in a queue. Subscriptions can be configured to expire and be deleted automatically after a certain time. [48]

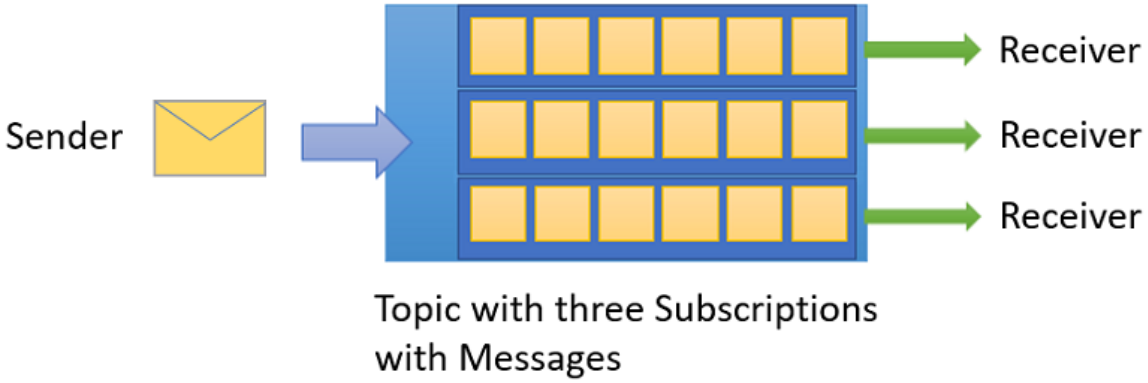

Figure 8: Azure Service Bus topic [48]

9 Overview of the virtual printer system for Print in City

Currently, the only way for users to upload jobs to Print in City is by using the website. The website accepts only PDF and JPEG files. This means that users need to convert all their documents into one of the supported formats themselves. For most documents, the user has to either export the document to PDF or use a PDF printer to "print" the document as PDF. This process adds an extra step to the users' workflow and discourages them from using Print in City.

These extra steps can be removed by integrating Print in City directly into the system print dialog as a printer. This is achieved by designing a web service that acts as a printer and forwards print jobs to Print in City.

9.1 Design requirements

The main requirement for the system is seamless integration with the printing system on the user's computer. It should support all the common operating systems currently in use. For desktops, this includes Microsoft Windows, Apple macOS, and Linux. For mobile devices, this includes Apple iOS and Android.

The user experience should be as similar as possible to the web version of Print in City. This means that the user should have similar print settings available to them, and they should be able to use the same authentication methods. The user must be able to choose the print point where they wish to pick up their job. The system must automatically calculate the price of the print job and inform the user if they have enough credit available to print to the selected print point. The user should also be able to add more credit to their account if needed.

The print parameters currently available for the user on the web are colour mode, page size, duplex mode, and the number of copies. The parameters should support the same values as the current web client. These values are listed in Table 2.

Table 2: Print parameters supported by Print in City and their possible values

Parameter	**Possible values**
Colour mode	Black and white, colour
Page size	A4, A3
Duplex mode	One-sided, Two-sided
Copies	1–N

Print in City currently uses a centralized single sign-on (SSO) solution, allowing users to use a single set of credentials to sign into their Print in City account in different regions. The SSO solution uses OpenID Connect to handle authentication. This SSO should also be implemented for the new system to provide a consistent experience for the users.

9.2 Design overview

The design of the virtual printer system consists of a web service that acts as a virtual printer and client applications that integrate the virtual printer with different operating systems.

The virtual printer service accepts print jobs from client applications and forwards them to Print in City. It also provides information about the virtual printer to clients on request.

The client applications are different depending on the operating system. For this thesis, prototype versions of client applications were developed for Windows and macOS. These two platforms were chosen because they are the biggest desktop platforms currently on the market and the most used by Print in City users.

10 Virtual printer service

The virtual printer service is the main service of the virtual printer system. It is responsible for communicating with clients. This includes providing clients information about the capabilities of the virtual printer and accepting jobs from clients.

The virtual printer service is also responsible for communicating with Print in City and forwarding submitted print jobs. Depending on the client application, the service may also handle the authentication of clients and conversion of printed documents.

Figure 9 shows an overview of the workflow that is used by the virtual printer service when processing an incoming print job.

The virtual printer service has three different methods for communicating with client applications. The primary communication protocol is IPP. Client applications use IPP to get information about the virtual printer and submit jobs to the printer. In addition to IPP, a REST API and Azure Service Bus are also used for communicating with certain clients. The REST API is used to download metadata about the virtual printer to advertise the printer on a computer. For example, it is used by the client application on macOS to download the TXT records needed to advertise the printer using Bonjour. Azure Service Bus is used to send notifications to client applications about the progress of the submitted print job. Azure Service Bus was chosen for this because IPP provides limited capabilities for two-way communication between the printer and client.

10.1 IPP communication

The virtual printer acts as a regular IPP printer and supports all the IPP operations and attributes required by the IPP 2.0 protocol. It also supports the additional parameters required for AirPrint support. The virtual printer is also backwards compatible with IPP versions 1.1 and 1.0.

A custom parser was written to parse the incoming IPP messages and create new IPP messages. This was done because there were no existing solutions designed to be as scalable as required for this application. Most of the existing solutions available on the market are designed to operate on an internal network and do not scale well for larger deployments. A custom solution also provides more customizability, which is useful since the virtual printer needs to operate differently than a regular print server.

The virtual printer supports all the print parameters specified in the design requirements (Table 2 in chapter 9.1). Client authentication is also supported, which is required for clients that use driverless printing protocols. Currently, only HTTP Basic authentication is supported since this is the only suitable method available in the current IPP standard. There is a proposed addition to the IPP standard that adds support for OAuth authentication to IPP, but it is currently not supported by most clients.

The PDLs supported by the printer are PDF, JPEG, Apple Raster and XPS. Support for PDF, JPEG and Apple Raster is required by AirPrint, and XPS is the default PDL for Windows clients. PDF is also the default PDL for Android and Linux clients.

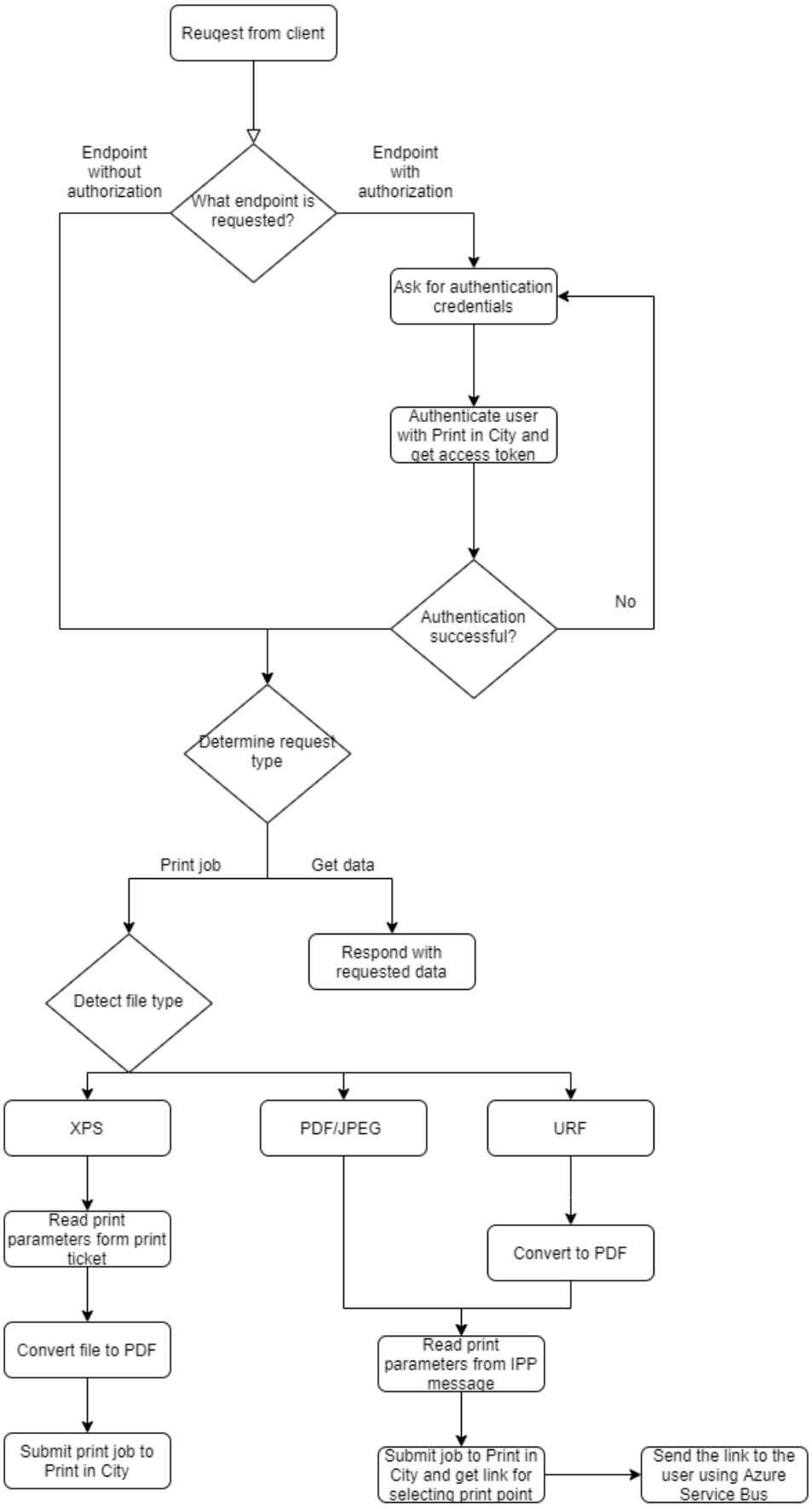

Figure 9: Print job processing workflow of the virtual printer service

10.2 Document conversion

The virtual printer is responsible for converting submitted print jobs into a format suitable for Print in City. Print in City supports documents in PDF and JPEG formats, but some client applications do not support these file formats or require that the printer supports additional file formats.

XPS files

The default file format for Windows print jobs is XPS. The use of other non-raster formats would require a custom driver filter. Windows also does not support IPP fully and embeds print parameters inside the XPS file instead of sending them as IPP attributes. The virtual printer needs to extract these print parameters from the XPS file and convert them into a format suitable for Print in City. The XPS file also needs to be converted to PDF before it can be submitted to Print in City.

Multiple different methods were considered for converting XPS files to PDF, and all the methods were tested using the XPS Documents Print Sample provided by Microsoft. This package includes test files that are designed to test the rendering capabilities of devices [49]. Out of all the tested methods, only Microsoft Print to PDF system managed to render all the test files correctly.

Microsoft Print to PDF is a virtual PDF printer that is included with all Windows installations starting from Windows 10. It can be used to print documents as PDF files from the Windows print dialog.

Apple Raster files

Apple Raster (URF) is a raster file format created by Apple to be used with the AirPrint. It is a proprietary file format – the exact specification is only available through Apple. Apple Raster is similar to other raster formats like PWG raster, with a few differences.

Since there are no third-party libraries available for C# that can parse Apple Raster files, a conversion library was written from scratch to convert Apple Raster files to PDF. The library converts each rester file page to an image and places the image on a page in a PDF file.

10.3 Communication with Print in City

The virtual printer communicates with Print in City using a REST API. It uses this REST API to forward print jobs to Print in City. Client applications handle all other communication with Print in City.

The virtual printer also handles user authentication for some clients. This is done using OpenID Connect.

11 Client applications

There are currently two client applications available for the virtual printer – a Windows application and a macOS application. There are plans to add support for Linux, iOS and Android in the future. Both of the available applications use IPP for communicating with the virtual printer.

11.1 Windows application

The Windows application consists of a printer driver and a driver companion application to support additional print options. These two components are linked together using a Device Metadata package that stores information about the virtual printer.

Printer driver

The Windows application uses a V4 printer driver to provide Windows information about the printer. The driver consists of four files: the printer information INF file, driver manifest file, generic printer description (GPD) file and a pipeline configuration file.

The printer information file consists of nine sections that define different attributes about the printer and the driver. An example of the printer information file can be seen in Appendix A – Printer information file example These sections are the version section, manufacturer section, model section, installer section, copy section, source disk section and strings section.

The version section contains information about the driver and the manufacturer of the driver. The parameters defined in the version section, their values and descriptions can be seen in Table 3.

Table 3: Version section parameters and values

Parameter name	Parameter value	Parameter description
Signature	$Windows NT$	Supported Windows version
Provider	Overall	Provider of the driver (usually the manufacturer)
CatalogFile	picdriver.cat	Name of the catalog file that contains signatures of driver files
ClassGUID	{00000000-0000-0000-0000-000000000000}	An unique GUID for this driver
Class	Printer	Defines this driver as a printer driver
DriverVer	02/03/2021,1.0.0.1	Driver version
ClassVer	4.0	Defines this driver as a V4 driver

The manufacturer section defines all the manufacturers that will be displayed to the user during the driver installation. There is only one manufacturer in this section, which is Print in City. All the platforms supported for this manufacturer are also listed here. Since this driver does not have any platform-specific features, it can be made available for all Windows platforms. These platforms are NTx86, NTamd64, NTarm and NTarm64.

The model section consists of multiple sub-sections. There is a sub-section for each manufacturer and platform defined in the Manufacturer section. For this driver, this means that there are four different sub-sections. Each sub-section contains information about the models available for this manufacturer and platform. For each defined model, an installer section is specified.

The installer section defines the operations for installing the driver. For this driver, the installer section contains only one sub-section since the installation procedure is the same for all platforms. This sub-section contains a single CopyFiles directive that references a sub-section in the copy section with a list of files to copy.

The copy section contains the actual list of files to be copied by the installer section and the destination of these files. The files can be divided into sub-sections, with each sub-section having at least two files. The DestinationDirs sub-section defines the destination for the CopyFiles operation. This destination is defined as 66000, which is the default driver directory for V4 printer drivers.

The source disk section defines the location of the source files for the driver. It consists of a SourceDiskFiles sub-section and multiple SourceDisksNames sub-sections.

The SourceDisksFiles sub-section defines all the source files and matches their location to a number that is referenced in the SourceDisksNames sub-sections.

There is a separate SourceDiskNames sub-section for each platform the driver supports. Each sub-section defines the folders that match the numbers set in the SourceDisksFiles section. This allows the use of different files for each platform without changing the name of the files.

The strings section defines any number of string variables that can be used in the file. This section can be used to define strings that need to be localized.

The driver manifest file defines general information about the driver. An example of the driver manifest file can be seen in Appendix B – Driver manifest file example. It consists of a single section that has five different parameters. These parameters, their values and descriptions can be seen in Table 4.

Table 4: Driver manifest file values [50]

Parameter	**Value**	**Description**
DriverCategory	PrintFax.Printer	Defines the category of the device
DataFile	Picdriver.gpd	The primary GPD or PPD file for the driver
PrinterDriverID	{00000000-0000-0000-0000-000000000000}	A unique ID that describes the driver
Flags	HostBasedDevice	Additional attributes associated with the driver
RequiredFiles	UNIRES.DLL, STDNAMES.GPD, MSXPSINC.GPD	Includes files from ntprint.inf or ntprint4.inf

The generic printer description (GPD) file is used to provide Windows information about the printer's capabilities. An example of a GPD file can be seen in Appendix C – Generic

printer description file. The file consists of a set of mandatory root elements that define metadata about the printer, followed by feature elements that define different printer capabilities. The mandatory elements and their values can be seen in Table 5.

Table 5: GPD file root level elements

Parameter	**Value**	**Description**
GPDFileName	picdriver.GPD	Name of the file
GPDFileVersion	1.0	GPD file version
GPDSpecVersion	1.0	GPD specification version, must be 1.0
Include	StdNames.gpd	Includes another .gpd file
Include	msxpsinc.gpd	Includes msxpsinc.gpd, this is required for V4 drivers
ModelName	PiC printer	Name of the printer
MasterUnits	PAIR(1200, 1200)	Printer master units, not important for a virtual printer but must be specified
PrinterType	PAGE	Required attribute, can be PAGE, SERIAL or TTY
MaxCopies	999	Maximum number of copies the printer can support
Command	CmdSendBlockData { *Cmd : "" }	Mandatory command

The file has five feature elements that define the capabilities of the printer. These capabilities match the ones defined in the design requirements. Table 6 shows the five elements and their values.

Table 6: Printer features in GPD file

Feature	**Values**
Orientation	PORTRAIT, LANDSCAPE
Resolution	600x600 DPI
PaperSize	A4, A3
Duplex	NONE, VERTICAL, HORIZONTAL
ColorMode	Mono, 24bpp

The file also contains cursor command definitions. These are used to control the movement of the cursor on some printers. The virtual printer does not use these cursor commands, so only the three required by Windows are defined.

The pipeline configuration file is used to configure the rendering pipeline of the printer. The virtual printer does not use a rendering pipeline. The XPS file generated by Windows is directly sent to the virtual printer. The pipeline configuration file is required for all drivers, but it may contain only an empty filters tag if no filtering is required.

Metadata file

The metadata file is a ZIP file that contains multiple XML files that provide additional information about the printer to Windows. This information includes the name and description of the printer in different languages, an icon for the printer, Universal Windows Platform applications associated with the printer and information about the printer's hardware.

The metadata file is required to associate a Universal Windows Platform application with the printer.

User Interface

The Windows application uses a combination of the system print dialog and a Universal Windows Platform (UWP) device application for the user interface. The main print settings are set using the system printing dialog. The values for these settings are provided by the driver. The UWP application is used for Print in City specific settings that are not supported by the system print dialog.

The UWP application has two main views. The first view is shown when the user starts the application. The application can be launched from the Start menu or the Printers & scanners settings.

The first view allows users to log in to their Print in City account and select the default print point they wish to use for printing. It also allows users to change their Print in City account settings. This view can be seen in Figure 10.

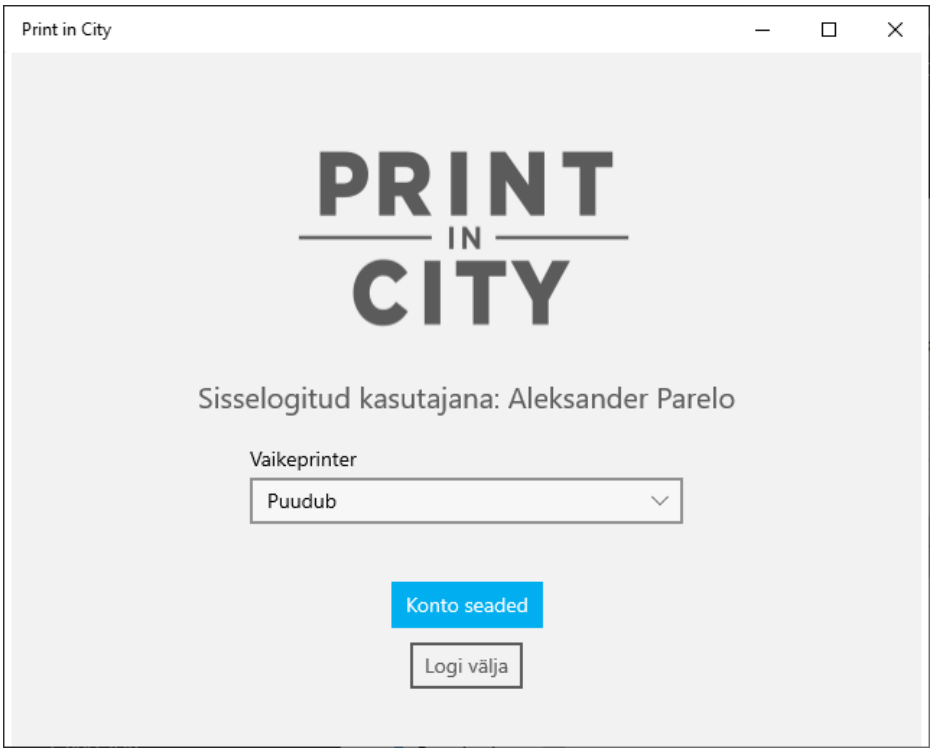

Figure 10: UWP application launched in standalone mode

The second view is launched by the system's printing process when a user submits a print job to the printer. The background process can also access the job data and the print ticket that contains the job parameters. This view allows the users to set additional Print in City specific print parameters, which are added to the print ticket. Figure 11 shows a diagram of the process.

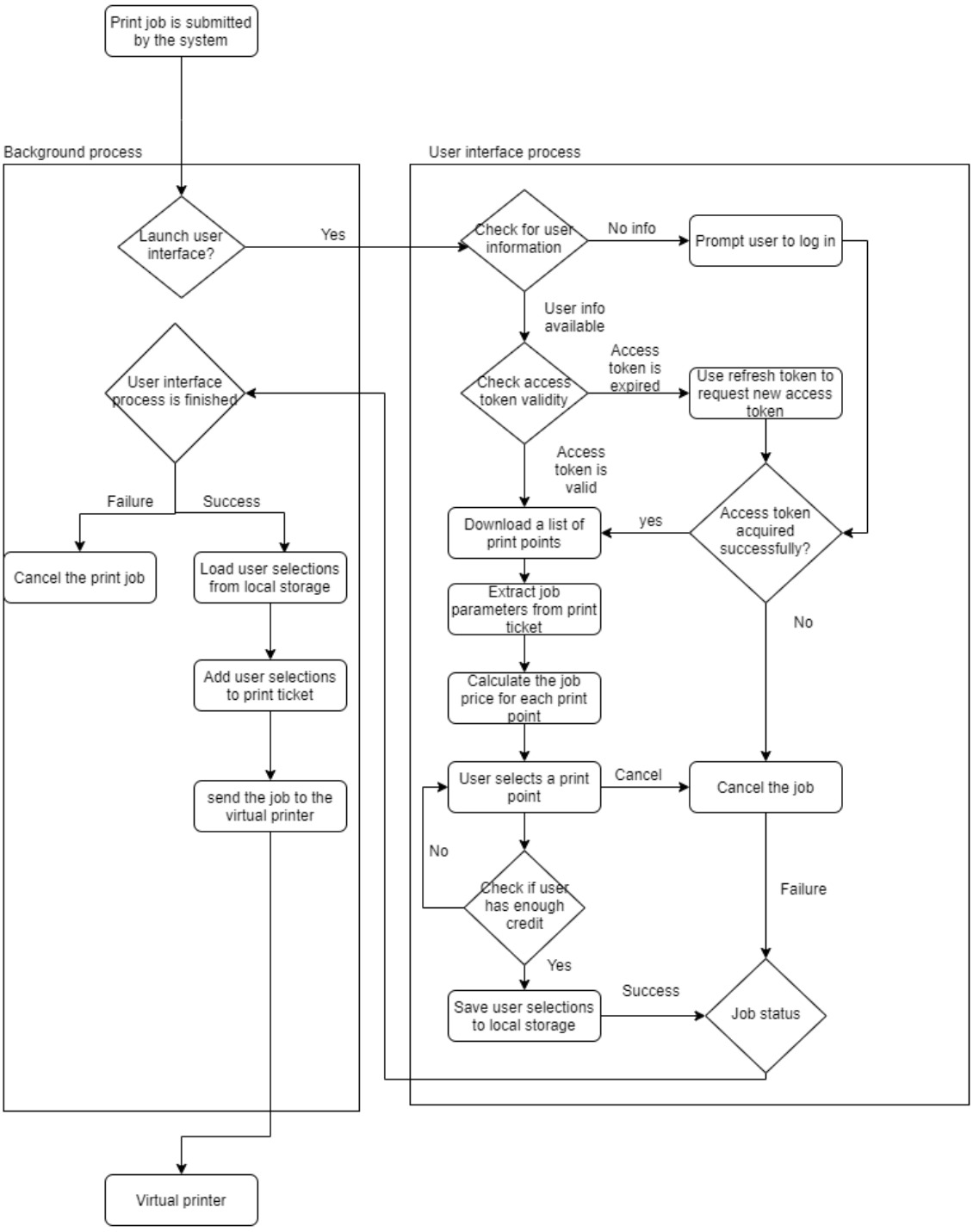

Figure 11: Print process overview of the Windows application

The background process can be used to launch a user interface. For Print in City, the user interface is launched for every print job. The user interface process first checks if the user

has already signed in to Print in City through the first view. If the user has not signed in, they are redirected to a sign-in page. If the user has already signed, then the validity of the authorization token is verified. If the token is not valid, then the refresh token is used to request a new token for the user.

If the token is valid, then the list of print points available to the user is requested from Print in City. The cost of the print job is then calculated for each print point. The print points and costs are then shown to the user. If the user has previously selected a default print point, then that point is pre-selected. The user can then either verify that the selected print point is correct or choose another print point. If they do not have enough credit to print at a specific print point, then that point is greyed out. Once the user has chosen a print point, they can click the print button to submit the job to Print in City. An example of this view can be seen in Figure 12.

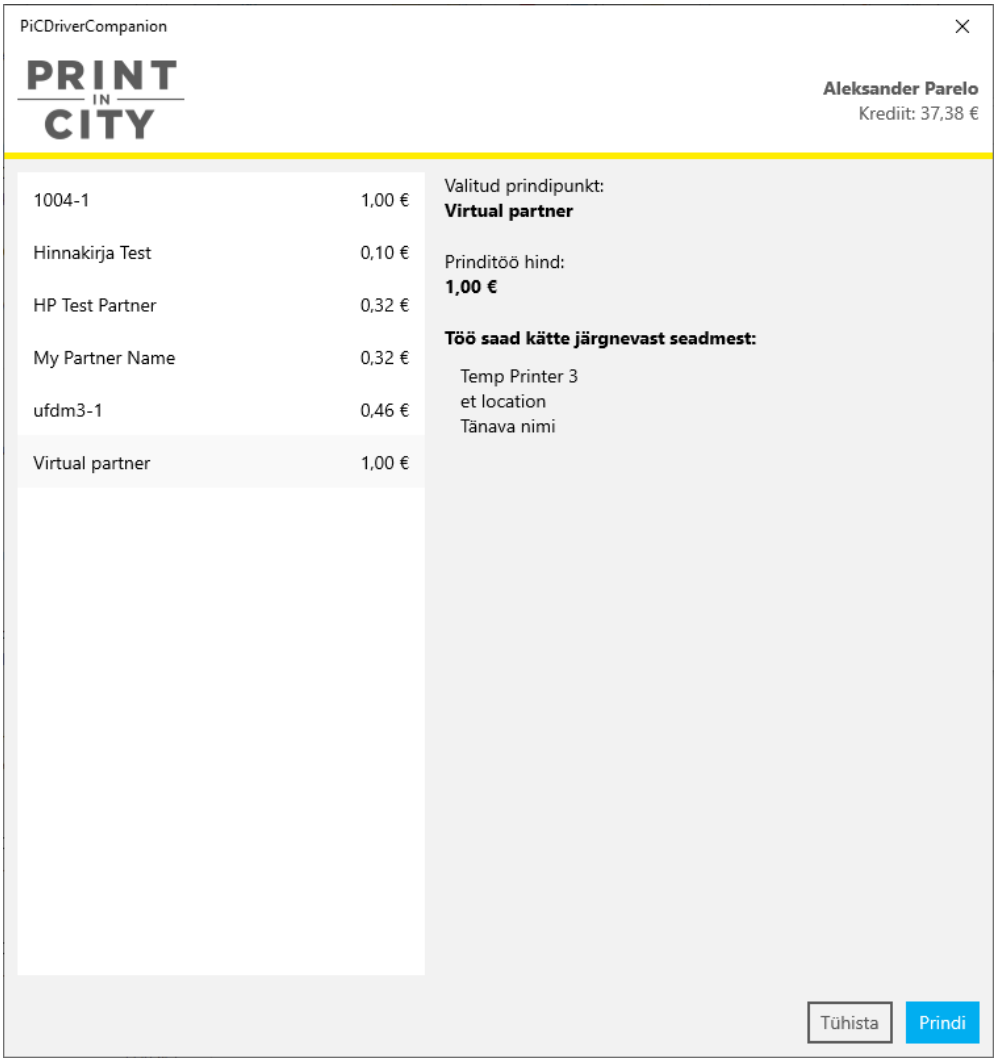

Figure 12: UWP application launched by the print process

After the user clicks the print button, the user interface process lets the background process know that it has finished and exits. The background process can now access the user data and continue processing the job. It writes the users authorization token and the selected print point into the print ticket file embedded in the XPS file sent out by the printer. It then sends the file to the virtual printer.

11.2 macOS application

The macOS application consists of a client application and an AirPrint printer. The client application is used to advertise the AirPrint printer to the operating system, allowing macOS to find the printer automatically. The client application also receives notifications from the virtual printer and displays them to the user. The AirPrint printer handles communication with the virtual printer and submitting print jobs.

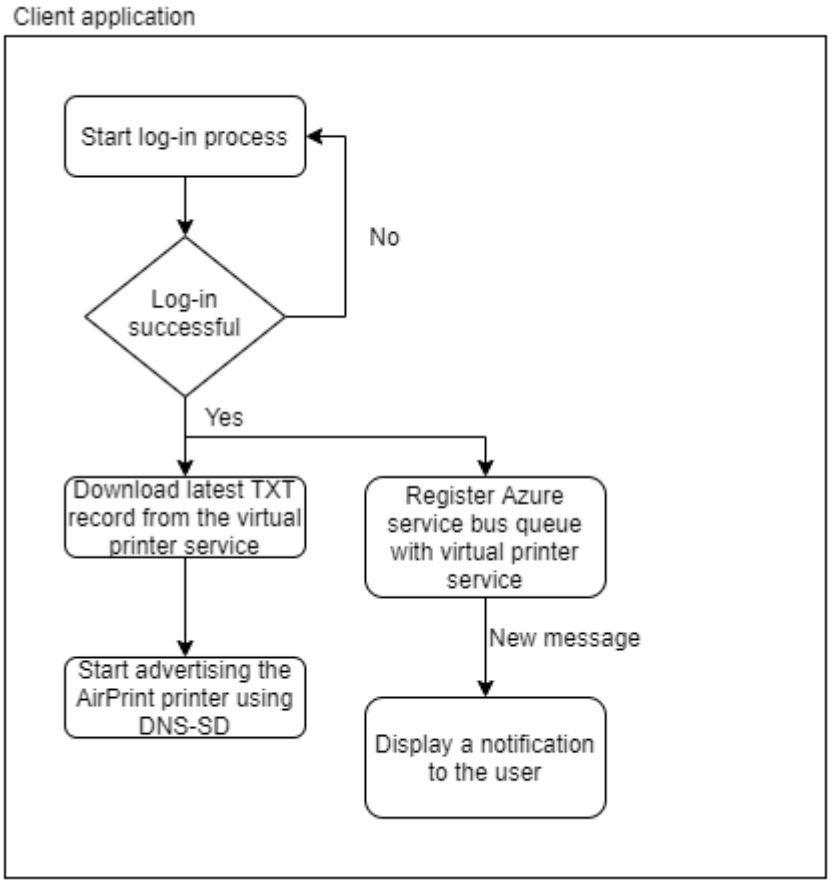

Figure 13: Client application workflow for macOS

Client application

The client application advertises the AirPrint printer on the local computer and receives and displays notifications from the virtual printer service. It also handles authentication to Print in City, which is required to send notifications to the user from the virtual printer service.

To use the macOS application, the user has to open the client application and log in to Print in City. Once the user has successfully logged in to Print in City, the application downloads an up to date TXT record from the virtual printer service. It then uses that record to advertise the printer to the computer using Bonjour on the loopback interface. An example of the TXT record can be seen in Table 7. The advertising must be done by the client application and cannot be done by the virtual printer because mDNS only works over the local network.

The client also subscribes to an Azure Service Bus queue created by the virtual printer service to receive notifications. This process is illustrated in Figure 13.

Table 7: TXT record values

Key	Value	Description
Adminurl	https://ipp-test.printincity.com	URL for the administration page of the printer
Air	username, password	Authentication methods supported by the printer
Kind	document	Type of printer
Note	Print in City printer	Note about the printer
Pdl	image/urf, application/octet-stream, application/pdf, image/jpeg	PDLs supported by the printer
Product	(PiC)	Postscript product string of the printer
Rp	Ipp/unix	Path where the printer can be accessed
Color	T	Is color printing supported
Duplex	T	Is duplex printing supported
Fax	F	Is fax supported
PaperMax	Tabloid-A3	Largest supported paper size
TLS	1.2	Maximum TLS version supported
URF	CP255, DM1, FN3-4-5-8-20, IS1-4, OB10, PQ4, RS600, SRGB24, V1.4, W8	Universal raster format capabilities supported by the printer
UUID	00000000-0000-0000-0000-000000000000	UUID of the printer

AirPrint printer

The AirPrint printer is used to submit print jobs from macOS to the virtual printer service. To use the AirPrint printer, the user needs to add it in the macOS System Preferences as a printer. Once the user adds the printer, macOS automatically downloads the necessary

metadata from the virtual printer service. The user is also prompted to log in to their Print in City account using a username and password when adding the AirPrint printer. This is a limitation of the macOS platform. Since the AirPrint printer and the client application have no way of communicating, the user must log in to both separately.

When the user prints a job using the AirPrint printer, the printer automatically forwards the print job to the virtual printer service with the authentication information and print parameters attached. The virtual printer service then parses these parameters and forwards them to Print in City. Print in City responds with a link that directs the user to a web page where they can select their desired print point. This link is then sent to the client application using Azure Service Bus, and the application displays the link to the user with a notification. The user can then click on it to complete the print process.

12 Conclusions

The aim of this thesis is to improve the overall user experience of Print in City users. The current browser-based web service only allows the users to upload PDF and JPEG files. This means that the users need to take extra steps before they can actually use the service. The virtual printer system developed in this thesis makes using Print in City as easy as using a regular printer.

This virtual printer system consists of a web service and client applications. Together these components integrate Print in City directly into the system print dialog and allow users to print to Print in City directly in any file format.

A fully functional virtual printer service was developed from scratch. This web service handles the communication with clients and preprocesses the received print jobs before sending them to Print in City. This preprocessing includes extracting and collecting the required print parameters for Print in City, such as the user ID and the requested print point. It also converts the received print jobs into a format supported by Print in City.

Prototypes of client applications were developed for Microsoft Windows and Apple macOS. These client applications use different technologies to communicate with the virtual printer service. The Windows application consists of a printer driver and a companion application. The printer driver is used to define the capabilities of the virtual printer and the companion application is used to set Print in City specific settings for print jobs. These Print in City specific settings include the print point where the user wishes to pick up their print job and the account information for the user.

The macOS application consists of an AirPrint printer and a companion application. The companion application is responsible for advertising the AirPrint printer on the client computer and accepting notifications from the virtual printer service. The AirPrint printer handles communication with the virtual printer service and submits jobs to the service.

This virtual printer system makes using public printers even more convenient, decreasing the need to own a personal printer. With the rise of sharing economy and environmental awareness, demand for such solutions is increasing rapidly. This means that the new virtual printer system could increase the user base of Print in City and bring in new revenue for the company.

In the future, more testing of the developed prototypes is required before the system can be released to the public. New platforms can also be added, for example, mobile operating systems like Apple iOS and Android. Support for Linux based desktop computers can also be added in the future. New ways of discovering Print in City printers could also make the user experience even better. These might include the use of iBeacons to advertise nearby Print in City printers to mobile users.

13 References

[1] “Print In City.” https://ee.printincity.com/places/ (accessed Apr. 09, 2021).

[2] “Eesti Raamatukoguhoidjate Ühing – Posts | Facebook,” Mar. 14, 2012. https://www.facebook.com/permalink.php?story_fbid=322875877770670&id=245435482167038 (accessed May 13, 2021).

[3] European Computer Manufacturers Association, “Standard ECMA-140: Document Printing Application.” European Computer Manufacturers Association, Jun. 1990. Accessed: May 14, 2021. [Online]. Available: https://www.ecma-international.org/wp-content/uploads/ECMA-140_1st_edition_june_1990.pdf

[4] L. McLaughlin, “Line printer daemon protocol,” RFC Editor, RFC1179, Aug. 1990. doi: 10.17487/rfc1179.

[5] C.-U. Manros, “The birth of the Internet printing protocol (IPP),” *StandardView*, vol. 6, no. 4, pp. 135–139, Dec. 1998, doi: 10.1145/338183.338184.

[6] R. deBry, R. Herriot, P. Powell, S. A. Isaacson, and T. Hastings, “Internet Printing Protocol/1.1: Model and Semantics.” https://tools.ietf.org/html/rfc2911 (accessed Mar. 24, 2021).

[7] R. Herriot, S. Butler, and J. Wenn, “Internet Printing Protocol/1.1: Encoding and Transport.” https://tools.ietf.org/html/rfc2910 (accessed Mar. 26, 2021).

[8] R. Herriot and H. Lewis, “Internet Printing Protocol (IPP): The ‘ippget’ Delivery Method for Event Notifications.” https://tools.ietf.org/html/rfc3996 (accessed Mar. 26, 2021).

[9] K. Ocke, R. Herriot, R. deBry, P. Zehler, and T. Hastings, “Internet Printing Protocol (IPP): The ‘collection’ attribute syntax.” https://tools.ietf.org/html/rfc3382 (accessed Mar. 26, 2021).

[10] R. Herriot, H. Lewis, C. Kugler, and T. Hastings, “Internet Printing Protocol (IPP): Job and Printer Set Operations.” https://tools.ietf.org/html/rfc3380 (accessed Mar. 26, 2021).

[11] R. Herriot and T. Hastings, “Internet Printing Protocol (IPP): Event Notifications and Subscriptions.” https://tools.ietf.org/html/rfc3995 (accessed Mar. 26, 2021).

[12] I. McDonald and M. Sweet, “Internet Printing Protocol (IPP) over HTTPS Transport Binding and the ‘ipps’ URI Scheme.” https://tools.ietf.org/html/rfc7472 (accessed Mar. 26, 2021).

[13] I. McDonald and M. Sweet, “Internet Printing Protocol/1.1: Model and Semantics.” https://tools.ietf.org/html/rfc8011 (accessed Mar. 24, 2021).

[14] The Printer Working Group, “Internet Printing Protocol Version 2.0 (IPP/2.0).” Aug. 31, 2009.

[15] The Printer Working Group, “IPP Version 2.0, 2.1, and 2.2.” Oct. 30, 2015.

[16] S. Zilles, “Rationale for the Structure of the Model and Protocol for the Internet Printing Protocol.” https://tools.ietf.org/html/rfc2568 (accessed Apr. 03, 2021).

[17] “Bonjour Concepts.” https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Articles/about.html#//apple_ref/doc/uid/TP40002458-SW1 (accessed Apr. 24, 2021).

[18] Apple Inc., “Bonjour Printing Specification,” *2015*, no. 1.2.1, p. 28.

[19] S Cheshire, M. Krochmal, and Apple Inc., “DNS-Based Service Discovery.” Internet Engineering Task Force (IETF), Feb. 2013. Accessed: Apr. 24, 2021. [Online]. Available: https://www.ietf.org/rfc/rfc6763.txt

[20] OpenPrinting, “Driverless Printing,” *OpenPrinting*. https://openprinting.github.io/driverless/ (accessed Apr. 15, 2021).

[21] OpenPrinting, “Driverless Printing Standards And their PDLS,” *OpenPrinting*. https://openprinting.github.io/driverless/01-standards-and-their-pdls/ (accessed Apr. 17, 2021).
[22] “About AirPrint,” *Apple Support*. https://support.apple.com/en-us/HT201311 (accessed Mar. 26, 2021).
[23] G. Gruman, “OS X Mountain Lion arrives: The top 25 features,” *InfoWorld*, Jul. 25, 2012. https://www.infoworld.com/article/2606197/os-x-mountain-lion-arrives--the-top-25-features.html (accessed Mar. 26, 2021).
[24] “Apple’s iOS 4.2 Available Today for iPad, iPhone & iPod touch,” *Apple Newsroom (United Kingdom)*. https://www.apple.com/uk/newsroom/2010/11/22Apples-iOS-4-2-Available-Today-for-iPad-iPhone-iPod-touch/ (accessed Mar. 26, 2021).
[25] Apple Inc., “AirPrint Specification version 1.8.” Apple Inc., 2017.
[26] The Printer Working Group, “IPP Everywhere,” Jan. 28, 2013. http://ftp.pwg.org/pub/pwg/candidates/cs-ippeve10-20130128-5100.14.pdf (accessed Apr. 17, 2021).
[27] The Printer Working Group, “IPP Everywhere v1.1,” May 15, 2020. https://ftp.pwg.org/pub/pwg/candidates/cs-ippeve11-20200515-5100.14.pdf (accessed Apr. 17, 2021).
[28] “IPP Everywhere™ - Printer Working Group.” https://www.pwg.org/ipp/everywhere.html (accessed Mar. 26, 2021).
[29] “Mopria Alliance.” https://mopria.org/mopria-alliance (accessed Apr. 17, 2021).
[30] “What is Mopria?” https://mopria.org/what-is-mopria (accessed Apr. 17, 2021).
[31] “Member Benefits.” https://mopria.org/member-benefits (accessed Apr. 17, 2021).
[32] “Wi-Fi Direct Specification,” p. 200, 2020.
[33] barrygolden, “Introduction to Printing - Windows drivers.” https://docs.microsoft.com/en-us/windows-hardware/drivers/print/introduction-to-printing (accessed Apr. 03, 2021).
[34] M. Schiffel, “The New Microsoft V4 Printer Driver Model: What Admins Need to Know,” *ThinPrint Blog*, Sep. 20, 2016. https://blog.thinprint.com/the-new-microsoft-v4-printer-driver-model/ (accessed Apr. 03, 2021).
[35] barrygolden, “Introduction to Printer Graphics DLLs - Windows drivers.” https://docs.microsoft.com/en-us/windows-hardware/drivers/print/introduction-to-printer-graphics-dlls (accessed Apr. 03, 2021).
[36] barrygolden, “V4 Printer Driver Rendering Architecture - Windows drivers.” https://docs.microsoft.com/en-us/windows-hardware/drivers/print/v4-driver-rendering-architecture (accessed Apr. 03, 2021).
[37] barrygolden, “V4 Driver UI Architecture - Windows drivers.” https://docs.microsoft.com/en-us/windows-hardware/drivers/print/v4-driver-ui-architecture (accessed Apr. 03, 2021).
[38] Microsoft, “XML Paper Specifictaion.” Microsoft, Oct. 18, 1006.
[39] Microsoft, “Print Services Protocols Overview.” Microsoft, Nov. 05, 2018. Accessed: Jan. 04, 2021. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-prsod/97fd9728-f83e-48d0-aa2c-79471fd9e00a
[40] “Print From Windows 10.” https://mopria.org/print-with-windows-10 (accessed Apr. 17, 2021).
[41] “Overview of CUPS.” http://www.cups.org/doc/overview.html (accessed Apr. 17, 2021).
[42] “Add a printer on Mac,” *Apple Support*. https://support.apple.com/guide/mac-help/add-a-printer-mh14004/11.0/mac/11.0 (accessed Apr. 17, 2021).

[43] “Using Network Printers.” http://www.cups.org/doc/network.html (accessed Apr. 17, 2021).
[44] “OpenPrinting/cups-filters,” *GitHub*. https://github.com/OpenPrinting/cups-filters (accessed Apr. 17, 2021).
[45] “AirPrint MDM payload settings for Apple devices,” *Apple Support*. https://support.apple.com/guide/mdm/airprint-payload-settings-mdm3b4cf515/web (accessed Apr. 17, 2021).
[46] “Print From Android.” https://mopria.org/print-from-android (accessed Apr. 17, 2021).
[47] “PrintService,” *Android Developers*. https://developer.android.com/reference/android/printservice/PrintService (accessed Apr. 17, 2021).
[48] axisc, “Azure Service Bus messaging overview - Azure Service Bus.” https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview (accessed Apr. 24, 2021).
[49] VSC-Service-Account, “XPS Documents Print Sample - Code Samples.” https://docs.microsoft.com/en-us/samples/microsoft/windows-driver-samples/xps-documents-print-sample/ (accessed Apr. 22, 2021).
[50] barrygolden, “V4 Driver Manifest - Windows drivers.” https://docs.microsoft.com/en-us/windows-hardware/drivers/print/v4-driver-manifest (accessed Apr. 24, 2021).

I. Appendix A – Printer information file example

```
[Version]
Signature="$Windows NT$"
Provider=%ProviderString%
CatalogFile=picdriver.cat
ClassGUID={00000000-0000-0000-0000-000000000000}
Class=Printer
DriverVer=02/03/2021,1.0.0.1
ClassVer=4.0

;
; Manufacturer section.
;
; This section lists all of the manufacturers
; that we will display in the Dialog box
;
[Manufacturer]
%ManufacturerName%=Overall, NTx86, NTamd64, NTarm, NTarm64

;
; Model sections
;
; Each section here corresponds with an entry listed in the
; [Manufacturer] section above. The models will be displayed in the order
; that they appear in the INF file.
;

[Overall.NTx86]
"PiCDriver"     = PICDRIVER, PICPrinter,

[Overall.NTamd64]
"PiCDriver"     = PICDRIVER, PICPrinter

[Overall.NTarm]
"PiCDriver"     = PICDRIVER, PICPrinter

[Overall.NTarm64]
"PiCDriver"     = PICDRIVER, PICPrinter

;
; Installer Sections
;
; These sections control file installation, and reference all files that
; need to be copied. The section name will be assumed to be the driver
; file, unless there is an explicit DriverFile section listed.
;
[PICDRIVER]
CopyFiles=PICDRIVER_FILES

;
; Copy Sections
;
; Lists of files that are actually copied. These sections are referenced
; from the installer sections, above. Only create a section if it contains
; two or more files (if we only copy a single file, identify it in the
; installer section, using the @filename notation) or if it's a color
; profile (since the DestinationDirs can only handle sections, and not
; individual files).
;

[PICDRIVER_FILES]
picdriver.gpd
picdriver-pipelineconfig.xml
picdriver_extension.xml
picdriver-manifest.ini
picdriver_events.xml
```

```
[DestinationDirs]
DefaultDestDir=66000
;
;  Source Disk Section
;  Location of source files
;

[SourceDisksFiles]
picdriver.gpd                   = 1
picdriver-pipelineconfig.xml    = 1
picdriver-manifest.ini          = 1
picdriver_extension.xml         = 1
picdriver_events.xml            = 1

[SourceDisksNames.x86]
1 = %Disk1%,,,

[SourceDisksNames.amd64]
1 = %Disk1%,,,

[SourceDisksNames.arm]
1 = %Disk1%,,,

[SourceDisksNames.arm64]
1 = %Disk1%,,,


;
; Localizable Strings
;
[Strings]
Disk1="."
ProviderString = "Overall"
ManufacturerName="Print in City"
```

II. Appendix B – Driver manifest file example

```
[DriverConfig]
DriverCategory=PrintFax.Printer
DataFile=picdriver.gpd

; Note: Please replace the GUID below when building a production driver.
PrinterDriverID={00000000-0000-0000-0000-000000000000}
Flags=HostBasedDevice
RequiredFiles=UNIRES.DLL,STDNAMES.GPD,MSXPSINC.GPD
```

III. Appendix C – Generic printer description file

```
*%
*% This file is a sample GPD demonstrating basic printer features/options
*%
*%

*%*************************************************************************
*%: The following root-level attributes should be modified to suit your printer
*%*************************************************************************
*GPDFileName: "picdriver.GPD"
*GPDFileVersion: "1.0"
*GPDSpecVersion: "1.0"
*Include:      "StdNames.gpd"
*%***********************************************
*% V4 GPD-based printer drivers must include msxpsinc.GPD file
*%***********************************************
*Include:      "msxpsinc.gpd"
*ModelName:    "PiC printer"
*MasterUnits:  PAIR(1200, 1200)
*PrinterType:  PAGE
*MaxCopies:    999
*Command: CmdSendBlockData { *Cmd : "" }


*PrintSchemaPrivateNamespaceURI: "http://www.microsoft.com/USBHostBasedSample"

*%*************************************************************************
*%                              Orientation
*%*************************************************************************
*Feature: Orientation
{
    *rcNameID: =ORIENTATION_DISPLAY
    *DefaultOption: PORTRAIT

    *Option: PORTRAIT
    {
        *rcNameID: =PORTRAIT_DISPLAY
    }

    *Option: LANDSCAPE_CC270
    {
        *rcNameID: =LANDSCAPE_DISPLAY
    }
}


*%*************************************************************************
*%                               Resolution
*%*************************************************************************
*Feature: Resolution
{
    *rcNameID: =RESOLUTION_DISPLAY
    *DefaultOption: Option1

    *Option: Option1
    {
        *Name: "600 x 600 " =DOTS_PER_INCH
        *DPI: PAIR(600, 600)
        *TextDPI: PAIR(600, 600)
        *SpotDiameter: 100
    }
}



*%*************************************************************************
*%                             Paper Size
```

```
*%****************************************************************************
*Feature: PaperSize
{
    *rcNameID: =PAPER_SIZE_DISPLAY
    *DefaultOption: A4


    *Option: A4
    {
        *rcNameID: =RCID_DMPAPER_SYSTEM_NAME  *%  1008
        *switch: Orientation
        {
            *case: PORTRAIT
            {
                *PrintableArea: PAIR(9000, 12600)
                *PrintableOrigin: PAIR(200, 200)
                *CursorOrigin: PAIR(0, 0)
            }
            *case: LANDSCAPE_CC270
            {
                *PrintableArea: PAIR(9000, 12600)
                *PrintableOrigin: PAIR(200, 200)
                *CursorOrigin: PAIR(0, 0)
            }
        }
    }

    *Option: A3
    {
        *rcNameID: =RCID_DMPAPER_SYSTEM_NAME  *%  1008
        *switch: Orientation
        {
            *case: PORTRAIT
            {
                *PrintableArea: PAIR(9000, 12600)
                *PrintableOrigin: PAIR(200, 200)
                *CursorOrigin: PAIR(0, 0)
            }
            *case: LANDSCAPE_CC270
            {
                *PrintableArea: PAIR(9000, 12600)
                *PrintableOrigin: PAIR(200, 200)
                *CursorOrigin: PAIR(0, 0)
            }
        }
    }
}

*%****************************************************************************
************
*%                                        Duplex Type
*%****************************************************************************
************
*Feature: Duplex
{
        *rcNameID: =TWO_SIDED_PRINTING_DISPLAY
        *DefaultOption: NONE
        *Option: NONE
        {
                *rcNameID: =NONE_DISPLAY
        }
        *Option: VERTICAL
        {
                *rcNameID: =FLIP_ON_LONG_EDGE_DISPLAY
        }
        *Option: HORIZONTAL
        {
                *rcNameID: =FLIP_ON_SHORT_EDGE_DISPLAY
```

```
        }
}


*%*****************************************************************************
***********
*%                                        Color Mode
*%*****************************************************************************
***********
*Feature: ColorMode
{
        *rcNameID: =COLOR_PRINTING_MODE_DISPLAY
        *Name: "Color mode"
        *DefaultOption: 24bpp
        *Option: Mono
        {
                *rcNameID: =MONO_DISPLAY
                *Name: "Black and white"
                *DevNumOfPlanes: 1
                *DevBPP: 1
                *Color?: FALSE
        }
        *Option: 24bpp
        {
            *rcNameID: =24BPP_DISPLAY
            *Name: "Color"
            *DevNumOfPlanes: 1
            *DevBPP: 24
            *DrvBPP: 24
            *PaletteSize: 256

        }
}



*%*****************************************************************************
*%                           Cursor Commands
*% The following cursor commands are mandatory
*%
*% Learn more: Cursor Commands
*% http://msdn.microsoft.com/en-us/library/ff547223(VS.85).aspx
*%*****************************************************************************
*Command: CmdCR { *Cmd : "" }
*Command: CmdLF { *Cmd : "" }
*Command: CmdFF { *Cmd : "" }
```

License

Non-exclusive licence to reproduce thesis and make thesis public

I, Aleksander Parelo

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Development of a virtual printer and print driver for Print in City

supervised by Urmas Tamm and Meelis Roos

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Aleksander Parelo

14/05/2021