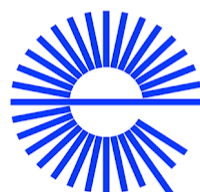




Emulation as a Service for Heritage Institutions

Test Report

Version 1.0 | October 2020



dutch digital
heritage
network





Contents

1. Introduction	4
1.2 Background	4
1.3 Preliminary Definitions	4
2. Basic Principles	9
2.2 Objects	9
2.3 Software	9
2.4 Environments	10
3. Deployment	11
3.2 Cloud Version	11
3.3 Installation notes for local version	12
4. Preparing the Environment for Use Cases	14
4.2 Regionaal Archief Alkmaar	14
4.3 Het Nieuwe Instituut	14
4.4 Beeld en Geluid	16
5. Results	18
6. Recommendations	20
7. Future Considerations	22
8. Appendix	24
9. Credits	36

1. Introduction

This report explores the concept of emulation in an archival context by examining what exactly the term means, providing an overview of a current framework for its implementation, and offering some suggestions for institutions or individuals that are looking to get started on this topic.

1.2 Background

The NDE Software Archiving project, conducted during the “intensiveringsperiode” 2019–2020, brings together research, best practices, and guidelines for Dutch heritage institutions looking to start with software preservation. Central to this research has been the exploration of possibilities for the use of emulation as part of preservation and access workflow. The ability to render and interact with outdated formats and obsolete software through emulated environments opens collections up to researchers and the public in a more engaging way. The issue of integrity, and faithfulness to the original, is key to archival practice. Given the difficulty of maintaining legacy hardware and software into the future, as new technologies continue to emerge, emulation can be seen as a worthwhile approach for sustainable access to such materials. As early as 2005, the Dutch National Archives and the National Library of the Netherlands were investigating and developing tools for emulation, as part of Dioscuri¹ project. Various other initiatives and research projects have been carried out on the topic but, with the ever advancing improvements in the technology and its implementation within current emulation projects, now is the time to investigate and test this approach in a Dutch context. This report is aimed at application managers who are interested in installing the Emulation as a Service (EaaS) framework, as well as general users who are looking to learn about creating object environments and how the framework operates.

1.3 Preliminary Definitions

Emulation

In its most basic description, emulation is the simulation of legacy computing hardware that works by allowing the execution of legacy software on contemporary computing hardware. There are several layers of complexity in relation to the way components contribute to providing access to legacy formats. For example, a file is generally dependent on a particular software package. In turn, this package will require a specific operating system in order to install and run. Lastly, this operating system is likely to be dependent on particular hardware specifications.

¹ <http://www.digitalpreservation.gov/series/edge/koninklijke.html>

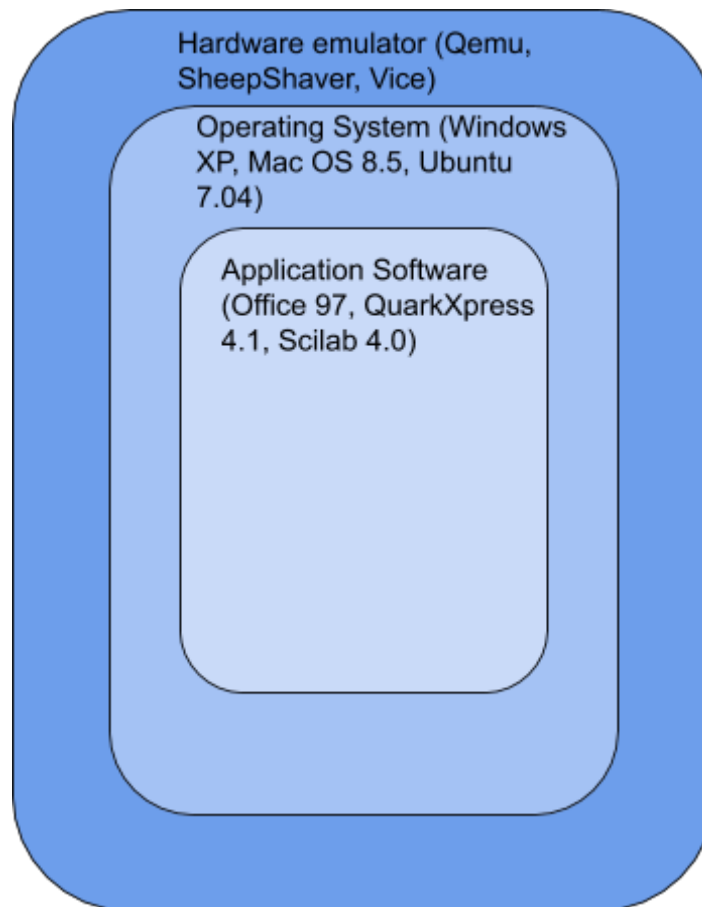


Fig. 1

The aim of emulation is to provide these hardware environments where the original hardware is no longer available. There are a host of open source emulators² available that can achieve this goal for most historical computers, as well as for arcade and console platforms. A common issue with use of emulators to date has been that they require a certain degree of technical expertise to set-up and configure.

EaaS & EaaSI

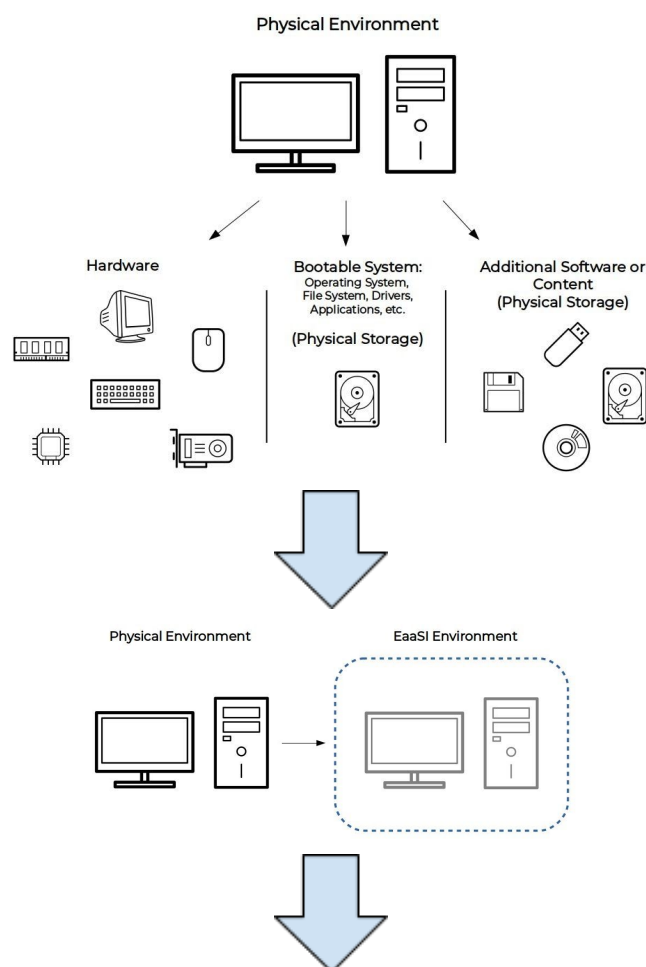
Emulation as a Service (EaaS) is a software framework that endeavours to provide long-term preservation and access to digital material through emulation. A key goal is to simplify the process and management of emulation components. The framework has been in development by the bwFLA team at the University of Freiburg since 2011 and now operates under the OpenSLX label. It makes use of abstract emulation components to standardize deployment and to hide individual emulator complexity.³ The user of the framework does not need to worry about the full technical workings of the emulator in use as the framework is designed to interact with the prepackaged, or containerised, emulators provided. These containers can be slotted in easily via the framework's front end without the need for users to understand what is going on in the background. The framework supports all major desktop systems and utilises a host of containerised, open source emulators including Qemu, Basilisk II, Sheepshaver and Vice.

² Qemu, SheepShaver, and VICE are examples of emulators that are able to render Windows, Mac, and Commodore 64 respectively

³ <http://eaas.uni-freiburg.de/eaas.html>

The Emulation as a Service Infrastructure (EaaSI) project, led by Yale University Library, has been working towards the development of technology and services to expand and scale the capabilities of the EaaS software since 2018. Much of their work has revolved around the establishment of a network of partner institutions to share the testing, research, and improvements of the framework. Among these institutions, software environments can be shared and reused where needed.

The framework uses environment configuration templates, stored in XML, to assemble the individual components that make up an emulated environment.⁴ These include the designated emulator for the session, the hardware settings required, a bootable disk image⁵ for the operating system being used, and then any additional software disk or file images that are being accessed. The way in which the recreated physical environment corresponds to the emulated environment can be seen below:



⁴ https://eaasi.gitlab.io/eaasi_user_handbook/guide/architecture.html

⁵ As well the operating system, the bootable disk image contains associated utilities and boot and recovery data. The image can be stored on memory sticks or similar devices and acts as a one stop location for installation of a particular operating system

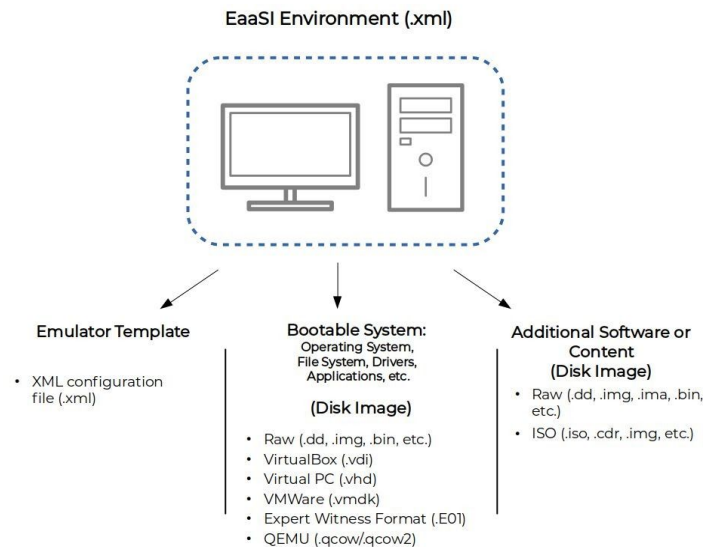


Fig. 2 System Architecture Diagram taken from the EaaS User Handbook

Methodology

For the purpose of this report, digital preservation analysts Eoin O'Donohoe and Claudia Rock of Beeld en Geluid (Sound and Vision) carried out the research, installation, testing, and evaluation of the EaaS framework. This was conducted both independently and with the assistance of project partners. The work explores the current state of the framework and its feasibility as an approach to preserving, and making accessible, obsolete software and software-dependent objects found within Dutch heritage institutions. Furthermore, the EaaS project was consulted on elements relating to user feedback and also acted as a marker for what a future network might look like. The process consisted of:

1. Familiarisation and experimentation with a preconfigured version of the framework hosted in the cloud
2. Installation of a local version on a dedicated machine to investigate the technical complexity of the framework
3. Research into the individual case studies from partner institutions and their specific needs in relation to the creation of suitable environments
4. Creating and testing environments in order to generate a feedback survey for partners

Goal

The goal of this study is to provide knowledge and recommendations on the feasibility of the EaaS framework for use within Dutch heritage institutions. Specifically, it focuses on the available use cases of the partner institutions and considers how they would operate within an emulated environment. Also, it gives advice and recommendations on the possibilities for future deployment and pricing.

Scope

As we began the research and testing process, two options for installing the EaaS framework were available to us: cloud installation and local installation. The cloud installation, which runs in Google Cloud, came preconfigured by the people at OpenSLX⁶ with several base environments, employing

⁶ The OpenSLX GmbH, founded 2006, is a University of Freiburg technology spin-off, developing long-term preservation solutions for complex digital content. <https://openslx.org/>

different emulators, and is ready to start adding software objects and data objects immediately. For the local installation, we used a dedicated Linux laptop, running Ubuntu version 18.04.4 LTS, with adequate specifications for testing purposes.⁷ We wanted to test this option to get a better idea of how the framework came together when installed from scratch. This enabled us to gain a greater understanding of the technology involved, how components of the framework interact with each other, and how disk images could be stored in a local manner.

Another installation option is the full implementation of a networked version of the framework, either internally or externally, but we did not pursue this as it would require thoughtful integration into an existing IT infrastructure and was beyond the needs of this study. We did, however, manage to experiment with the duplication of preconfigured environments made available from the EaaSI project across a standard internet connection. The primary focus of investigating such a framework is to make software and software-dependent files more easily accessible to the end users. The many considerations that take place in relation to the preservation of such material, including metadata and storage, will be addressed elsewhere in the project. Finally, the issue of licensing for obsolete software and the debate surrounding the legality of emulation is an important area that will be addressed elsewhere in the project and are therefore out of the scope of this report.⁸

⁷ https://eaasi.gitlab.io/eaasi_user_handbook/install/requirements.html

⁸ Much work has been carried out in this area in the United States in relation to their Fair Use policy. Without a similar policy in place in Europe it will likely be necessary for institutions to try to advocate for a similar, or alternative, agreement with software companies or even at an EU legislative level. Nevertheless, the Code of Best Practices In Fair Use For Software Preservation offers some interesting ideas for what might be possible in the future: <https://www.arl.org/wp-content/uploads/2018/09/2019.2.28-software-preservation-code-revised.pdf>

2. Basic Principles

The basic principles of the EaaS framework are based around the ideas of Objects, Software, and Environments. The separation of each of these elements makes the management of material simple. While environments generally make up the foundation of each emulation session, it is important to start from the Object level when understanding the process.

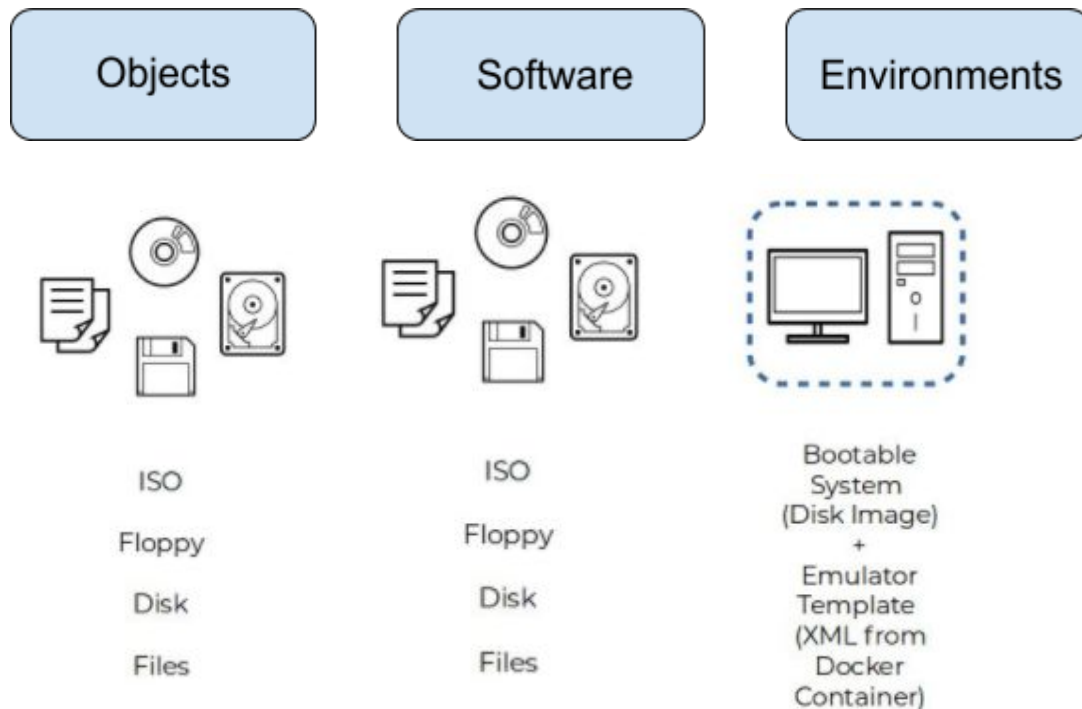


Fig. 3 Objects, Software and Environments

2.2 Objects

An object can be anything from a piece of software or game to a virtual disk or datafile. Objects form the first layer of the environment creation process and can be imported into the framework using the designated page. This could be the image of an installation CD or an archive file packaging software components or a collection of specific file formats. The user is provided with the option of adding an object to the framework, designating it to one of four categories; an ISO (optical disc image), a Floppy, a Disk, or a File. Depending on the size of the object in question, adding it may take a minute or more. Once uploaded, the object appears in the Local Object Archive with whatever title the user provided.

2.3 Software

Objects can be promoted to software in order to keep separation between software applications and other datafile objects. The environment creation process relies on multiple components, ranging from operating systems, commercial and open source software applications, device drivers, etc., in order to work. By promoting these objects to software it is possible to keep a clearer separation between the components that comprise the emulated environments and the files which are themselves the target of emulation. In this step, the user can associate additional metadata with the object, including the rendering environment, licensing information, and whether or not the object should be designated as an operating system. Software can easily be attached to an architecture, as defined by the specific emulator, and run to create new environments.

2.4 Environments

Environments are where users engage with the majority of the objects. They are ways of saving combinations of specific emulated computer hardware, and specific operating systems and software for reuse. The EaaS framework has three different types of environments: Base Environments, Virtual Machines, and Object Environments. A Base Environment is the combination of a specific architecture and a bootable operating system, e.g., x86_64 + Windows 98 Second Edition. Once created, these environments provide the foundation for other software applications. The Virtual Machines tab of the Environments section acts as a crossover point. Here, the user can find their stored base environments, but also any further derivative environments they create. A derivative environment generally consists of a Base Environment and an additional piece of software that has been installed and configured to the users' needs. It is these derivative environments that are used to access specific objects, such as file sets. Objects can be associated with a particular environment from the Objects tab, allowing the user to launch an environment directly from the chosen object. Files are wrapped as ISOs for ease of use and can be found in the emulated systems CD-ROM drive. Once the user is happy with the rendering of the objects within the environment, they can make another derivative consisting of all three of the above elements; the Base Environment, the necessary software to access the file, and the files themselves. This is known as an Object Environment and allows for quick and easy access to objects.

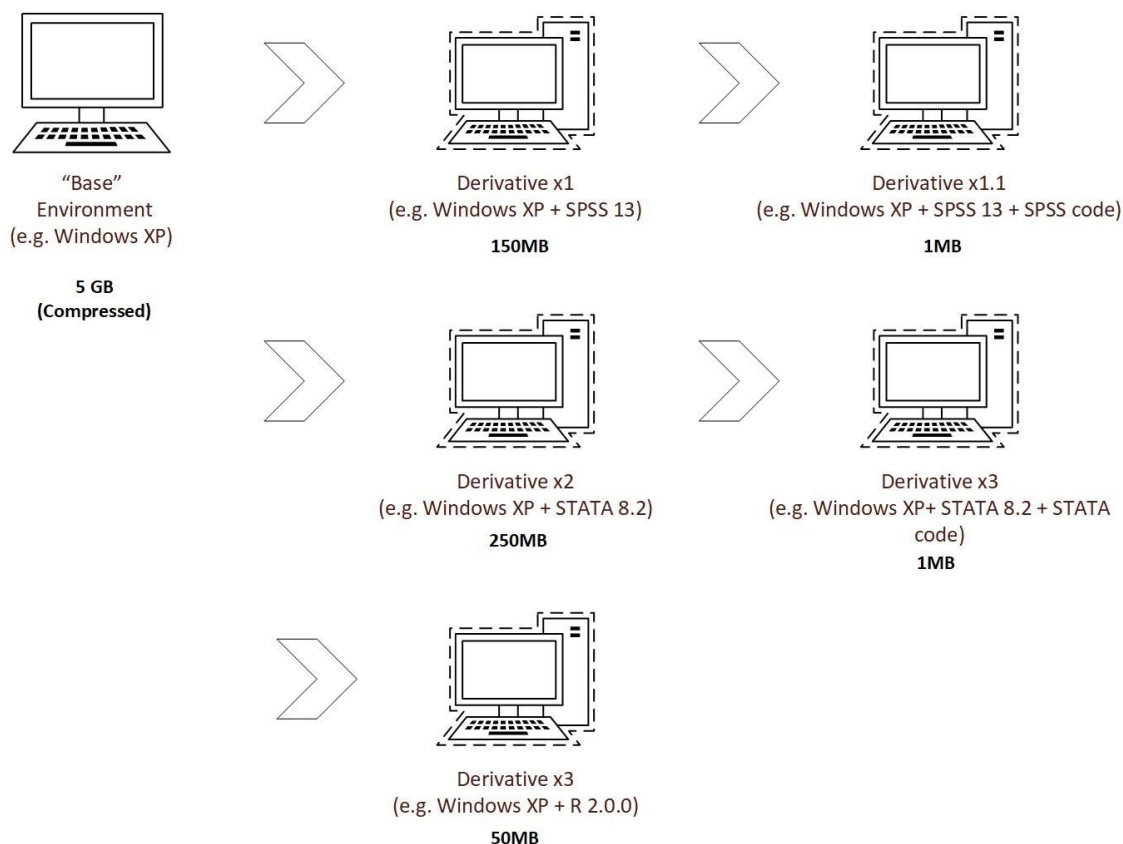


Fig. 4 Diagram of "base" environment and how derivatives can be built up.⁹

For further information relating to these concepts please consult the EaaS User handbook. To see the process in action you can visit [here](https://eaasi.gitlab.io/eaasi_user_handbook/resources/workflow.html).¹⁰

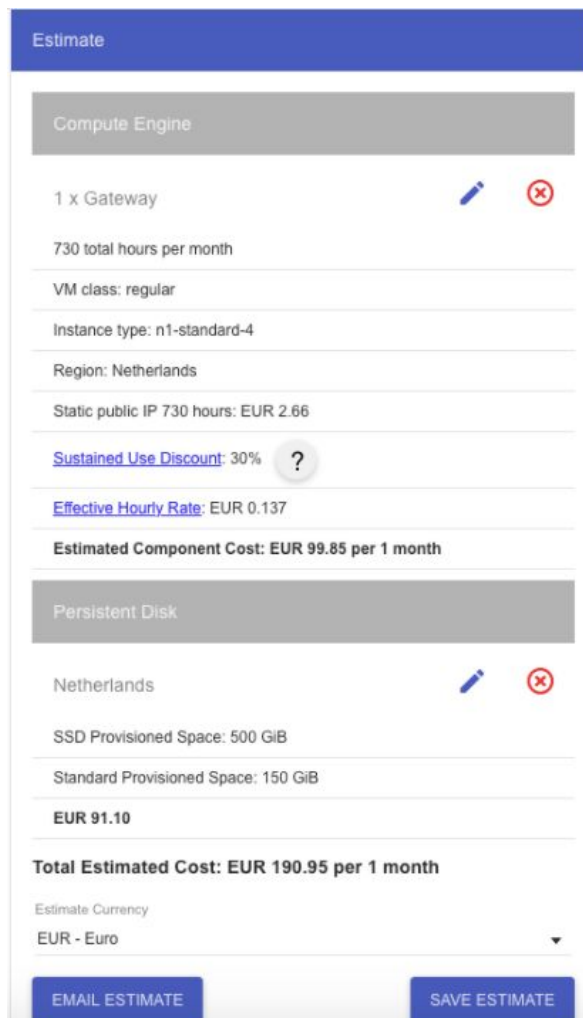
⁹ https://eaasi.gitlab.io/eaasi_user_handbook/overview/architecture.html

¹⁰ https://eaasi.gitlab.io/eaasi_user_handbook/resources/workflow.html

3. Deployment



3.2 Cloud Version

The cloud version of the framework was hosted using the Google Cloud platform and was fully preconfigured and installed by the OpenSLX team. The option to have OpenSLX take care of the more technical backend side of things allowed us to focus on the creation and management of environments, software, and objects using the front end of the framework. For the purpose of this study we have made use of a set capacity that kept the pricing fairly standard month on month. Our basic package included allowance for continuously running four virtual CPUs, allowing four simultaneous instances of the framework at the same time. This package also included 500GB of storage and 16GB of RAM. The estimated cost for this setup is projected at an effective hourly rate of €0.137 per hour according to the Google Cloud Calculator. Combined with the cost of storage, the monthly price is estimated at €190.95 before taking into account 30% sustained use discount. Our current pricing comes in at an average monthly cost of €140 per month.



Estimate

Compute Engine

1 x Gateway  


730 total hours per month

VM class: regular

Instance type: n1-standard-4

Region: Netherlands



Static public IP 730 hours: EUR 2.66

[Sustained Use Discount](#): 30% 

[Effective Hourly Rate](#): EUR 0.137

Estimated Component Cost: EUR 99.85 per 1 month

Persistent Disk

Netherlands  

SSD Provisioned Space: 500 GiB

Standard Provisioned Space: 150 GiB

EUR 91.10

Total Estimated Cost: EUR 190.95 per 1 month

Estimate Currency
EUR - Euro

[EMAIL ESTIMATE](#) [SAVE ESTIMATE](#)

Fig. 5 Google Cloud pricing calculator

Depending on scaling strategy, it is possible for additional CPUs to be added on demand when needed. This is done in set steps, in the case of the above example, steps of 4 CPUs once the next threshold is reached. This will effectively double the hourly rate while in use but will revert back to

the starting price when not in use. A sixteen-CPU setting is recommended for high traffic scenarios, where a lot of clients are expected. In this case it would scale up in sixteen CPU steps and would keep one 16 core instance running continuously. The sustained use discount is applied depending on the amount of time each CPU is running, with greater savings for increased use. This means that the user only pays for the number of minutes that they use in an instance, and the provider automatically gives the best price. There's no reason to run an instance for longer than needed.¹¹

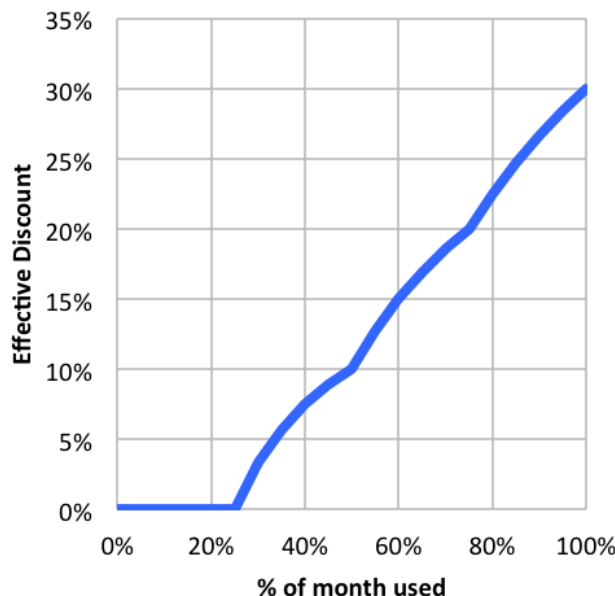


Fig. 6 Taken from Google Cloud documentation.

3.3 Installation notes for local version¹²

Although the majority of the testing of use cases was carried out using the cloud version, we decided that it would be beneficial to attempt our own installation of the framework on a local machine. This process is documented in several sources in both the EaaS and EaaSI user handbooks, but these instructions do not fully account for installing on one single machine. Through discussions with the OpenSLX team, a new local installer was created for testing purposes.

Installing a standalone version of the framework required some prerequisites before getting started. In order to simplify deployment of EaaSI, all components have been containerised as Docker instances. This acts as a one stop location where all resources for installation can be gathered by the installer. In order to access these Docker instances, the intended machine must have Docker installed. Docker allows developers and users to package and run applications in a loosely isolated environment called a container.¹³ A project's Docker Registry contains a collection of images that can be combined to run a specific application. In the case of EaaSI, the installation templates, as well as the individual emulators, exist as images in the EaaSI registry.

¹¹ <https://cloud.google.com/compute/docs/sustained-use-discounts>

¹² These notes should be viewed as a case study in their own right. As the EaaSI project is continuously evolving it means that these instructions are likely to change with time. As of March 2020 a new version of EaaSI has been released which has integrated the single machine test version. Full documentation on this can be found here: https://eaasi.gitlab.io/eaasi_user_handbook/overview/demo.html

¹³ <https://docs.docker.com/get-started/overview/>

The following steps detail the installation of an EaaSI instance on a single machine. A supported Linux operating system should be installed on the target machine. Currently Ubuntu 16.04, Ubuntu 18.04, and CentOS 7 distributions are supported. At least 10 GB of free disk space are needed for a minimal EaaSI installation. Additional disk space is required to run emulators and store disk images. Through trial and error, as well as discussions with the developers, it was discovered that a simplified, test instance of the framework would be beneficial for institutions looking to get a sense of its capabilities. The installation process makes use of an automatic installer but requires some basic interaction with the command line. The general steps include:

1. Clone the EaaSI-Installer from the Gitlab repository¹⁴ to a location on the target machine.
2. Via the command line, navigate to the EaaSI-Installer folder and use the following command to prepare the machine for installation:

```
./scripts/prepare.sh --local-mode
```

3. Define the target machine simply by copying a file template. Use the following command:

```
cp ./config/localhost.yaml.template ./artifacts/config/hosts.yaml
```

4. EaaSI deployment configuration must be defined in the eaasi.yaml file. This again can be copied from an existing template using the following command:

```
cp ./config/eaasi.yaml.template ./artifacts/config/eaasi.yaml
```

5. When everything is configured correctly, the installation process can be started by running:

```
./scripts/deploy.sh
```

Once installed, the EaaSI instance can be accessed from the local host in a web browser.

¹⁴ <https://gitlab.com/eaasi/eaasi-installer/-/tree/master>

4. Preparing the Environment for Use Cases

For the purpose of testing our use cases we worked in the cloud version of the framework. This ensured that everything was configured correctly from the outset and allowed us to receive rapid support from the OpenSLX team if required. We also wanted to receive testing feedback from the project partners, allowing them to access the cloud instance via a dedicated link.

4.2 Regionaal Archief Alkmaar

The Regional Archive Alkmaar has two video game collections: the company archive of game developer M2H and the archive of Joost Honig, hacker and game developer in the 1980s and founder of the 1001 Crew. Both collections have been made available via the institution's website as a download, but it is the user's responsibility to install and configure a suitable emulator or environment in order to play these games. This creates a barrier for less technical users and leads to potentially inaccurate or inconsistent results. For our test we focused on the Commodore 64 cassette files. These vary from playable games to interactive demos and intros and cover several different file extensions, including .d64, .t64, .prg, and .tap.

The first step was to identify the necessary emulator for these particular files. The Vice¹⁵ emulator, which is provided within the EaaSI framework, is intended to run old 8-bit computers, including the Commodore 64, and does not require any additional operating system or software. The user manual of the Vice emulator covered all of the example file formats we wished to work with, so this proved to be the perfect starting point. The Vice emulator settings can be adjusted to optimise video and sound quality, as well as the frame rate and mapping of keyboard controls. The various test files we selected were saved as derivative environments of a base Commodore 64 environment, allowing us to start an emulated session directly to the specific game with all the correct settings.

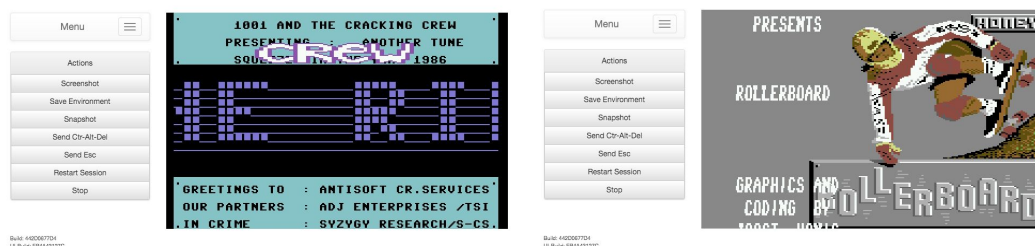


Fig. 7 Example files from Regionaal Archief Alkmaar

The files we chose for testing purposes covered as broad a spectrum as we could find. We used at least one of each file extension. The test cases ranged from basic visual animations to intros featuring text, animation, and audio. In some cases there was an interactive element with keyboard commands, and in one particular case we tested a fully controllable game.

4.3 Het Nieuwe Instituut

The collection of Het Nieuwe Instituut (HNI) contains drawings, photographs, and models from the archives of Dutch architects and urban planners. Many items are born digital, covering a range of file formats. The format provided by Het Nieuwe Institute as a use case was the QuarkXpress Data File.

¹⁵ <https://vice-emu.sourceforge.io/>

Name	Date Modified	Size	Kind
report.html	22 Oct 2019 at 11:07	7 KB	HTML
CD Front Fase 08 BA fase 02 DD 30.03.2004.qxd	4 Oct 2006 at 09:52	1.1 MB	Document
TP176_aq_nl.qxd	5 Sep 2005 at 15:21	4.5 MB	Document
las vegas.qxd	3 Feb 2004 at 13:40	2.8 MB	Document
booklet1.qxd	31 Jan 2003 at 13:35	15.8 MB	Document
HUNTING3.qxd	26 Mar 2002 at 16:18	20.6 MB	Document
COVER.qxd	24 Jan 2001 at 11:52	141 KB	Document
ACQ-010-VPRO.qxd	5 Jan 2000 at 18:44	2.6 MB	Document
story.qxd	5 Mar 1999 at 00:07	9.3 MB	Document
SLOT.QXD	31 Mar 1998 at 13:45	1.8 MB	Document

Fig. 8 Batch of test files

The batch of files we received from HNI consisted of thirty-nine .qxd files (QuarkXPress Data File), dating from the year 1998 to 2005. Our first impression was that the .qxd format would require additional investigation and research in order to determine the specific version of files we were dealing with, information that would be vital to set up the necessary environment in which to access these files. Our initial searches led us to the understanding that the .qxd extension was discontinued after QuarkXPress version 5, released in 2002. This gave us a rough idea of what we were working with, but with only the year of creation to go on, we could not be sure of the exact version of QuarkXPress used to make the files. The files were examined with the National Archive's DROID File Profiling Tool,¹⁶ which makes use of the associated PRONOM¹⁷ file format registry. Unfortunately, the only result available was a generic placeholder entry that confirmed the file format was QuarkXpress, but no further information relating to the specific version was available. Luckily, the discovery of a trial version of FlightCheck¹⁸ software allowed us to be more precise. FlightCheck is a compliance and quality assessment tool for print publishing materials and covers a broad range of desktop publishing files and formats. Simply running the files through this software produced a small report with important metadata, including the file version (most, it turned out, were version 4 and some version 3.33). This research was a necessary part of our work and highlighted the gap in file-profiling tools available today. The importance of knowing what is in your collection, having good metadata, and contributing knowledge to the wider archival community cannot be overstated. Our findings were included in the PRONOM research week of November 2019 and, with the work of many other organisations, there are now unique entries for several QuarkXpress versions within the PRONOM registry.

1	Filename	Version (show in FlightCheck)
3	booklet1	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 80044180 00000000
5	COVER	4.1 00004949 58505233 41204120 44430000 00000000 1D3357CB 9C016787 08090A0A
6	HUNTING3	4.1 00004949 58505233 41004100 44430000 00000000 1D3295C8 80044180 00000000
7	las vegas	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 01044180 00000000
12	085_belichtingboek_01	4.1 00004949 58505233 41004100 44430000 00000000 1D3251C8 80044180 00000000
13	085_ruimteboek_01	4.1 00004949 58505233 41004100 44430000 00000000 1D32D5C8 80044180 00000000
16	Ateliers_A5	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 89044140 00000001
17	ateliers-vierkant	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 81044140 00000000
19	BARCELON	4.1 00004949 58505233 41004100 44430000 00000000 5D3255E0 88044180 00000001
22	book-final	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 88044180 00000001
23	booklet1	4.1 00004949 58505233 41004100 44430000 00000000 1D3215C8 80044180 00000000
27	cover	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 88044180 00000001
31	HNY 2	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 88044180 00000001
32	HUNTING3	4.1 00004949 58505233 41004100 44430000 00000000 1D3295C8 80044180 00000000
34	MVRDVmaps	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 00044180 00000000
38	potsdamboek01	4.1 00004949 58505233 41004100 44430000 00000000 1D3255C8 80044180 00000000

Fig. 9 File identification research for PRONOM

¹⁶

<https://www.nationalarchives.gov.uk/information-management/manage-information/policy-process/digital-continuity/file-profiling-tool-droid/#:~:text=DROID%20stands%20for%20Digital%20Record,wide%20range%20of%20file%20formats.>

¹⁷ <https://www.nationalarchives.gov.uk/PRONOM/Default.aspx>

¹⁸ <https://markzware.com/products/flightcheck/>

This new information gave us the platform to start working on the necessary environment for the files. We acquired a copy of version 4 of QuarkXpress for MacOS and also the necessary operating system, MacOS 8.5. It was important to use a contemporary version of the software, as close to the version in which the original document was created, in order to achieve the highest level of integrity. The emulator selected was SheepShaver,¹⁹ which is optimised for classic MacOS applications. First, the base environment needed to be configured. This operating system disk image was attached to the relevant architecture as provided by the emulator. From here, the system could be booted and configured. The saved environment offered a platform on which we could install our additional software, just as with any modern computer system. The virtual disk image of QuarkXpress version 4 was loaded into the base environment, the setup file was deployed, and the software was successfully installed onto the emulated system. This could then be saved as a new environment in its own right, with the user simply updating the new derivative's label to something such as "MacOS 8.5 + QuarkXpress 4.1". As is the layered nature of the EaaS framework, this new environment was ready to accept and process our QuarkXpress Data Files.



Fig. 10 QuarkXpress 4.1 installed in framework

4.4 Beeld en Geluid

The Beeld en Geluid collection contains a broad spectrum of Dutch audiovisual and new media objects. In terms of software, the archive holds a diverse collection of computer games, including Commodore 64 and PC titles. The prospect of utilising the EaaS framework for something more graphically intensive than 8 bit games, as a way to test the emulator performance, led us to selecting a number of PC games from the 1990s.

In a similar manner to the creation of the QuarkXpress environment, it was necessary to first create a suitable base environment onto which the game media could be installed. The emulator we used for this test case was Qemu,²⁰ which supports a multitude of computer architectures, including legacy PC options. In the environment creation stage of the EaaS framework, the user is presented with various options for computer architectures. These are based on the emulators installed. For our purposes, we tested both the '90s PC and 2000s PC systems. It is these foundations upon which we

¹⁹ <https://sheepshaver.cebix.net/>

²⁰ <https://www.qemu.org/>

were again able to boot up the relevant operating systems, installing and saving versions of Windows 95, Windows 1998, Windows 2000, and WindowsXP. Matching the year of release of each game to a corresponding environment allowed us to install the content from a virtual disk image and, again, create a new derivative of the operating system plus game. Similarly to the Commodore 64 examples, the internal settings of each game needed to be adjusted in order to get the best performance, particularly with the game controls and display aspect ratio.



Fig. 11 *Grachten Racer*, 2000.

5. Results

For each of the above use cases a feedback workflow and survey was designed. The purpose of this test is to evaluate and gain feedback on the key operations of using the framework, but also the quality and accuracy of the emulated file. For ease of access the test made use of the cloud instance of the framework with the necessary base environments included.

The four areas of functionality the survey assessed were:

1. Importing a new object
2. Running an instance with object attached
3. Saving this instance as a new object environment for easy access in future
4. Evaluating the performance and integrity of the test files

These areas of interaction provide an opportunity to get a good sense of the look and feel of the framework and also to see the level of quality of emulation. Feedback on the first three points was generally positive, with users noting that the framework navigation and controls were easy to use. Two testers highlighted that the user interface labels differed from those noted in the workflow provided. This was discovered to be an issue with the framework not having translation capabilities when running in a Dutch version of Google Chrome.

The results of the performance and integrity of the emulated files varied upon the intensity of the session. Noticeably, the more graphically intensive files suffered from more lag when rendering. For both the Regionaal Archief Alkmaar and Beeld en Geluid use cases, the areas of evaluation fell under the headings of Graphics, Audio, and Interactivity. For Het Nieuwe Instituut's QuarkXpress use case, the survey focussed on general interactivity and processes. Each test criteria was evaluated using a rating system of "Poor/ does not work", "Moderate/ works partly", "Accurate/ works correctly". These ratings are mostly subjective to the testers from each institution, based on their in-depth knowledge of the content. The results below show the feedback for each case study:

Regionaal Archief Alkmaar (Commodore 64 files)

Test File	Graphics		Audio				Interactivity	
	Image Quality (Colour/Detail)	Framerate/Speed	Music Quality	Music Sync	Sound Effects Quality	Sound Effects Sync	Control Responsiveness	Overall Look/Feel
Another_Tune.t64	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	N/A	accurate / works correctly
Warrior - 1001 Crew.PRG	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	N/A	moderate / works partly
1001 Crew - The Final Edge.d64	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	moderate / works partly
1001_Joke.t64	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly
Rollerboard.d64	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly

Fig. 12 Regionaal Archief Alkmaar Feedback

Almost all Commodore files tested displayed an accurate level of rendering. In a couple of cases the image appeared to be slightly cropped with some minor information towards the edges being clipped. As this isn't the case for all files, it is possible that these examples may require further adjustments within the VICE emulator settings.

Het Nieuwe Instituut (QuarkXpress Files)

Test File					
	Control Reponsiveness	Opening file	Editing file	Saving file (to Desktop)	Overall Look/Feel
Booklet1	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly
HUNTING3	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly
AteliersA5	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly
Bookje06012001	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly	accurate / works correctly
Flight Forum xp4_#1	accurate / works correctly	moderate / works partly	accurate / works correctly	accurate / works correctly	accurate / works correctly

Fig. 13 Het Nieuwe Instituut Feedback

The overall accuracy of the QuarkXpress use case was quite positive. The operating system features and software ran smoothly and the basic interactions with the files worked well. Some minor issues were encountered that may require further research, including the rendering of specific fonts and language settings. The version of QuarkXpress available for the test was an English-language version, while the files in question contained Dutch. A warning that the font may not be rendered correctly was displayed upon opening several files. To date, a Dutch language pack or Dutch version of this software has not been found, but one may be available either from the developers or from private publishing houses that previously used these packages. A final comment on an issue of missing link to images within a particular file; it was noted that this was not an issue with the emulation environment but instead was due the individual images not being archived along with the document in the same package.

Beeld en Geluid (PC Games)

Test File	Graphics		Audio				Interactivity	
	Image Quality (Colour/Detail)	Framerate/Speed	Music Quality	Music Sync	Sound Effects Quality	Sound Effects Sync	Control Reponsiveness	Overall Look/Feel
Grachten Racer	accurate / works correctly	poor / does not work	accurate / works correctly	moderate / works partly	accurate / works correctly	moderate / works partly	moderate / works partly	poor / does not work
Redcat Basisschool	accurate / works correctly	poor / does not work	accurate / works correctly	moderate / works partly	accurate / works correctly	moderate / works partly	poor / does not work	moderate / works partly
A2 Racer	moderate / works partly	moderate / works partly	moderate / works partly	poor / does not work	accurate / works correctly	poor / does not work	poor / does not work	poor / does not work

Fig. 14 Beeld en Geluid feedback

The PC games testing displayed the most issues due to the added graphical requirements. As can be seen in the feedback, less graphically challenging software performs better in the framework whereas the PC games show considerable lag, which affects the overall look and feel. It was suggested that this may be dependent on internet speed when using the cloud instance, but there was no significant improvement when using a local version. It is important to understand that emulators continue to develop and improve and that, even as a reference point for otherwise inaccessible material, they offer a valuable option for heritage institutions.

Overall, the testing carried out above demonstrates that emulation through the EaaSI framework is a viable option, although there are varied results in terms of quality from one use case to another due to the differences in computing resources required. The vast array of collections, and workflows, within heritage institutions is sure to present both opportunities and challenges going forward, but the ongoing development and support of the software by OpenSLX, and the extensive implementation work carried out by the team at Yale University Library, provides a valuable launchpad for getting started.

6. Recommendations

Where emulation is seen as, or hoped to be, a viable option for access, there are certain steps to take. Overall, based on the test cases and investigations to date, several scenarios for use of the framework have emerged. Though the testing was conducted on a relatively small scale, the exploration of different versions of deployment opens up new possibilities for institutions to make their digital material available to their end user, whether that is internal staff, researchers, or members of the public. While licensing considerations for obsolete software needs to be further researched and advocated for, the two distinct options of cloud deployment and local deployment offers institutions different ways to integrate the framework into their infrastructure. This could resemble traditional onsite reading room access at dedicated terminals whereby visitors to the institution could access preconfigured environments for specific collections. It could also have a wider reach online with the cloud deployment option, offering a password protected platform for users to engage with content. The particular route taken will be influenced both by the size of the collection to be made available as well as the size and frequency of access requests. As mentioned above, the choice between local and cloud deployment and storage, primarily comes down to an institution's plan for access and who the end users will be. A further consideration is the current infrastructure of the institution. Is local storage and terminal access currently available or will they need to be added? Does the quantity of material being made accessible warrant purchasing cloud storage?

It is advised that, as a starting point, users identify a particular collection that would suit an emulation pilot project at their institution. This will require staff to conduct research into a sample set of files they hope to use, including identification and sourcing of the components required to create environments. The intensity of the computing resources or graphical requirements needed to interact with the sample files should be considered and tested for suitability before committing to a larger file set. Once identified, the questions for these institutions should then be:

1. Consider the requirements of end users and the specific needs of each type.
 - Who will be the likely designated audience for the collection and what are their preferences for access? This question can be answered with capacity building surveys and questionnaires that gain feedback from particular groups.
2. Examine the existing access practices of the institution and how a software-dependent collection could be incorporated.
 - How does your institution currently provide access to its collection? Is there a high visitor rate in person or is there a strong online presence? Identify what ways you provide access well already and explore how this can be tailored to the new collection.
3. Consider the collection's size, access frequency, and potential need to scale up in the future.
 - Starting small, and getting the framework setup and running properly, will allow for further expansion as needed. Again, designated audience feedback can help provide insights into demand for access.
4. Consider the current IT infrastructure and who would be responsible for the framework.
 - As a framework that is under ongoing development it is likely that a line of communication will be necessary between institution and developers. Decisions around deployment and access may be of interest to an institution's IT department, particularly with regard to deploying on a network or for online access.

5. Consider and budget for a small scale trial for a defined period.
 - Organising and managing a small test case can be the best way to get a sense of the possibilities of the framework. Work out the minimum hardware/storage/cloud resources needed.

These questions should help with the process of thinking through the initial steps of how a collection might benefit from an emulation pilot.

7. Future Considerations

The examination of the use cases in this report has focussed primarily on the viability of working with such a framework from a technical and functional perspective. The wider concept of emulation itself, however, requires a number of other questions to be answered.

First, as with any collection, documentation of software and environments is a vital consideration. The difficulty in terms of describing such objects stems from their complexity. Often there will be several layers of dependencies that contribute to make up a particular environment, with seemingly endless variables that can drastically alter integrity. There is no concrete schema for documenting software, but several platforms are available to facilitate documentation in a linked manner. Wikidata, the preferred choice of the EaaS project, as well as The National Archive's PRONOM registry, already include descriptions of common obsolete software packages and files. These initiatives make use of UIDs and common vocabulary to describe entries and are largely dependent on crowdsourcing. Both resources were investigated during the course of this research and further, active participation in adding to such registries is seen as a valuable and insightful practice. Questions for further research are: Which existing schemas can provide crosswalk potential for the purposes of documenting software collections? What work has already been carried out in this area? What resources can aid this process (eg. Wikidata, PRONOM)?

Second, the legal challenges surrounding the emulation and access to licenced software material are extremely troublesome. Most commercial software, including operating systems, was originally sold under licence from a specific company. Each licence would specify how the product could be used, usually limiting its availability to a single user or machine. Even as software becomes obsolete, and no longer in common use, these licences dictate how and when a piece of software can be used. The ever-evolving nature of the field of technology presents heritage institutions with a real dilemma when it comes to preserving software objects. The urgency for their protection is not helped by current European copyright laws. This is a potential stumbling block for access to collections and an ever-increasing concern for heritage institutions as they continue to acquire technology-based cultural material. Questions that need to be addressed include: What is the viability of a fair use model²¹, such as exists in the US? How this may be advocated for within the Netherlands? Who are the key decision makers? An alternative, and perhaps short term, approach would be to see about approaching individual companies regarding licensing and use within an institutional context. This is something that might be worthwhile for something as broadly applicable as operating systems.

Third, there is the question of how to progress with all of this work in a collaborative and constructive manner. Engaging in these conversations in wider group settings can highlight potential crossover areas in research for partner institutions. Is it possible to further explore these topics with a unified approach between Dutch heritage institutions? What might this look like? The template of the EaaS project is a good example of institutions pulling together in this manner and a lot of the groundwork has already been carried out, including conducting surveys on some of the 'getting started' questions. Sharing environments between institutions is possible using OAI-PMH Synchronisation, which is built into the framework, allowing members of the EaaS network to benefit from the collective work of all member institutions. This is something that could be mirrored in the Netherlands, allowing institutions to gain a greater understanding of not only their own software-dependent collections but the broader area of research and innovation.

We hope the scenarios presented in this paper provide a better overview of how an emulation framework such as EaaS might be used to access obsolete software and software-dependent data

²¹ <https://www.copyright.gov/fair-use/more-info.html>

in a heritage context. Furthermore, we hope that the questions posed around the wider area of research into software collections' management act as a starting point for greater discussions, collaboration, and action going forward.

8. Appendix

Workflow Regionaal Archief Alkmaar

Testing of Commodore 64 emulation using EaaSI framework

The testing of specific Commodore 64 files will use a cloud instance of the EaaSI framework for ease of access. The EaaSI cloud instance includes a preconfigured Commodore 64 environment which utilises the VICE emulator. This is ready to load d64, t64, PRG, etc. files directly without the need for an additional operating system layer. For an idea of how EaaSI links hardware templates with operating systems to create base environments please see this example [workflow²²](#). This example shows the core principles of creating an environment, adding software and object files, and how these interact with each other. Below is a more specific workflow developed for the needs of this survey.

The 4 areas of functionality we wish to assess are:

1. Importing a new Object
2. Running an instance of the Commodore 64 environment with Object attached
3. Saving this instance as a new Object Environment for easy access in future
4. Evaluating the performance and integrity of the test files

The suggested test cases are listed below but we welcome feedback on additional files.

Test files

Another_Tune.t64

Warrior - 1001 Crew.PRG

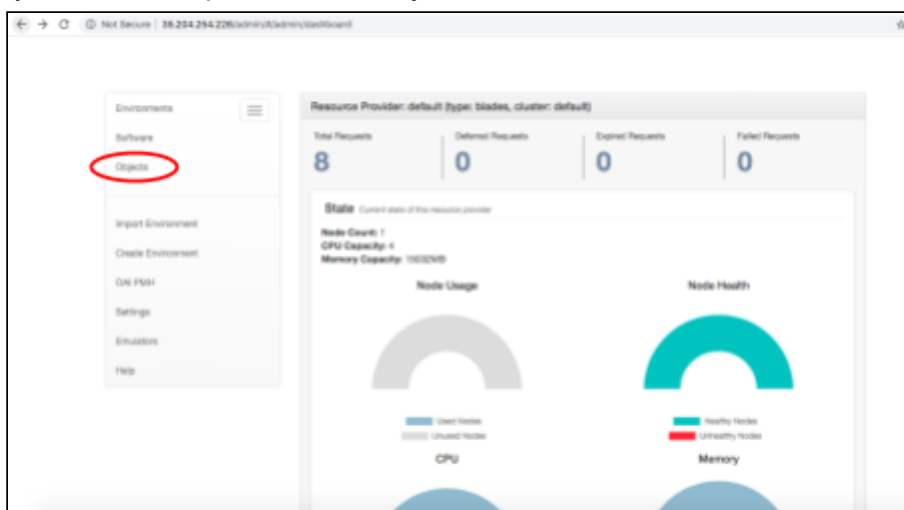
1001 Crew - The Final Edge.d64

1001_Joke.t64

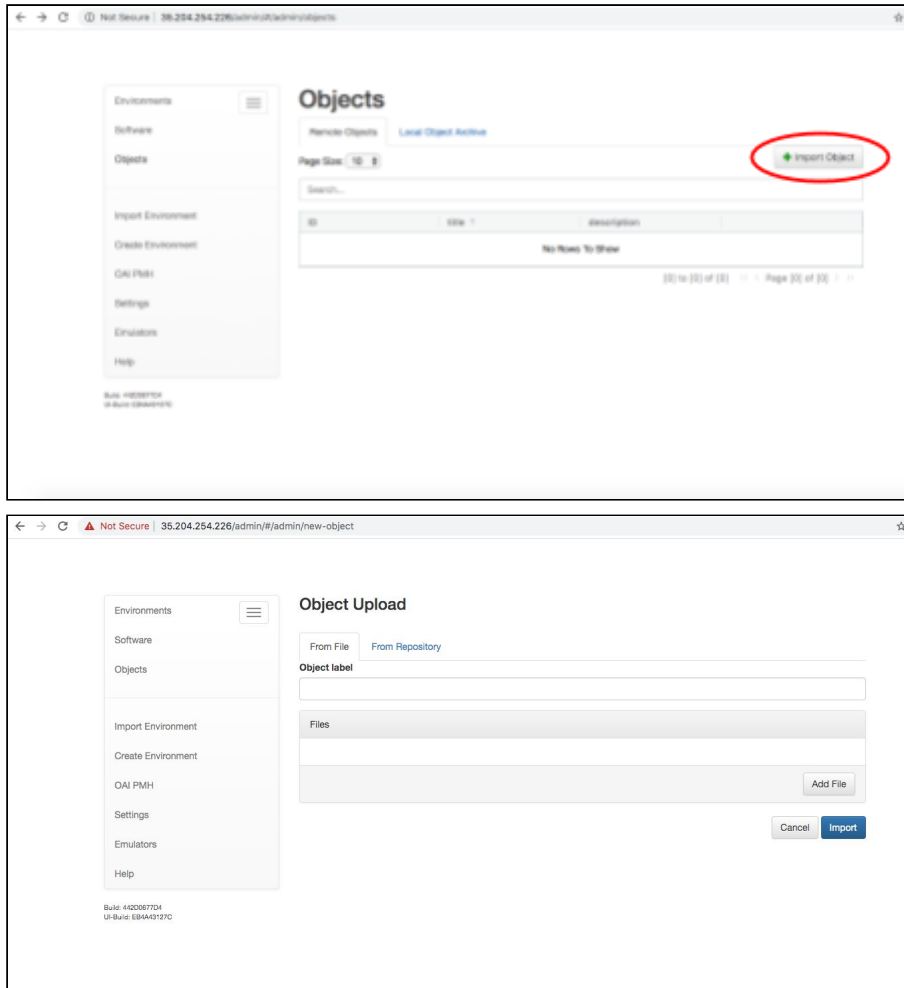
Rollerboard.d64

Workflow

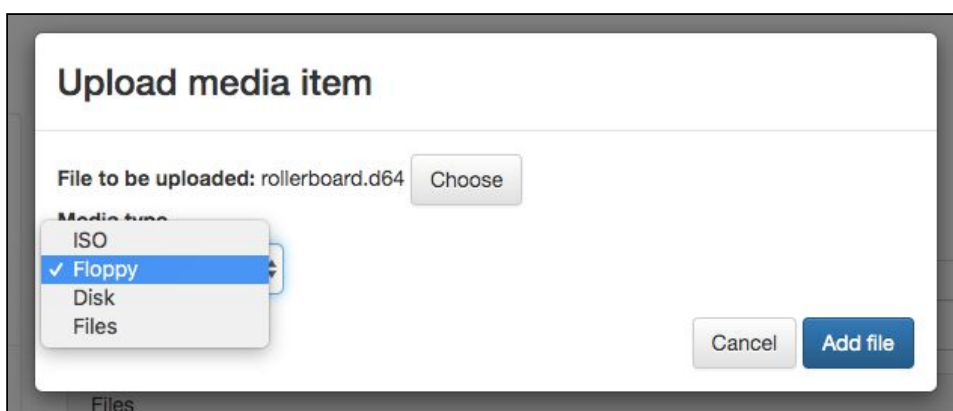
1. The first step is to import a new object. From the main dashboard, navigate to the “Objects” tab. To add a new object, use the “Import Object” button. All new objects imported from a local file system will end up in the “Local Object Archive”.



²² https://eaasi.gitlab.io/eaasi_user_handbook/resources/workflow.html



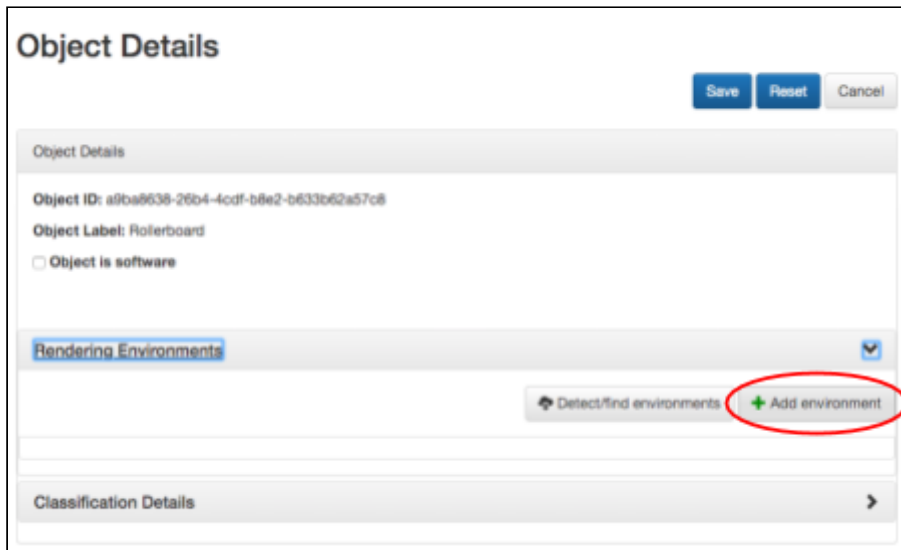
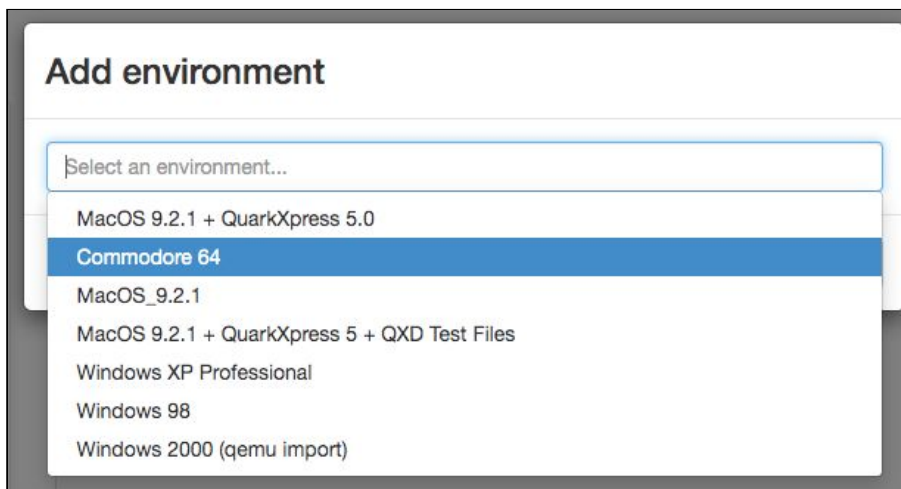
From the “Object Upload” page you can add your file of choice. The “Add File” button opens a dialogue box that prompts the user for input. Select the desired file and then choose the media type from the drop-down menu. The media type for Commodore 64 files is “Floppy.”



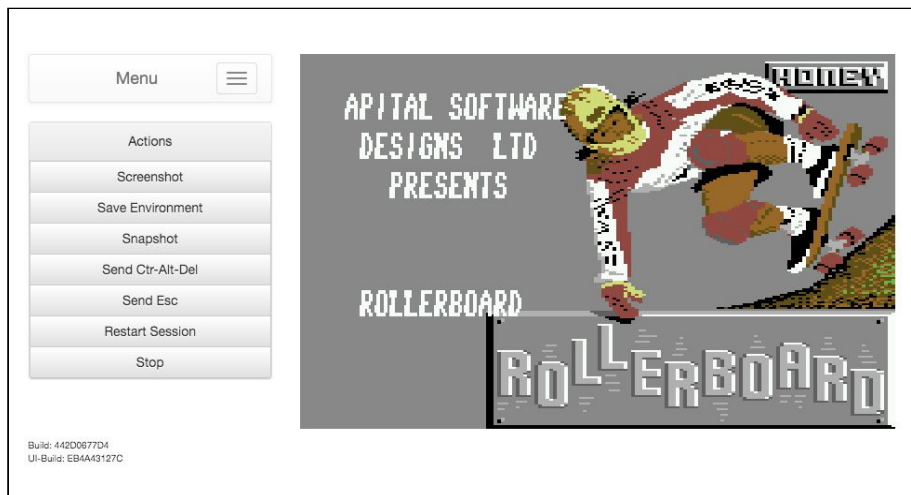
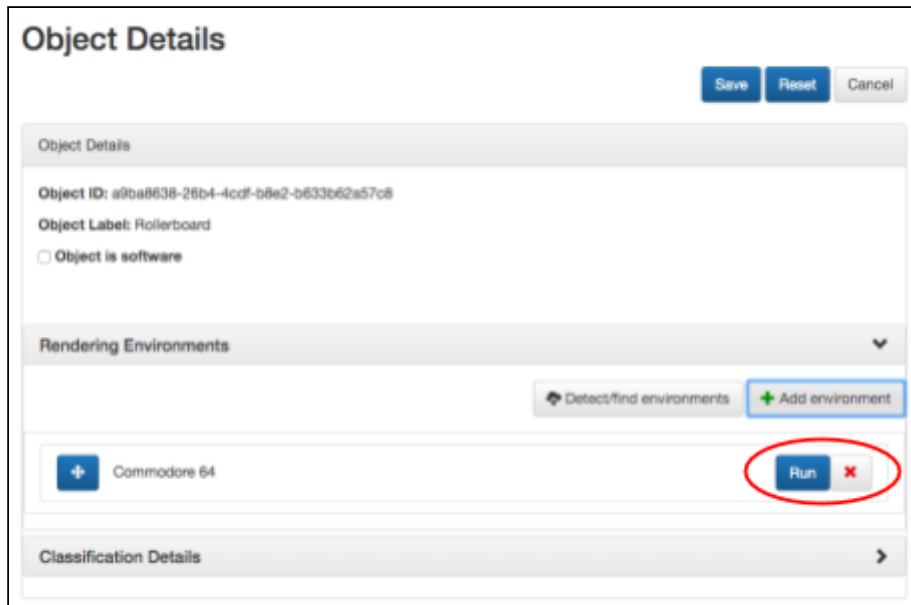
Back at the “Object Upload” page, add an Object Label for the file and click the “Import” button. You will see the system working on the import and, if successful, be returned to the “Objects” page. There should also be a green success notification in the top right corner of the screen. To double check that the import worked, click into the Local Object Archive to see if the new object is listed.

2. Now that you have a new Commodore 64 object imported, you can test it out in an emulation instance. Starting from the Local Object Archive, you can assign specific rendering environments to your objects. Locate your object in the list and select from the “Choose action” drop-down menu “Choose action → Details”

The Object Details page has a drop-down section for Rendering Environments. If you know which environment you want to use for these particular objects you can use the “Add Environment” button to select the correct one, in this case the Commodore 64 environment.

Adding the Commodore 64 environment presents the user with the option to run the object in an emulation instance directly from the Object Details page. The “Run” button will prepare a new emulation session. From here the user can interact with and assess the object.



Please note that out of all the test cases the Rollerboard example appears to be the only one that doesn't start with the standard controls configured. These, along with other VICE settings, can be adjusted within the emulation session using the F12 key to open the menu. This is probably useful to explore and figure out yourself but please contact us if it proves troublesome.

3. When you are happy with the look and feel of the emulated environment you can save it for future use. This saves the trouble of having to reassign rendering environments each time you launch a new session.

To the left of the emulated screen is a menu. You can use the "Save Environment" button to create a new Object Environment that links the Commodore 64 environment with the object in question. This action will prompt you to make sure "to shutdown the guest operating system before creating a snapshot". For the Commodore environments you can ignore this and choose to continue. A dialogue box allows you to add some descriptive information about the new environment, and when saved the user is returned to the Environments page.

Save changes

Name

Commodore 64 + Rollerboard

Description

H1H2H3H4H5H6Ppre”

BBIUS≡≡CC∅

≡≡≡≡≡≡

<>🖼️🔗📺Words: 3Characters: 22

Rollerbaord game added

Save

Cancel

The new environment can be located in the “Object Environments” tab and can easily be launched via the Run Environment button in the Actions drop-down menu.

Environments

[Virtual machines](#)
[Object Environments](#)

Number of Environments: 2

Page Size: 10

	Name ↑	ID	Own...	ObjectID	Actions
<input type="checkbox"/>	Commodore 64 + Rollerboard	ab353...	shared	a9ba8638-26b4-4...	Choose action ▾
<input type="checkbox"/>	MacOS 9.2.1 + QuarkXpress 5 + QXD Test Fil...	f5768...	shared	bb9a2524-1b5e-4e...	Run Environment Details Delete Add Software

[1] to [2] of [2] < >

Workflow Het Nieuwe Instituut

Testing of QuarkXpress emulation using EaaS framework

The testing of specific QuarkXpress files will use a cloud instance of the EaaSI framework for ease of access. The EaaSI cloud instance includes a preconfigured Quarkxpress environment which utilises the Sheepshaver emulator in combination with images of MacOS 8.5 and QuarkXpress 4.1. This environment is ready to add QXD files for use with the QuarkXpress software. For an idea of how EaaSI links hardware templates with operating systems to create base environments, please see this example [workflow](#)²³. This example shows the core principles of creating an environment, adding software and object files, and how these interact with each other. Below is a more specific workflow developed for the needs of this survey.

²³ https://eaasi.gitlab.io/eaasi_user_handbook/resources/workflow.html

The 4 areas of functionality we wish to assess are:

1. Importing a new Object
2. Running an instance of the Commodore 64 environment with Object attached
3. Saving this instance as a new Object Environment for easy access in future
4. Evaluating the basic operations and integrity of the test files

The suggested test cases are listed below but we welcome feedback on additional files. Please note that these files are a mix of Windows native files and Mac native files. The cross compatibility of the QuarkXpress software allows for interaction with the files, but the limited availability of software images means that the test environment will be using a Mac version.

Test files

Booklet1

HUNTING3

AteliersA5

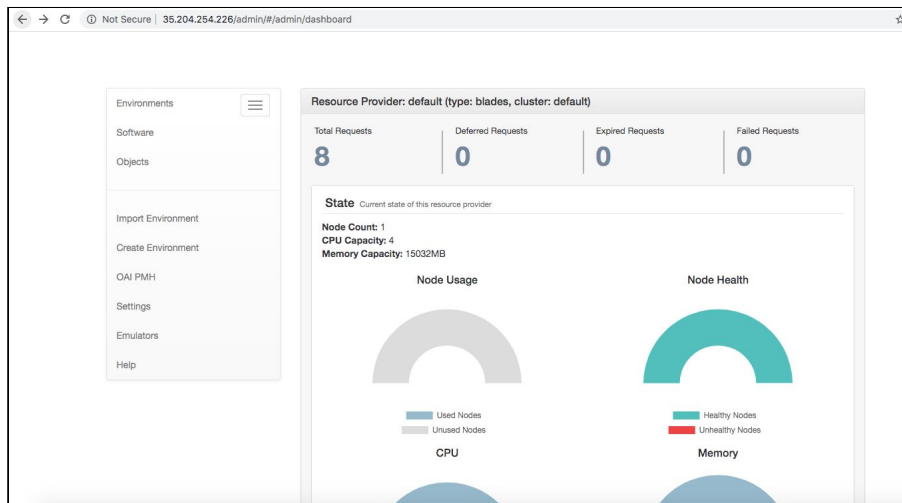
Bookje06012001

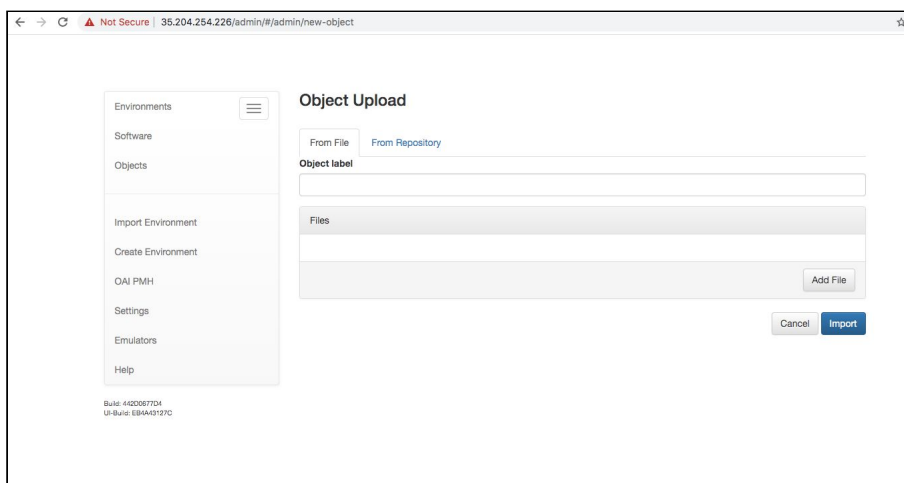
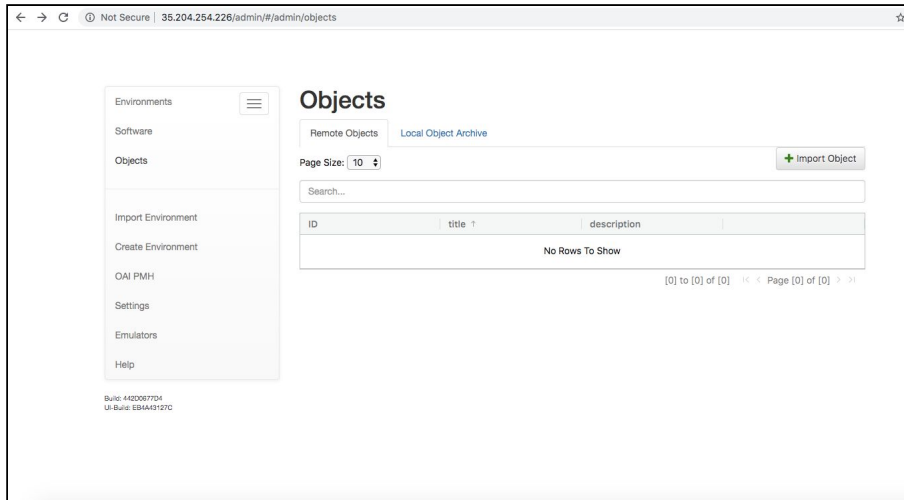
Flight Forum xp4_#1

Workflow

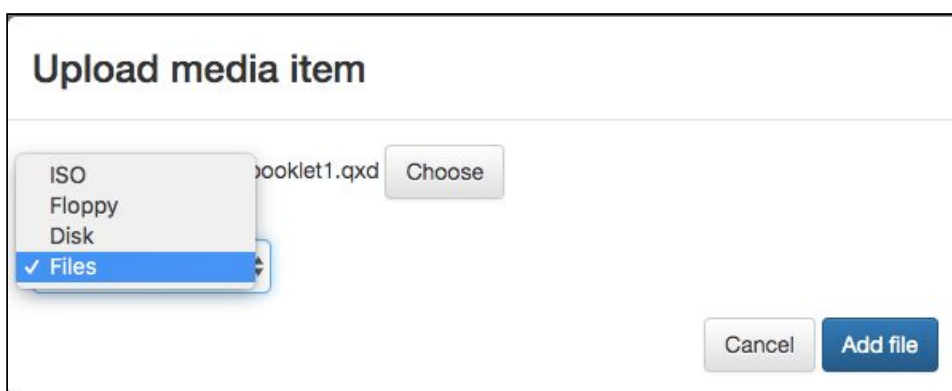
1.

The first step is to import a new object. From the main dashboard, navigate to the “Objects” tab. To add a new object use the “Import Object” button. All new objects imported from a local file system will end up in the “Local Object Archive”.





From the “Object Upload” page, add your file of choice. The “Add File” button opens a dialogue box which prompts the user for input. Select the desired file and then choose the Media Type from the drop-down menu. The Media Type for the individual QXD files is “Files”.



The “Add File” operation can be repeated to include multiple files in the one import. The EaaSI framework will package all file uploads into an ISO image for ease of use within the emulated environment.

Object Upload

From File [From Repository](#)

Object label

Files

- **booklet1.qxd** -- MediaType: Q82753 [delete](#)
- **HUNTING3.qxd** -- MediaType: Q82753 [delete](#)
- **Ateliers_A5.qxd** -- MediaType: Q82753 [delete](#)

[Add File](#)

[Cancel](#) [Import](#)

Back at the “Object Upload” page, add an Object Label for the file and click the “Import” button. You will see the system working on the import and, if successful, be returned to the “Objects” page. There should also be a green success notification in the top right corner of the screen. To double check that the import worked, click into the Local Object Archive to see if the new object is listed.

2.

Now that you have the QXD files imported as an object, you can test it out in an emulation instance. Starting from the Local Object Archive, you can assign specific rendering environments to our objects. Locate your object in the list and select from the “Choose action” drop-down menu: “Choose action → Details”

The Object Details page has a drop-down section for Rendering Environments. If you know which environment you want to use for these particular objects you can use the “Add Environment” button to select the correct one, in this case the “MacOS_8.5 + QuarkXpress_4.1” environment.

Object Details

[Save](#) [Reset](#) [Cancel](#)

Object Details

Object ID: 9df749ae-3cd4-45b2-8aa7-3445a54cd12c

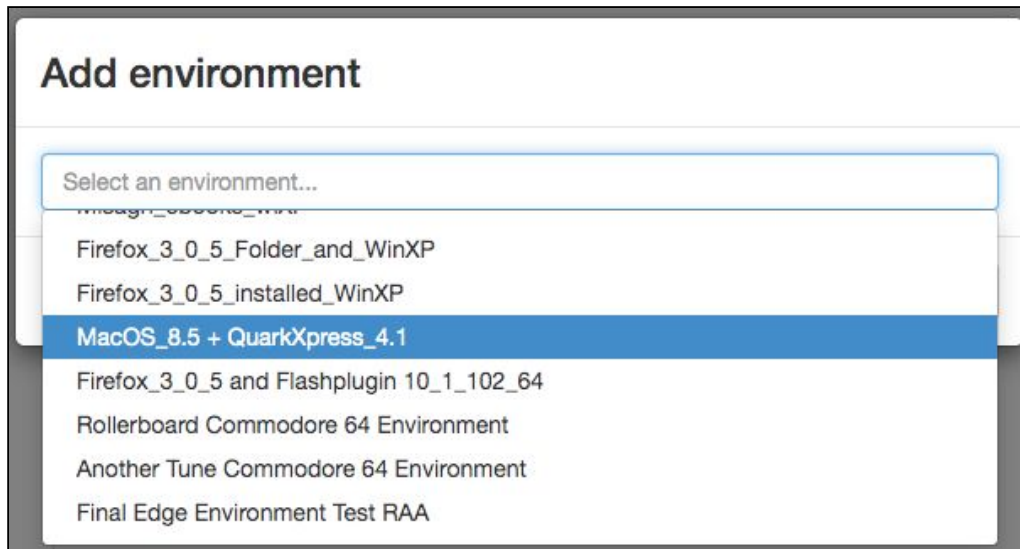
Object Label: test

☐ Object is software

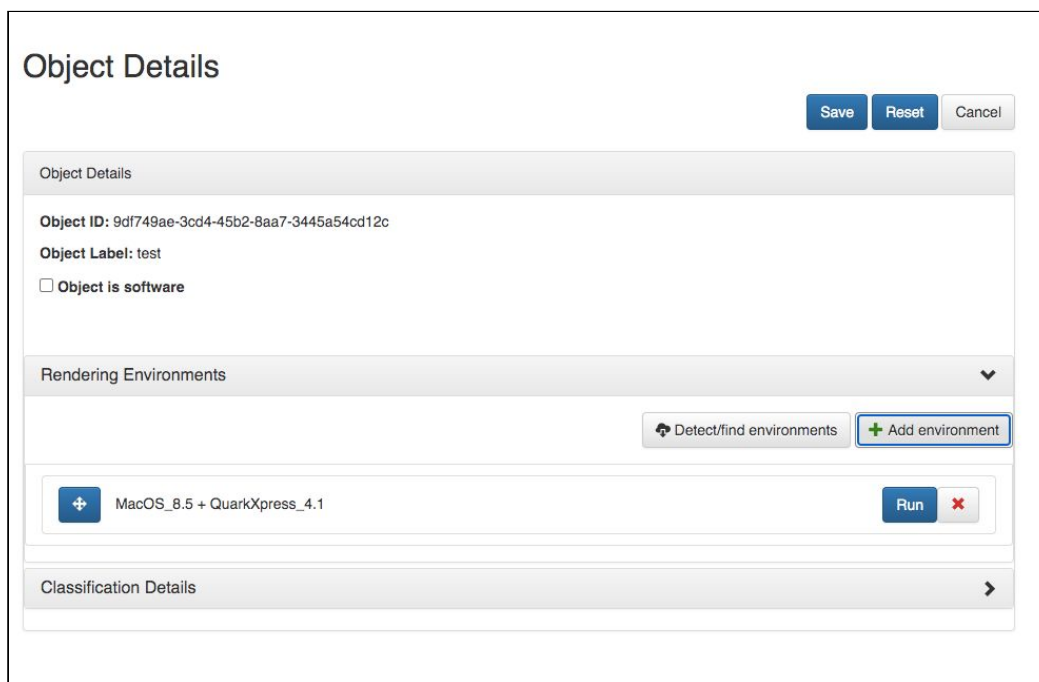
[Rendering Environments](#)

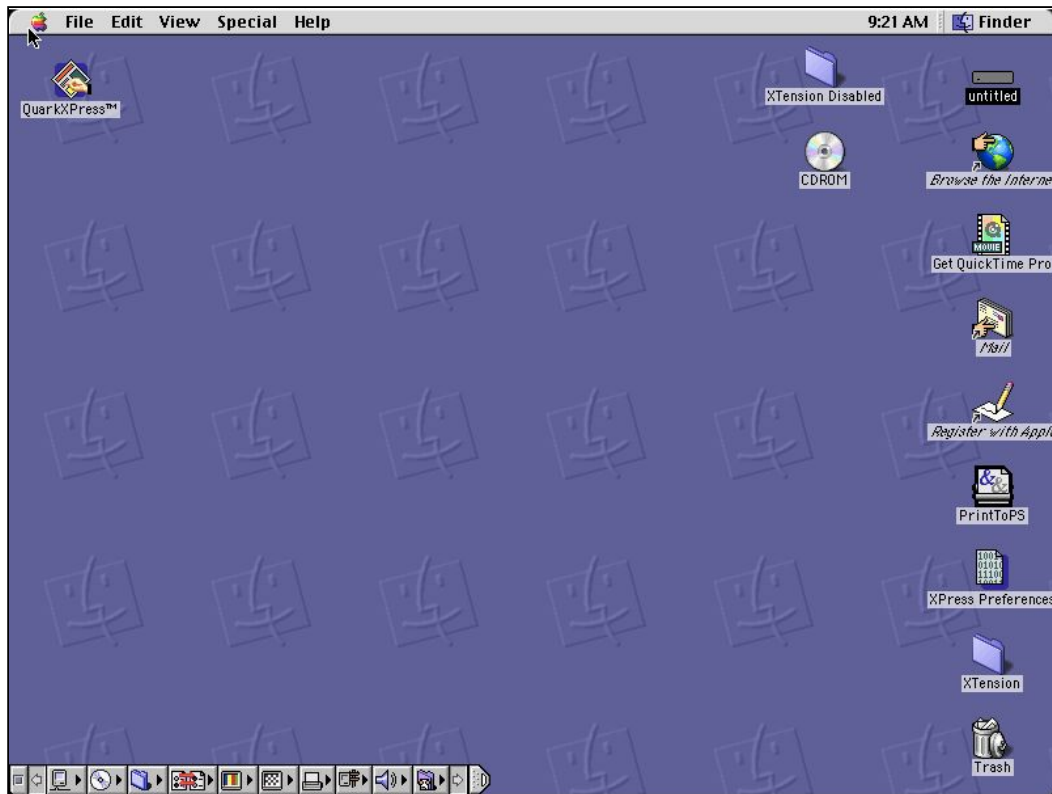
[Detect/find environments](#) [+ Add environment](#)

Classification Details [>](#)



Adding the “MacOS_8.5 + QuarkXpress_4.1” environment presents the user with the option to run the object in an emulation instance directly from the Object Details page. The “Run” button will prepare a new emulation session. Once running, please ensure to click inside the emulation window. This will enable relative mouse control. To leave the relative mouse control press the “Esc” key. From here the user can interact with and assess the object.



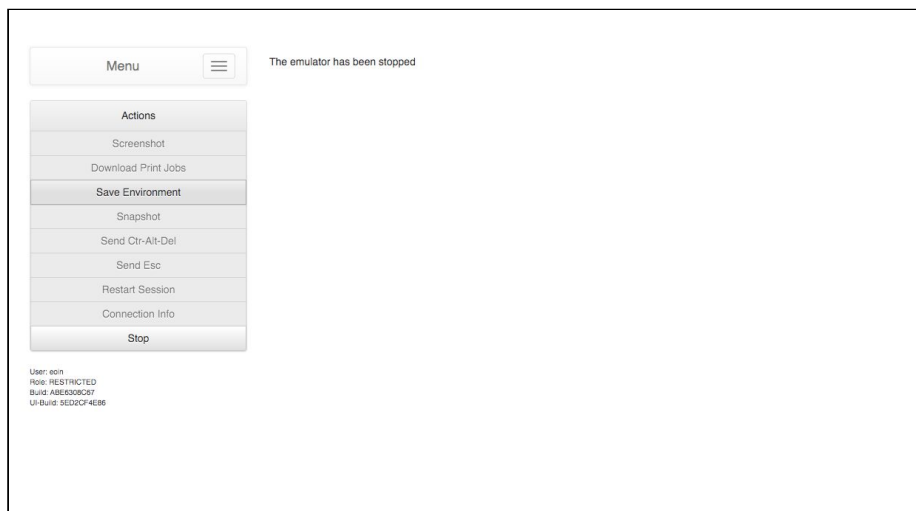
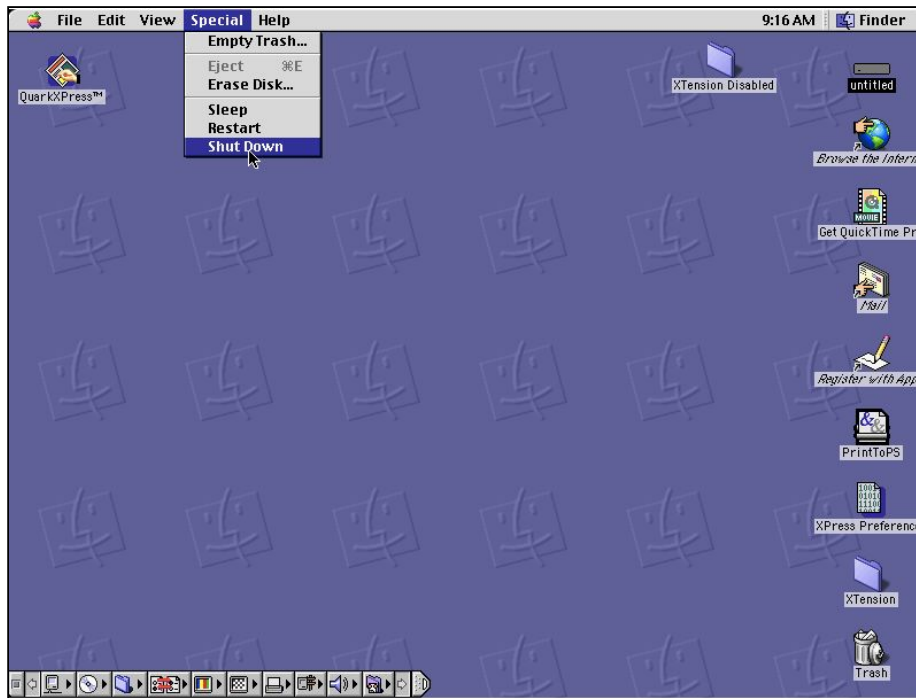


The files are best accessed by first launching the QuarkXpress software and then using the “File” -> “Open” method. The location of the files will be “CDROM”. From this point you will be able to use all the tools available in QuarkXpress. Some toolbars may not appear as default, so spend some time exploring the menu bar along the top of the screen.

3.

When you are happy with the look and feel of the emulated environment, you can save it for future use. This saves the trouble of having to reassign rendering environments each time you launch a new session.

To save a new Object Environment you first need to cleanly shut down the emulated environment. This can be done through the “Special” menu. Once the emulated Operating System has been shut down you will be notified that the emulator has stopped and presented with the option to “Save Environment” in the left hand menu.



When “Save Environment” is selected, a dialogue box is displayed to allow you to add some descriptive information about the new environment. When this is saved, the user is returned to the Environments page. For future alterations to the environment, or to ensure that changes to files persist, you can follow the same steps, choosing to save the environment as a revision or as a new Object Environment.

Save changes

Name




MacOS_8.5 + QuarkXpress_4.1 + Test_Files

Description

H1 H2 H3 H4 H5 H6 P pre ”

B **I** **U** **S** **≡** **≡** **↺** **↻** **⊗**

≡ **≡** **≡** **≡** **≡** **≡**

</>    Words: 3 Characters: 17

Test files added.

Save **Cancel**

The new environment can be located in the “Object Environments” tab and can easily be launched via the Run Environment button in the Actions drop-down menu.

Environments

[Virtual machines](#) [Object Environments](#)

Number of Environments: 8

Page Size: 10

Search...

	Name ↑	ID	Own...	ObjectID	Actions
<input type="checkbox"/>	Another Tune Commodore 64 Environment	ed043...	shared	8dc0ca41-b6d3-4...	Choose action ▾
<input type="checkbox"/>	Final Edge Environment Test RAA	3311aa...	shared	07242252-5ce0-4...	Choose action ▾
<input type="checkbox"/>	Firefox_3_0_5 and Flashplugin 10_1_102_64	7166b...	shared	f69fcb26-aec8-40...	Choose action ▾
<input type="checkbox"/>	Firefox_3_0_5_Folder_and_WinXP	80d35...	shared	cbe633bd-fed7-46...	Choose action ▾
<input type="checkbox"/>	Firefox_3_0_5_installed_WinXP	8a7a5...	shared	cbe633bd-fed7-46...	Choose action ▾
<input type="checkbox"/>	MacOS_8.5 + QuarkXpress_4.1 + Test_Files	282de...	shared	679aef8d-d31f-43...	Choose action ▾
<input type="checkbox"/>	Misagh_ebooks_wiXP	39e711...	shared	14e12c80-a554-48...	Run Environment Details Delete Add Software
<input type="checkbox"/>	Rollerboard Commodore 64 Environment	1f848...	shared	16b3e871-81c2-4c...	

[1] to [8] of [8] < >

9. Credits

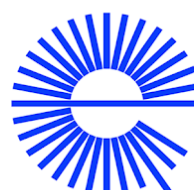
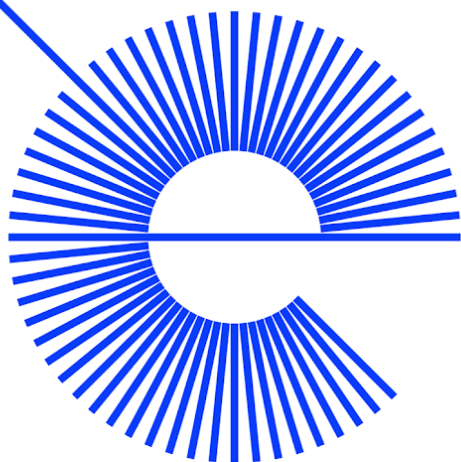
Eoin O'Donohoe, author of this report, is a Digital Preservation Analyst at Beeld en Geluid. He is currently working on areas of research that aim to lower the threshold for institutions looking to make a start on software archiving.

eodonohoe@beeldengeluid.nl

About this publication

This report was published by the Dutch Digital Heritage Network (NDE) in October 2020.
For further information, see: netwerkdigitaalerfgoed.nl

If you have any queries or comments about the contents of the report, please feel free to email them to: info@netwerkdigitaalerfgoed.nl



**dutch digital
heritage
network**